Deep Learning – Assignment 1

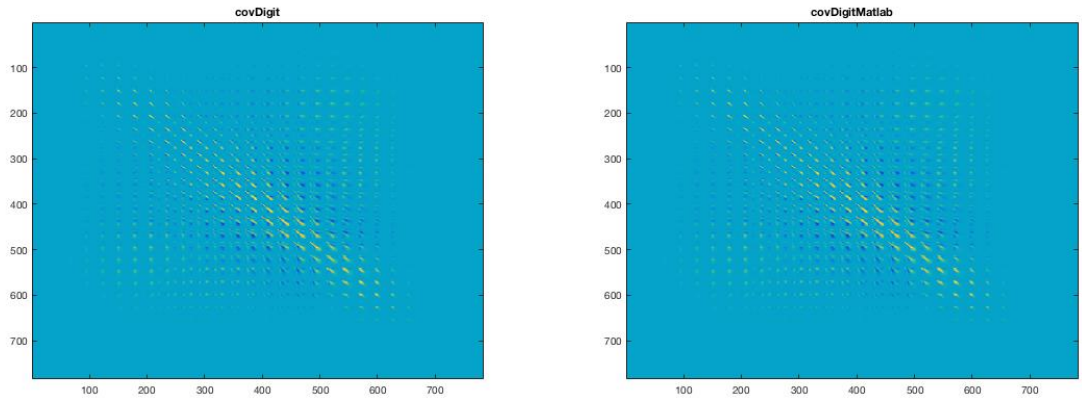
Flavie VAMPOUILLE (MSc in DS&BA - ESSEC/CENTRALE)

1 Data modeling: PCA and K-means

1.1 PCA

Covariance matrix

```
%% calcul covariance matrix
covDigits = 0;
DtrainMinusMean = Dtrain - repmat(meanDigit',nsamples,1);
for k=1:nsamples
    covDigits = covDigits + (DtrainMinusMean(k,:)'*DtrainMinusMean(k,:)) /
(nsamples-1);
end

%% make sure covDigitsMatlab = your covDigits
figure,imagesc(covDigitsMatlab); title('covDigitMatlab');
figure,imagesc(covDigits); title('covDigit');
```



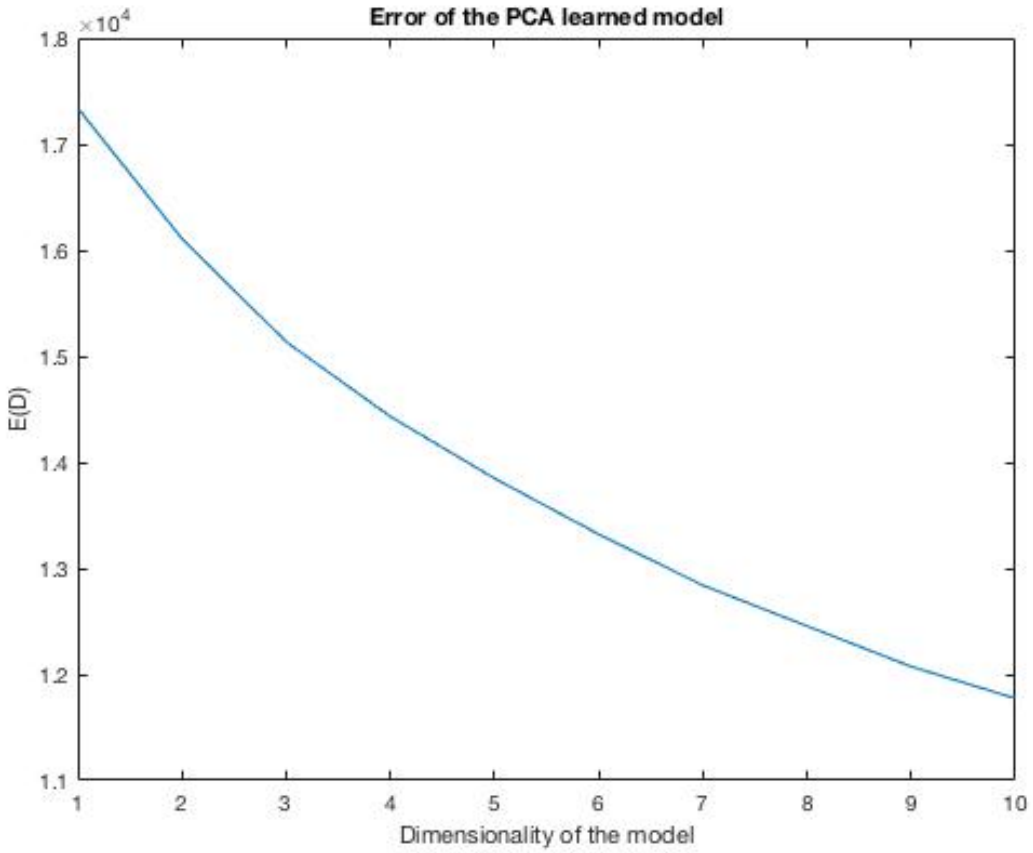

```
%% calcul check
sum ( sum ( abs(covDigitsMatlab - covDigits) ) )
```

When we check the difference between the two covariance matrices we obtain something very close to zero (2.7617e-11).

Plot E(D)

```
%% 11 first eigenvectors and eigenvalues
[eigvec,eigvals] = eigs(covDigitsMatlab,11) ;

%% Number of images in Dtest
ntest = size(Dtest,1) ;
```

```
%% Centered Dtest matrix
DtestCentered = Dtest - repmat(meanDigit',ntest,1);

%% expansion coefficients : Cnd = <In,eigvec(d)> / norm(eigvec(d))^2
Cnd = zeros(ntest,11) ;
for d=1:11
    for n=1:ntest
        Cnd(n,d) = dot(DtestCentered(n,:)',eigvec(:,d)) /
norm(eigvec(:,d))^2 ;
    end
end

%% Quality of the learned model
E = zeros(1,11) ;
for D=1:11
    for n=1:ntest
        S = meanDigit ;
        for d=1:D
            S = S + Cnd(n,d)*eigvec(:,d) ;
        end
        E(D) = E(D) + norm ( Dtest(n,:)' - S ) ;
    end
end

%% plot E(D)
figure ;
E = E(1:10);
input = [1:1:10] ;
plot ( input , E )
title('Quality of the PCA learned model') ;
xlabel('Dimensionality of the model') ;
ylabel('E(D)') ;
```



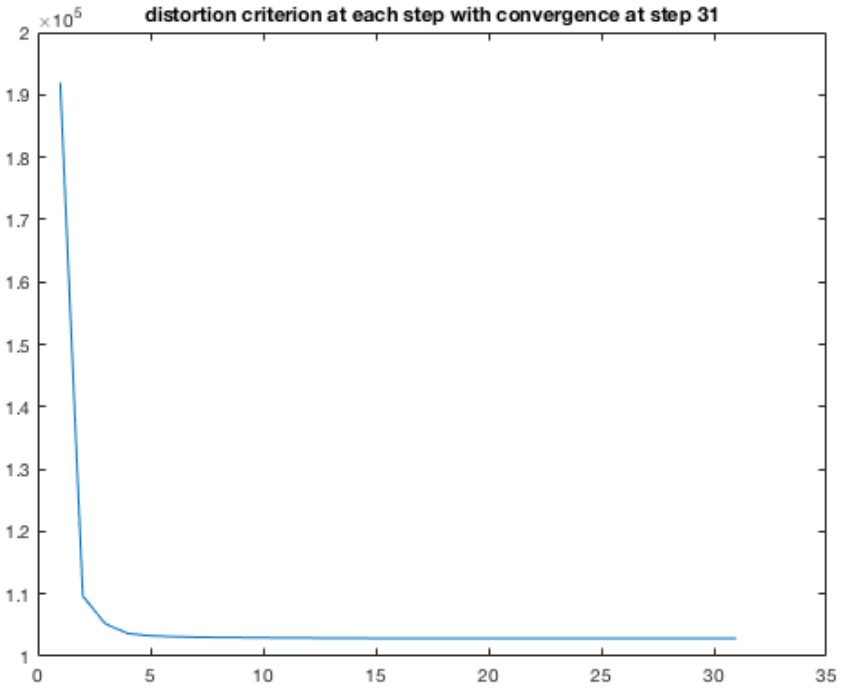
1.2 K-means

----------Functions ----------

```
% get initial centers randomly
function ao_centers = get_initial_Centers(ai_K, ai_Data)
w_index = randi(size(ai_Data,1),1,ai_K) ;
ao_centers = ai_Data(w_index,:) ;
end

% evaluate distances between data and centers
function ao_distances = evaluate_distances( ai_centroid , ai_Data)
ao_distances = zeros (size(ai_Data,1),size(ai_centroid,1)) ;
for i = 1 : size(ai_Data,1)
    for k = 1 : size(ai_centroid,1)
        ao_distances(i,k) = norm(ai_Data(i,:) - ai_centroid(k,:))^2 ;
    end
end
end

% affect to cluster depending on minimal distance et return the distortion
criterion
function [w_label,distortion] = minimal_distance(ai_distances)
distortion = 0 ;
[M,w_label] = min(ai_distances,[],2) ;
distortion = sum (M) ;
end

% update center at each iteration
function ao_centroid = update_centers(ai_K , ai_data, w_label)
ao_centroid = zeros (ai_K,size(ai_data,2)) ; n = zeros (ai_K,1) ;
for k = 1 : ai_K
    for i = 1 : size(ai_data,1)
            if w_label(i) == k
                n(k) = n(k) + 1 ;
                ao_centroid(k,:) = ao_centroid(k,:) + ai_data(i,:) ;
            end
    end
    ao_centroid(k,:) = ao_centroid(k,:) / n(k) ;
end
end

% kmeans function
function [ao_centroid,distortion,distortionCost,converge] = Kmeans( ai_K ,
ai_Data , ai_numIteration )

ao_centroid = get_initial_Centers(ai_K, ai_Data) ;
distortionCost = zeros(ai_numIteration) ;

% local copy
w_Data = ai_Data ;

% initialize labels
w_label = zeros(size(ai_Data,1)) ;

for w_iter = 1:ai_numIteration
    % 1 / Evaluate distance to each centroid
    w_distance = evaluate_distances(ao_centroid, w_Data) ;
    % 2 / Find the minimum value
    test_label = w_label ;
    [w_label,distortion] = minimal_distance(w_distance) ;
```

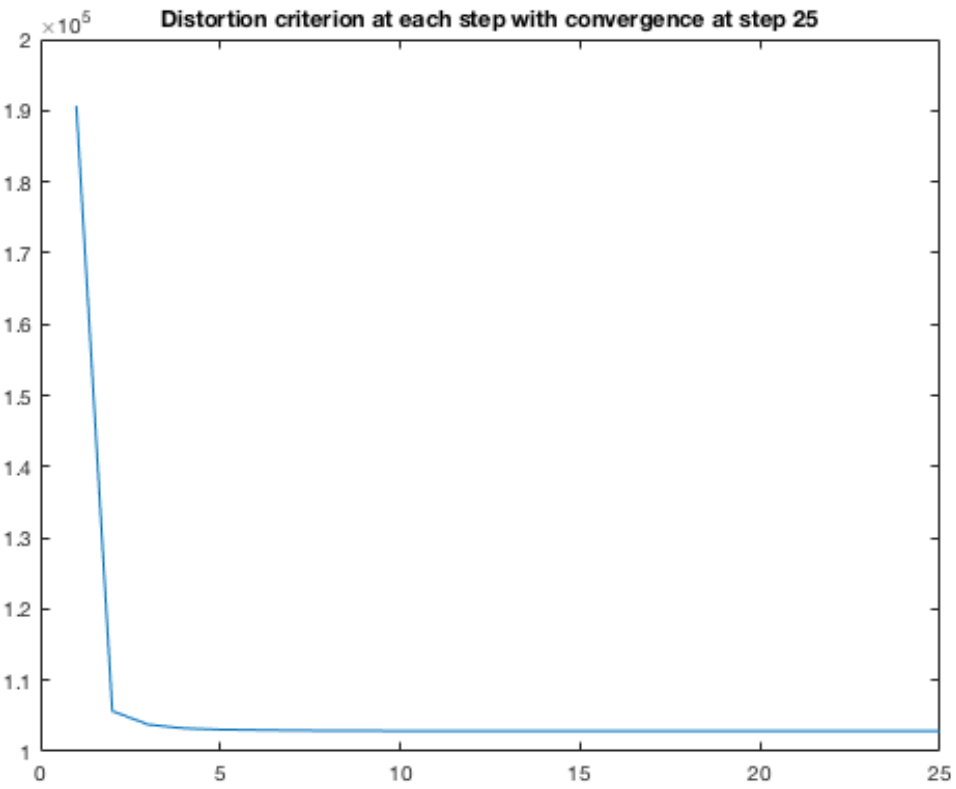
```
    distortionCost(w_iter) = distortion ;
    % disp(['The distortion criterion at step ' num2str(w_iter) ' is : '
num2str(distortion)])
    if test_label == w_label
        converge = w_iter ;
        break
    end
    % 3 / Update Centroid Value
    ao_centroid = update_centers(ai_K, w_Data, w_label) ;
end
end
```

```
%% number of centroids
w_K = 2 ;

%% number max of iterations
w_numIteration = 100 ;

%% return the w_K group centers, print criterion distortion at each step
[ao_centroid,distortion,distortionCost,converge] = Kmeans( w_K , Dtrain ,
w_numIteration ) ;

%% plot the ditortion at each step
input = (1:converge) ;
distortionCost = distortionCost(distortionCost~=0);
plot (input,distortionCost) ;
title(['distortion criterion at each step with convergence at step '
num2str(converge)]) ;
```

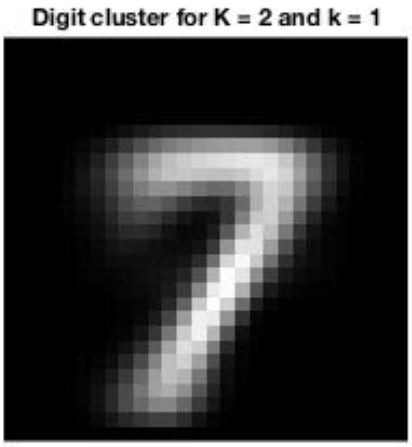


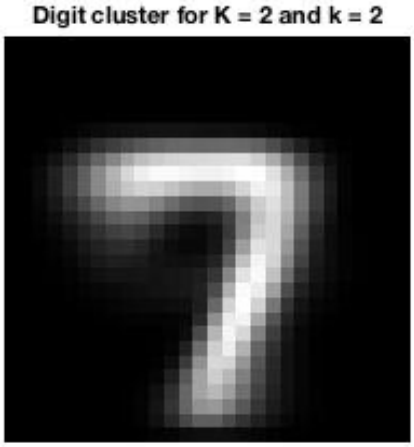
We can see that the distortion criterion decreases as the k-means algorithm progresses. The best distortion on Dtrain for K = 2 is in this case 102872.0868 and the cluster affectations stop changing at step 31.

```
%% Run 10 times k-means functions with different initial centers and keep
the best solution

bestDistortion = Inf(1) ;
for t = 1:10
    [finalCentroid,distortion,distortionCost,converge] = Kmeans( w_K ,
Dtrain , w_numIteration ) ;
    if distortion < bestDistortion
        bestDistortion = distortion ;
        bestCentroid = finalCentroid ;
        bestDistortionCost = distortionCost ;
        bestConverge = converge ;
    end
end

disp ( ['The best distortion on Dtrain for K = 2 is : '
num2str(bestDistortion)] )

input = (1:bestConverge) ;
bestDistortionCost = bestDistortionCost(bestDistortionCost~=0);
plot (input,bestDistortionCost) ;
title(['Distortion criterion at each step with convergence at step '
num2str(bestConverge)]) ;
```



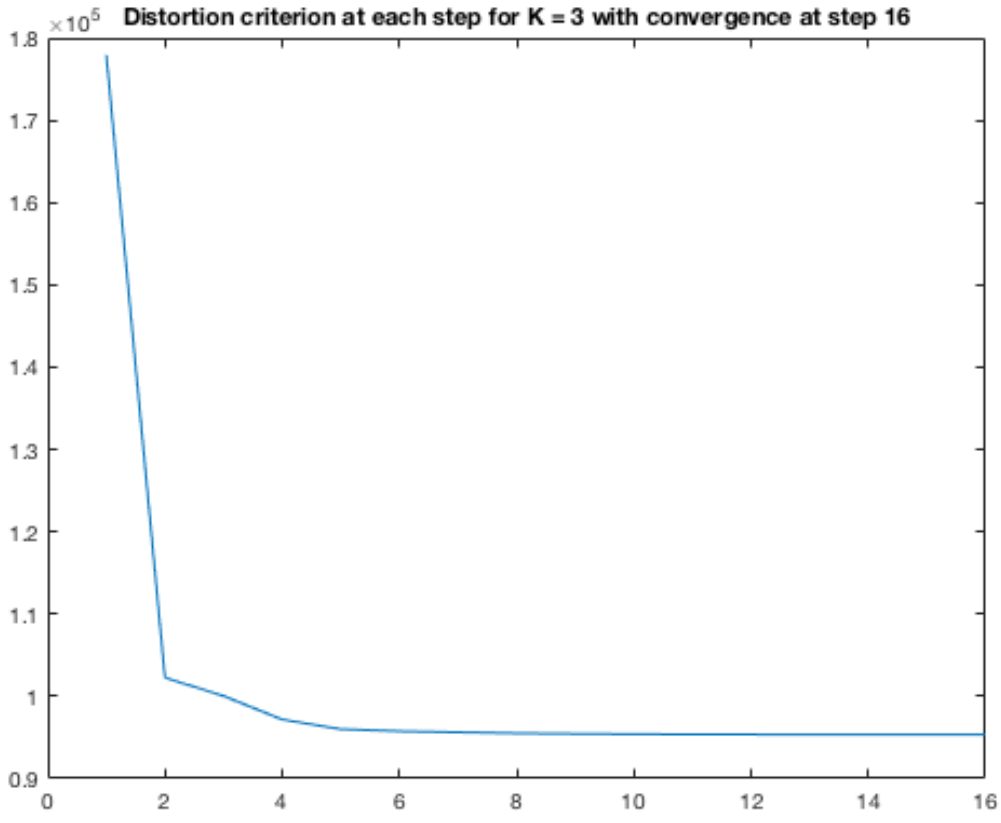

```
%% resulting digit cluster

figure ;
for k = 1 : w_K
    subplot(1,w_K,k) ;
    ClusterImage = reshape(bestCentroid(k,:),[28,28]) ;
    imshow(ClusterImage,[]) ;
    title (['Digit cluster for K = ', num2str(w_K), ' and k = ',
num2str(k)] ) ;
end
```



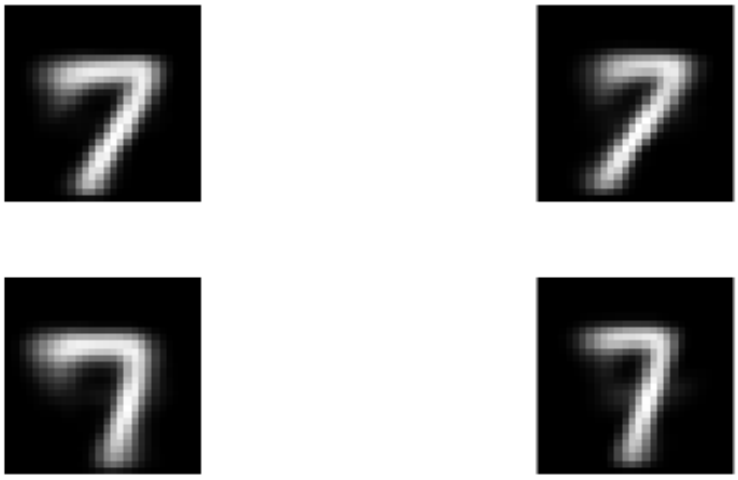


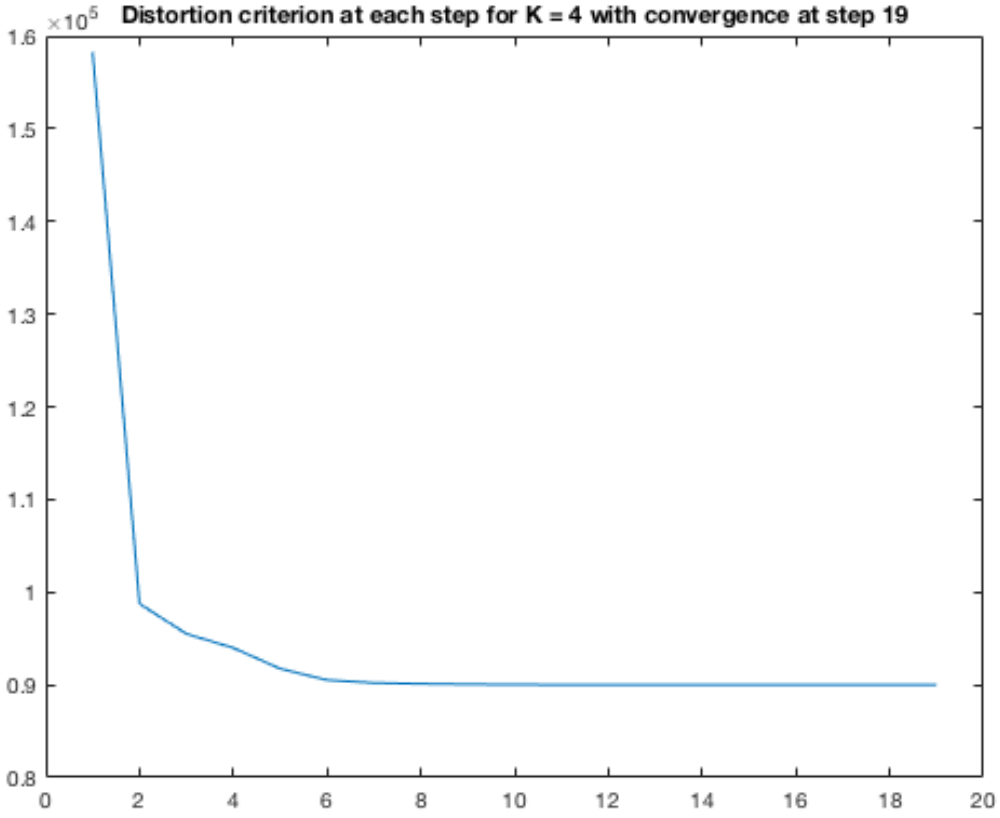

```
%% Repeat the process for K = 3, 4, 5, 10, 50, 100

allCentroid = cell([6,1]) ;
distortionDtrain = [] ;
n = 1 ;

for w_K=[3,4,5,10,50,100]

bestDistortion = Inf(1) ;
for t = 1:10
    [finalCentroid,distortion,distortionCost,converge] = Kmeans( w_K ,
Dtrain , w_numIteration ) ;
    if distortion < bestDistortion
        bestDistortion = distortion ;
        bestCentroid = finalCentroid ;
        bestDistortionCost = distortionCost ;
        bestConverge = converge ;
    end
end

disp ( ['The best distortion on Dtrain for K = ' num2str(w_K) ' is : '
num2str(bestDistortion)] )

figure ;
input = (1:bestConverge) ;
bestDistortionCost = bestDistortionCost(bestDistortionCost~=0);
plot (input,bestDistortionCost) ;
title(['Distortion criterion at each step for K = ' num2str(w_K) ' with
convergence at step ' num2str(bestConverge)]) ;

distortionDtrain = [distortionDtrain,bestDistortion] ;
allCentroid{n} = bestCentroid ;
n = n+1 ;

figure ;
for k = 1 : w_K
    subplot(round(sqrt(w_K)+1),round(sqrt(w_K)),k) ;
    ClusterImage = reshape(bestCentroid(k,:),[28,28]) ;
    imshow(ClusterImage,[]) ;
end

end
```

K = 3

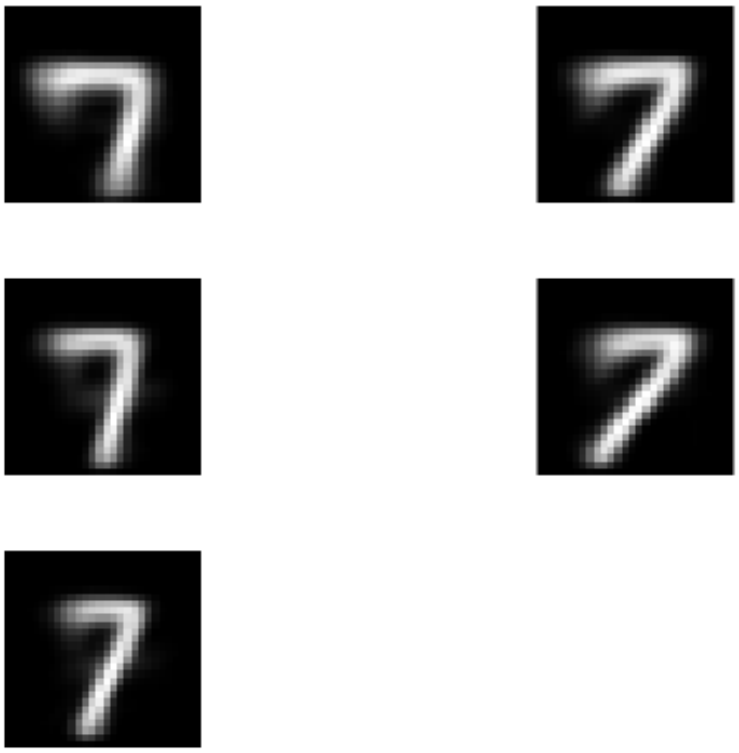

Distortion criterion at each step for K = 3 with convergence at step 16
×10^5
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0
2
4
6
8
10
12
14
16

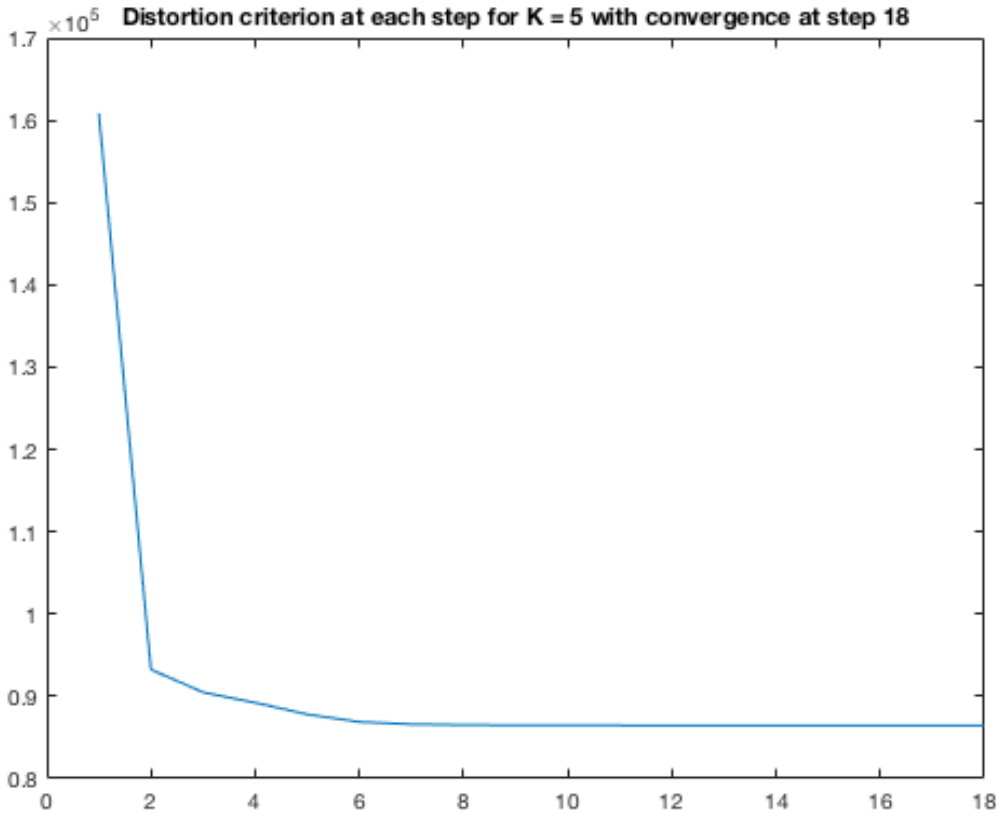
K = 4



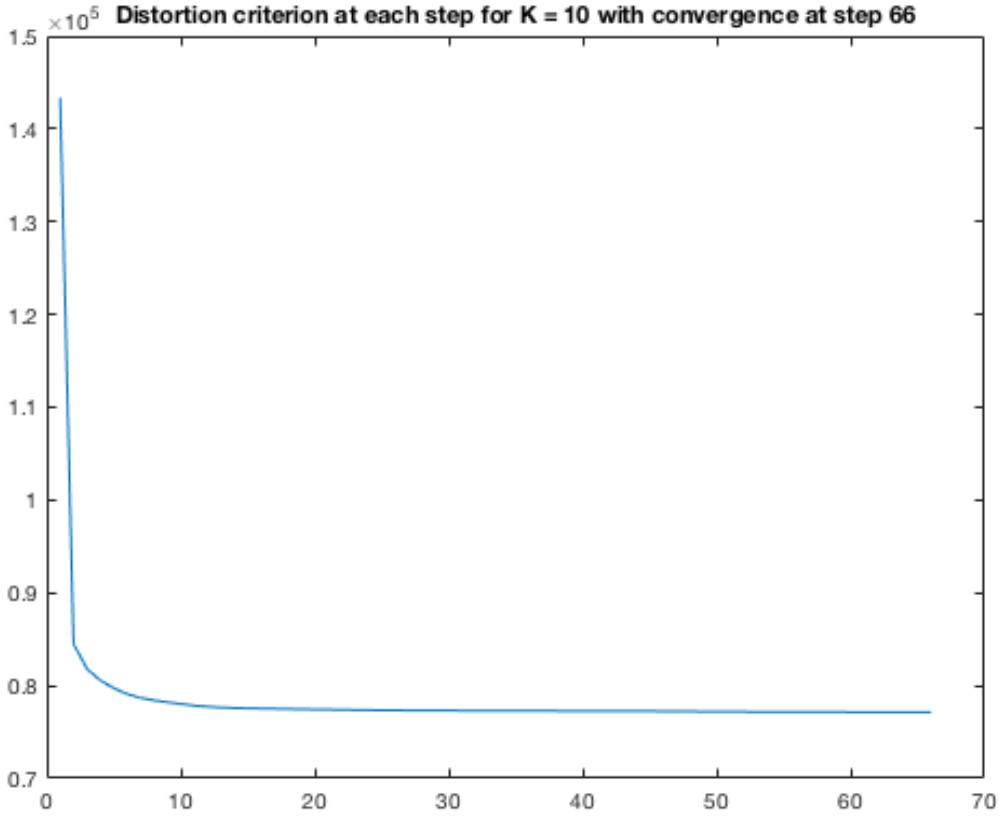

Distortion criterion at each step for K = 4 with convergence at step 19
×10^5
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0
2
4
6
8
10
12
14
16
18
20

K = 5

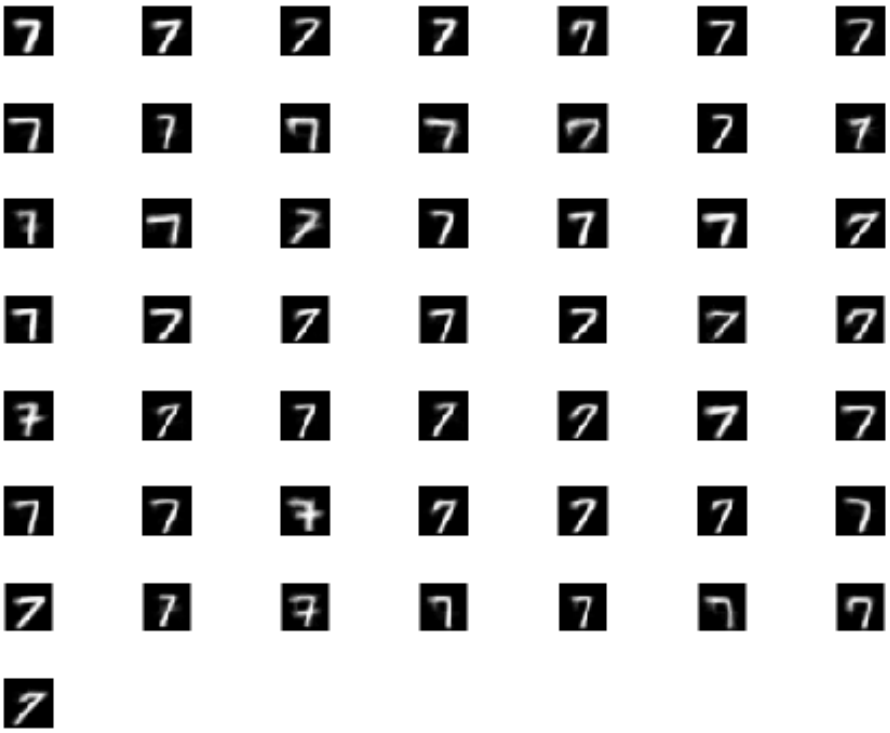


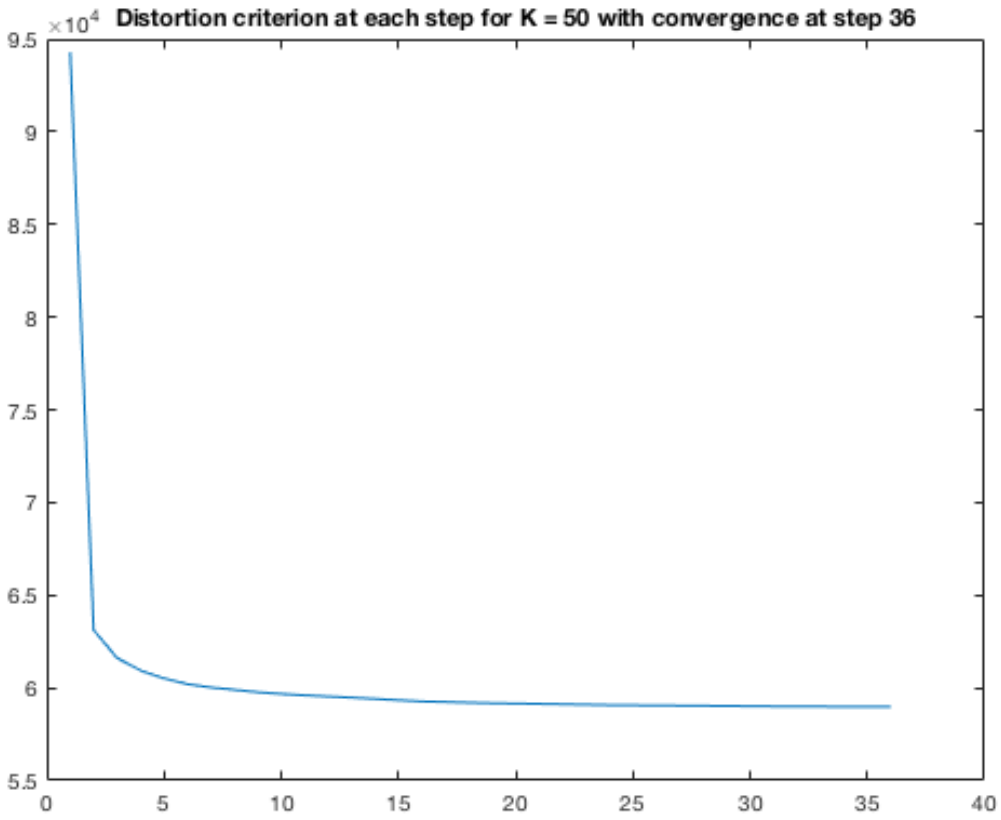

Distortion criterion at each step for K = 5 with convergence at step 18
×10^5
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0
2
4
6
8
10
12
14
16
18

K = 10

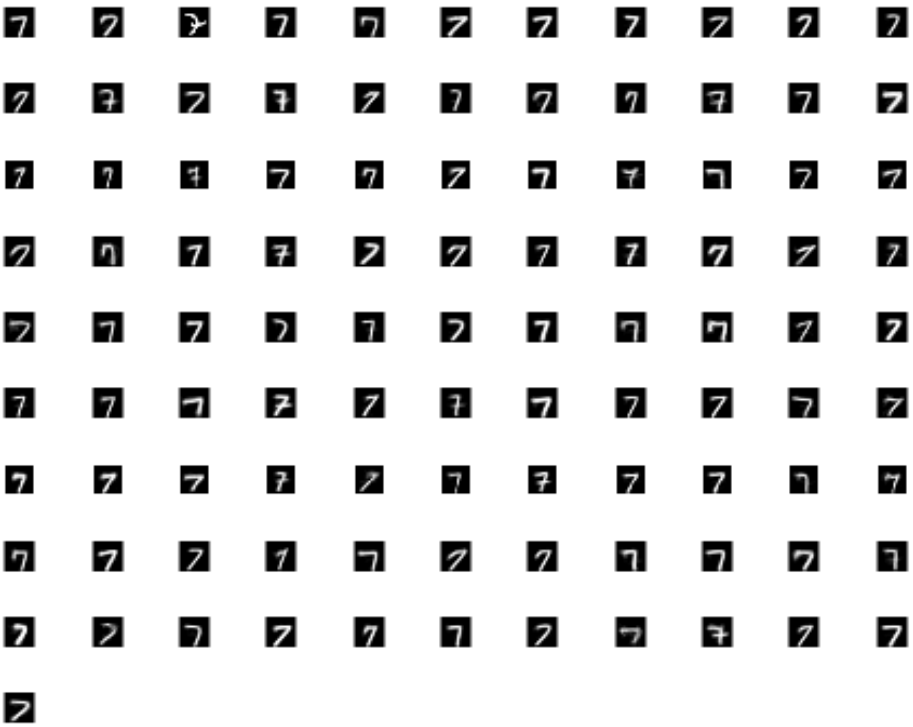

Distortion criterion at each step for K = 10 with convergence at step 66
×10^5
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0
10
20
30
40
50
60
70

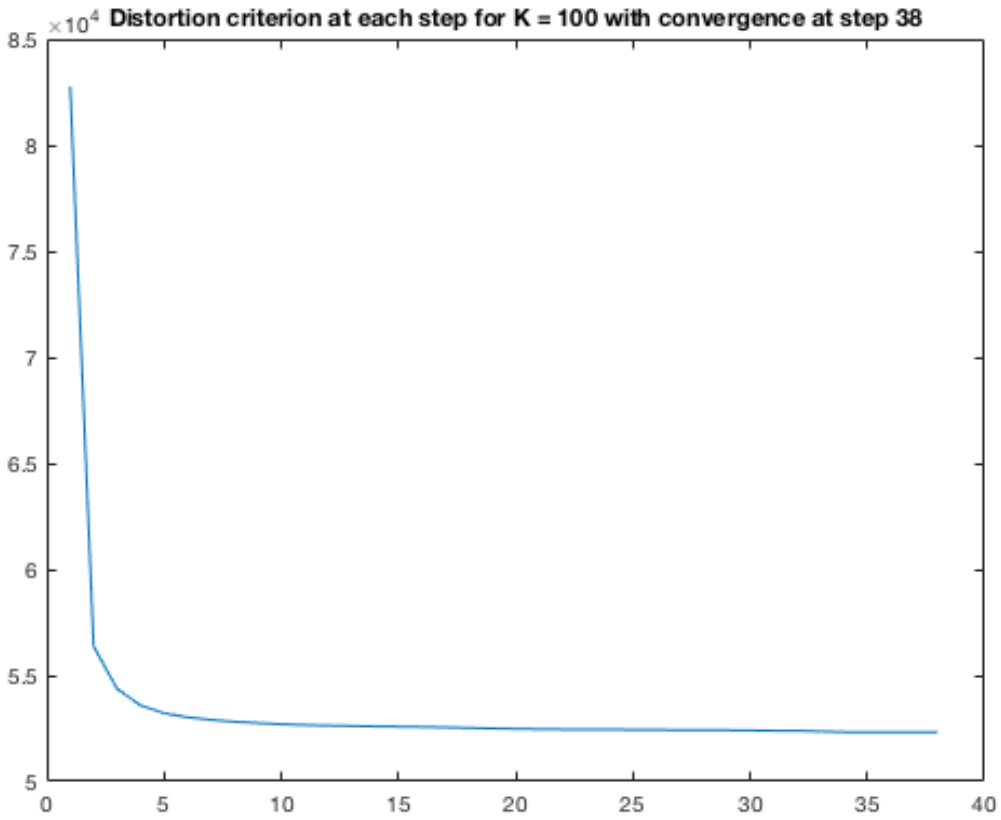
K = 50




Distortion criterion at each step for K = 50 with convergence at step 36
×10^4
9.5
9
8.5
8
7.5
7
6.5
6
5.5
0
5
10
15
20
25
30
35
40

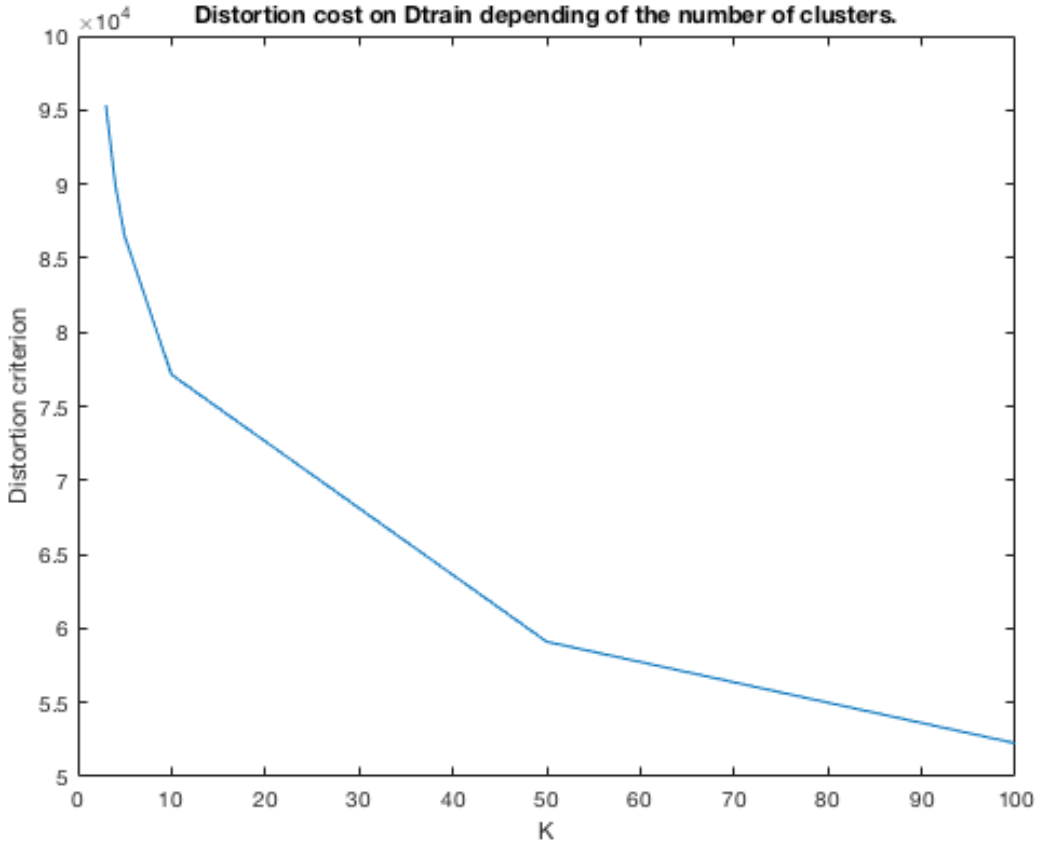
K = 100




Distortion criterion at each step for K = 100 with convergence at step 38
×10^4
8.5
8
7.5
7
6.5
6
5.5
5
0
5
10
15
20
25
30
35
40

```
The distortion cost on Dtrain for K = 3 is : 95350.0688
The distortion cost on Dtrain for K = 4 is : 89987.8759
The distortion cost on Dtrain for K = 5 is : 86471.686
The distortion cost on Dtrain for K = 10 is : 77134.2852
The distortion cost on Dtrain for K = 50 is : 59111.4601
The distortion cost on Dtrain for K = 100 is : 52252.5654
```

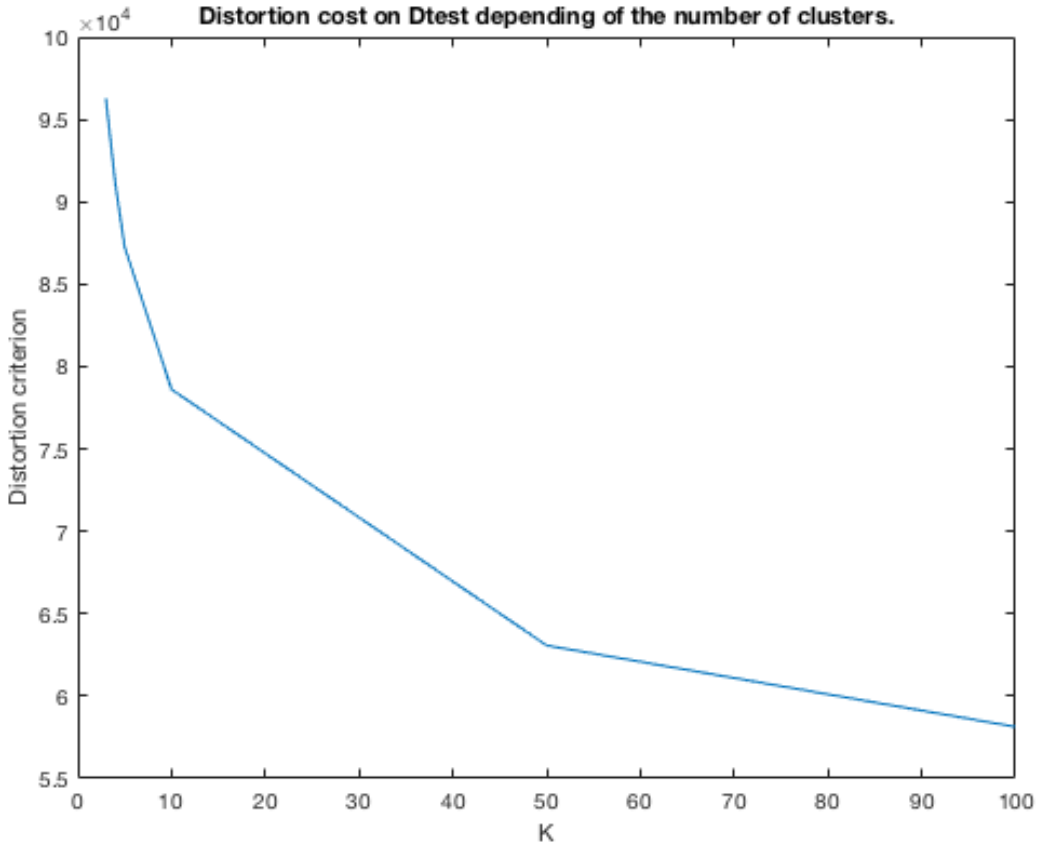
```
%% plot the distortion cost on Dtrain

image ;
input = [3,4,5,10,50,100] ;
plot (input,distortionDtrain) ;
title ('Distortion cost on Dtrain depending of the number of clusters.') ;
xlabel('K') ;
ylabel('Distortion criterion') ;
```



We observe that the distortion cost decreases as the number of cluster increases. The distortion cost being define as the sum of the distances of each point to the nearest center it is evident that this will happen. The more center we have, the smaller each distance will be.

```
%% Report the distortion cost on the testing data

n = 1 ;
distortionDtest = [] ;
for w_K=[3,4,5,10,50,100]
    distances = evaluate_distances(allCentroid{n},Dtest) ;
    [w_label,distortion] = minimal_distance(distances) ;
    disp ( ['The distortion cost for K = ' num2str(w_K) ' clusters on Dtest
is : ' num2str(distortion)] )
    distortionDtest = [distortionDtest,distortion] ;
    n = n+1 ;
end

image ;
input = [3,4,5,10,50,100] ;
plot (input,distortionDtest) ;
title ('Distortion cost on Dtest depending of the number of clusters.') ;
xlabel('K') ;
ylabel('Distortion criterion') ;
```

```
The distortion cost for K = 3 clusters on Dtest is : 96272.5491
The distortion cost for K = 4 clusters on Dtest is : 91143.4123
The distortion cost for K = 5 clusters on Dtest is : 87277.0387
The distortion cost for K = 10 clusters on Dtest is : 78617.2818
The distortion cost for K = 50 clusters on Dtest is : 63075.5821
The distortion cost for K = 100 clusters on Dtest is : 58138.2963
```



We see before that for a PCA model, the bigger the number of dimensions is, the smaller the error on the test set will be. So for bigger dimensionality, we have a better quality for the learned model. This will always happen because with higher dimension we keep more of the original information. However, if there is collinearity in the first information, the last eigenvalues can be null (or very small) and the model will no more improve (or very slightly) in adding dimensions. So we can keep only the first dimensions with the bigger eigenvalues.

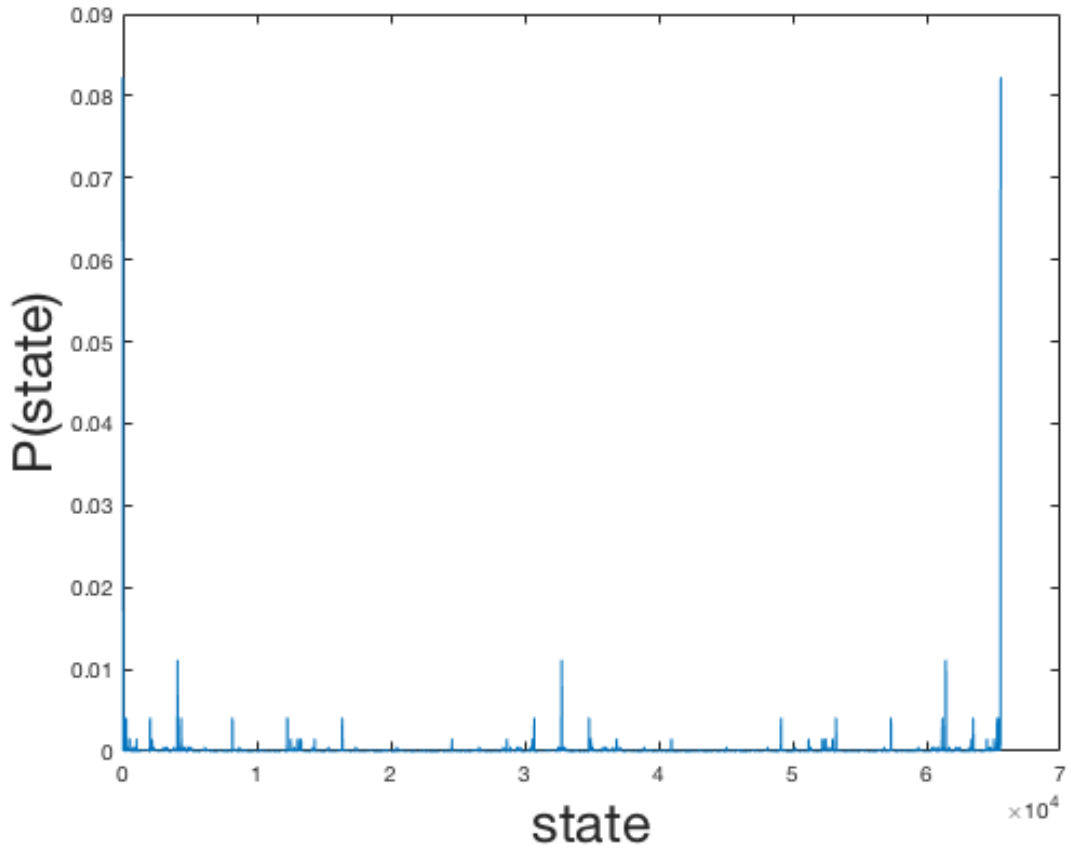
In the k-means method the more cluster we take, the lower our cost function will be. However, the model will not be necessary better and there will be a number K of cluster for which it will be optimum. There exist several methods to evaluate the best number of cluster.

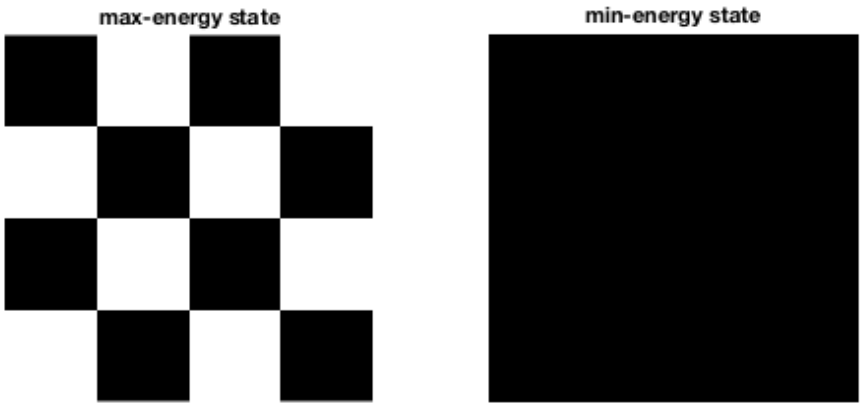
2 Ising model: MCMC & learning

2.1 Part-1: Brute-force evaluations

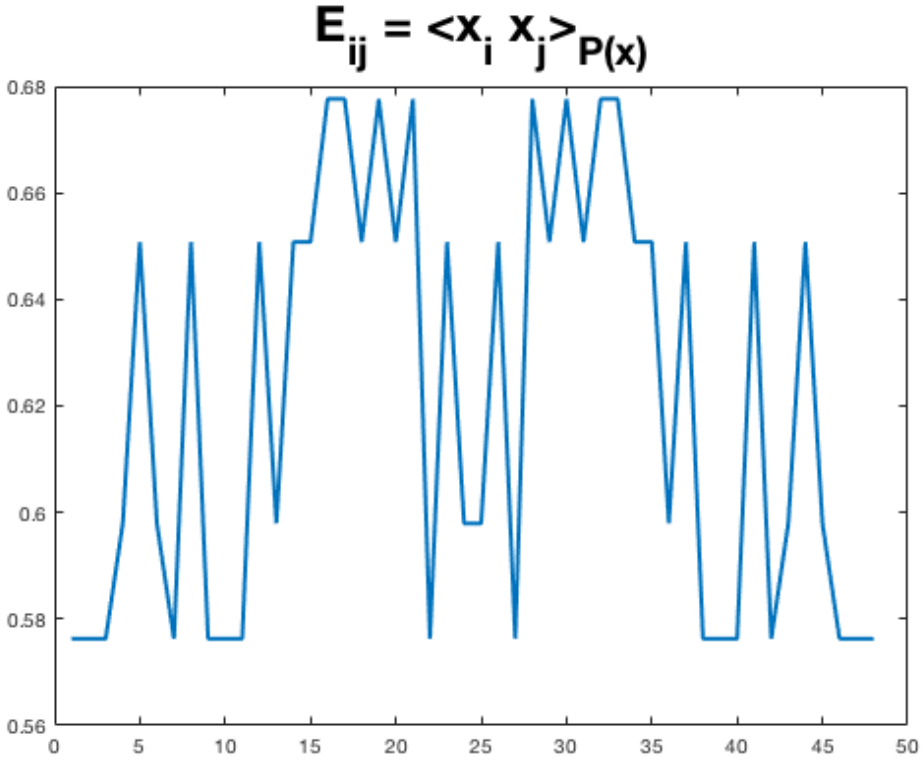
```
%% construct [1 x 2^{nnodes}] vector ener such that ener(1,i) measures the
energy of states(:,i)
ener        = -.5*sum((states*Js).*states,2) ;

%% compute partition function
Z           = sum(exp(-ener)) ;

%% compute probability of state i in [1 x 2^{nnodes}] vector P
P           = exp(-ener)/Z;
```

Probability P(x ;J) that the network is in any state x





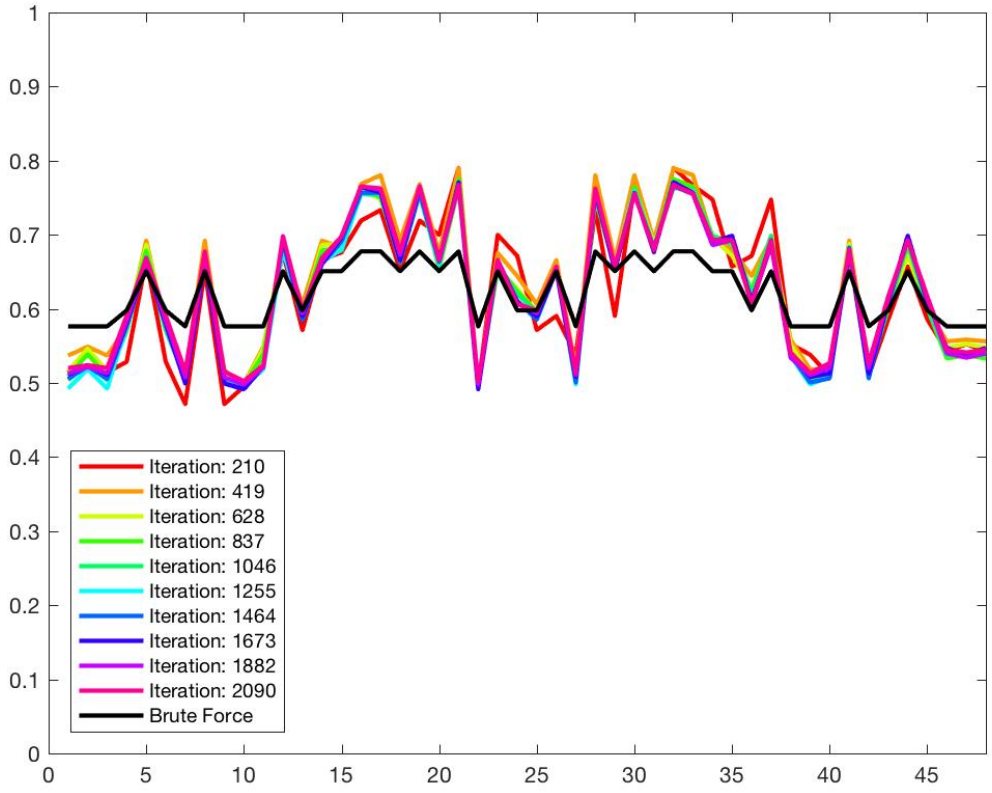

```
%% Brute-force computation of feature expectations
%% E_{ij} = <x_i x_j>_{P(x)} = sum_{x} P(x) [x_i x_j]

for k=1:nedges
        E_ij(k) = sum(P.*states(:,is(k)).*states(:,js(k))) ;
end
```

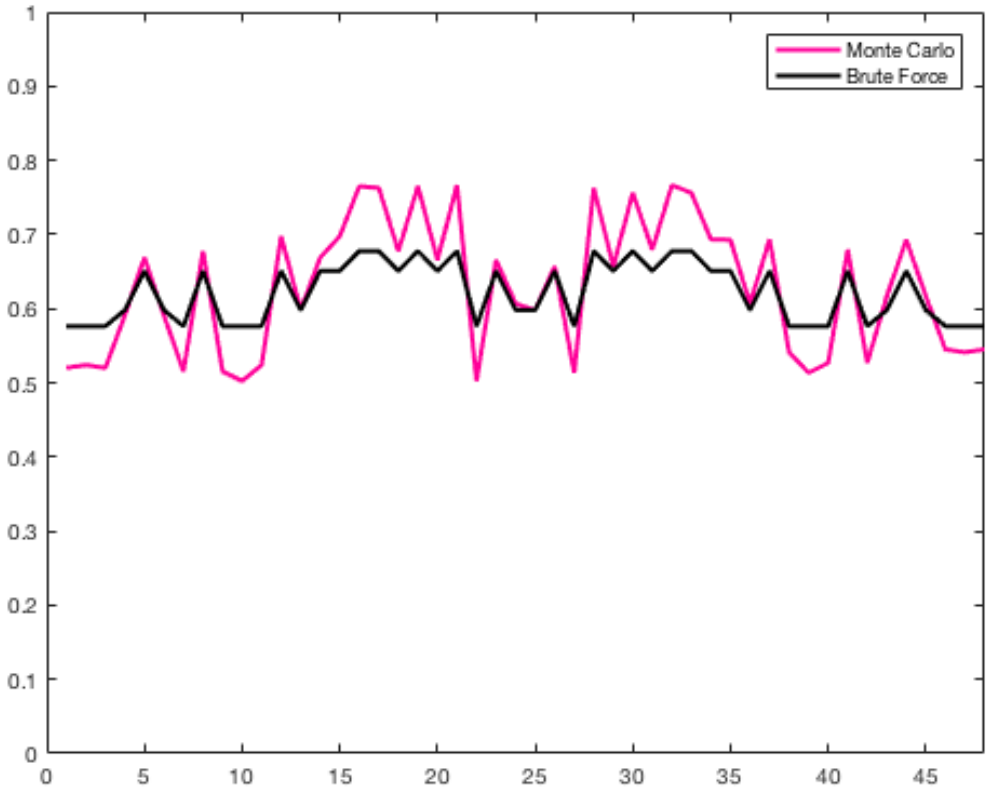


2.2 Part-2: MCMC evaluations

```
%% Gibbs sampling, inner loop

%% energy if we assign state +1 to 'position'
ener_p1 =    - sum ( neighsCon.*X ) ;

%% ditto for state -1
ener_m1 =    + sum ( neighsCon.*X ) ;

%% posterior probability that 'position' will be +1
p1          = exp(-ener_p1) ./ ( exp(-ener_m1) + exp(-ener_p1) ) ;

%% decide whether to set 'position' to +1 or -1
state       = double(p1>rand(1)) - ( 1 - double(p1>rand(1)) ) ;
```

Comparison between Monte Carlo estimate and brute-force evaluation of E_ij



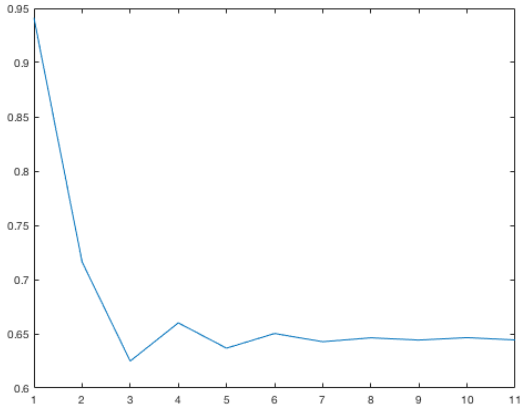


2.3 Part-3 : Parameter Estimation

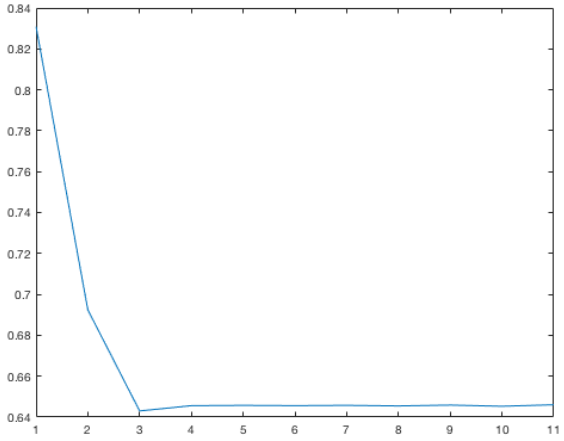
Use of the gradient of the likelihood of Boltzmann-Gibbs distribution with respect to J, starting at J = .5

In the gradient descent method we have $J_{k+1} = J_k - \lambda.\nabla \log L(P(x))$.
The $\lambda > 0$ is a small number that forces the algorithm to make small jumps. That keeps the algorithm stable and its optimal value depends on the function.
Here we test value for $\lambda =$ 0.04, 0.03 and 0.02.

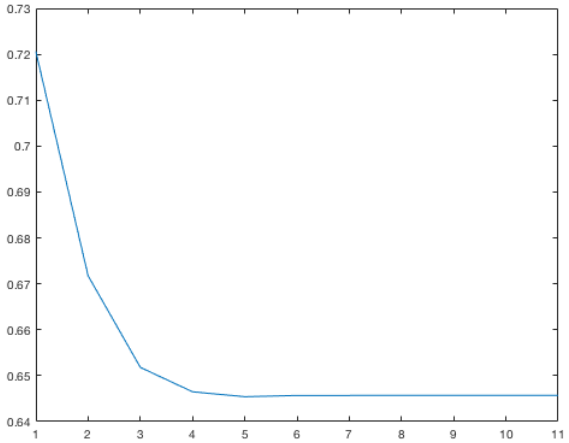
$\lambda = 0.04$



λ = 0.03



λ = 0.02

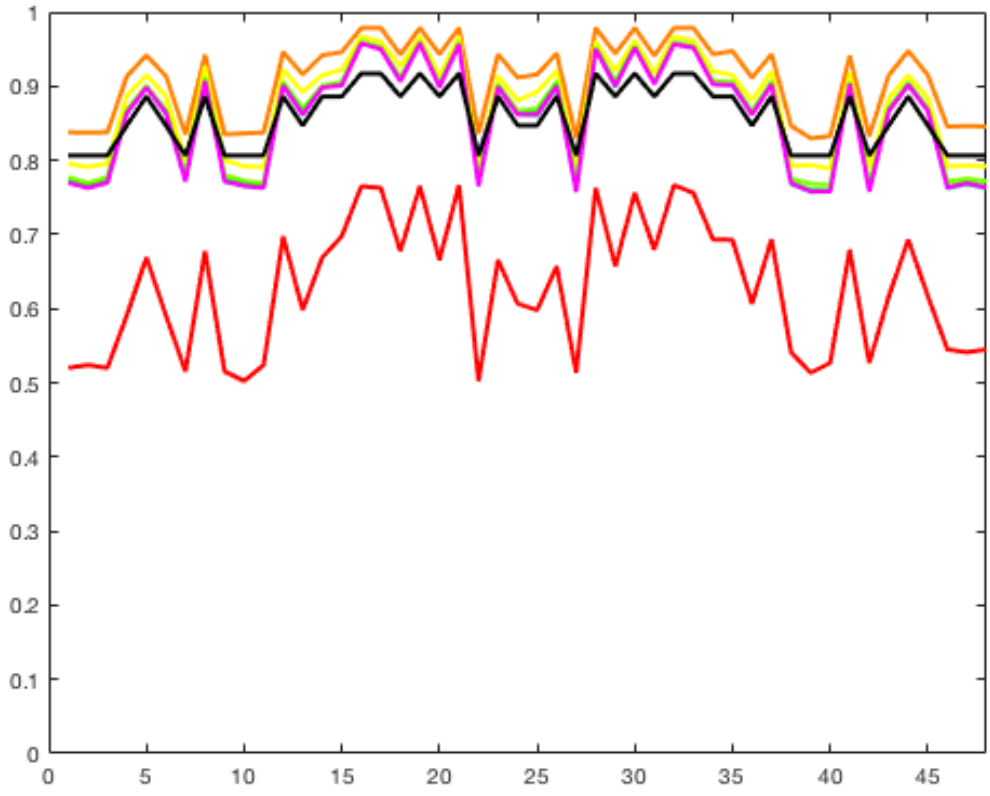


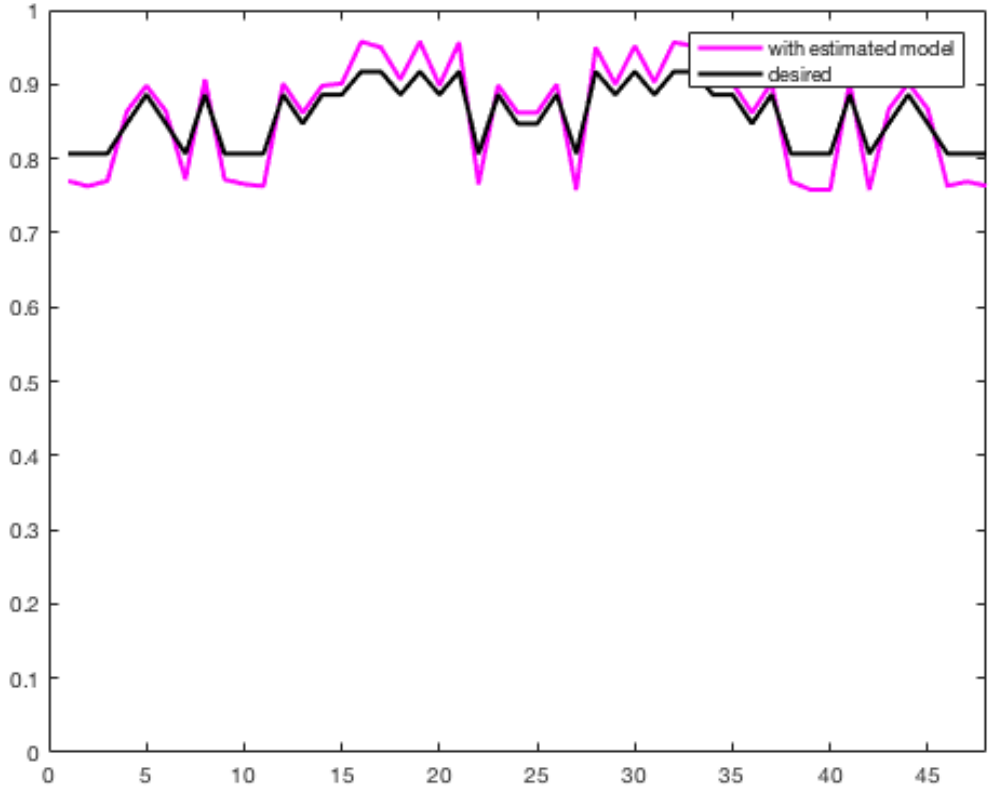
As λ = 0.02 seems good enough we keep this value to run the model.

```
J_ = J_ - 0.02*sum( avg_final - E_ij ) ;
```

We obtain : `J_ = 0.6455`

We obtain the following graphs:


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45


with estimated model
desired
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45