K-Means and EM - HW2

Karthik Ravindra Rao and Praneet Kalluri

February 17, 2017

1 Algorithms

1.1 K-Means

Following functions are written for the K-Means Algorithm

getrandomcentrioids(*listpoints*, *k*) : Selects k random centroids from the available list of data points

stopcheck(*oldcentroids*, *updatedcentroids*, *iterations*) : Termination condition for k-means. Algorithm terminates when centroid in n-1th step is same as the nth step centroid

pointlabel(*listpoints*, *centroids*) : Assigns data points to the closest centroid

makearray(*listvariable*) : Helper function to convert from dictionary to list

calcentroid(*listvar*) : Calculates the centroid for the list of datapoints

getnewcentroid(*object*) : Recalculates the centroid for a cluster

printclusters() : main function to call the functions

main(*listpoints*, *centroids*) : Prints and plots the clusters in the desired manner

The Output obtained for K-Means is shown in Figure-1. The centroids obtained are mentioned below.

```
Centroids for K-means:
{'x': -1.0393701035692309, 'y': -1.2380392655538461}
{'x': 5.1729039155918359, 'y': 4.1359136770612261}
{'x': 0.49711036355555549, 'y': 1.266963754611111}
```

1.2 EM for clustering using a GMM

Following functions are written for the EM Algorithm

stopcheck(*clusters*, *oldclusters*) : Terminating condition for the algorithm

weightedmean(*listpoints*) : Calculates the weighted mean of each cluster by taking into consideration soft-membership of each point

weightedcovar(*listpoints*, *riclist*) : Calculates the co-variance matrix using the array of soft-membership of each point with respect to the cluster

newclusters(*listpoints*) : Updates standard deviation, mean and amplitude of each cluster after each iteration

covariance(*listpoints*) : Calculates the initial co-variance matrix of each cluster without weight

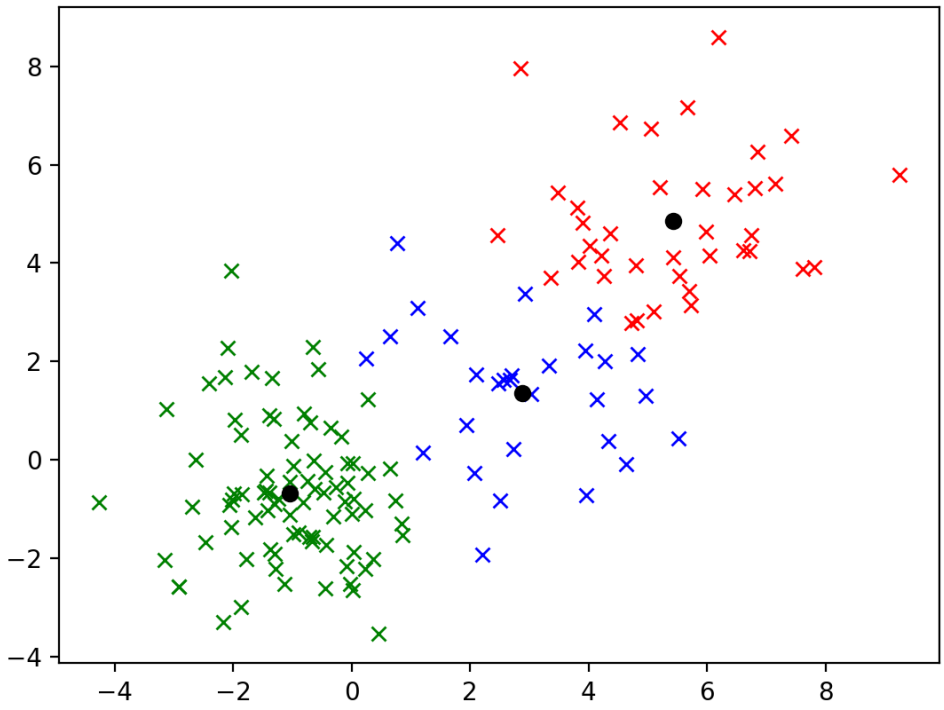

Figure 1: Cluster Obtained for K-Means

meancal(*listpoints*) : Calculates the initial mean of each cluster without weight

ricinitialization(*listpoints*) : Randomly initializes soft membership of each point with respect to each cluster to begin the algorithm

splitcontent(*listpoints*) : Helper method to calculate the initial parameters of the clusters

ric(*clusters*, *listpoints*) : Calculates the soft membership of each point with respect to clusters

main(*listpoints*, *centroids*) : Main function to call other functions and coordinate the data flow

The Outputs obtained for EM using GMM are mentioned below.

```
Mean:  [-0.94254700013885129, -0.7719695409795132] Covariance:  [[ 1.09461435 -0.04079337]
 [-0.04079337  1.68404941]] Amplitude:  0.521879786894
Mean:  [4.5494457059982665, 3.2285096745683166] Covariance:  [[ 3.39797759  2.57105296]
 [ 2.57105296  4.99869074]] Amplitude:  0.403147739541
Mean:  [0.22042129662400683, 2.9338991440633113] Covariance:  [[ 6.49188751  5.11468536]
 [ 5.11468536  6.98699934]] Amplitude:  0.0749724735652
```

2 Software Familiarization

2.1 Scikit-learn

K-Means and GMM can be implemented using Scikit as follows

Figure 2: Original Image 96,615

```
K-Means
-------
X = np.array(data_points)
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
kmeans.labels_

kmeans.predict([[0, 0], [4, 4]])

kmeans.cluster_centers_

------------------------
EM using GMM
-----------

GaussianMixture(n_components=1, covariance_type='full', tol=0.001, reg_covar=1e-06,
max_iter=100, n_init=1, init_params='kmeans', weights_init=None,
means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0,
verbose_interval=10)[source]
```

2.2 Improvements for Implemented Algorithm

- The Algorithm written for the purpose of this assignment has a restriction of creating 3 clusters. This should be made flexible to increased to desirable number of clusters
- A predict function needs to be written where test data can be predicted without manual-tracing which is currently employed
- This algorithm is restricted to data points in 2 dimensions. This should be expanded over to multiple dimensions.

3 Applications

3.1 K-Means for Color Quantization

K-Means can be used to perform a pixel-wise Vector Quantization (VQ) of an image, reducing the number of colors required to show the image drastically, while preserving the overall appearance quality. Figure 2, 3 and 4 obtained from Scikit Learn clearly elaborate this application.

3.2 Response Theory Models

In psychometrics [NCME] *a field of study concerned with the theory and technique of psychological measurement*, EM used for item parameters and latent abilities of item response theory models [NCME] *a paradigm for the design, analysis, and scoring of tests, questionnaires, and similar instruments measuring abilities, attitudes, or other variables.* It is the preferred method for developing scales in the in the Graduate Record Examination (GRE) and Graduate Management Admission Test (GMAT).

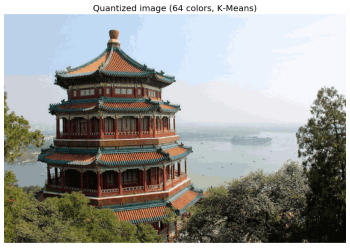

Figure 3: Quantized Image K-Means 64 colors

Figure 4: Quantized Image Random 64 Colors

4 Responsibility division between Teammates

We, Karthik and Praneet have brain-stormed and implemented the assignment with equal contribution for every line of code.

5 References

http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html
http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
http://stanford.edu/~cpiech/cs221/handouts/kmeans.html
http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html#sphx-glr-auto-examples-cluster-plot-color-quantization-py https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm#Applications https://en.wikipedia.org/wiki/Item_response_theory http://www.ncme.org/ncme/NCME/Resource_Center/Glossary/NCME/Resource_Center/Glossary1.aspx?hkey=4bb87415-44dc-4088-9ed9-e8515326a061#anchorP