Pattern Recognition Practical 5

Group 24: Maikel Withagen (s1867733) Steven Bosch (s1861948)

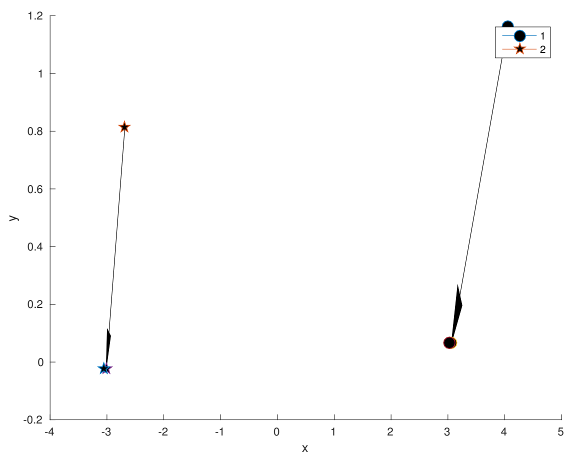
October 15, 2015

Assignment 1 k-means clustering, quantization error, gap statistic

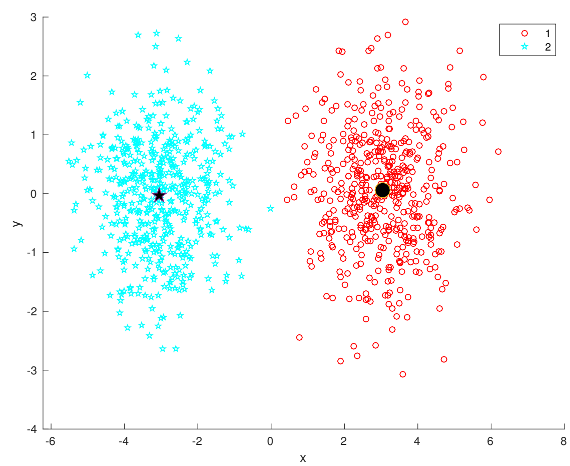
1

Using the code given in the Appendix(kmeans.m and runKMeans.m), we created the plots shown in figures 1, 2 and 3.

Figure 1: Results for k=2

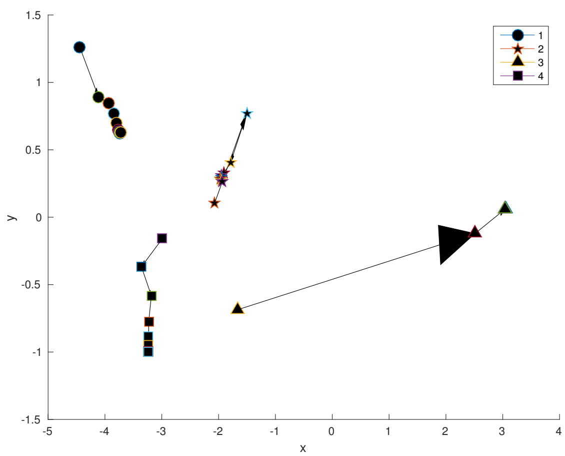


(a) intermediate positions of the cluster means, with their progress indicated by the arrows.

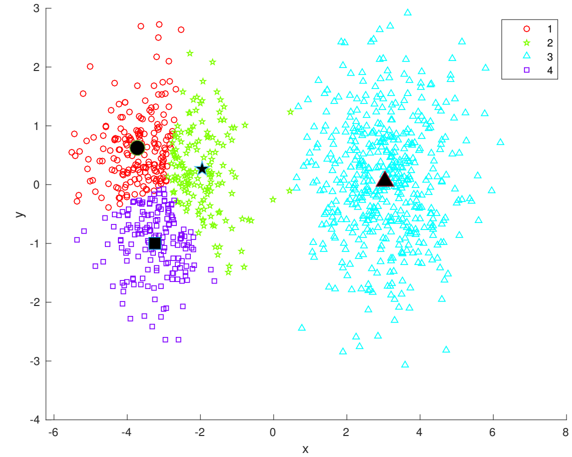


(b) The final cluster means with their assigned data points)

Figure 2: Results for k=4

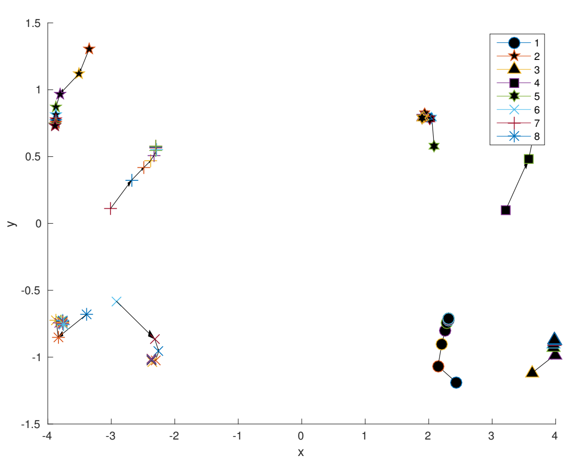


(a) intermediate positions of the cluster means, with their progress indicated by the arrows.

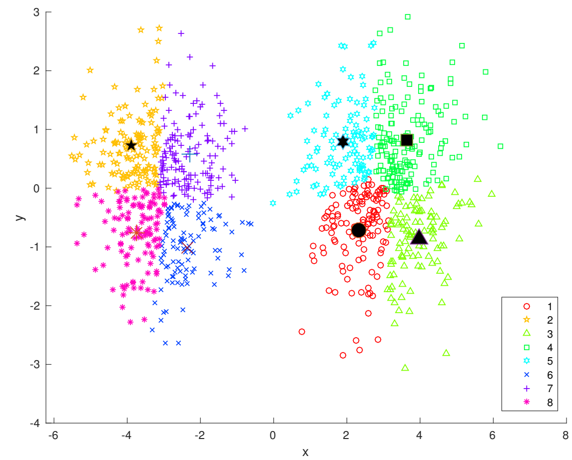


(b) The final cluster means with their assigned data points)

Figure 3: Results for k=8



(a) intermediate positions of the cluster means, with their progress indicated by the arrows.



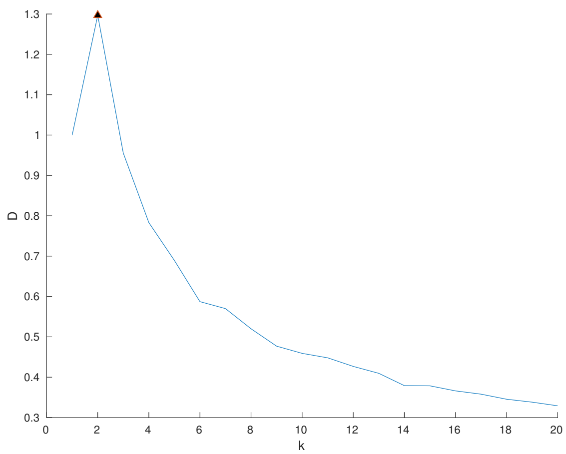
(b) The final cluster means with their assigned data points)

We we can clearly see that the data form two clusters. Therefore figure 1a shows the quickest convergence to the final cluster centers. Usually it takes about two epochs for the cluster centers to converge, as is shown in the figure. Figure 1b shows that these centers form in the places which the human eye observes to be the correct centers. When we choose k as 4, as shown in figure 2, we can see that, dependent on the initialization, sometimes one main cluster gets divided into three subclusters and the other remains one cluster, and sometimes the two clusters get separated into two clusters each. The number of epochs it takes to reach convergence is high compared to a run using $k = 2$. This can be explained by the fact that the data are not naturally separated into four clusters but in two, so the distances between the data points within a main cluster are small. This causes the algorithm to take longer to find a convergence, since the cluster centers switch often during the clustering process. Finally figure 3 shows the clustering for $k = 8$, which takes the longest amount of epochs to converge, because of the same reasoning. It separates both of the clusters into four subclusters.

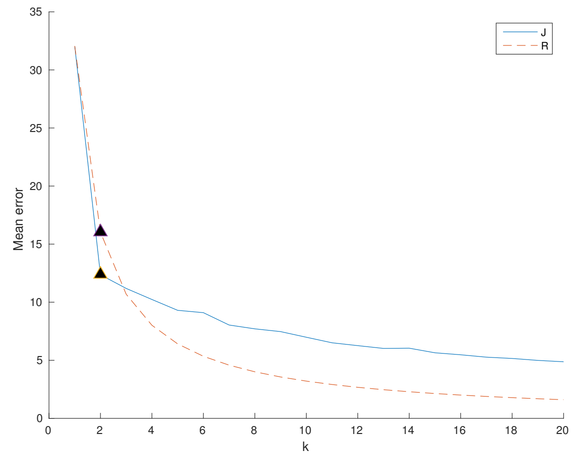
2

Using the code given in the appendix (kmeans.m and runKMeans.m) we computed the quantization errors and D-function given in figure 4.

Figure 4: Results for $kmax = 20$. The triangles give k_{opt}



(a) The D-function.



(b) The quantization error function $J(k)$ and the reference function $R(k)$.

Figure 4 shows the $D(k)$, $J(k)$ and $R(k)$ for a $kmax$ of 20. Here $k_{opt} =_k D(k)$ is the k for which the difference between the error function and the reference function is the largest. Figure 4 shows that k_{opt} can be found at a k of 2, which was expected because it could clearly be seen by a human that the data are divided into two main clusters. $D(k)$ decreases as k increases, clusters that have no natural division need to be divided by even more clusters, which causes them to shift a lot before finally reaching convergence.

Assignment 2 Batch Neural gas vs k-means

Appendix

../Code/kmeans.m

```
function [qError] = kmeans(dat, k, writeOutput)
% K-means clustering algorithm
close all;
shapes = 'op^shx+*dv<>.';

% Init the prototypes to a random point
prototypes = zeros(k,ndims(dat));
for i = 1:k
    newPoint = dat(randi(length(dat)),1:2);
    while (sum(pdist2(prototypes, newPoint) == 0) ~= 0)
        newPoint = dat(randi(length(dat)),1:2);
    end
    prototypes(i,:) = newPoint;
end

%Init the first figure
figure(1)
hold on;
```

```
xlabel('x');
ylabel('y');

for i = 1 : size(prototypes, 1)
    plot(prototypes(i,1),prototypes(i,2),'Marker', shapes(i), 'MarkerSize', 10, 'MarkerFaceColor', 'black')
end


% Perform k-means
loop = 1;
while(loop == 1)
    loop = 0;

    for point = 1 : length(dat)
        dat(point,3) = find(pdist2(dat(point,1:2), prototypes) == min(pdist2(dat(point,1:2), prototypes)),1);
    end

    for prototype = 1 : size(prototypes, 1)
        newMean = mean(dat(dat(:,3) == prototype,1:2));
        if newMean ~= prototypes(prototype,:)
            loop = 1;
        end
        plot_arrow( prototypes(prototype,1),  prototypes(prototype,2), newMean(:,1), newMean(:, 2));
        prototypes(prototype,:) = newMean;
        plot(newMean(1),newMean(2),'Marker', shapes(prototype), 'MarkerSize', 10, 'MarkerFaceColor', 'black')
    end


end

% Calculate the quantization error
qError = 0;
for i = 1 : size(prototypes, 1)
    qError = qError + sum(pdist2(prototypes(i,:), dat(dat(:,3) == i,1:2)));
end

% More figure stuff
legend(strtrim(cellstr(num2str((1:k)'))'));
if writeOutput == 1
    print(sprintf('../Report/Fig1_k%d', k), '-depsc');
end
figure(2)
hold on;
gscatter(dat(:,1),dat(:,2),dat(:,3),[],shapes, 5)


for i = 1 : size(prototypes, 1)
    plot(prototypes(i,1),prototypes(i,2),'Marker', shapes(i), 'MarkerSize', 13, 'MarkerFaceColor', 'black')
end

xlabel('x');
ylabel('y');
if writeOutput == 1
    print(sprintf('../Report/Fig2_k%d', k), '-depsc');
end
```

../Code/runKMeans.m

```
load('kmeans1.mat', 'kmeans1');

```

```
error= zeros(1,10);
kmax = 20;
J = zeros(1, kmax);
R = zeros(1,kmax);

% Run for 1 to kmax clusters
for k = 1 : kmax
    k
    % Run it 10 times for every cluster and calculate the mean error and
    % reference
    for i = 1:10
        error(i) = kmeans(kmeans1,k, 0);
    end
    J(k) = mean(error)/10;
    R(k) = J(1) * k^(-2/ndims(kmeans1));
end

D = R ./ J;

% Plot D
[maxVal maxInd] = max(D);
figure(3)
hold on;
plot(D);
plot(maxInd, maxVal,'Marker', '^', 'MarkerSize', 6, 'MarkerFaceColor', 'black')
xlabel('k');
ylabel('D');
print(sprintf('../Report/Fig3'), '-depsc');

% Plot J and R
figure (4)
hold on ;
plot(J);
plot(R, '--');
plot(maxInd, J(maxInd),'Marker', '^', 'MarkerSize', 10, 'MarkerFaceColor', 'black')
plot(maxInd, R(maxInd),'Marker', '^', 'MarkerSize', 10, 'MarkerFaceColor', 'black')
xlabel('k');
ylabel('Mean error');
legend('J', 'R');
print(sprintf('../Report/Fig4'), '-depsc');
```