K_medians.m

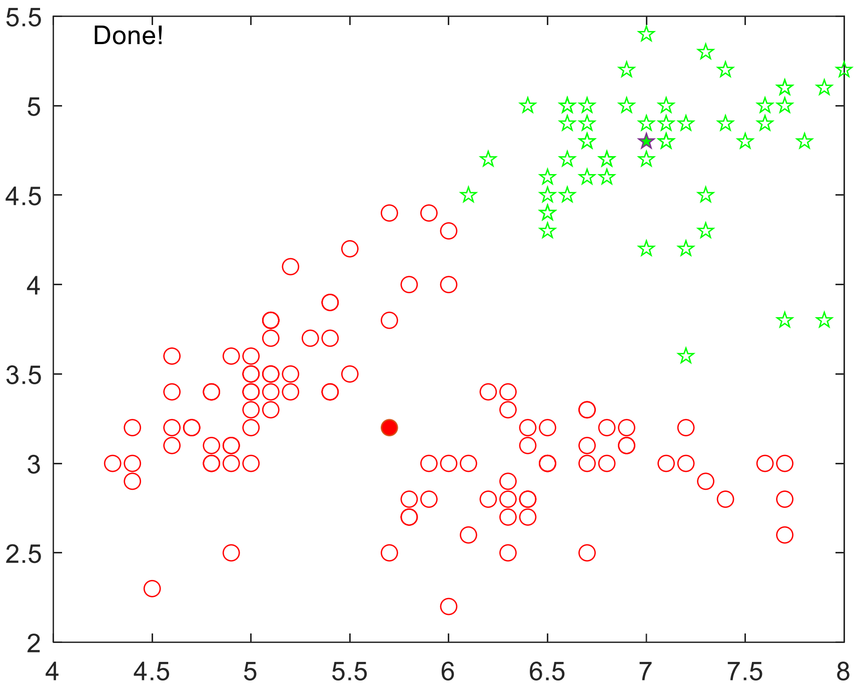
```
function [membership, centres] = k_medians(X, n_cluster)
% X: the data matrix, rows are data points and columns are features
% n_cluster: number of cluster

if n_cluster > 4
    disp ('You have set too many clusters.');
    disp ('Set the number of clusters to be 1-4.');
    disp ('The program and visualization allow for up to 4 clusters.');
    return;
end

% Initialize the figure
figure('position', [200, 200, 600, 500]);

% Get the number of data points and number of features
[n_sample, n_feat] = size(X);

% Randomly initialize the starting cluster centres.
rng('shuffle');
up_bound = max(X);
lw_bound = min(X);
% "centres" is an n_cluster-by-n_feat matrix.
centres = lw_bound + (up_bound-lw_bound).*rand(n_cluster, n_feat);

disp('Start K-means clustering ... ');

% Initialization:
% In the begining, all data points are in cluster 1
% The "old_membership" variable is an n_sample-by-1 matrix.
% It saves the cluster id that each data point belongs to.
% Again, in the begining, all data points are in cluster 1
old_membership = ones(n_sample, 1);

% Display the initial cluster membership for all datapoints
% and the initial cluster centres
show(X, old_membership, n_cluster, centres, 'Cluster centres initialized!')

while true
    distance = pdist2(X, centres,"cityblock");

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % You need to add code here.
    % E step: Assign data points to closest clusters.
    % Specifically, for each data point, find closest
    % cluster centre, and assign the data point
    % to that cluster.
    [~, membership] = min(distance, [], 2);

    %Show the result of the E step.
    show(X, membership, n_cluster, centres, 'E step finished: Datapoints re-assigned!')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % You need to add code here.
    % M step: Update cluster centres based on the new assignment.
    for j = 1:n_cluster
        centres(j, :) = median(X(membership == j, :));
    end
    %Show the result of the M step.
    show(X, membership, n_cluster, centres, 'M step finished: Cluster centers updated!')

    % Stop if no more updates.
    if sum(membership ~= old_membership)==0
        show(X, membership, n_cluster, centres, 'Done! ');
        break;
    end

    old_membership = membership;
end
end

function show(X, c_pred, n_cluster, centres, txt)
    symbol = ['ro'; 'gp'; 'bd'; 'k^'; 'r*'];
    hold off;

    for i = 1:n_cluster
        marker = mod(i,5);
        if i > 4
            disp('Total number of clusters exceeds 4, some symbols in the plot are reused!');
        end
        plot(X(c_pred==i, 1), X(c_pred==i, 2), symbol(marker,:));
        hold on;
        plot(centres(i, 1), centres(i, 2), symbol(marker,2), 'MarkerFaceColor',symbol(marker,1));
    end
    text(4.2, 5.4, txt);
    drawnow;

    pause(2);
end
```

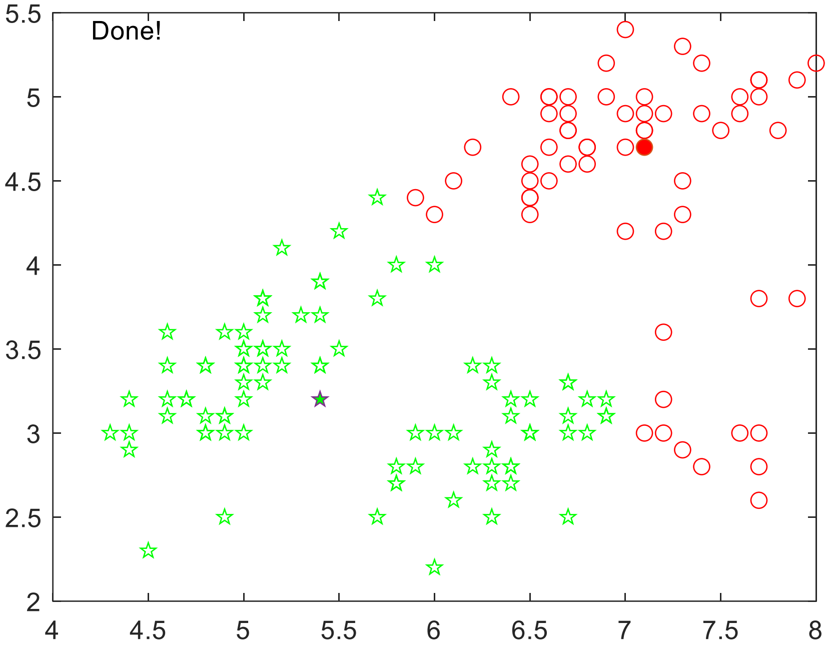
k_medians_main.m

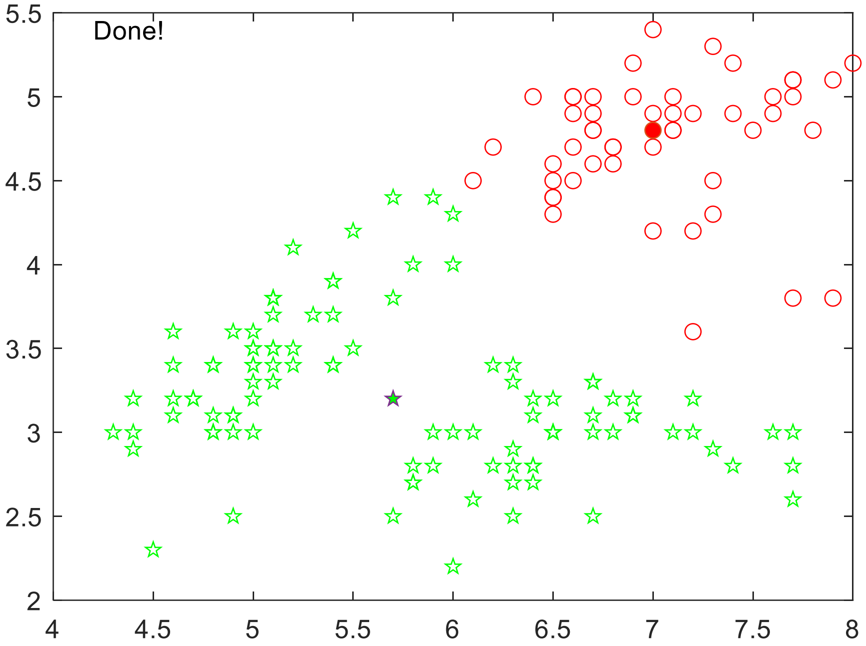
```
clear all;
load data;
k_medians(data, 2);
k_medians(data, 2);
k_medians(data, 2);
k_medians(data, 3);
k_medians(data, 3);
k_medians(data, 3);
k_medians(data, 4);
k_medians(data, 4);
k_medians(data, 4);
```

OUTPUT

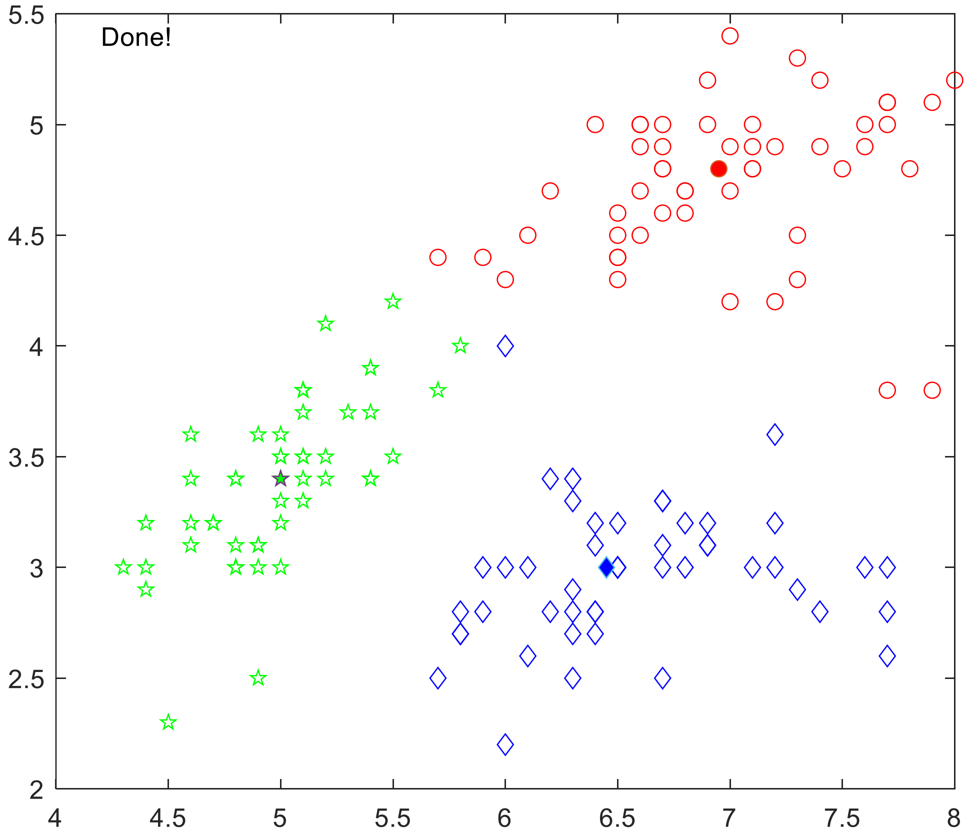
K=2

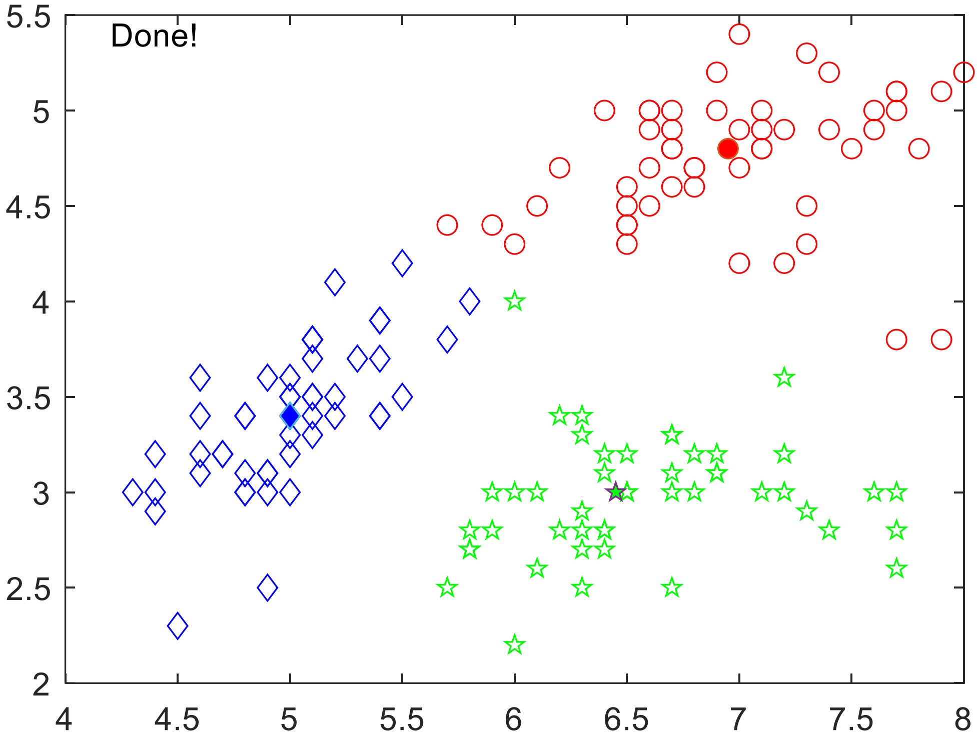

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

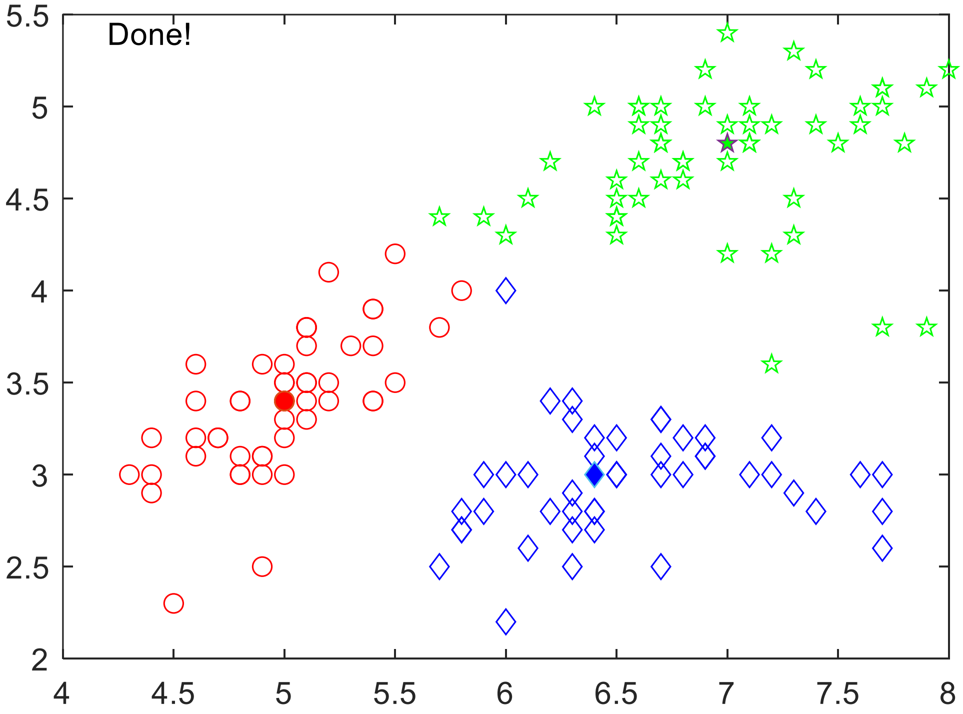

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

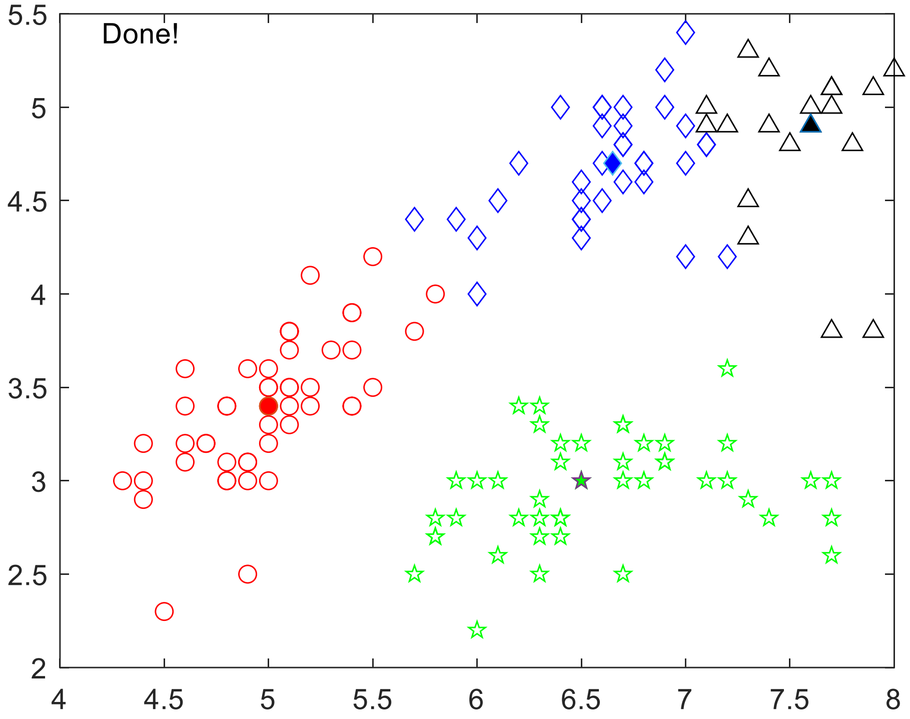
K=3

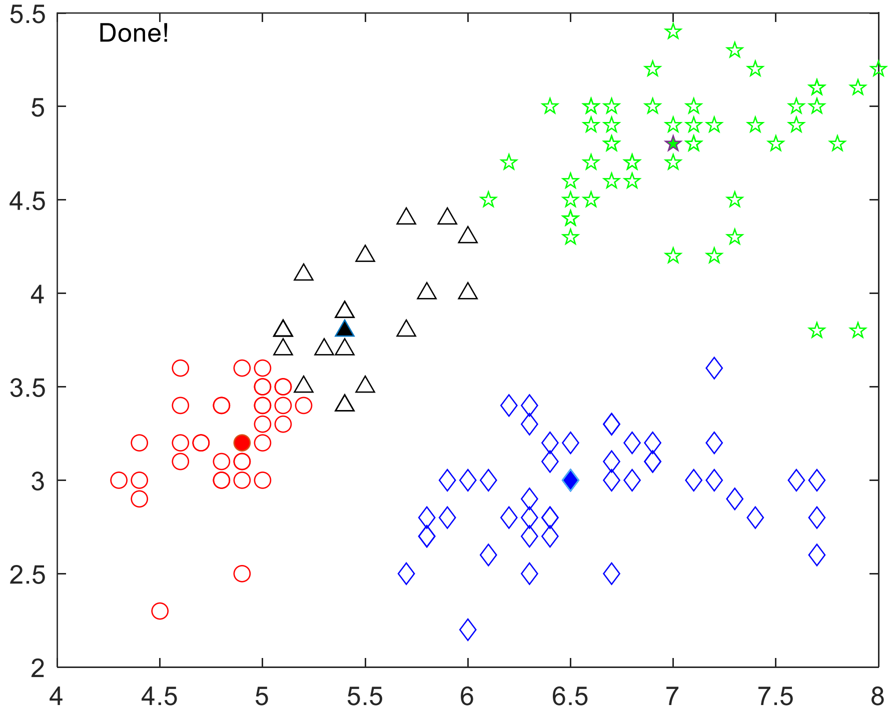

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

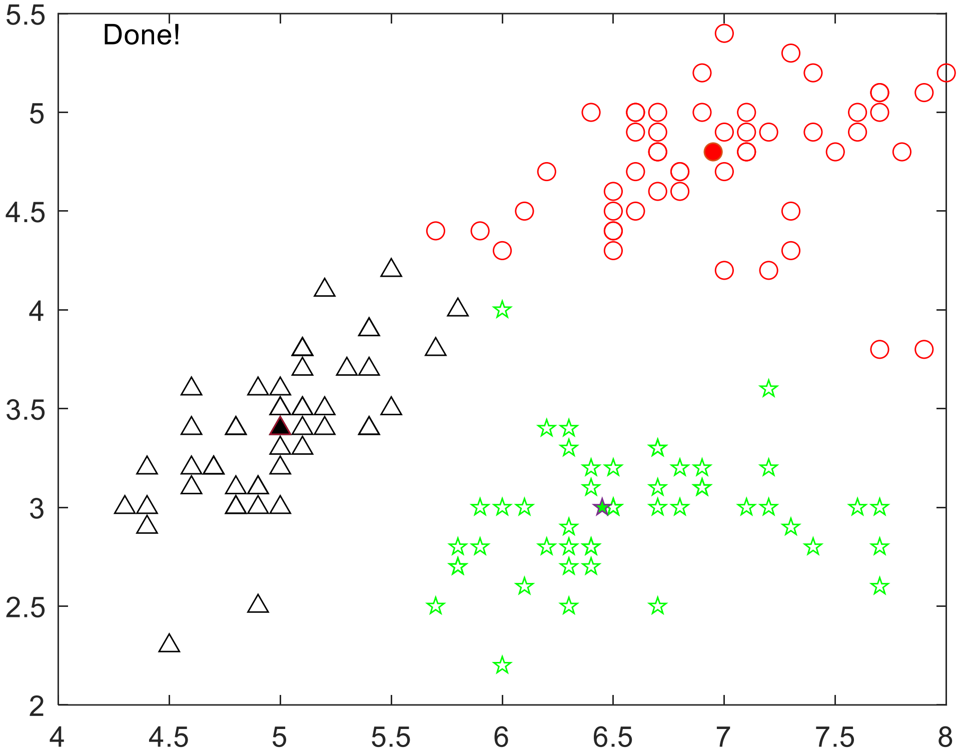

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

K=4


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8



k_medoids.m

```
function [membership, centres] = k_medoids(X, n_cluster)
% X: the data matrix, rows are data points and columns are features
% n_cluster: number of cluster

if n_cluster > 4
    disp ('You have set too many clusters.');
    disp ('Set the number of clusters to be 1-4.');
    disp ('The program and visualization allow for up to 4 clusters.');
    return;
end

% Initialize the figure
figure('position', [200, 200, 600, 500]);

% Get the number of data points and number of features
[n_sample, n_feat] = size(X);

% Initialize the starting cluster centres with fixed values.
% centres = [5.5, 4; 4.5, 3.2; 6.5, 3.5];

% Randomly initialize the starting cluster centres.
rng('shuffle');
centres = datasample(X,n_cluster);


disp('Start K-medoids clustering ... ');

% Initialization:
% In the begining, all data points are in cluster 1
% The "old_membership" variable is an n_sample-by-1 matrix.
% It saves the cluster id that each data point belongs to.
% Again, in the begining, all data points are in cluster 1
old_membership = ones(n_sample, 1);
```

```matlab
% Display the initial cluster membership for all datapoints
% and the initial cluster centres
show(X, old_membership, n_cluster, centres, 'Cluster centres initialized!')


while true
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % You need to add code here.
    % Calculate squared Euclidean distances
    % between every data point and every cluster centre.
    % Please put your results in an
    % n_sample-by-n_cluster matrix named as "distance".
    % You may find the Matlab function pdist2 to be useful here.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    distance = pdist2(X, centres,"cityblock");

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % You need to add code here.
    % E step: Assign data points to closest clusters.
    % Specifically, for each data point, find closest
    % cluster centre, and assign the data point
    % to that cluster.
    % Save your assignment to the variable "membership",
    % which is an n_sample-by-1 vector and each row saves
    % the cluster membership of a datapoint.
    % (See the description of "old_membership" as well.)
    % You should use here the "distance" computed above.
    % You may find the function "min" useful here,
    % but feel free not to use it and write whatever code
    % you think can implement the above requirement.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [~, membership] = min(distance, [], 2);
    %Show the result of the E step.
    show(X, membership, n_cluster, centres, 'E step finished: Datapoints re-assigned!')

    for j = 1:n_cluster
         point = X(membership == j,:);
         pointSize = size(point,1);
         disS = zeros(1,pointSize);
         for i = 1:pointSize
             disS(1,i) = sum(pdist2(point,point(i,:),"cityblock"));
         end
        [~, min_num] = min(disS);
        centres(j,:) = point(min_num,:);
    end

    %Show the result of the M step.
    show(X, membership, n_cluster, centres, 'M step finished: Cluster centers updated!')

    % Stop if no more updates.
    if sum(membership ~= old_membership)==0
        show(X, membership, n_cluster, centres, 'Done! ');
        break;
    end

    old_membership = membership;
end
```

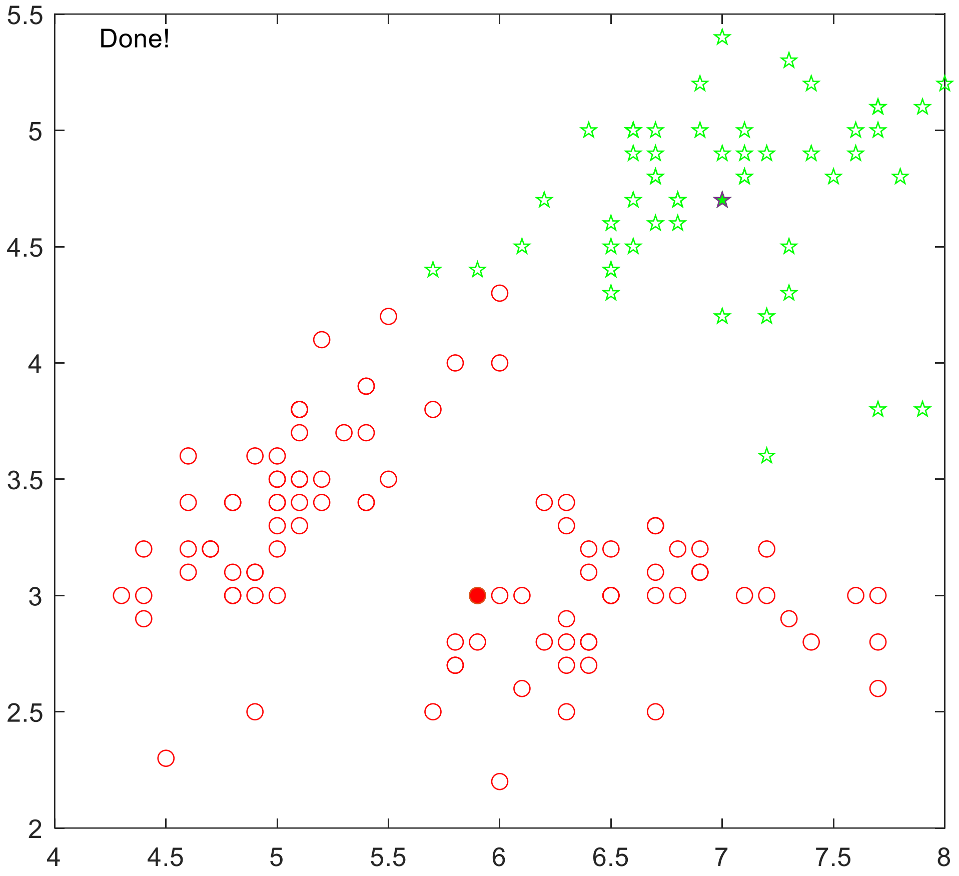
```
end

function show(X, c_pred, n_cluster, centres, txt)
    symbol = ['ro'; 'gp'; 'bd'; 'k^'; 'r*'];
    hold off;

    for i = 1:n_cluster
        marker = mod(i,5);
        if i > 4
            disp('Total number of clusters exceeds 4, some symbols in the plot
are reused!');
        end
        plot(X(c_pred==i, 1), X(c_pred==i, 2), symbol(marker,:));
        hold on;
        plot(centres(i, 1), centres(i, 2), symbol(marker,2),
'MarkerFaceColor',symbol(marker,1));
    end
    text(4.2, 5.4, txt);
    drawnow;

    %Pause some time here.
    %Used to show figure with enough time.
    %You can change the pause time.
    pause(0.5);
end
```

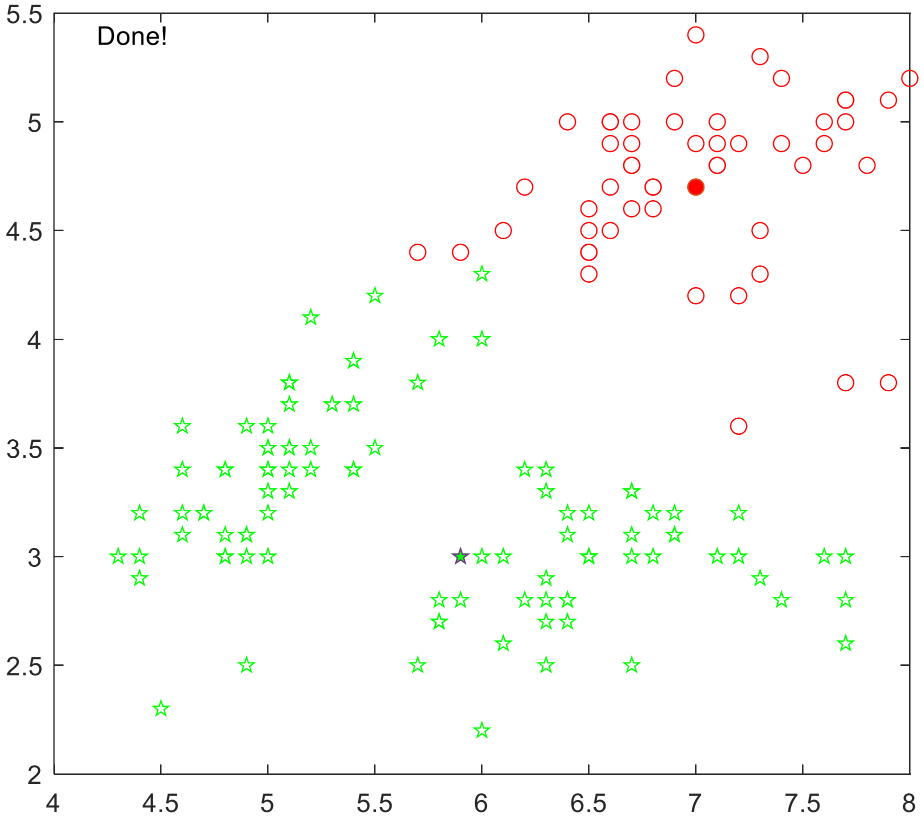
k_medoids_main.m

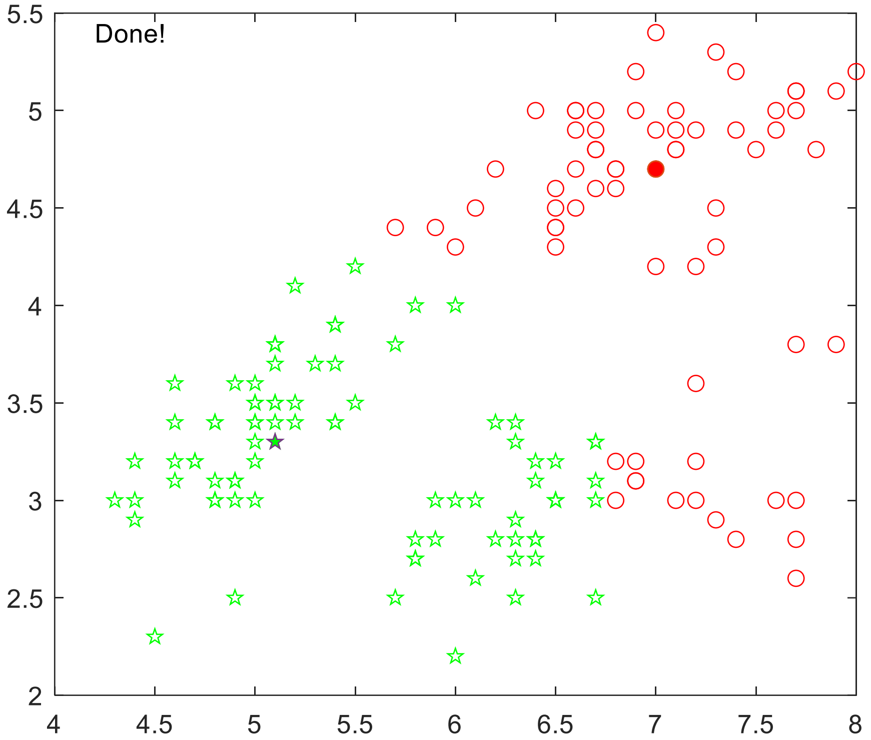
```
clear all;
load data;
k_medoids(data, 2);
k_medoids(data, 2);
k_medoids(data, 2);
k_medoids(data, 3);
k_medoids(data, 3);
k_medoids(data, 3);
k_medoids(data, 4);
k_medoids(data, 4);
k_medoids(data, 4);
```

OUTPUT

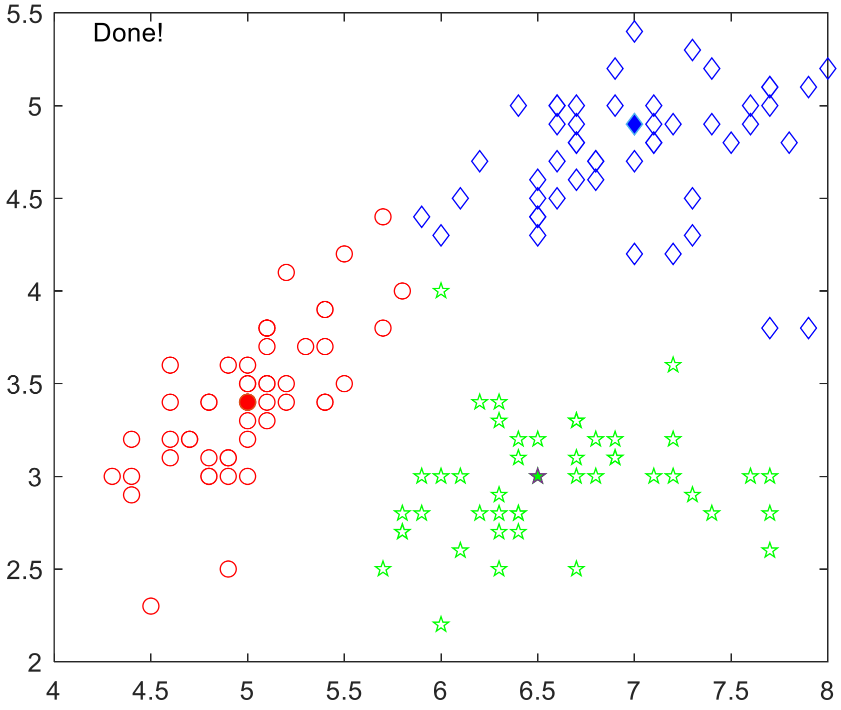
K=2

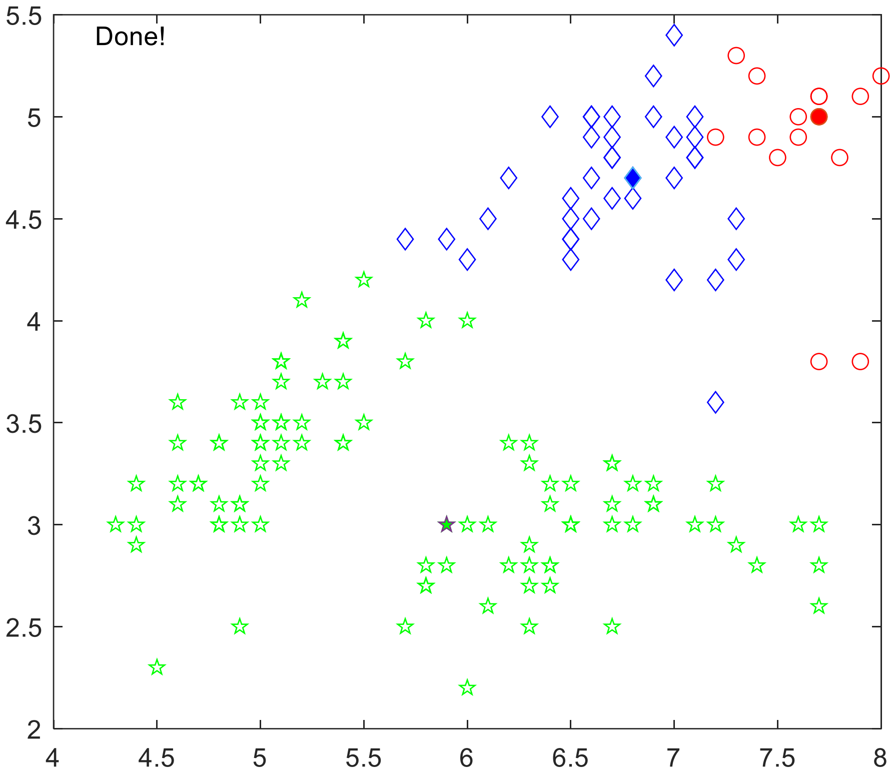

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

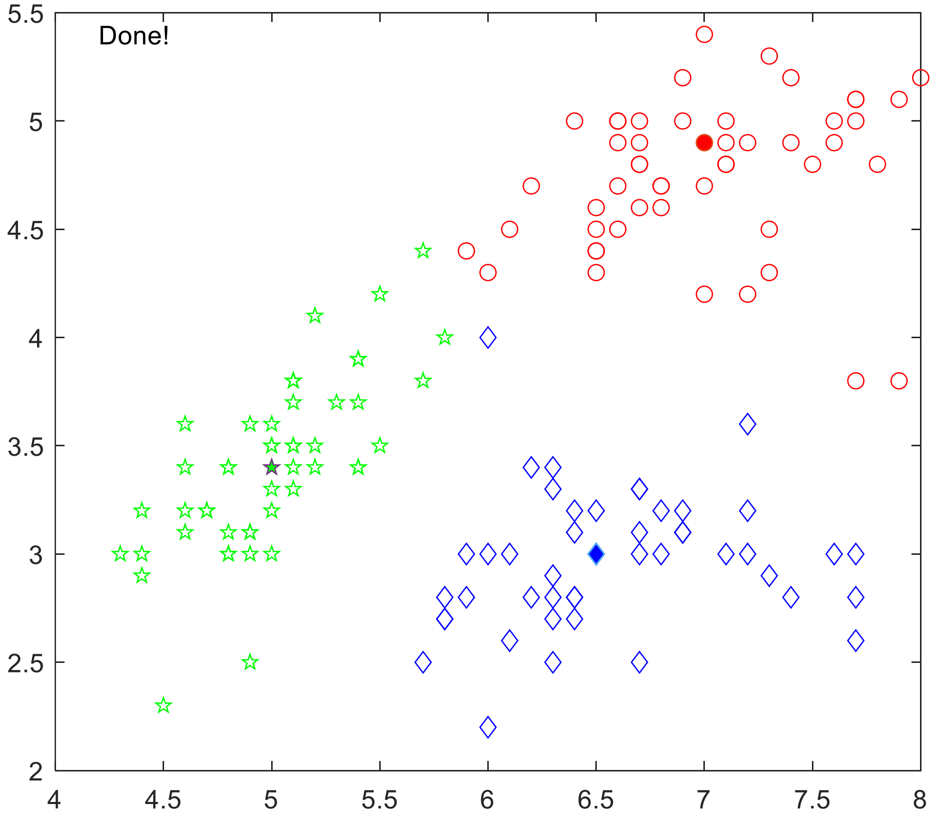

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

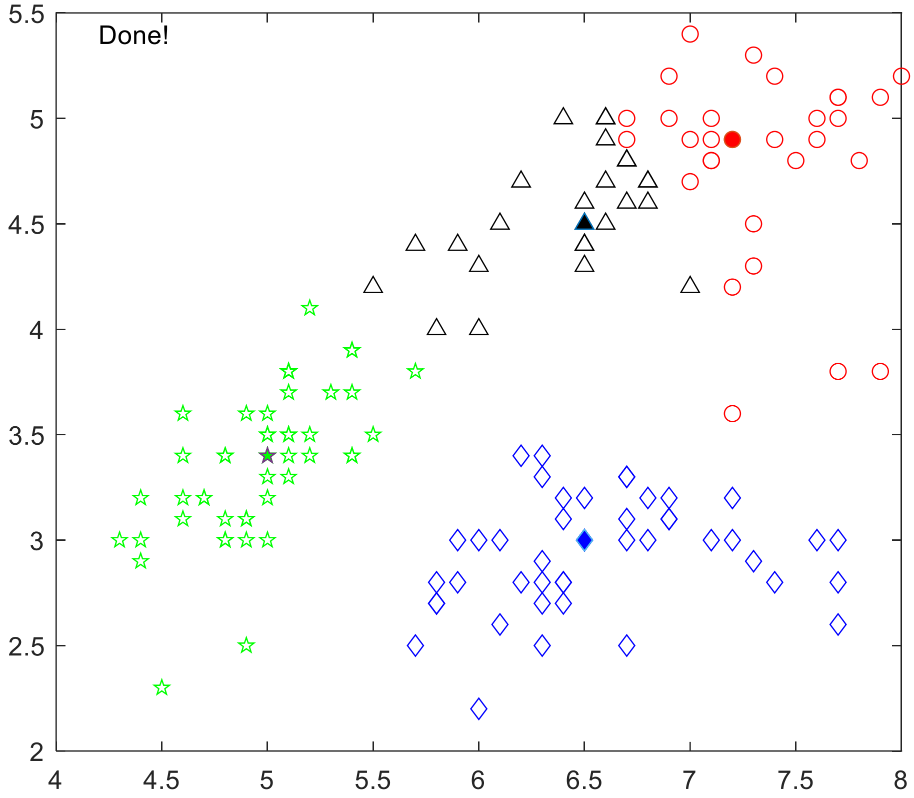
K=3

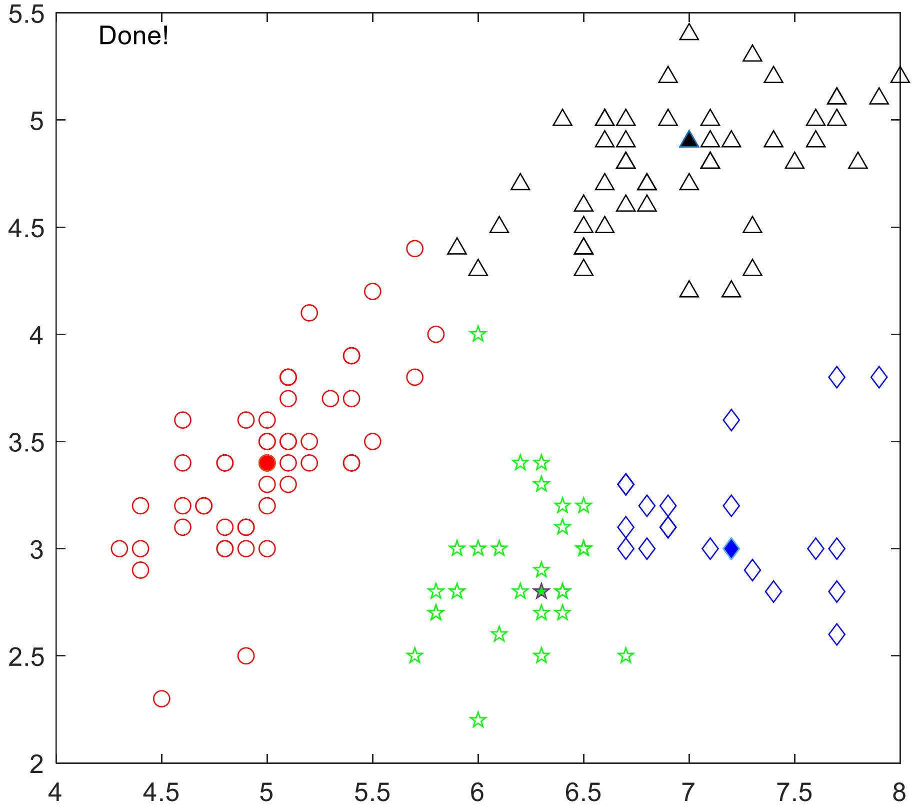

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

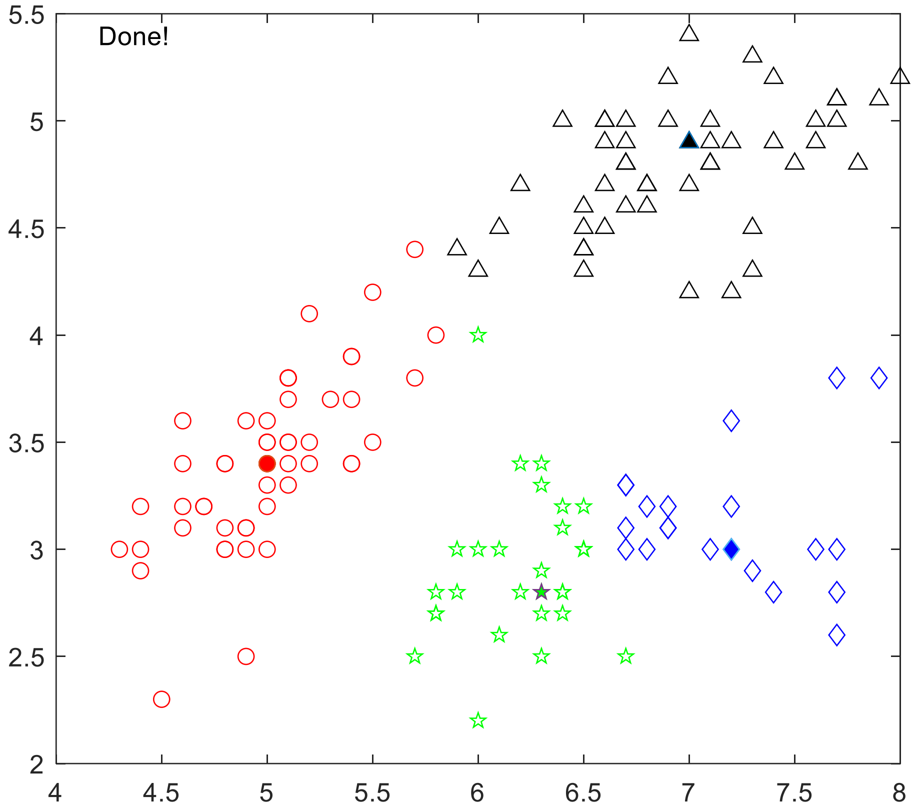

Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8

K=4


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8


Done!
5.5
5
4.5
4
3.5
3
2.5
2
4
4.5
5
5.5
6
6.5
7
7.5
8