EECS 4404

Assignment 3

Name: Bochao Wang

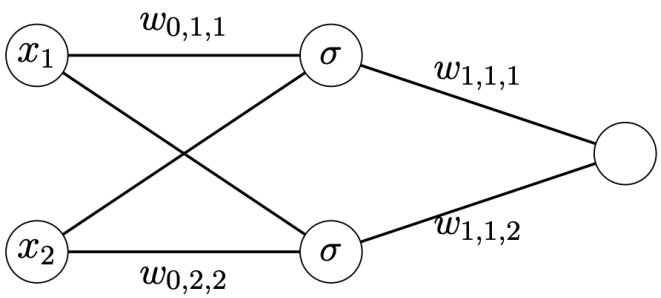
Student ID: 215237902

Prism: bochao

Date: March. 31th

1. Backpropagation

Consider a neural network with one hidden layer containing two nodes, input dimension 2 and output dimension 1. That is, the fist layer contains two nodes $v_{0,1}$, $v_{0,2}$, the hidden layer has two nodes $v_{1,1}$, $v_{1,2}$, and the output layer one nodes $v_{2,1}$. All nodes between consecutive layers are connected by an edge. The weights between node $v_{t,i}$and$v_{t+1,j}$ is denoted by $w_{t,j,i}$ as (partially) indicated here: The nodes in the middle layer apply a differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$, which has derivative σ'.



(a) The network gets as input a 2-dimensional vector $x = (x_1, x_2)$. Give an expression for the output $N(x)$ of the network as a function of x_1, x_2 and all the weights.

- Solve:
  - $o_{1,1}(\boldsymbol{x}) = \sigma(x_1 w_{0,1,1} + x_2 w_{0,2,1})$
  - $o_{1,2}(\boldsymbol{x}) = \sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2})$
  - $N(\boldsymbol{x}) = o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2} = \sigma(x_1 w_{0,1,1} + x_2 w_{0,1,2}) w_{1,1,1} + \sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2}) w_{1,1,2}$

(b) Assume we employ the square loss. Give an expression for the loss $l(N(\cdot), (\boldsymbol{x}, t))$ of the network on an example $(\boldsymbol{x}, t)$ (again, as a function of x_1, x_2, t and all the weights).

- Solve:
  - $l(N(\cdot), (\boldsymbol{x}, t)) = \frac{1}{2}||N(x) - t||^2$
  - $l(N(\cdot), (\boldsymbol{x}, t)) = \frac{1}{2}||(\sigma(x_1 w_{0,1,1} + x_2 w_{0,1,2}) w_{1,1,1} + \sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2}) w_{1,1,2}) - t||^2$

(c) Consider the above expression of the loss as a function of the set of weights
$L(w_{0,1,1}, w_{0,2,1}, w_{0,1,2}, w_{0,2,2}, w_{1,1,1}, w_{1,1,2}) = l(N(\cdot), (\boldsymbol{x}, t))$. Compute the 6 partial derivatives

- Solve:
  - $\frac{\partial L}{\partial w_{1,1,1}} = ((\sigma(x_1 w_{0,1,1} + x_2 w_{0,1,2}) w_{1,1,1} + \sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2}) w_{1,1,2}) - t)(\sigma(x_1 w_{0,1,1} + x_2 w_{0,1,2}) w_{1,1,1} + \sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2}) w_{1,1,}$
    $= ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t)((\sigma(x_1 w_{0,1,1} + x_2 w_{0,1,2}) w_{1,1,1})' + (\sigma(x_1 w_{0,2,1} + x_2 w_{0,2,2}) w_{1,1,2})')$
    $= ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) o_{1,1}$
  - $\frac{\partial L}{\partial w_{1,1,2}} = ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) o_{1,2}$
  - $\frac{\partial L}{\partial w_{0,1,1}} = ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) \sigma'(a_{1,1}) x_1 w_{1,1,1}$
  - $\frac{\partial L}{\partial w_{0,1,2}} = ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) \sigma'(a_{1,1}) x_2 w_{1,1,1}$
  - $\frac{\partial L}{\partial w_{0,2,1}} = ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) \sigma'(a_{1,2}) x_1 w_{1,1,2}$
  - $\frac{\partial L}{\partial w_{0,2,2}} = ((o_{1,1} w_{1,1,1} + o_{1,2} w_{1,1,2}) - t) \sigma'(a_{1,2}) x_2 w_{1,1,2}$

2. k-means clustering

(a) Implement the k-means algorithm and include your code in your submission

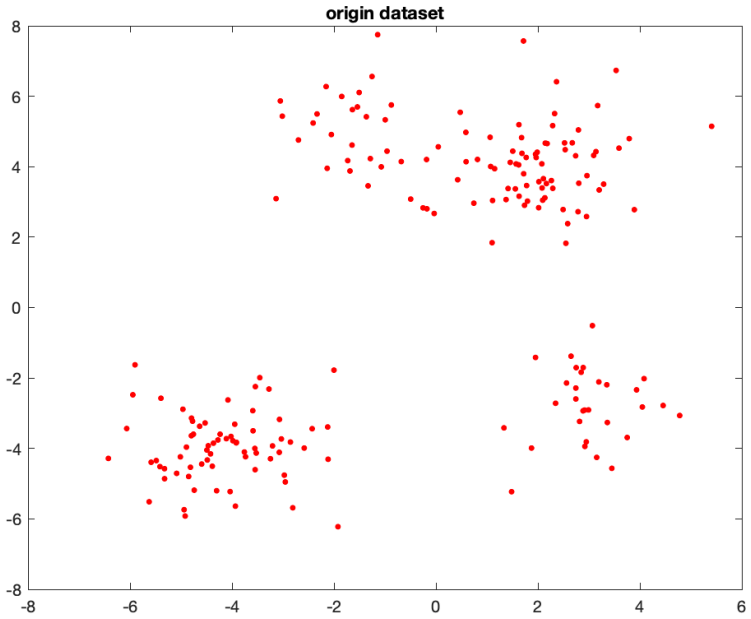
- `k_means_alg.m`

```
function [C, cost] = k_means_alg(D,k,init,init_centers)
% N: the number of points
```

```
% d: the dimension of each point
[N, d] = size(D);

% C : indecate each points belong to which class (k=1,2,..or K)
C = zeros(N,1);

% three ways to init:
% 1. simply set the inital centers by hand
% 2. choose k datapoints uniformly at random from the dataset
% 3. choose the first center uniformly at random, and then choose
%    each next center to be the datapoint that maximizes the sum of
%    (euclidean) distances from the previous datapoints
centers = zeros(k, d); % init k-centers
if (strcmpi(init, 'manual'))

    % generate the random number in [p_min,p_max] area
    centers = init_centers;
elseif (strcmpi(init, 'uniform'))
    for i = 1:k
        j = unidrnd(N); % choose index j uniformly at random
        centers(i,:) = D(j,:); % choose jth points uniformly at random
    end
elseif (strcmpi(init, 'euclidean'))
   % 1st center
   j = unidrnd(N); % choose index j uniformly at random
   centers(1,:) = D(j,:); % choose jth points uniformly at random as first center
   % 2nd center
   previous_point = centers(1,:);
   % cpmpute the eucliden distances
   distances = zeros(N,1);
   for n_p = 1:N
       distances(n_p,:) = norm(previous_point-D(n_p,:));
   end
   % find the max distance point
   max_i = find(distances==max(distances),1);
   % set this point as point
   centers(2,:) = D(max_i,:);
   % next centers
   for i = 3:k
       previous_centers = centers(1:i-1,:);
       % cpmpute the sum of eucliden distances
       distances = zeros(N,1);
       [n_c,~]=size(previous_centers);
       for c = 1:n_c
           for n_p = 1:N
               distances(n_p,:) = distances(n_p,:) + norm(previous_centers(c,:)-D(n_p,:));
           end
       end
       % find the max distance point
       max_i = find(distances==max(distances),1);
       % set this point as point
       centers(i,:) = D(max_i,:);
   end
end
% repeat until convergence
itr = 0;
max_itr = 100;
while 1
   % compute nearest point index
   for i = 1:N
```

```
        dists = zeros(k,1);
        for j = 1:k
            dists(j,:) = norm(D(i,:)-centers(j,:));
        end
        % mark this point which class belongs to
        C(i,1) = find(dists==min(dists),1);
    end
    % store old centers
    old_centers = centers;
    % update the center
    for i = 1:k
        neares_i = find(C(:,1)==i);
        centers(i,:) = mean(D(neares_i,:));
    end
    % stop when centers are not uptated
    if isequal(old_centers,centers)
        break;
    end
    % prevent from dead loop
    itr = itr + 1;
    if itr > max_itr
        break
    end
end
b_cost = 0;
for i = 1:N
    b_cost = b_cost + norm(D(i,:) - centers(C(i,:),:))^2;
end
cost = b_cost/N;
```

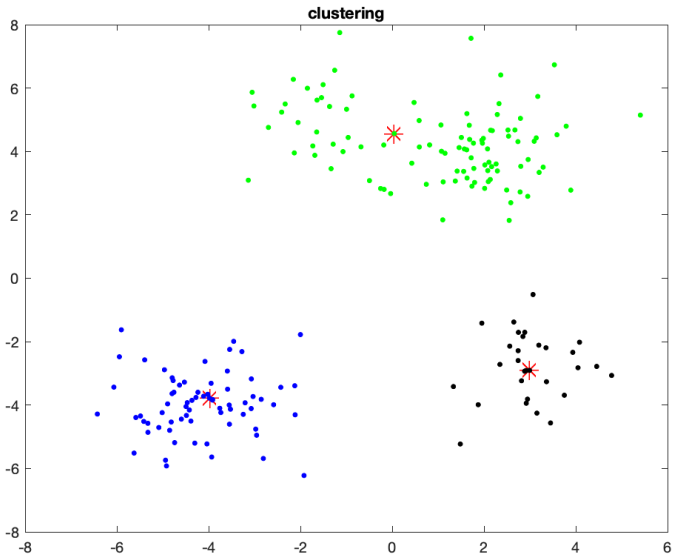
(b) Load the first dataset `twodpoints.txt`. Plot the datapoints.



Which number of clusters do you think would be suitable from looking at the points?

- **3** clusters is suitable from looking at the points.
- 1st instance

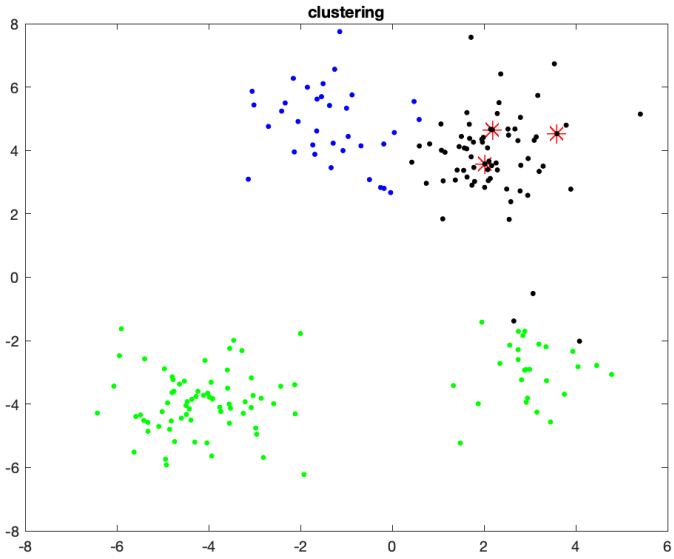
```
% init by hand
init_centers = zeros(3,d);
init_centers(1,:)=[0.04607 4.565];
init_centers(2,:)=[-3.983 -3.781];
init_centers(3,:)=[2.993 -2.907];
% set k clusters, and init method
k = 3;
init = 'manual';
% clustering
[cluster_i,~,~] = k_means_alg(twoD,k,init,init_centers);
```

- 

- 2st instance

```
% init by hand
init_centers = zeros(3,d);
init_centers(1,:)=[2.0188 3.574];
init_centers(2,:)=[2.181 4.6575];
init_centers(3,:)=[3.5908 4.5265];
% set k clusters, and init method
k = 3;
init = 'manual';
% clustering
[cluster_i,~,~] = k_means_alg(twoD,k,init,init_centers);
```

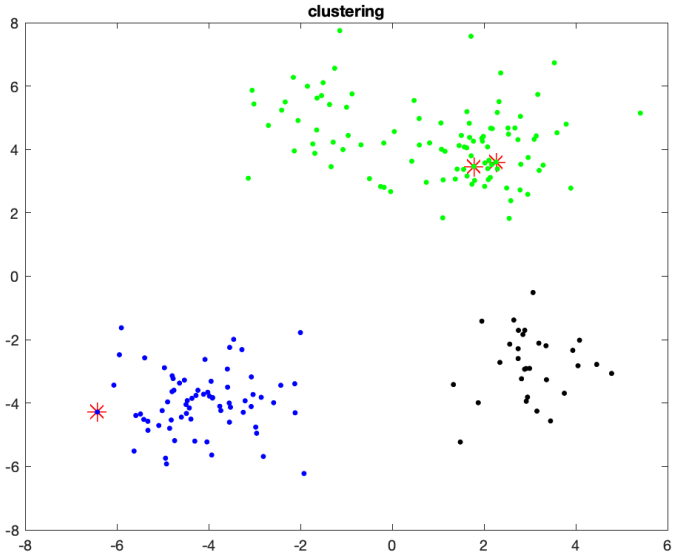
-



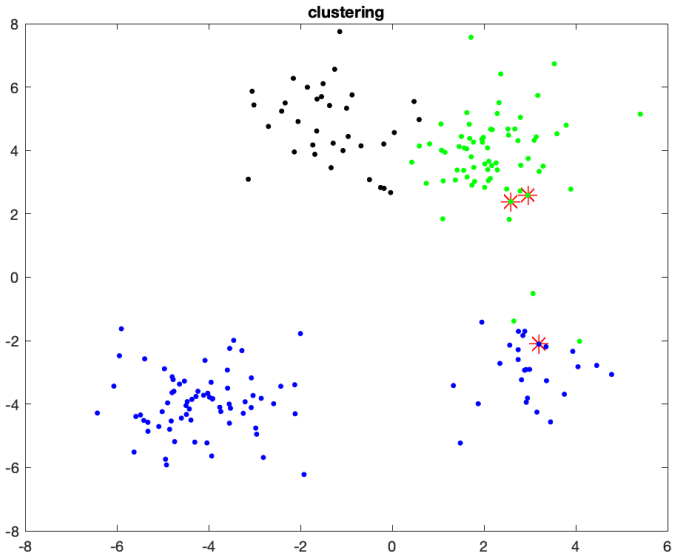
- When we inintialize the centres by hand, it would effect the clustering result, espessicially we initialize centres in same group. The result is very sancetive with choosing by hand.

(c) Can the above phenomenon also happen if the initial centers are chosen uniformly at random? What about the third way of initializing? For each way of initializing, run the algorithm several times and report your findings.

- Initial centers uniformly at random
- Fig 1

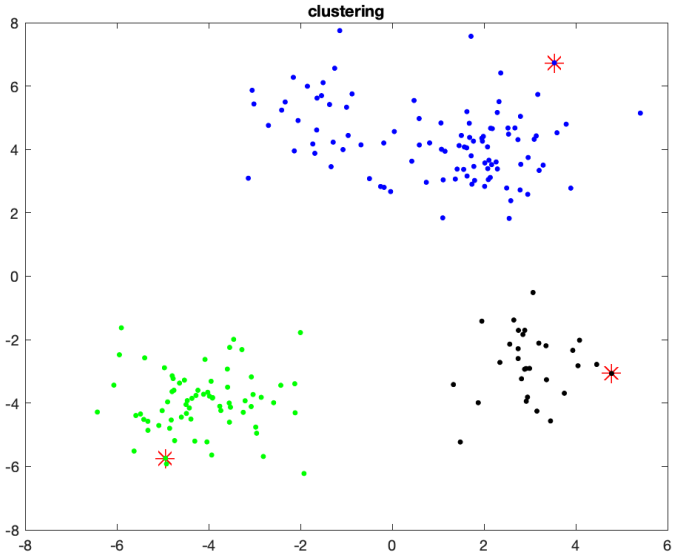


- Fig 2

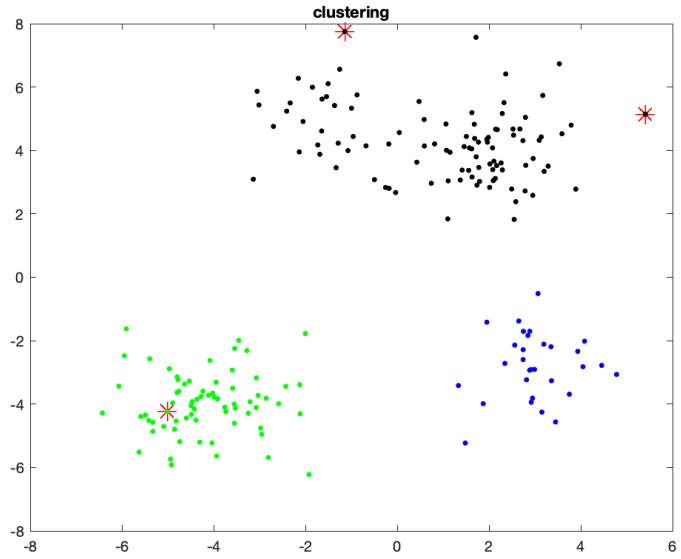


- When the initial centers are chosen uniformly at random, almost time the points would cluster as same as fig 1, in a few time, it would be different (like fig 2). The result of clustering would not very sencitive as previous part which choose by hand, but it would also happen miss clustering sometimes.

- Third way of initializing
- Fig 3



- Fig 4

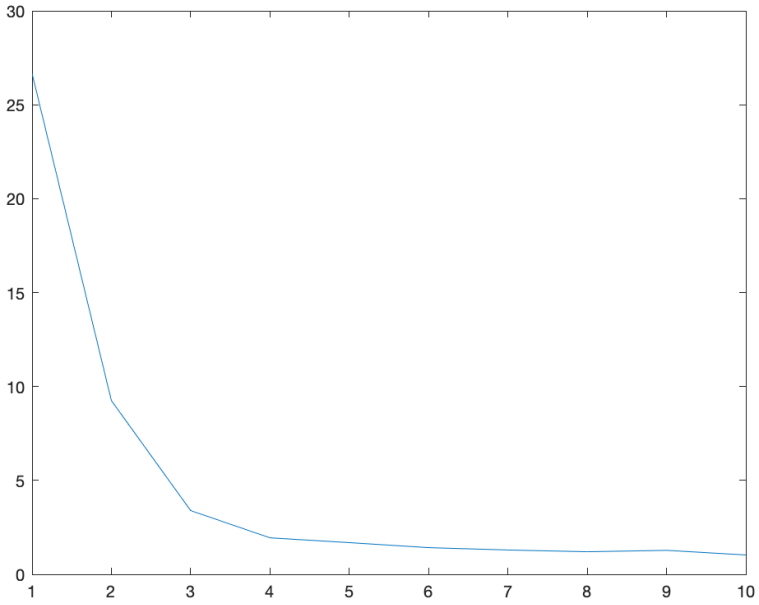


- When we use third way of initializing, the culstering (fig 3) would always cluster correctly, even if sometimes initial two centre in same group, it would be more tolerant than second approach.

- For each way of initializing, run the algorithm several times and report your findings.
  - The first way is very sensitive with centres position which would effect the results very distinctly. The second way, sometime got the expected clustering, in a few tiems, it would got different clustering like Fig 2, which would split the points on the top of the picture, but would not split the bottom. it would more tolorant than first method, but it still be sensitive. The third would always give us expected result. It much more tolerant with initlal centers in third method than previous two method.

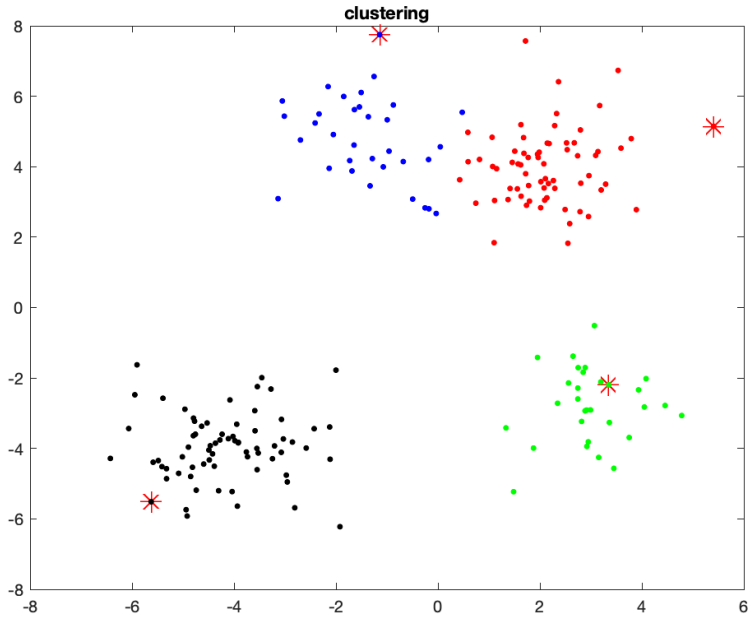
(d) From now on, we will work with the third method for initializing cluster centers. Run the algorithm for $k = 1, \ldots 10$ and plot the k-means cost of the resulting clustering

-



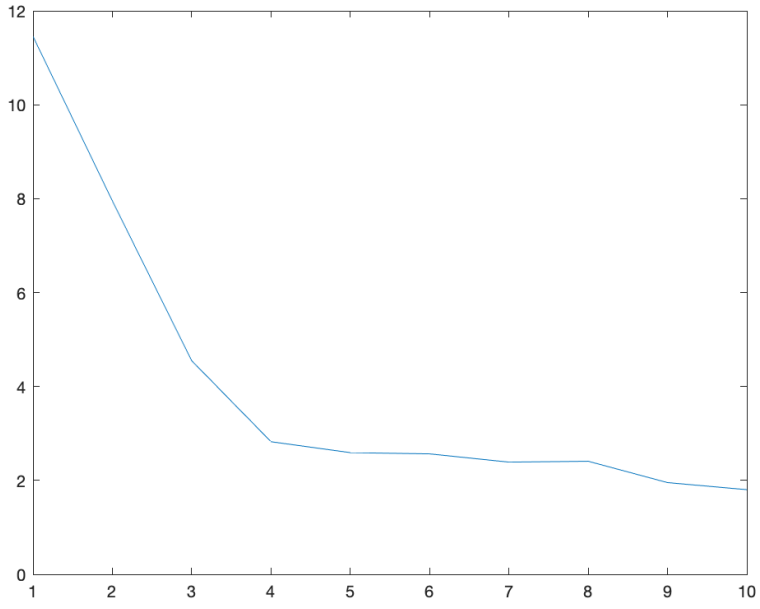
- What do you observe? How does the cost evolve as a function of the number of clusters? How would you determine a suitable number of clusters from this plot (eg in a situation where the data can not be as obviously visualized).

- From $k = 1$ to $k = 4$, the cost would decrease more acutely, when k is larger than 4, the cost would keep balance relatively, which decreases very slowly. When $k = 4$, the cost is minimal relatively.
- we can choose k such that minimizes the cost. In this part, we can choose k =4
- Like fig 5:
- Fig 5



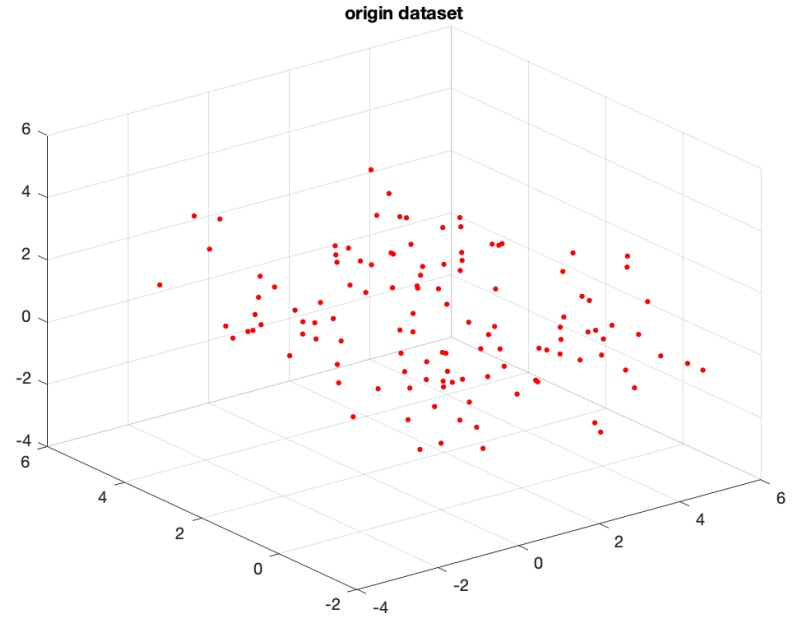
(e) Repeat the last step (that is, plot the cost as a function of $k = 1, \ldots, 10$) for the next dataset `threedpoints.txt`. What do you think is a suitable number of clusters here? Can you confirm this by a visualization?

- `threedpoints.txt` cost of $k = 1, \ldots, 10$
-

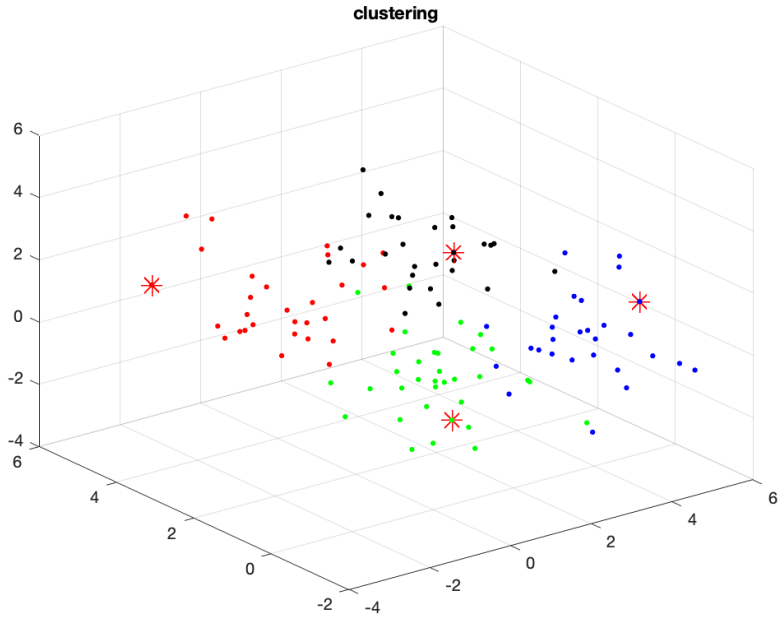


- By the plot of the cost, we should choose $k = 4$, it hard to get the 4-culstering by original data points visualization.

- Origin dataset

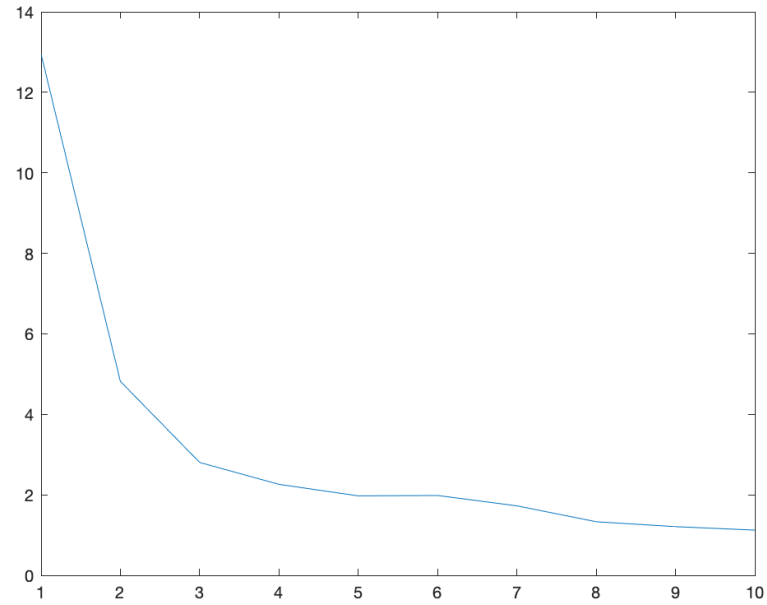


- $k = 3$



(f) Load the UCI “seeds” dataset from the last assignment and repeat the above step.

- From how the the k-means cost evolves, what seems like a suitable number of clusters for this dataset?

- 

  - Based on the plot of cost, it would be **3 clusters** for this dataset.
  - we choose **k = 3** in next part.
- How could you use this for designing a simple 3-class classifier for this dataset? What is the empirical loss of this classifier?
  - we can choose a three initial centers by hand from each class {1, 2, 3}
  - The `k_means_alg.m` would see part (a)
  - 3rd approach we use third method to initialize the initial centres, set $k = 3$

```
% load data
clc;clear;close all;
seedD=load('seeds_dataset.txt');
[~, d] = size(seedD);
% % init centers
% init_centers = zeros(3,d-1);
% for i = 1:3
%     D=seedD((seedD(:,d)==i),:);
%     init_centers(i,:) = D(unidrnd(70),1:d-1);
% end
% remove last col
label=seedD(:,d);
seedD=seedD(:,1:d-1);
[N, d] = size(seedD);
% set k clusters, and init method
k = 3;
init = 'euclidean';
min_loss = realmax;
for i = 1:500
    % clustering
    [cluster_i,~, ~] = k_means_alg(seedD,k,init,0);
    Diff = label ~= cluster_i;
    % compute the emp binary loss
    loss = sum(Diff);
    % find the minimal loss
    if loss < min_loss
        min_loss = loss;
        opt_classifier = cluster_i;
    end
```

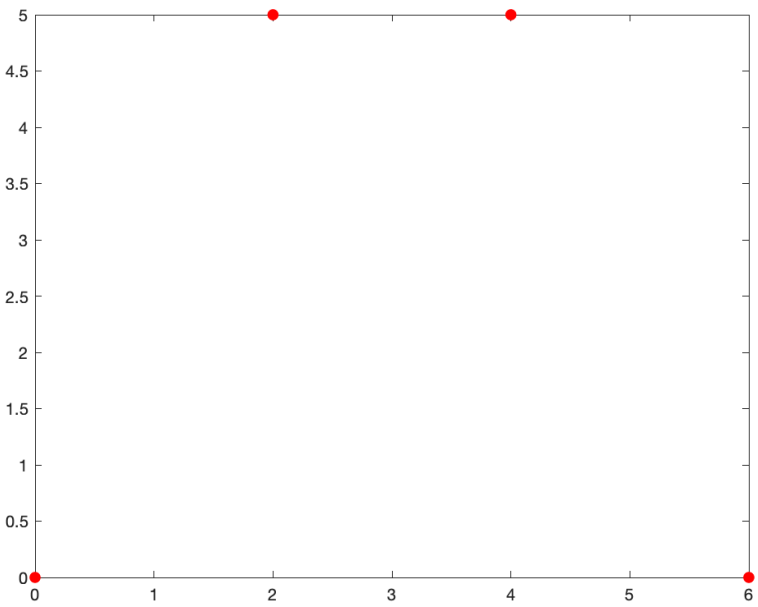
```
end
min_loss
opt_classifier;
```

- The cost is `22` approximately in several times running.

(g) Design a simple (two-dimensional) dataset where the 2-means algorithm with the third initialization method will always fail to find the optimal 2-means clustering. Explain why it will fail on your example or provide plots of the data with initializations and costs that show that 2-means converges to a suboptimal clustering.

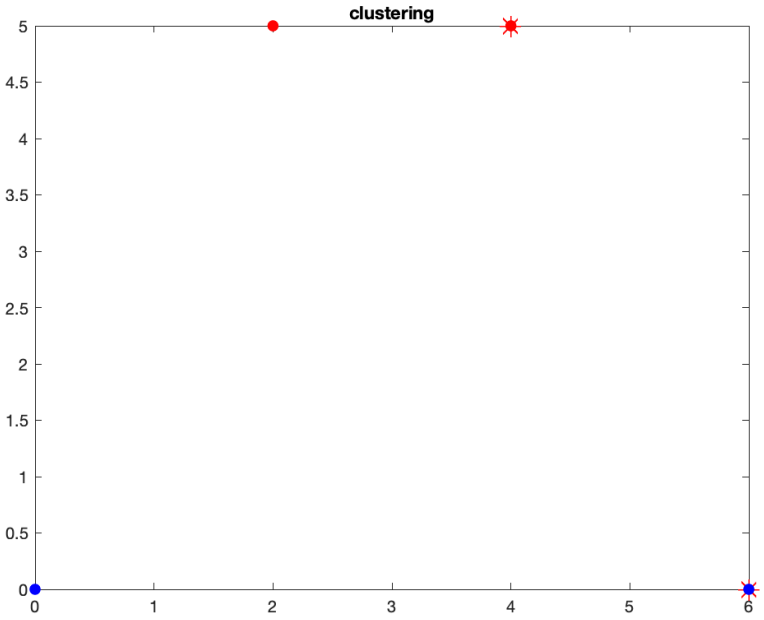
```
clc;clear;close all;

x = [0 6 2 4];
t = [0 0 5 5];
D = [x',t'];
figure(1)
plot(x,t,'r.','MarkerSize', 20)
% set k clusters, and init method
k = 2;
init = 'euclidean';
% clustering
[cluster_i,cost,init_c] = k_means_alg(D,k,init,0);
% plot clustering in different color
colors = {'blue','red'};
figure(2)
plot(init_c(:,1), init_c(:,2), 'r*', 'MarkerSize', 12);
hold on
for i = 1:k
    hold on
 plot(x(find(cluster_i==i)),t(find(cluster_i==i)),'.','color',colors{i},'MarkerSize', 20);
end
title('clustering');
cost
```

- Fig g.1

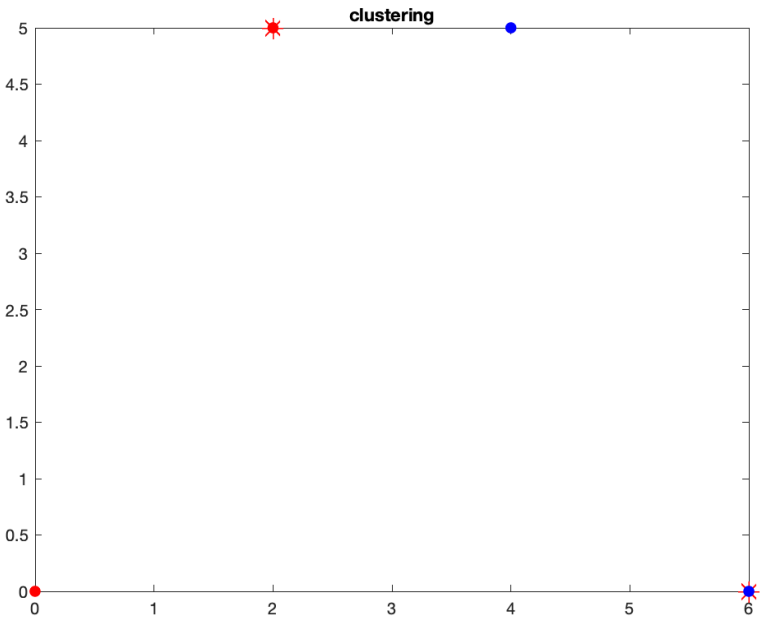


- We choose dataset $D = \{(0,0),(6,0),(2,5),(4,5)\}$ like plot above (fig g.1).

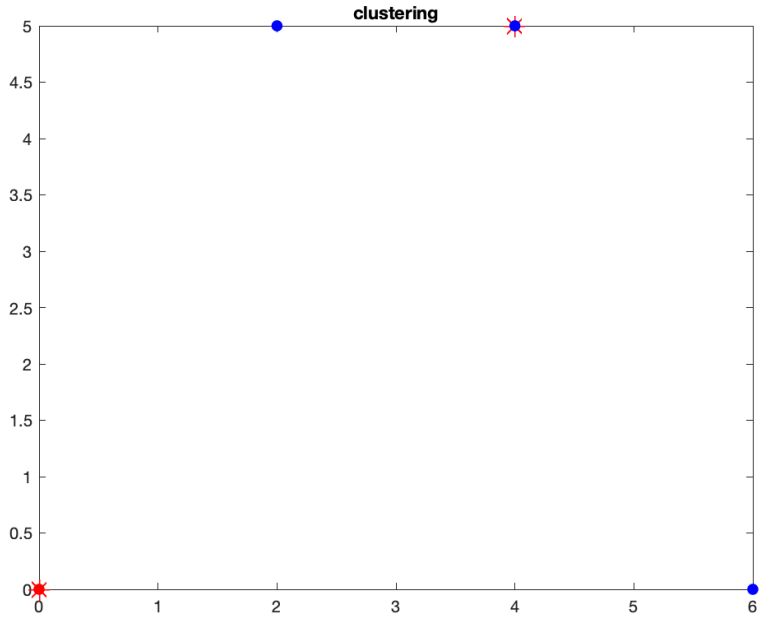
- Fig g.2



- When we use first or second method, we can got minimal $cost = 5$, and cluster like above (fig g.2)
- **However**, when we **use third approace** to initialize the centres, we always got the cost is larger than 5, the plot like fig g.3 ($cost = 7.25$) or fig g.4 ($cost = 6.17$) which always fail to find the optimal 2-means clustering.
- Fig g.3



- Fig g.4



- The reason is the third approach would always initial second centre such that far away the first centre. Thus, in 2-mean clustering, second centre position would be effected by first centre position. In our dataset, $A(0,0)$, $B(6,0)$, $C(2,5)$, $D(4,5)$, AD and BC are longest. in a satuation, when it choose C as first center, it must choose B as second center. B would cluster the nearse point D, and C would cluster the neatest point A (fig g.3) $cost = 7.25$, which faile to find optimal culster ($cost_{opt} = 5$). On the ohter senario when it choose D as first centre, it also choose A as second centre. For now, D is near B and C, so clustering them (fig g.4) $cost = 6.1667$, which faile to optimal cluster ($cost_{opt} = 5$).
- Because the number of clusters is too few only 2-mean, and in this dataset some distance would same, it hard to get optimal clustering. We can initial first centre uniformly random choose in a real number area, rather than in dataset. Such way, would decrease the secetive with first centre initiallized.