Table of Contents


```
clear all;
close all;
```

In this script, you need to implement three functions as part of the k-means algorithm.

These steps will be repeated until the algorithm converges:

```
 % 1. initialize_centroids
 % This function sets the initial values of the centroids

 % 2. assign_vector_to_centroid
 % This goes through the collection of all vectors and assigns them
to
 % centroid based on norm/distance

 % 3. update_centroids
 % This function updates the location of the centroids based on the
collection
 % of vectors (handwritten digits) that have been assigned to that
centroid.
```

Initialize Data Set

These next lines of code read in two sets of MNIST digits that will be used for training and testing respectively.

```
% training set (1500 images)
```

```
train=csvread('mnist_train_1500.csv');
trainsetlabels = train(:,785);
train=train(:,1:784);
train(:,785)=zeros(1500,1);

% testing set (200 images with 11 outliers)
test=csvread('mnist_test_200_woutliers.csv');
% store the correct test labels
correctlabels = test(:,785);
test=test(:,1:784);

% now, zero out the labels in "test" so that you can use this to
 assign
% your own predictions and evaluate against "correctlabels"
% in the 'cs1_mnist_evaluate_test_set.m' script
test(:,785)=zeros(200,1);
```

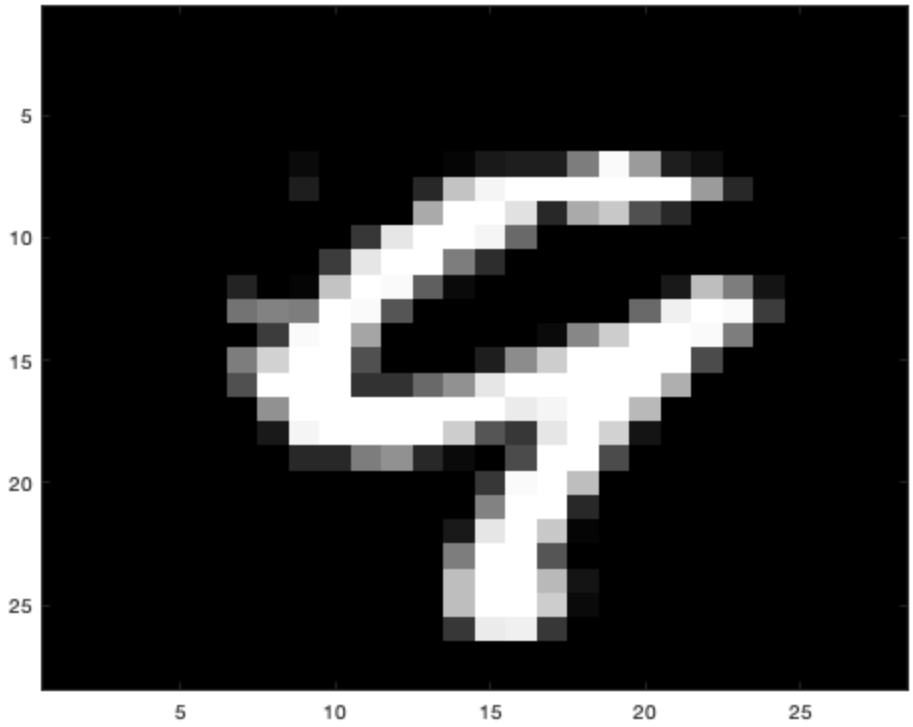
After initializing, you will have the following variables in your workspace:

1. train (a 1500 x 785 array, containins the 1500 training images) 2. test (a 200 x 785 array, containing the 200 testing images) 3. correctlabels (a 200 x 1 array containing the correct labels (numerical meaning) of the 200 test images

To visualize an image, you need to reshape it from a 784 dimensional array into a 28 x 28 array.

to do this, you need to use the reshape command, along with the transpose operation. For example, the following lines plot the first test image

```
figure;
colormap('gray'); % this tells MATLAB to depict the image in grayscale
testimage = reshape(test(1,[1:784]), [28 28]);
% we are reshaping the first row of 'test', columns 1-784 (since the
 785th
% column is going to be used for storing the centroid assignment.
imagesc(testimage'); % this command plots an array as an image.  Type
 'help imagesc' to learn more.
```



After importing, the array train consists of 1500 rows and 785 columns.

Each row corresponds to a different handwritten digit (28 x 28 = 784) plus the last column, which is used to index that row (i.e., label which cluster it belongs to. Initially, this last column is set to all zeros, since there are no clusters yet established.

This next section of code calls the three functions you are asked to specify

```
k = 50; % set k
max_iter = 30; % set the number of iterations of the algorithm
```

The next line initializes the centroids. Look at the initialize_centroids()

function, which is specified further down this file.

```
centroids=initialize_centroids(train,k);
```

Initialize an array that will store k-means cost at each iteration

```
cost_iteration = zeros(max_iter, 1);
```

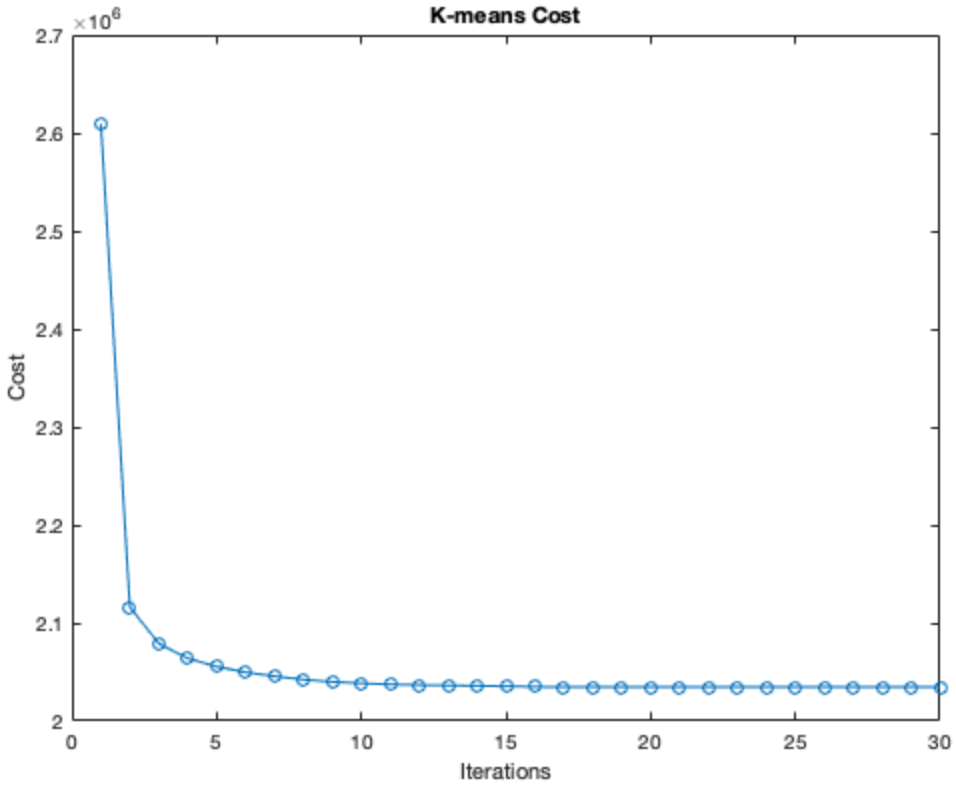
This for-loop enacts the k-means algorithm

```
for iter=1:max_iter

    % find distances and assign centroid to each row vector in train
    distances = zeros(k,1);
    for index=1:length(train)

        % method to assign a vector to a centroid
        % parameters are (row from train, set of centroids)
        [i, minDist] = assign_vector_to_centroid(train(index,:),
 centroids);

        % use return values from previous function to assign the
 correct
        % centroid to this image
        train(index,785) = i;

        % update cost function
        cost_iteration(iter) =  cost_iteration(iter) + minDist;
    end

    % update centroids (each new centroid is the average of the
 vectors in
    % its group)
    centroids = update_Centroids(train,k);
end
```

This section of code plots the k-means cost as a function of the number

of iterations

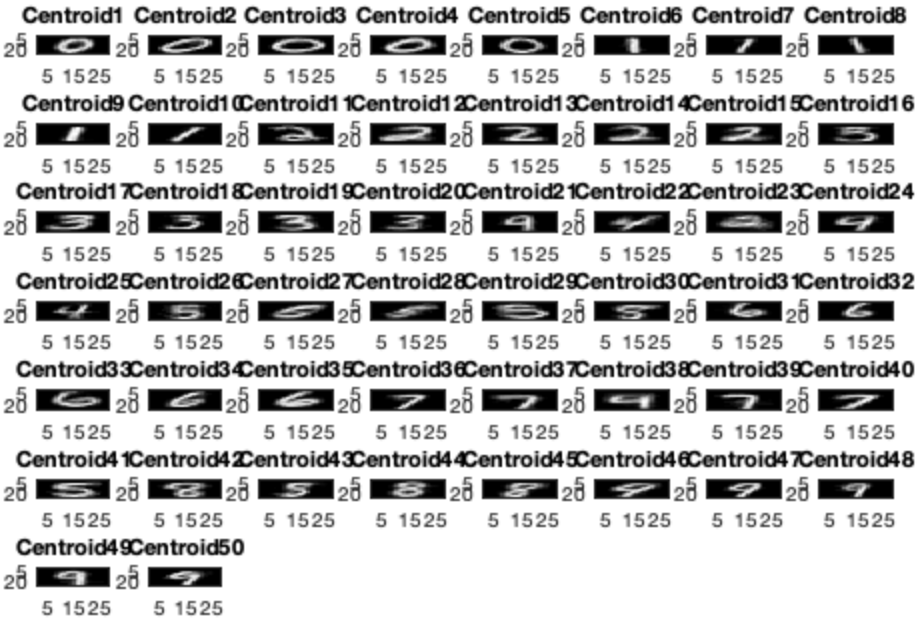
```
figure;
plot(1:max_iter,cost_iteration,'-
o','MarkerIndices',1:length(cost_iteration));
xlabel('Iterations');
ylabel('Cost');
title('K-means Cost');
```



This next section of code will make a plot of all of the centroids

Again, use help functionname to learn about the different functions that are being used here.

```
figure;
colormap('gray');

plotsize = ceil(sqrt(k));

for ind=1:k

    centr=centroids(ind,[1:784]);
    subplot(plotsize,plotsize,ind);

    imagesc(reshape(centr,[28 28])');
    title(strcat('Centroid ',num2str(ind)))

end
```



Function to initialize the centroids

This function randomly chooses k vectors from our training set and uses them to be our initial centroids There are other ways you might initialize centroids. **Feel free to experiment.** Note that this function takes two inputs and emits one output (y).

```
function y = initialize_centroids(data,num_centroids)
zero_indices = [4, 5, 8, 33, 53];
zero_labels = [0, 0, 0, 0, 0];

one_indices = [12, 36, 76, 293, 442];
one_labels = [1, 1, 1, 1 ,1];

two_indices = [27, 31, 40, 48, 71];
two_labels = [2, 2, 2, 2, 2];

three_indices = [7, 10, 75, 96, 106];
three_labels = [3, 3, 3, 3, 3];

four_indices = [14, 81, 83, 169, 243];
four_labels = [4, 4, 4, 4, 4];

five_indices = [3, 50, 87, 186, 235];
five_labels = [5, 5, 5, 5, 5];

six_indices = [19, 88, 93, 105, 119];
```

```
six_labels = [6, 6, 6, 6, 6];

seven_indices = [9, 39, 59, 65, 125];
seven_labels = [7, 7, 7, 7, 7];

eight_indices = [21, 41, 44, 162, 193];
eight_labels = [8, 8, 8, 8, 8];

nine_indices = [45, 49, 69, 77, 122];
nine_labels = [9, 9, 9, 9, 9];

centroid_indices = [zero_indices, one_indices, two_indices,
 three_indices, four_indices, five_indices, six_indices,
 seven_indices, eight_indices, nine_indices];
centroid_labels = [zero_labels, one_labels, two_labels, three_labels,
 four_labels, five_labels, six_labels, seven_labels, eight_labels,
 nine_labels];

centroids=data(centroid_indices(1:num_centroids),:);
centroids(:, 785) = centroid_labels;

y=centroids;

end
```

Function to pick the Closest Centroid using norm/distance

This function takes two arguments, a vector and a set of centroids It returns the index of the assigned centroid and the distance between the vector and the assigned centroid.

```
function [index, vec_distance] =
 assign_vector_to_centroid(data,centroids)

    k = size(centroids, 1);
    distances = zeros(k,1);
    values = 1:k;

    for cenIn=1:k
        distances(cenIn)= norm(data(1:length(data)-1) -
 centroids(cenIn,1:(size(centroids,2)-1)));
    end


    % return index as the centroid number
     index = values(distances == min(distances));

     % return vec_distance as the minimum distance
     vec_distance = min(distances);

end
```

Function to compute new centroids using the mean of the vectors currently assigned to the centroid.

This function takes the set of training images and the value of k. It returns a new set of centroids based on the current assignment of the training images.

```
function new_centroids=update_Centroids(data,K)

% create new centroid matrix
    % set all centroids to zero
    % centroids have k rows and colums are the same size of the
 columns in
    % data
    % size(data,2)) = take the number of columns in data

    centroids = zeros(K,size(data,2));


    for centroid=1:K
        centroids(centroid, 1:size(centroids,2)-1) =
 mean(data( (data(:,785) == centroid) , 1:size(centroids,2)-1 ));
    end

    new_centroids = centroids;

end
```

Published with MATLAB® R2019a