70-511: Statistical Programming
Programming Assignment 2 - k-Means Clustering
Step by Step Directions

 # Initialization
1. Print header info to screen

2. Get input/output file names and number of clusters

3. Read file:
-use open() and a list comprehension to strip all lines of ending char (using rstrip method) and convert to floats

4. Create variables to store centroids, clusters, and point assignments. Initially, pick one point (centroid) per cluster:
-create and initialize a variable to store centroids for each cluster: a mapping (dict) from range(k) to data[0:k]
-create and initialize another variable to store all points for each cluster: a mapping (dict) of range(k) to k empty lists
-use zip when creating the dict
-create and initialize a dict mapping points to clusters
-create a variable to store old point assignments (from previous iteration)

 # Algorithm
5. Repeat the following:
a) Save current point assignment into old point assignment variable (create a new dict from current assignment variable)
b) Place each point in the closest cluster (you should make a function that does this)
c) Update the locations of centroids of the k clusters (make a function for this also)
d) Reinitialize the clusters variable to empty lists

 # Output
6. Print the point assignments

DETAILED HINTS

 # Initialization

1. Print header info to screen

Use print function.

2. Get input/output file names and number of clusters

Use raw_input. For number of clusters (k), make sure to use int() to convert to an integer.

3. Read file:

-use open() and a list comprehension to strip all lines of ending char (using rstrip method) and convert to floats

I basically showed this in the video:

```
data = [float(x.rstrip()) for x in open(input_file)]
```

4. Create variables to store centroids, clusters, and point assignments. Initially, pick one point (centroid) per cluster:

-create and initialize a variable to store centroids for each cluster: a mapping (dict) from range(k) to data[0:k]

```
centroids = dict(zip(range(k),data[0:k])
```

-create and initialize another variable to store all points for each cluster: a mapping (dict) of range(k) to k empty lists -use zip when creating the dict

```
clusters = dict(zip(range(k),[[] for i in range(k)]))
```

-create and initialize a dict mapping points to clusters

First, think about how points are represented - as numbers in a list, with each number having an index value. So you can do a mapping between each of the index values (0..k-1) and the cluster to which the point having this index value belongs. Initially, the points won't belong to any cluster, so you can just map to dummy values. Later, you will update these with appropriate values (after each iteration).

-create a variable to store old point assignments (from previous iteration)

Since there is no previous assignment on first iteration, just assign an empty dict to an old_point_assignments variable

 # Algorithm

5. Repeat the following:

a) Save current point assignment into old point assignment variable (create a new dict from current assignment variable)

This should create a copy of the dict holding the point assignments into the old_point_assignments variable

b) Place each point in the closest cluster (you should make a function that does this)

Make a function called assign_to_clusters that takes as input: data, clusters, centroids, and point_assignments. The function should go through each point and index of that point in data (use enumarate()) and find the closest centroid. Then add that point to the list of points for that clusters in the clusters variable. Also, do:

```
point_assignments[j] = closest_index
```

c) Update the locations of centroids of the k clusters (make a function for this also)

Make a function that takes as input: data, clusters, and centroids. It should go through each list contained in clusters variable and recompute the centroid by averaging over all the points in that list (use sum() and len() functions). After computing, update the corresponding centroids value.

d) Reinitialize the clusters variable to empty lists

```
clusters = dict(zip(range(k),[[] for i in range(k)]))
```

 # Output

6. Print the point assignments

Go through all values in point_assignments and print them.