Pattern Recognition Practical 5

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

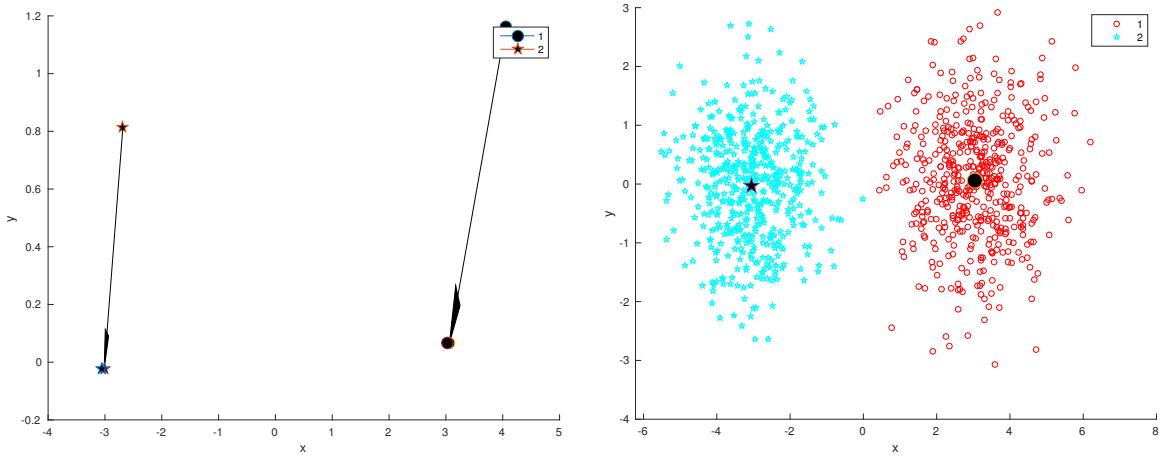
October 15, 2015

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

1

Using the code given in the Appendix(A B), we created the plots shown in figures 1, 2 and 3.

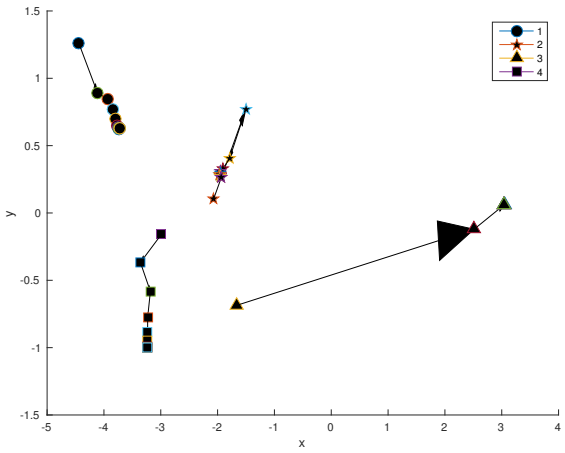
Figure 1: Results for k=2



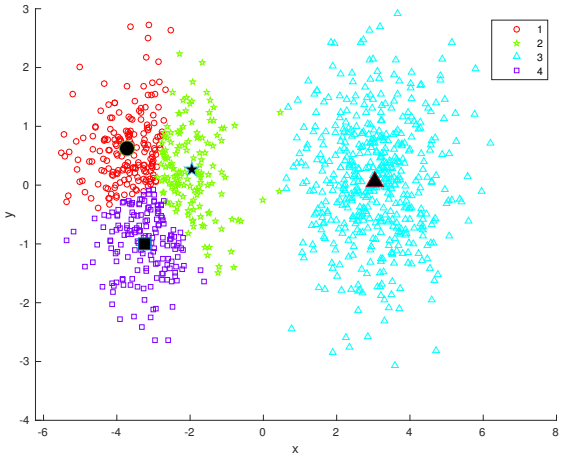
(a) intermediate positions of the cluster means, with their progress indicated by the arrows.

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

Figure 2: Results for k=4

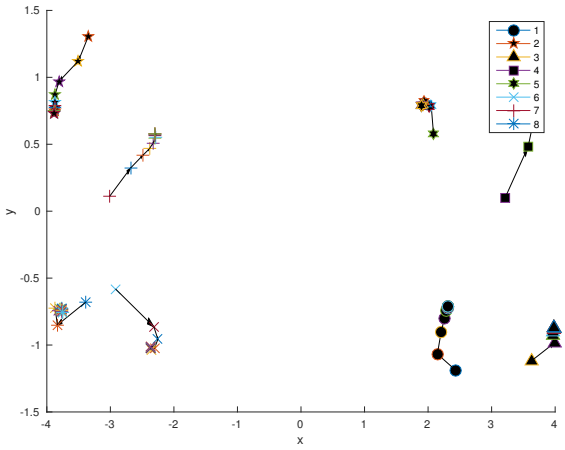


(a) intermediate positions of the cluster means, with their progress indicated by the arrows.

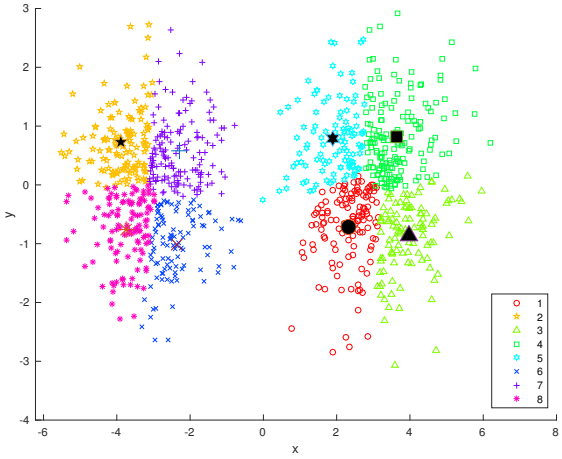


(b) The final cluster means with their assigned data points

Figure 3: Results for k=8



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



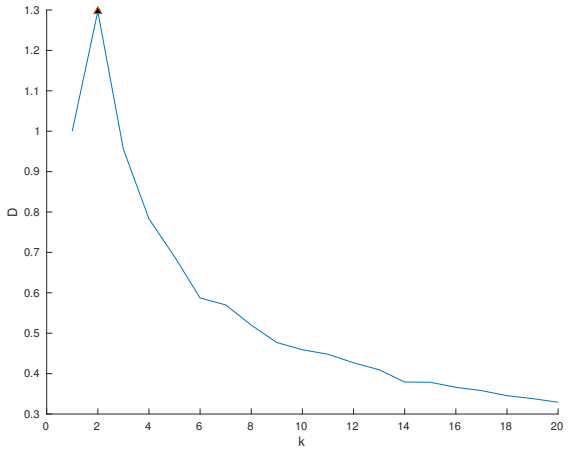
(b) The final cluster means with their assigned data points

We (humans) can clearly see that the data form two clusters. Therefore figure 1a, which has a k of 2, 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.

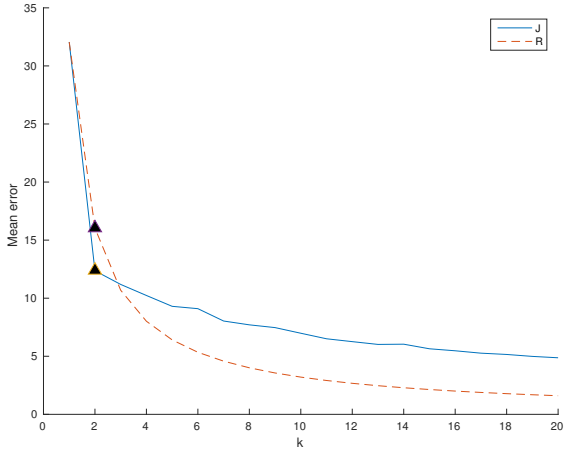
2

Using the code given in the appendix (A B) 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 $k_{opt} = \text{argmax}_k D(k)$ 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.

3

a

See the part in A which is commented "Kmeans plusplus initialization".

b

We computed the following means and standard deviations given in 1.

Table 1: Mean and standard deviation of quantization error for $k = 100$.

Algorithm	mean q-error	sd q-error
$kmeans++$	10.0310	0.3502
$kmeans$	10.6001	0.4268

c

We calculated the following p-value using a an unpaired one-tailed two-sample t-test (see run.m at the bottom): $2.3816e^{-5}$. This p-value is significant (¡0.05), which means that we have enough certainty to reject the null hypothesis that there is no difference in performance between the two algorithmes. Hence, we assume a difference between the performance of the two algorithmes ($kmeans++$ performs better).

Assignment 2 Batch Neural gas vs k-means

1

Code batchNG.m (C)in the appendix gives our implementation of the Neural gas algorithm.

2

Figure 5: K-Means results

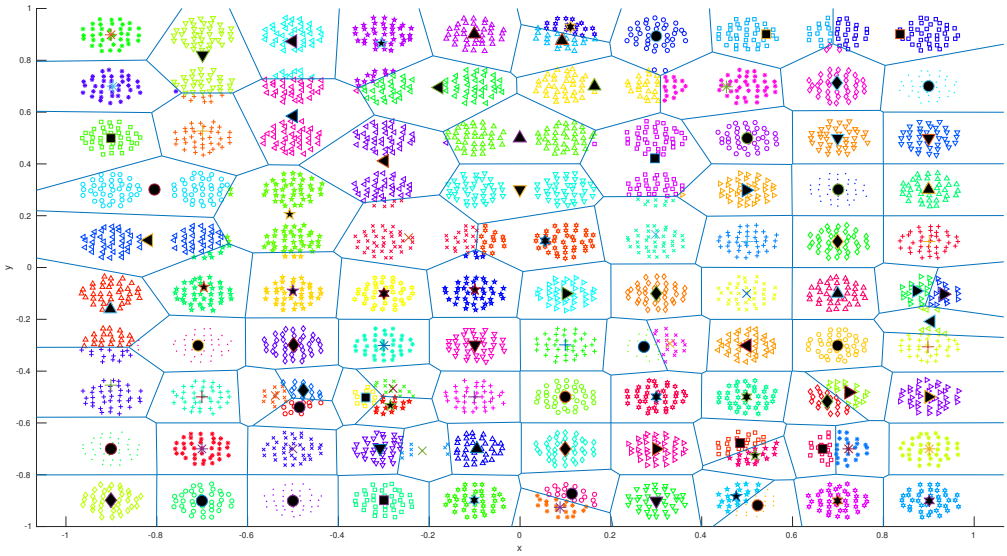
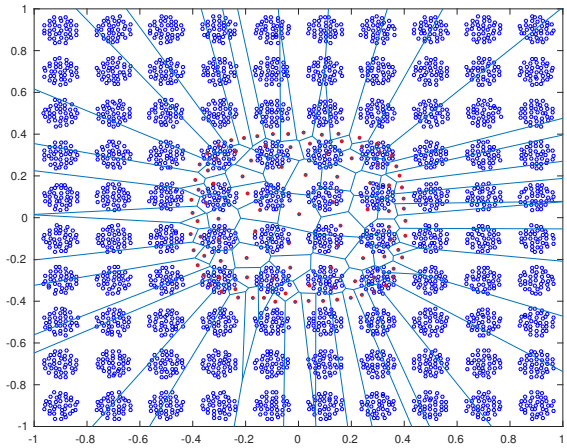
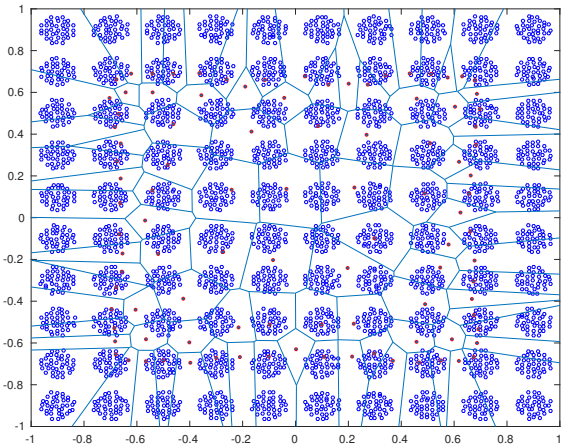


Figure 6: The final cluster means with their assigned data points and the cluster boundaries

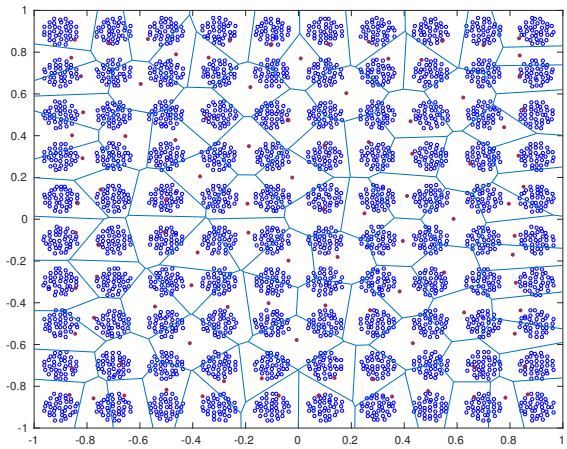
Figure 7: Batch Neural Gas results on epochs 20, 100, 200 and 500



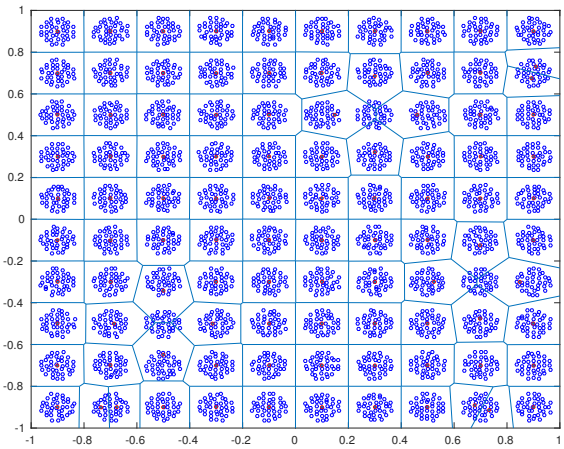
(a) Epoch 20



(b) Epoch 100



(c) Epoch 200



(d) Epoch 500

3

The first thing to notice when comparing the two algorithms is their speed. KMeans took only 10 epochs to establish a final result. We let Batch Neural gas run for 500 epochs, and it still hasn't converged properly. In terms of performance, Batch Neural Gas performs much better than KMeans. As you can see in Figure 6, A lot of the perfectly seperable clusters are clustered under a wrong prototype. The horizontal and vertical symmetry of the data-points has caused KMeans to assign two halves of two different clusters under the same prototype. Batch Neural Gas (as can be seen in Figure 7) slowly converges to the correct positions, thereby mimicing the behavior of a gas and thus living up to its name.

In conclusion we can say that in terms of speed KMeans can be considered faster than Batch Neural Gas. However, in terms of performance, Batch Neural Gas outperforms KMeans. The best choice of algorithm therefore depends on you personal preference for speed vs performance.

Appendix

A kmeans.m

../Code/kmeans.m

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

dat = checkerboard;
k = 100;
prototypeSelector = 2;
writeOutput = 1;
if prototypeSelector == 0
    % 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

end
if prototypeSelector == 1
    % Init the prototypes
    prototypes = zeros(k,ndims(dat));
    newPoint = dat(randi(length(dat)),1:2);
    distances = pdist2(newPoint, dat(:,1:2));
    prototypes(1,:) = newPoint;

    for i = 2:k
        probabilities = distances.ˆ2;
        % Choose a random number in the range of sum(probabilities)
        newPointIndex = find(cumsum(probabilities) > rand*sum(probabilities),1);
        newPoint = dat(newPointIndex,:);
        prototypes(i,:) = newPoint;

        % Calculate the new distances (select min value)
        distances = min(distances, pdist2(newPoint, dat(:,1:2)));
    end
end
if prototypeSelector == 2
    sbrace = @(x,y)(x{y});
    fromfile = @(x)(sbrace(struct2cell(load(x)),1));
    prototypes=fromfile('clusterCentroids.mat');
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(max(mod(i,size(shapes,2)),1)), '
        MarkerSize', 10, 'MarkerFaceColor', 'black')
end

% Perform k-means
loopCounter = 0;
```

```
57  loop = 1;
58  while(loop == 1)
59      loop = 0;
60      % Uncomment to show loop counter :D
61      loopCounter = loopCounter + 1
62
63      for point = 1 : length(dat)
64          dat(point,3) = find(pdist2(dat(point,1:2), prototypes) == min(pdist2(dat(point,1:2),
                prototypes)),1);
65      end
66
67      for prototype = 1 : size(prototypes, 1)
68          newMean = mean(dat(dat(:,3) == prototype,1:2));
69          if newMean ~= prototypes(prototype,:)
70              loop = 1;
71          end
72          plot_arrow( prototypes(prototype,1), prototypes(prototype,2), newMean(:,1), newMean
                (:, 2));
73          prototypes(prototype,:) = newMean;
74          plot(newMean(1),newMean(2),'Marker', shapes(max(mod(prototype,size(shapes,2)),1)), '
                MarkerSize', 10, 'MarkerFaceColor', 'black')
75      end
76  end
77
78  % Calculate the quantization error
79  qError = 0;
80  for i = 1 : size(prototypes, 1)
81      qError = qError + sum(pdist2(prototypes(i,:), dat(dat(:,3) == i,1:2)));
82  end
83
84  % More figure stuff
85  %legend(num2str(1:k))
86  if writeOutput == 1
87      print(sprintf('../Report/Fig1_KMeans'), '-depsc');
88  end
89  figure(2)
90  hold on;
91  gscatter(dat(:,1),dat(:,2),dat(:,3),[],shapes, 5, 'off')
92  voronoi(prototypes(:,1),prototypes(:,2))
93  for i = 1 : size(prototypes, 1)
94      plot(prototypes(i,1),prototypes(i,2),'Marker', shapes(max(mod(i,size(shapes,2)),1)), '
            MarkerSize', 13, 'MarkerFaceColor', 'black')
95  end
96
97  xlabel('x');
98  ylabel('y');
99  if writeOutput == 1
100     print(sprintf('../Report/Fig2_KMeans'), '-depsc');
101 end
```

B run.m

../Code/run.m

```
1  load('kmeans1.mat', 'kmeans1');
2  load('checkerboard.mat', 'checkerboard')
3
4  error= zeros(1,10);
5  kmax = 20;
6  J = zeros(1, kmax);
7  R = zeros(1,kmax);
8
```

```
% 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, 2, 1, 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');

% Perform the kmeans++ test
k = 100;
nRuns = 20;
error_without = zeros(1,nRuns);
error_with    = zeros(1,nRuns);
for i = 1:nRuns
    datestr(now)
    i
    error_without(i) = kmeans(checkerboard, k, 0, 0);
    error_with(i)    = kmeans(checkerboard, k, 1, 0);
end
error_without = error_without/nRuns;
error_with = error_with/nRuns;

mean(error_without)
std(error_without)
mean(error_with)
std(error_with)

% using an unpaired one-tailed two-sample t-test

% With should give a smaller average, Without larger ->
% Tail right = x larger than y
% The variances are unequal
[h, p] = ttest2(error_without, error_with, 'Tail', 'right','Vartype','unequal');
p

% Compare batch Neural Gas with our kmeans
```

```
batchNG(checkerboard,100,500)
kmeans(checkerboard,100,2,1)
```

C batchNG.m

../Code/batchNG.m

```
function [prototypes] = batchNG(Data, n, epochs, xdim, ydim)

% Batch Neural Gas
%   Data contains data,
%   n is the number of clusters,
%   epoch the number of iterations,
%   xdim and ydim are the dimensions to be plotted, default xdim=1,ydim=2

%% TEst
% dat = kmeans1;
% n = 100;
% epochs = 1;

error(nargchk(3, 5, nargin));  % check the number of input arguments
if (nargin<4)
    xdim=1; ydim=2;   % default plot values
end;

[dlen,dim] = size(Data);

%prototypes =  % small initial values
%% or
sbrace = @(x,y)(x{y});
fromfile = @(x)(sbrace(struct2cell(load(x)),1));
prototypes=fromfile('clusterCentroids.mat');


lambda0 = n/2; %initial neighborhood value
% lambda
lambda = lambda0 * (0.01/lambda0).^([0:(epochs-1)]/epochs);
% note: the lecture slides refer to this parameter as sigma^2
%       instead of lambda

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Action

for i=1:epochs,
    datestr(now)
    i
    D_prototypes = zeros(n,dim);   % difference for vectors is initially zero
    D_prototypes_av = zeros(n,1);        % the same holds for the quotients

    for j=1:dlen,  % consider all points at once for the batch update

        % sample vector
        x = Data(j,:); % sample vector
        X = x(ones(n,1),:);  % we'll need this

        % neighborhood ranking

        % Distances (sorted) to each prototype
        % 1st column = distance
        % 2d column = prototype index
        distances = sortrows([pdist2(x, prototypes); 1:n].');
        % 3d column = rank
```

```
56          ranks = [distances.'; 1:n].';
57          % Sort according to prototype index
58          ranks = sortrows(ranks, 2);
59
60          % 1-BMU, 2-BMU, etc. (hint:sort)
61          %find ranking,h,H
62
63          % accumulate update
64          D_prototypes = D_prototypes + (exp((-1*ranks(:,3))/lambda(i)) * x);
65          D_prototypes_av = D_prototypes_av + exp((-1*ranks(:,3))/lambda(i));
66      end
67      D_prototypes = D_prototypes ./ [D_prototypes_av D_prototypes_av];
68
69      % update
70      prototypes = D_prototypes ;
71
72      %plot after 20, 100, 200, and 500 epochs
73  %     if 1,
74      if sum(i == [20, 100, 200, 500])
75          figure();
76          fprintf(1,'%d / %d \r',i,epochs);
77          hold off
78          plot(Data(:,xdim),Data(:,ydim),'bo','markersize',3)
79          hold on
80          plot(prototypes(:,xdim),prototypes(:,ydim),'r.','markersize',10,'linewidth',3)
81          % write code to plot decision boundaries
82          %     ...
83          %     plot decision boundaries here
84          %     ...
85          %pause
86          %or
87          voronoi(prototypes(:,xdim),prototypes(:,ydim));
88          drawnow
89          print(sprintf('../Report/NeuralGas_%d', i), '-depsc');
90      end
91  end
92  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```