COMPM012 - Coursework 3

Esben A. Sørig

November 27, 2014

1 Practical

1.1 k-means implementation

```
function [clusterings, centers] = mykmeans(X, k)
    % Randomly initialize centers of clusters
    c = datasample(X, k, 'Replace', false);
    r = repmat(0, size(X, 1), k);
    oldr = 1; % something that is not equal to r initially

    dist = @(x, y) norm(x-y);

    % Loop as long as the clustering is changing
    while ~isequal(r, oldr)
        oldr = r;

        % Assign points to clusters
        for i = 1:size(X,1)
            cluster = 1;
            for j = 1:k
                if dist(X(i, :), c(j, :)) <  dist(X(i, :), c(
    cluster, :))
                    cluster = j;
                end
            end

            r(i, :) = [repmat(0, 1, cluster-1) 1 repmat(0, 1, k-
    cluster)];
        end

        % Update center positions
        for i = 1:k
            npoints = 0;
            c(i, :) = 0;
            for j = 1:size(r, 1)
```

```
                c(i, :) = c(i, :) + r(j, i)*X(j, :);
                npoints = npoints + r(j, i);
            end
            c(i, :) = c(i, :)/npoints;
        end
    end

    centers = c;
    % Output formatting (vector with cluster index for each row
    in X)
    clustering = repmat(0, size(X,1), 1);
    for i = 1:size(X,1)
        for j = 1:k
            if r(i, j) == 1
                clustering(i) = j;
                break;
            end
        end
    end

    clusterings = clustering;
end
```

1.2 k-means test on data generated from three gaussians

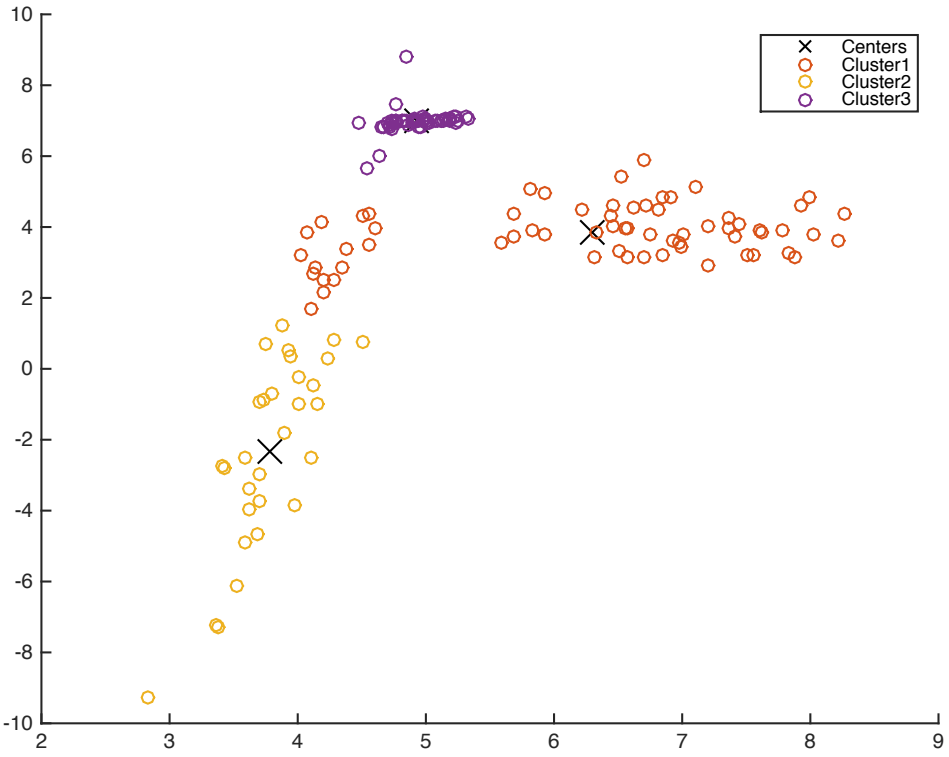
Firstly, we generate the data and run the k-means algorithm on it.

```
% Generate data
data = genData2;

% Put data for each cluster in a seperate list
cluster1 = [];
cluster2 = [];
cluster3 = [];
[clusterings, centers] = mykmeans(data, 3);

for i = 1:size(data, 1)
    if clusterings(i) == 1
        cluster1 = [cluster1; data(i, :)];
    end
    if clusterings(i) == 2
        cluster2 = [cluster2; data(i, :)];
    end
    if clusterings(i) == 3
        cluster3 = [cluster3; data(i, :)];
    end
end
```

We can now plot the clusters the algorithm has found

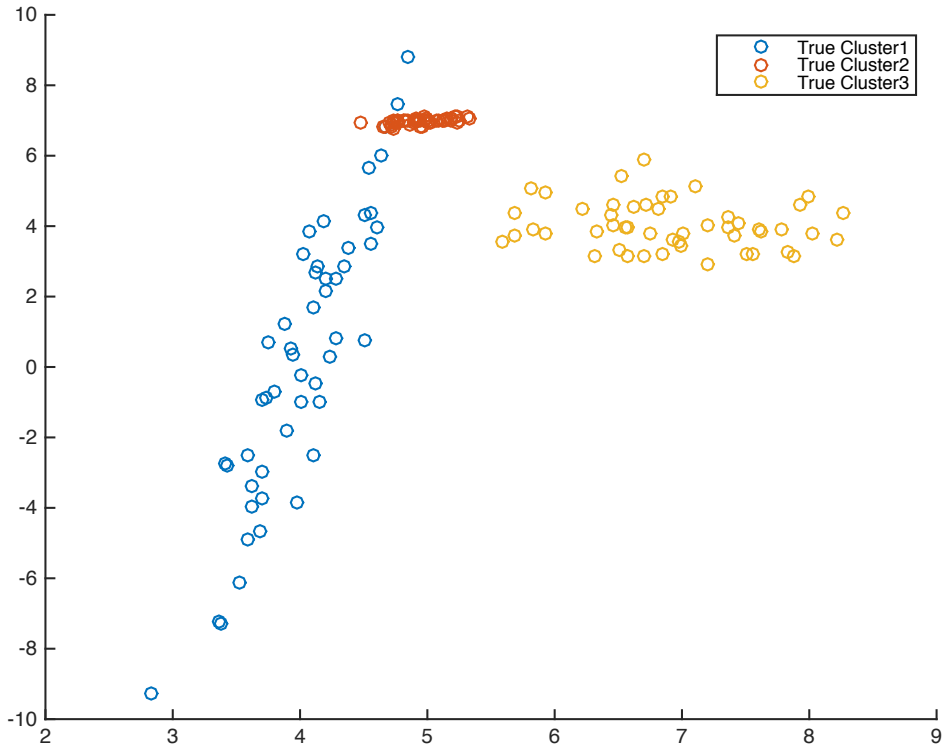
```
figure
```

```
hold on

scatter(centers(:, 1), centers(:, 2), 200, 'xblack');
scatter(cluster1(:, 1), cluster1(:, 2));
scatter(cluster2(:, 1), cluster2(:, 2));
scatter(cluster3(:, 1), cluster3(:, 2));
legend('Centers', 'Cluster1', 'Cluster2', 'Cluster3');
hold off
```



Comparing this to the true classifications

```
figure
hold on
scatter(data(1:50, 1), data(1:50, 2));
scatter(data(51:100, 1), data(51:100, 2));
scatter(data(101:150, 1), data(101:150, 2));
legend('True Cluster1', 'True Cluster2', 'True Cluster3');
hold off
```



We clearly get an intuition of how the k-means algorithm is performing on the data. Please see appendix 4.1 for the code that creates the animation of the convergence to a solution for this dataset.

We measure the mean error and standard deviation of the error of the clustering over a 100 runs of the algorithm.

```
%% Error measurements
errors = [];
for j = 1:100
    [clusterings, centers] = mykmeans(data, 3);

    % keep track of the classifications of each true cluster
    firstcluster = [0,0,0];
    secondcluster = [0,0,0];
    thirdcluster = [0,0,0];
    for i = 1:50
        firstcluster(clusterings(i)) = 1 + firstcluster(
    clusterings(i));
    end
    for i = 51:100
```

```
        secondcluster(clusterings(i)) = 1 + secondcluster(
    clusterings(i));
    end
    for i = 101:150
        thirdcluster(clusterings(i)) = 1 + thirdcluster(
    clusterings(i));
    end
    % We assume that the mode of the classifications of a true
    cluster is the class of the true cluster. Therefore, the
    error of a cluster is the frequency of other classifications
    appearing for that cluster.
    firstcluster = sort(firstcluster);
    secondcluster = sort(secondcluster);
    thirdcluster = sort(thirdcluster);
    misclassifications = firstcluster(1) + firstcluster(2) +
    secondcluster(1) + secondcluster(2) + thirdcluster(1) +
    thirdcluster(2);
    errors = [errors misclassifications/150];
end

meanerror = mean(errors)
stddeviationerror = std(errors)
```

Which returns:

```
meanerror =
    0.1438
stddeviationerror =
    0.0155
```

2 Questions

2.1 Dataset with local minima

2.2 Argument that the centroid is the minimizer of the sum of squared distances

2.3 Proof of convergence in finite amount of steps

3 Extension

3.1 k-means segmentation

3.2 (p, k)-means

4 Appendix

4.1 Movie generation code

```
%% Movie
X = data;
k = 3;
% Randomly initialize centers of clusters
c = datasample(X, k, 'Replace', false);
r = repmat(0, size(X, 1), k);
oldr = 1; % something that is not equal to r initially
movie = [];

dist = @(x, y) norm(x-y);

% Loop as long as the clustering is changing
while ~isequal(r, oldr)
    oldr = r;

    % Assign points to clusters
    for i = 1:size(X,1)
        cluster = 1;
        for j = 1:k
            if dist(X(i, :), c(j, :)) <  dist(X(i, :), c(cluster
    , :))
                cluster = j;
            end
        end

        r(i, :) = [repmat(0, 1, cluster-1) 1 repmat(0, 1, k-
    cluster)];
    end
```

```
27
28      %MOVIE GENERATION
29      % Split the data into the three clusters
30      cluster1 = [];
31      cluster2 = [];
32      cluster3 = [];
33
34      for i = 1:size(X, 1)
35          if r(i,1) == 1
36              cluster1 = [cluster1; X(i, :)];
37          end
38          if r(i,2) == 1
39              cluster2 = [cluster2; X(i, :)];
40          end
41          if r(i,3) == 1
42              cluster3 = [cluster3; X(i, :)];
43          end
44      end
45
46      % Plot the centers and the three clusters
47      hold on
48      scatter(c(:, 1), c(:, 2), 200, 'xblack');
49      if size(cluster1, 1) ~= 0
50          scatter(cluster1(:, 1), cluster1(:, 2));
51      end
52      if size(cluster2, 1) ~= 0
53          scatter(cluster2(:, 1), cluster2(:, 2));
54      end
55      if size(cluster3, 1) ~= 0
56          scatter(cluster3(:, 1), cluster3(:, 2));
57      end
58      legend('Centers', 'Cluster1', 'Cluster2', 'Cluster3');
59      hold off
60      movie = [movie getframe];
61      clf;
62      % MOVIE GENERATION OVER
63
64      % Update center positions
65      for i = 1:k
66          npoints = 0;
67          c(i, :) = 0;
68          for j = 1:size(r, 1)
69              c(i, :) = c(i, :) + r(j, i)*X(j, :);
70              npoints = npoints + r(j, i);
71          end
72          c(i, :) = c(i, :)/npoints;
73      end
74  end
75  movie2avi(movie, 'k-means.avi', 'fps', 1);
```