K-Means

Separate data points into K clusters with no other information.
Inputs:
X - D-by-N matrix of N points in D dimensions.
K - Integer number of clusters to detect.
Outputs:
mu - D-by-K matrix with the learned cluster centroids.
labels - Length N vector with integer (1, 2, ..., K) class assignments.

```
function [mu, labels] = km(X, K)
    [~, N] = size(X);

    %initialize clusters by random points
    idx = randperm(N,K);
    mu = X(:,idx);

    labels = zeros(N, 1);
    while true
        %Declare temp labels
        tlabels = labels;

        %E-step
        DIST = zeros(K,N);
        for i = 1:K
            DIST(i,:) = vecnorm(X - mu(:,i));
        end
        [~,labels] = min(DIST);

        if isequal(tlabels,labels)
            return
        end

        %M-step
        for i = 1:K
            mu(:,i) = mean(X(:,labels == i), 2);
        end
    end
end
```

Published with MATLAB® R2020b

compute energy (cost) given X, mu, labels

calculate performance of k-mean clustering using sum of square cost (between data pts and its respective means)

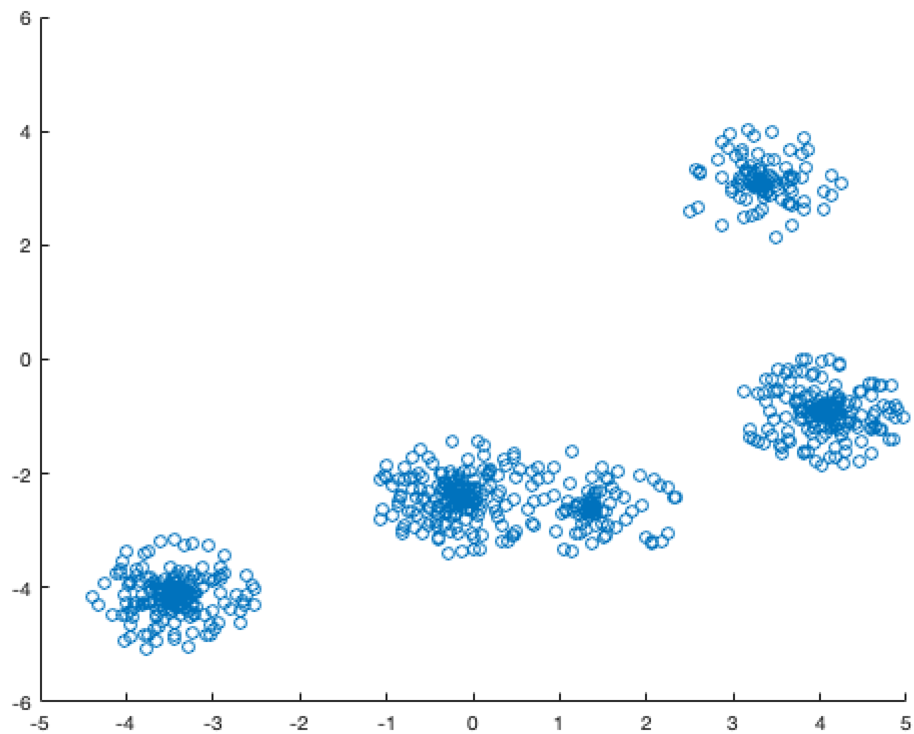
```
function [E] = energy(X, mu, labels)
    [~,K] = size(mu);
    E = 0;

    for i = 1:K
        E = E + sum(vecnorm(X(:,labels == i) - mu(:,i)).^2);
    end
end
```

Published with MATLAB® R2020b

Contents

3.2.1 Generate 5 pointclouds & examine data distribution

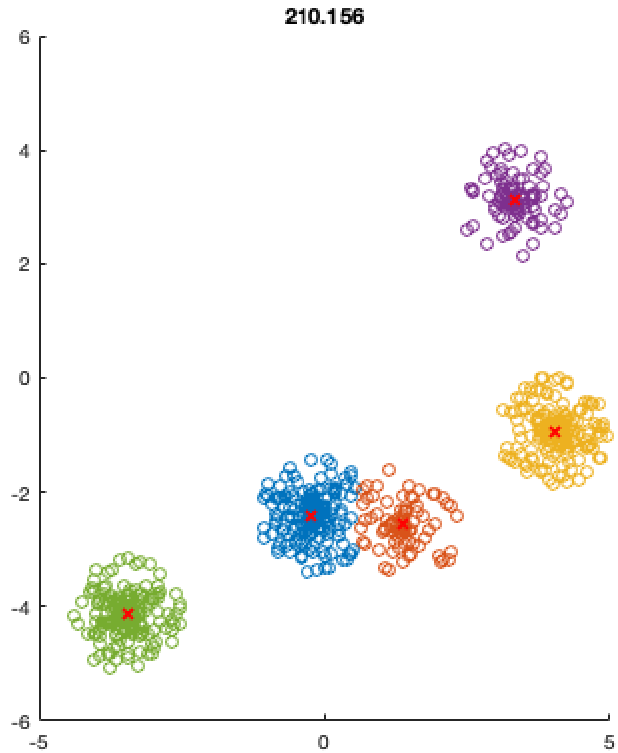
```
X = pointclouds();

%visualize distribution of data in 5 point clouds
scatter(X(1,:),X(2,:));
```

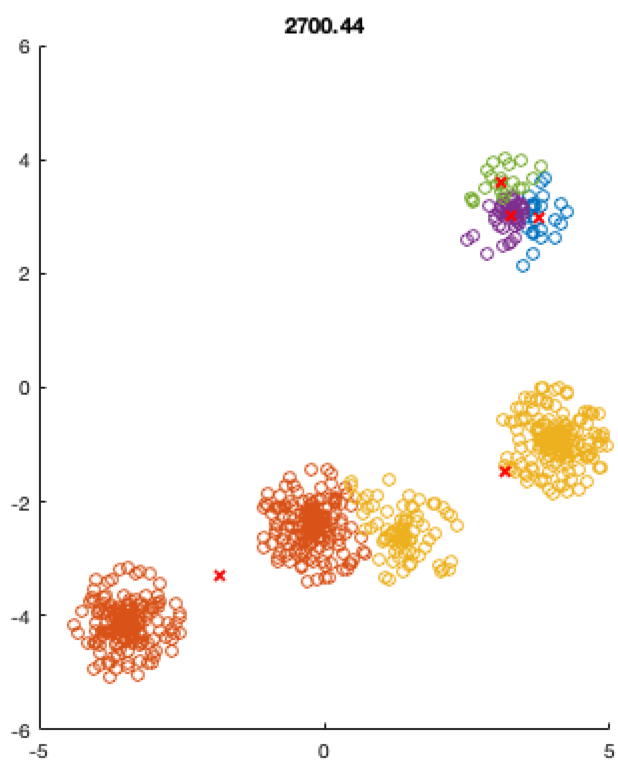


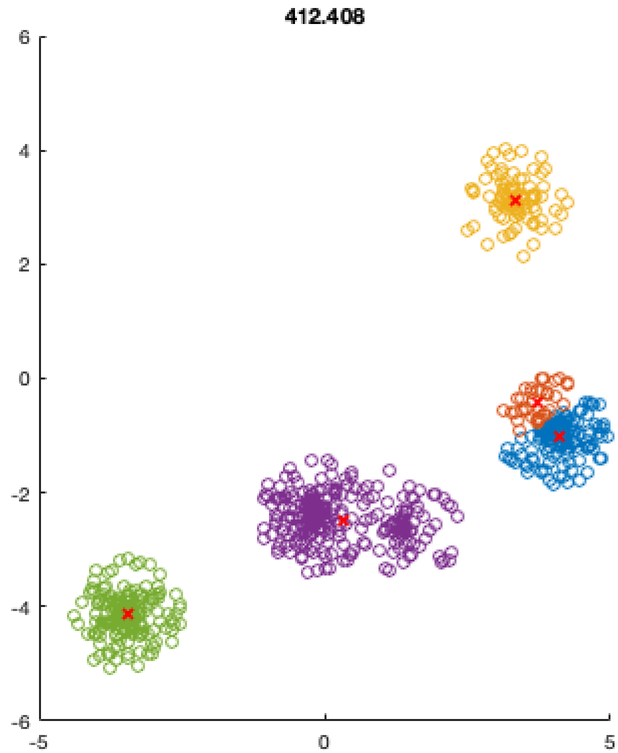
3.2.2, 3.3 k-means on pointclouds() (5 clusters)

```
K = 5;

%data points in different clusters are colour coded
%red 'x' represents means of clusters
for iter = 1:3
    [mu, labels] = km(X,K);

    figure;
    for i = 1:K
```

```
        scatter(X(1,labels == i), X(2,labels == i)); hold on;
    end
    scatter(mu(1,:), mu(2,:), 'rx', 'linewidth', 2);

    %energy helps quantify how well one iteration does w.r.t another (and
    %with other types of point clouds)
    %the smaller energy (cost) the better.
    title(energy(X,mu,labels));
    daspect([1 1 1]);
end
```


210.156
6
4
2
0
-2
-4
-6
-5
0
5


2700.44
6
4
2
0
-2
-4
-6
-5
0
5

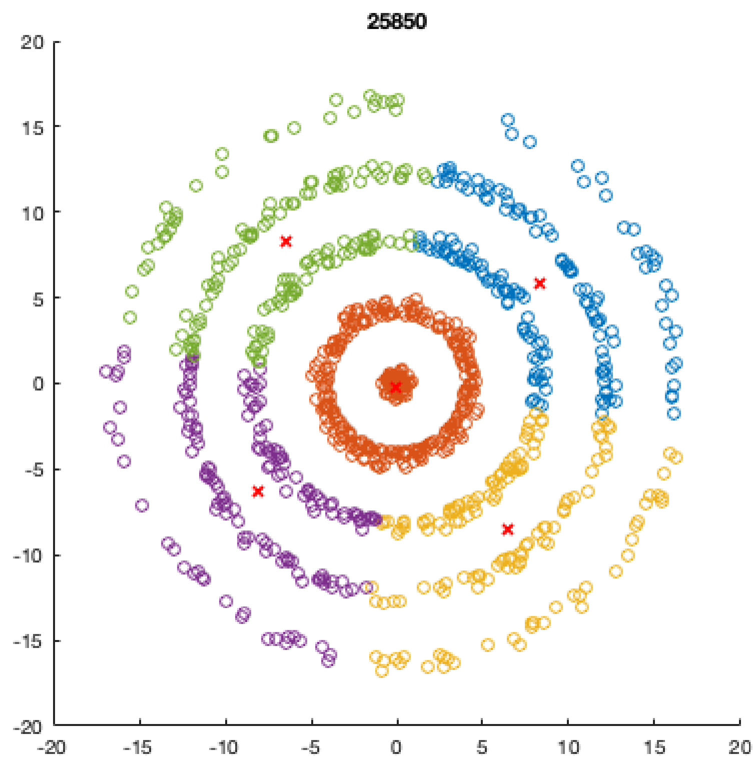


3.4 k-means on pointrings() (5 clusters)

```
X = pointrings();
K = 5;

for iter = 1:3
    [mu, labels] = km(X,K);

    figure;
    for i = 1:K
        scatter(X(1,labels == i), X(2,labels == i)); hold on;
    end

    scatter(mu(1,:), mu(2,:), 'rx', 'linewidth', 2);
    title(energy(X,mu,labels));
    daspect([1 1 1]);
end
```

25691.5
20
15
10
5
0
-5
-10
-15
-20
-20 -15 -10 -5 0 5 10 15 20

25721.7
20
15
10
5
0
-5
-10
-15
-20
-20 -15 -10 -5 0 5 10 15 20



3.5 Why k-means performed better in pointclouds() than pointrings()

```
%basic idea is: the way k-means works is grouping by proximity to a common mean
%mu for each cluster so the distance between points within a cluster to its
%cluster center is minimized whereas distances between points across different
%clusters is maximized. So it is perfect for pointclouds() where points are
%sampled randomly from 5 distant balls of some given radius.

%The problem with data cloud sampled from rings is that each ring does not
%have a well defined center that all points are "bunched up" on in euclidean
%space. So its quite hard for k-means to identify the rings as distinct groups.
%In short, rings are not linearly separable in euclidean space.
```

3.6 k-means on plane image

```
%Vectorizing image data into X, each pixel being a vector in R^3
[X,I,dims] = im2rgb('plane_small.png');

[mu, labels] = km(X,10);
```

3.7 k-means on plane (result)

```
%Observation: notice how there is now only a few colours (10) in image instead
%of a large variety. However, the modified image is still a good representation
%as k-means bring only similar colours to a common one, so larger differences in
%colours are still recognized and distinguished whereas small nuances may not
%be separated (but smaller nuances is less obvious to our eyes).

figure;
imshow(rgb2im(mu(:,labels), dims));
```

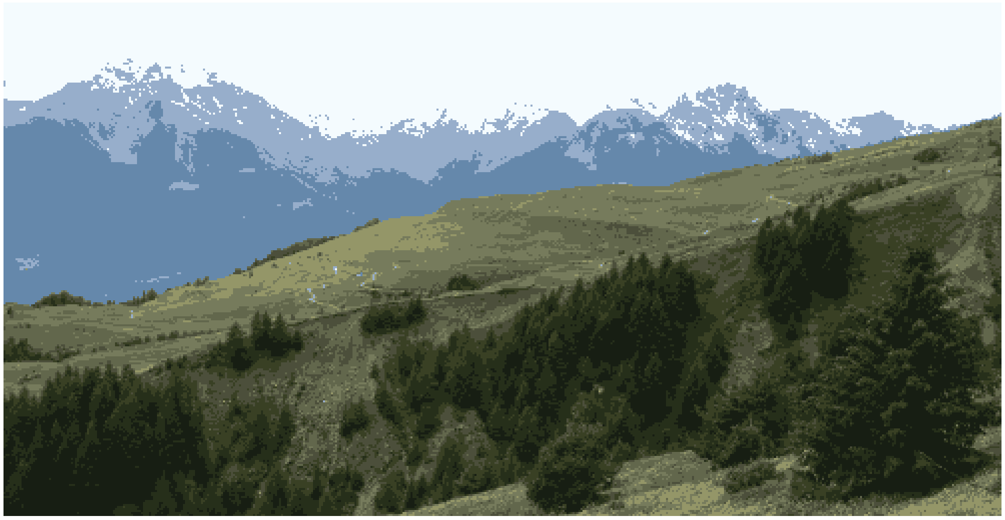
```
title(energy(X,mu,labels));

figure;
imshow(I);
```

551.287

extra for mountain

```
[X,I,dims] = im2rgb('mountains_small.png');

[mu, labels] = km(X,10);

%Observation: I feel like k-means actually works pretty well on Mountains
%since original image has clear distinct shades of colours: dark green, light
%green, white, light blue, dark blue, brown, etc. Though it usually fails to
%recognize the brown mud as its own colour
figure;
imshow(rgb2im(mu(:,labels), dims));
title(energy(X,mu,labels));

figure;
imshow(I);
```

581.469



Published with MATLAB® R2020b