Table of Contents

```
% INSTRUCTIONS:
% 1) save this file into the directory that holds your work
% 2) make sure it runs without error
% 3) publish this file to PDF.  There are two ways to do this:
%    a) You can either use the 'Publish' tab, being sure to edit the
%    publishing options so the output format is 'pdf', or
%    b) run the commands below in the command window, one at a
%    time: (without the comment symbol!)
%    options.format = 'pdf';
%    publish('runAllHW9',options);
%
% 4) There should then be a PDF file in the subdirectory 'html'
%    that you can upload to TurnItIn. Make sure it has everything you
 need

% prepare to run...
clear all  % clear out all variables
close all  % close any open figures
```

KmeansCore

```
type('KmeansCore.m')


function ClusInfo = KmeansCore(dMat,Par)

%{
Inputs:
-dMat:  data matrix, size nFeat x nSamp
-Par: parameter structure, with fields:
    Par.nClus:  number of clusters
    Par.maxIter:  max iterations to run algorithm
    Par.isMedian:  update method: mean or median

Output:
-ClusInfo:  output structure, with fields:
    ClusInfo.clusIds: vector of cluster assignments, length nSamp
    ClusInfo.centerVecs: matrix of cluster center vectors,
size nFeat x nClus
    ClusInfo.sumDist:  summed total distance to assigned clusters,
length # iterations

Alyssa Rose HW10  04-15-2018
%}

%% initialzing values
centerVecs = InitializeCenters(dMat,Par.nClus);
isConverged = false;
k = 1;

%% loop
while isConverged == false
    % make a copy of our current best guess at cluster centers...
    centerVecsPrev = centerVecs;
    % calculate distance to each cluster center
    distVec = [];
    for g = (1:Par.nClus)
        distVec = [distVec; ComputeDistance(dMat, centerVecs(:,g))];
    end
    % reassign each vector to the nearest center
    ixClusIDs = ReassignClusters(distVec);
    % get summed distances across all clusters
    sumDist(k) = GetSummedDistances(distVec,ixClusIDs);
    % update the cluster centers
    centerVecs = UpdateClusterCenters(dMat,ixClusIDs,Par.nClus);
    % check: are we done?
    isConverged =
 CheckConvergence(centerVecs,centerVecsPrev,k,Par.maxIter);
    % call convergence check; pass in centerVecsPrev, other inputs
    k = k+1;
end


%% copy results to output structure
ClusInfo.sumDist = sumDist;
ClusInfo.clusIds = ixClusIDs;
ClusInfo.centerVecs = centerVecs;
return
```

InitializeCenters

```
type('InitializeCenters.m')
```

```
function centerVecs = InitializeCenters(dMat,nClus)
%{
assigns centerVec nClus # of random columns from dMat; nClus is
an integer, dMat is a matrix; centerVec size nFeat x nClus

Alyssa Rose  HW10  04-11-2018
%}
```

```
%finds dimensions dMat
[m,n] = size(dMat);
%creates random variable dMat column by nClus size
p = randperm(n, nClus);
%creates center vecs
centerVecs = dMat(:,p);
end
```

ComputeDistance

```
type('ComputeDistance.m')
```

```
function dist = ComputeDistance(dMat, centerVec)
%{
computes Euclidian distance from each data vector to the vector
 centerVec
Inputs: dMat is data matrix, size nFeat x nMeas
        centerVec a vector, size nFeat x 1
Output: dist is 1 x nMeas vector of Euclidian distances

Alyssa Rose  HW10  04-11-2018
%}

%finds Euclidian dist. using vector math
dist = sqrt(sum((dMat-centerVec).^2));
end
```

ReassignClusters

```
type('ReassignClusters.m')
```

```
function ixClusIDs = ReassignClusters(dist)
%{
assigns each cluster to the center it is closest to
Input: dist- matrix of distances to cluster centers, size nClus x
 nMeas
Output: ixClusIDs: vector of assigned cluster indices, length nMeas

Alyssa Rose  HW10  04-11-2018
%}

%finds index of mins (closest to center pt)
[~, ixClusIDs] = min(dist);
end
```

GetSummedDistances

```
type('GetSummedDistances.m')
```

```
function sumDist = GetSummedDistances(dist,clusIDs)
```

```
%{
Sums up the distance from every data vector to its assigned cluster
center.
Inputs: dist is nClus x nMeas matrix of distances to cluster centers
clusIds is the cluster number assigned to each example (length nMeas)
Output: sumDist is a scalar (1x1) holding summed distances from each
 measurement
to its corresponding cluster center

 Alyssa Rose  HW10  04-11-2018
%}

%preallocate/initialize
[~,nMeas]= size(dist);
sumDist = 0;
%creates sumDist
for k= 1:nMeas
    sumDist = sumDist + dist(clusIDs(k), k);
end
end
```

UpdateClusterCenters

```
type('UpdateClusterCenters.m')
```

```
function centerVecs = UpdateClusterCenters(dMat,ixClus,nClus,isMedian)
%{
compute new cluster centers, based on most recent cluster assignments
Inputs:
-dMat:  data matrix, size nFeat x nMeas
-ixClus:  vector of cluster # assigned to each example, length nMeas
-nClus:  total number of clusters in problem
Output:
-centerVecs: updated cluster center vectors, size nFeat x nClus

Alyssa Rose HW10 04-15-2018
%}
[m,~] = size(dMat);
centerVecs = zeros(m,nClus);
for h = (1:nClus)
    centerVecs(:,h) = mean(dMat(:,ixClus==h),2);
end
return
```

KmeansWrapper

```
type('KmeansWrapper.m')
```

```
function [GoodClusInfo,BadClusInfo] = KmeansWrapper(dMat,Par,nRepeats)
```

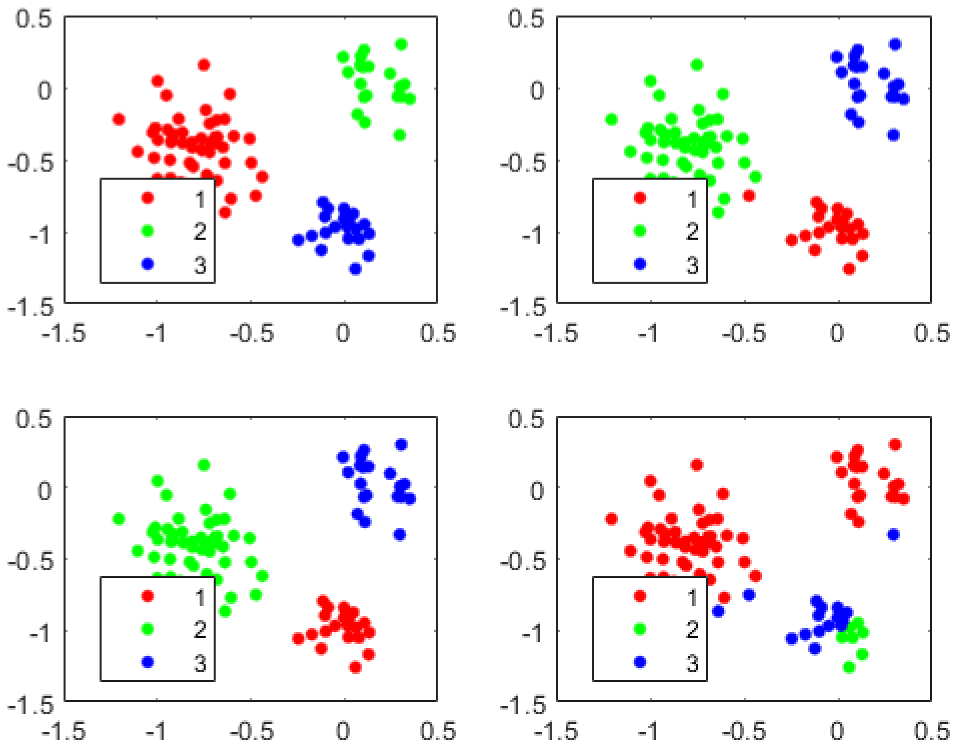
```
%{
wrapper code to run KmeansCore repeatedly and find best and worst
results, as measured by the final summed distance for each run
Inputs:
-dMat, Par: same as KmeansCore (see help there)
-nRepeats:  number of times to repeat kmeans
Outputs:
-GoodClusInfo:  output of KmeansCore that has the minimum
    summed distance between each vector and its assigned center
-BadClusInfo:  output of KmeansCore that has the maximum
    summed distance between each vector and its assigned center

Alyssa Rose HW10 04-15-2018
%}
%% pre allocating
trueSumDist = zeros(1,nRepeats);
ClusInfoCell = cell(nRepeats,3);

%% calls Kmeans core nRepeats time and stores best sumDist
for q = 1:nRepeats
    ClusInfo = KmeansCore(dMat,Par);
    ClusInfoCell(q,:) = struct2cell(ClusInfo);
    trueSumDist(q) = ClusInfo.sumDist(end);
end

%% creates fields and finds best/worst data
fields = {'sumDist','clusIds','centerVecs'};
[~,good] = min(trueSumDist);
[~, bad] = max(trueSumDist);

%% creates cell and converts to struct for readability
GoodClusInfo = cell2struct(ClusInfoCell(good,:),fields,2);
BadClusInfo = cell2struct(ClusInfoCell(bad,:),fields,2);
return
```

Test3clus

```
type('Test3clus.m')

fprintf('\n---------------------------------------------\n');
Test3clus
```

```
load threeClusTest.mat
%{
sript to test KMeans wrapper and core

Alyssa Rose HW10 04-15-2018
%}

%% creates structs and extracts good/bad info
Par = struct('nClus', 3, 'maxIter',100);
ClusInfo = KmeansCore(data2d,Par);
```

```
[GoodClusInfo,BadClusInfo] = KmeansWrapper(data2d,Par,20);

%% plotting
figure,
subplot(2,2,1)
gscatter(data2d(1,:),data2d(2,:),trueCluster)

subplot(2,2,2)
gscatter(data2d(1,:),data2d(2,:),ClusInfo.clusIds)

subplot(2,2,3)
gscatter(data2d(1,:),data2d(2,:),GoodClusInfo.clusIds)

subplot(2,2,4)
gscatter(data2d(1,:),data2d(2,:),BadClusInfo.clusIds)

%% Discussion: "why doesn't it group successfully?"
%{
The data isn't precise since their is a limit on how may times
it is allowed to run. If the program implemented for more
iterations, then the reassignments would be more accurate.
%}

%% Discussion: "KmeansWrapper comments, discuss the changes"
%{
The first two subplots of Test3clus appear to be close to actual
 grouping
with minimal error.
In the last two subplots, it is easy to distinguish the "good" and
 "bad"
data as the good data has minimal error and the bad data appears to be
entirely random and ungrouped.
%}

---------------------------------------------
```

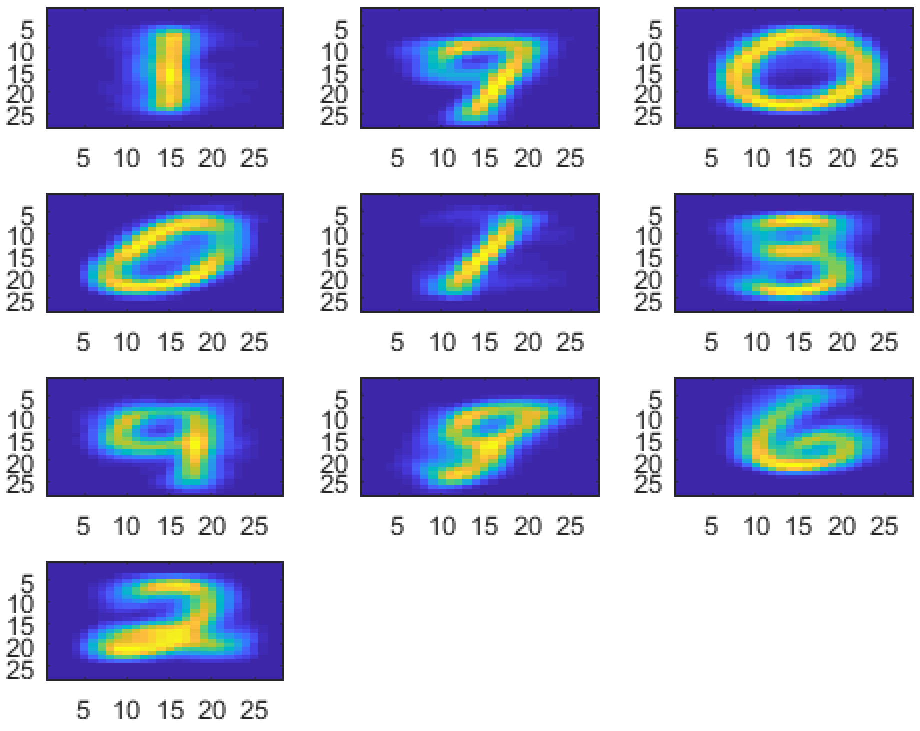


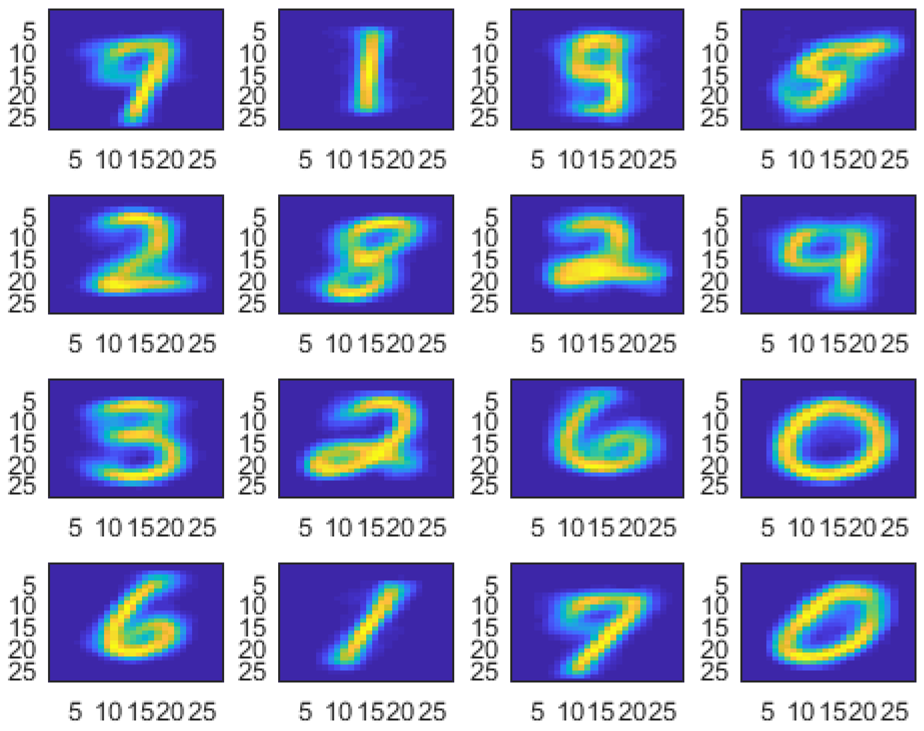
TestMNIST

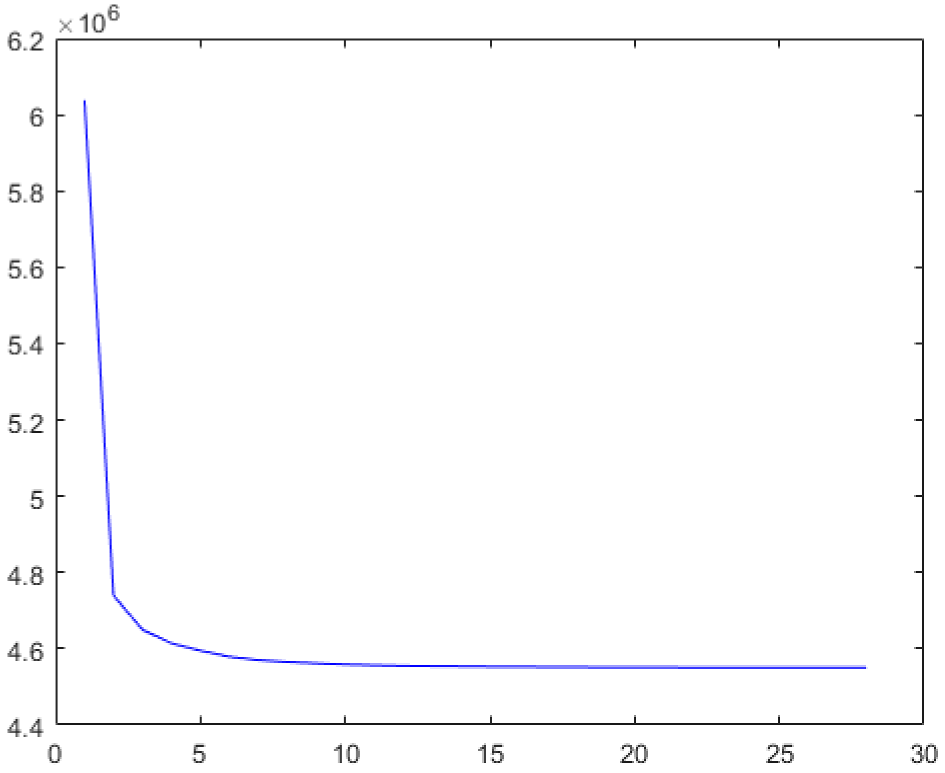
```
type('TestMNIST.m')

fprintf('\n---------------------------------------------\n');
TestMNIST


load mnistData.mat
%{
analyzes MNIST data and groups with varyiang # clusters

Alyssa Rose HW10 04-15-2018
%}

%% creates struct for nClus = 10
ClusInfo1 = struct('nClus',10,'maxIter',100)
ClusInfo10 = KmeansCore(imagesData,ClusInfo1);

%% plotting nClus = 10
figure(2)
for h = 1:10
    subplot(4,3,h)
    newImg = reshape(ClusInfo10.centerVecs(:,h),[28,28])';
    imagesc(newImg);
end
```

```
%% creates struct for nClus = 16
ClusInfo2 = struct('nClus',16,'maxIter',300)
ClusInfo16 = KmeansCore(imagesData,ClusInfo2);

%% plotting nClus = 16
figure(3)
for h = 1:16
    subplot(4,4,h)
    newImg = reshape(ClusInfo16.centerVecs(:,h),[28,28])';
    imagesc(newImg);
end

% plots line of sumDist values
figure(4)
plot(ClusInfo16.sumDist,'b-')

%% Discussion 1
%{
The plot shows how the 'guess' of sumDist improves on each
iteration, ie the code is reassigning the data pts to their
clus center more accurately on each run. As it improves,the value
for sumDist decreases on each run.
%}

%% Discussion 2
%{
Choosing 16 clusters allows for better grouping of the data. If
the number of clusters increases, it is easier to create smaller
groups that are more similar to each other instead of general
 grouping.
%}

%% Discussion 3
%{
Handwritten digits could be identified by clustering them based
on about 5-10 clusters for each number (to account for large
 discrepancies
in how people write numbers). Therefore it would be easier to extract
 and
identify the numbers that are similar and create databases for
 identifying
the most common ways people write #'s and by grouping them in their
respective clusters.
%}

---------------------------------------------

ClusInfo1 =

  struct with fields:

      nClus: 10
    maxIter: 100
```

```
ClusInfo2 =

  struct with fields:

      nClus: 16
    maxIter: 300
```




5
10
15
20
25
5 10 15 20 25


×10^6
6.2
6
5.8
5.6
5.4
5.2
5
4.8
4.6
4.4
0
5
10
15
20
25
30

Published with MATLAB® R2017b