Journal of Computations & Modelling, vol.2, no.4, 2012, 1-23
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2012

Development of a hybrid K-means-expectation maximization clustering algorithm

Adigun Abimbola Adebisi[1], Omidiora Elijah Olusayo[2], Olabiyisi Stephen Olatunde[3], Adetunji Abigael 'Bola[4] and Adedeji Olayinka Titilayo[5]

Abstract

Data clustering is a common technique for statistical data analysis in including machine learning, data mining, pattern recognition, image analysis and bioinformatics. The computational task of classifying the data set into *k* clusters is often referred to as *k*-clustering. K-Means and Expectation-Maximization algorithms are part of the commonly employed methods in clustering of data in relational databases. Results findings in some related work revealed that both algorithms have been found to be characterized with shortcomings. K-means was established not to guarantee convergence while Expectation-Maximization's quick and premature convergence doesn't assure optimality of results. As such, both algorithms are not efficient and effective enough; hence, there arises a need for a proposed algorithm that could both guarantee convergence and optimality of

---

[1] Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria, email: adigunademidun@yahoo.co.uk
[2] E-mail: eoomidiora@lautech.edu.ng
[3] E-mail: solabiyisi@lautech.edu.ng
[4] E-mail: abadetunji@yahoo.com,
[5] E-mail: adedeji_oluyinka@yahoo.com

Article Info: *Received* : June 12, 2012. *Revised* : August 12, 2012
Published online : December 30, 2012

results in discovering knowledge in very large database. A hybrid of K-means and Expectation-Maximization (KEM) was developed for this purpose.
The proposed hybrid approach was evaluated with both K-means and Expectation-Maximization algorithms using time (rate of convergence), space complexities and the number of iterations. The computational results showed that the developed algorithm guarantees both optimality and convergence over K-Means and Expectation Maximization clustering algorithms.

Keywords: K-Means, Expectation Maximization, hybrid K-Means-Expectation Maximization and, Data Clustering

1 Introduction

Clustering algorithms partition a dataset into several groups such that points in the same group are close (similar) to each other and points across groups are far (different) from each other [2].

The commonly used clustering techniques include AGNES (Agglomerative Nesting), DIANA (Divisive Analysis), CLARA (Clustering Large Applications), PAM (Partitioning Around Medoids), BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), K-Means and EM (Expectation-Maximization) algorithms [1].

The k-means algorithm [3] is a very popular algorithm for data clustering because of its simplicity. Originally developed for and applied to the task of vector quantization, k-means has been used in a wide assortment of applications. It has been proven to be a good approach to classify data.
However, K-means does not assure the best representation or fit for the data in the model because it uses distances from the centers of clusters to determine which sample belongs to which class. Also, it has been shown that, with K-Means, there

is no guarantee for optimal clustering, since the convergence depends on the initial number of clusters selected. In addition, K-means is not considered as the best choice for clustering due to its time performance and requirements. K-means typically requires that clusters be spherical, that the data be free of noise and that its operation be properly initialized. This makes it inefficient for major industrial clustering problems [6].

EM is a model based approach to solving clustering problems. It is an iterative algorithm that is used in problems where data is incomplete or considered incomplete. Unlike distance based or hard membership algorithms (such as K-Means), EM is known to be an appropriate optimization algorithm for constructing proper statistical models of the data. EM is widely used in applications such as computer vision, speech processing and pattern recognition. EM aims at finding clusters such that maximum likelihood of each clusters parameters is obtained.

In EM, each observation belongs to each cluster with a certain probability. EM clusters data, in a manner different than K-means. EM starts with an initial estimate for the missing variables and iterates to find the maximum likelihood (ML) for these variables. Maximum likelihood methods estimate the parameters by values that maximize the sample's probability for an event.

EM is typically used with mixture models. Unlike in K-means, in clustering via EM, the number of clusters that are desired are predetermined. It is initialized with values for unknown (hidden) variables. Since EM uses maximum likelihood it most likely converges to local maxima, around the initial values. Hence selection of initial values is critical for EM. However, the EM algorithm works well on clustering data when the number of clusters is known [4].

This approach combines the two clustering algorithms above to come up with a hybrid clustering algorithm for better clustering. The initial clusters centers are found using K-means algorithm. These give centers that are widely spread within the data. EM takes these centers as its initial variables and iterates to find the local maxima. Hence, clusters that are well distributed using K-means and clusters that are compact using EM are obtained. Educational database, which is a good benchmark for classification problems, is used to test the algorithms.

2 Research methodology

2.1 K-Means

K-means which uses weighted average was used instead of the ordinary mean, to get new cluster centers. Let $\{x_1,...,x_p\}$ be a set of *P* real numbers. The number of iteration is given as *r*. The weighted average (WA) is given by [5].

$$\mu^{(r+1)} = \sum_{p=1,p} w_p^{(r)} x_p, \qquad r = 0,1,... \tag{1}$$

An initial mean is taken and a Gaussian is centered over the mean and weight, w_p is obtained for x_P. Feature vectors are assigned to each cluster point, empty or small sets are eliminated. Cluster centers are replaced with weighted averages and feature vectors are reassigned. This process is repeated until convergence.

K-Means performance function:

The algorithm partitions the data set into *K* clusters, $S = (S_1,\ldots\ldots,S_K)$, by putting each data point into the cluster represented by the center nearest to the data point. K-Means algorithm finds a local optimal set of centers that minimizes the total within cluster variance, which is K-means performance function [5]:

$$\mathrm{Perf}_{\mathrm{KM}}(\mathrm{X},\mathrm{M}) = \sum_{\mathrm{k}=1}^{\mathrm{K}} \sum_{\mathrm{x}\in \mathrm{S}_\mathrm{i}} \|\mathrm{x} - \mathrm{m}_\mathrm{k}\|^2 \tag{2}$$

where the *k*th center, m_k, is the centroid of the *k*th partition. The double summation in (2) can instead be expressed as a single summation over all data points, adding only the distance to the nearest center expressed by the *MIN* function below [5]:

$$\mathrm{Perf}_{\mathrm{KM}}(\mathrm{X},\mathrm{M}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{MIN}\{\|\mathrm{x}_\mathrm{i} - \mathrm{m}_\mathrm{l}\|^2 \;|\mathrm{l} = 1,...,\mathrm{K}\} \tag{3}$$

The K-Means algorithm starts with an initial set of centers and then iterates through the following steps:

1. For each data item, find the closest m_k and assign the data item to the k^{th} cluster. (The current m_k's are not updated until the next phase).
2. Recalculate all the centers. The kth center becomes the centroid of the kth cluster.
3. Iterate through 1 and 2 until the clusters no longer change significantly

After each phase, the performance value never increases and the algorithm converges to a local optimum.

The algorithm should reach a stable partition in a finite number of steps for finite datasets. The cost per iteration is $O(K \cdot dim \cdot N)$.

2.2 Expectation Maximization (EM) clustering algorithm

The EM algorithm is an unsupervised clustering method, that is, it doesn't require a training phase, based on mixture models. It follows an iterative approach (sub-optimal), which tries to find the parameters of the probability distribution that have the maximum likelihood of its attributes.

The algorithm's inputs are the data set (x), the total number of clusters (M), the accepted error to converge (e) and the maximum number of iterations. First, the algorithm goes through the initialization stage, where the parameter vectors are initialized and then iterates through both the Expectation and Maximization stages until convergence. For each iteration, first is executed the E-Step (E-xpectation), that estimates the probability of each point belonging to each cluster, followed by the M-step (M-aximization), that re-estimates the parameter vector of the probability distribution of each class. The algorithm terminates when the distribution parameters converge or reach the maximum number of iterations.

Initialization

Each class j, of M classes (or clusters), is constituted by a parameter vector (θ), composed by the mean (μj) and by the covariance matrix (*P*$_j$), which represent the features of the Gaussian probability distribution (Normal) used to characterize the observed and unobserved entities of the data set x.

$$\theta(t) = \mu_j(t)\ ,\ P_j(t)\ ,\ j = 1...M \tag{4}$$

On the initial instant (t = 0), the implementation can generate randomly the initial values of mean (μ_j) and of covariance matrix (P_j). The EM algorithm aims to approximate the parameter vector (θ) of the real distribution. Another alternative offered by MCLUST is to initialize EM with the clusters obtained by a hierarchical clustering technique.

E-Step

This step is responsible to estimate the probability of each element belong to each cluster ($P(C_j \mid x_k)$). Each element is composed by an attribute vector (x_k). The relevance degree of the points of each cluster is given by the likelihood of each element attribute in comparison with the attributes of the other elements of cluster C_j [4].

$$P(C_j|x) = \frac{\left|\sum_j(t)\right|^{-1/2} \exp^{n_j} P_j(t)}{\sum_{k=1}^{M} \left|\sum_j(t)\right|^{-1/2} \exp^{n_j} P_k(t)} \tag{5}$$

M-Step

This step is responsible for the estimation of the parameters of the probability distribution of each class for the next step.

First is computed the mean (μj) of classes j obtained through the mean of all points in function of the relevance degree of each point.

$$\mu_j(t+1) = \frac{\sum_{k=1}^{N} P(C_j|x_k)x_k}{\sum_{k=1}^{N} P(C_j|x_k)} \tag{6}$$

Secondly, the covariance matrix for the next iteration is computed by applying the Bayes Theorem which implies that *P*(*A* | *B*) = *P*(*B* | *A*) * *P*(*A*)*P*(*B*), based on the conditional probabilities of the class occurrence.

$$\sum_j (t+1) = \frac{\sum_{k=1}^{N} P(C_j | x_k)(x_k - \mu_j(t))(x_k - \mu_j(t))}{\sum_{k=1}^{N} P(C_j | x_k)} \tag{7}$$

Thirdly, the probability of occurrence of each class is computed through the mean of probabilities (C_j) in function of the relevance degree of each point from the class.

$$P_j(t+1) = \frac{1}{N}\sum_{k=1}^{N} P(C_j | x_k) \tag{8}$$

The attributes represent the parameter vector, θ that characterize the probability distribution of each class that will be used in the next algorithm iteration [4].

Steps for the implementation of EM

The steps for the implementation of EM are as follows;

A guess has to be initialized for the mean and standard deviation.

Initialization step: initialize the hypothesis $\theta^0 = \left(\mu_1^0, \mu_2^0, \ldots, \mu_k^0\right)$

$$\theta_k^0 = \mu_k^0 \tag{9}$$

Where K is the current number of Gaussians, σ is the standard deviation, θ0 is the estimate at 0th iteration, μ is the mean.

• Expectation step: estimate the expected values of the hidden variables z_{ij} (mean and standard deviation) using the current hypothesis $\theta_t = \left(\mu_1^t, \mu_2^t, \ldots, \mu_K^t\right)$ [4]

$$E(z_{ik}) = \frac{\exp\left[-\frac{(x_i - \mu_k^t)^2}{2\sigma^2}\right]}{\sum_{j=1}^{K} \exp\left[-\frac{(x_i - \mu_k^t)^2}{2\sigma^2}\right]} \tag{10}$$

Where *t* is the number of the iteration, *E(Z$_{ik}$)* is the expected value for the hidden variables (namely mean and standard deviation), k is the dimension, σ is the standard deviation.

• Maximization step: provides a new estimate of the parameters.

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{n} E(z_{ik})x_i}{\sum_{i=1}^{n} E(z_{ik})} \tag{11}$$

Convergence step: if $\left\|\theta^{t+1} - \theta^{t}\right\| < e$, stop (finish iteration); otherwise, go back to the e-step [4].

Convergence Test

After each iteration is performed, a convergence test which verifies if the difference of the attributes vector of an iteration to the previous iteration is smaller than an acceptable error tolerance, given by parame ter. Some implementations use the difference between the averages of class distribution as the convergence criterion.

```
If (|| θ (t + 1) − θ (t) || < Q)
  Stop
else
  call E-Step
end;      (where Q is the acceptable error tolerance)
```

The algorithm has the property of, at each step, estimating a new attribute vector that has the maximum local likelihood, not necessarily the global, which reduces its complexity. However, depending on the dispersion of the data and on its volume, the algorithm can also be terminated due to the maximum number of iterations defined.

Performance function of Expectation Maximization

Unlike K-Means in which only the centers are to be estimated, the EM algorithm estimates the centers, the co-variance matrices, $\sum_K$ and the mixing probabilities, *p(mk)*.

The performance function of the EM algorithm is [56]

$$\mathrm{Perf}_{EM}(X, M, \Sigma, p) = -\log\left\{\prod_{x\in S}\left[\sum_{k=1}^{K} p_k \cdot \frac{1}{\sqrt{(2\pi)^D \det(\Sigma_K)}} \cdot \mathrm{EXP}(-(x-m_k)\Sigma_k^{-1}(x-m_k)^T)\right]\right\} \quad (12)$$

where the vector $p = (p_1, p_2, ..., p_K)$ is the mixing probability. *EM* algorithm is a recursive algorithm with the following two steps:

E-Step: Estimating "the percentage of *x* belonging to the *k*th cluster"

$$p(m_k|x) = \frac{p(x|m_k)\cdot p(m_k)}{\sum_{x\in S} p(x|m_k)\cdot p(m_k)} \quad (13)$$

where *p(x|m)* is the prior probability with Gaussian distribution, and *p(m$_k$)* is the mixing probability.

$$p(x|m_k) = \frac{1}{\sqrt{(2\pi)^D \det(\Sigma_K)}} \cdot \mathrm{EXP}(-(x-m_k)\Sigma_k^{-1}(x-m_k)^T) \quad (14)$$

M-Step: With the fuzzy membership function from the E-Step, find the new center locations, new co-variance matrices and new mixing probabilities that maximize the performance function [5].

$$m_k = \frac{\sum_{x\in S} p(m_k|x)\cdot x}{\sum_{x\in S} p(m_k|x)},$$

$$.\ \Sigma_k = \frac{\sum_{x\in S} p(m_k|x)\cdot (x-m_k)^T(x-m_k)}{\sum_{x\in S} p(m_k|x)}, \quad (15)$$

$$p(m_k) = \frac{1}{|S|}\sum_{x\in S} p(m_k|x)$$

2.3 KEM (K-means Expectation Maximization) algorithm

The KEM algorithm, like the EM algorithm is divided into two (2) stages: the initialization stage and the iterative stage. In the initialization stage, the weighted average variation of the K-means algorithm is used to classify the data into the number of clusters desired, since this method works better in the choice of the initial centroids than using the simple average K-means.

Pseudocode for the KEM algorithm is given below:

```
• Choose the number of cluster, k.
• Randomly generate k clusters and determine the cluster centers,
• Assign each point to the nearest cluster center.
• Recompute the new cluster centers using weighted averages.
• Generate an initial model M’ = $(C_1, C_2, ..., C_k)$
  repeat
• // (re-) assign points to clusters
  Compute $P(x|C_i)$, P(x) and $P(C_i|x)$ for each object x from D
  and each cluster (= Gaussian) $C_i$
• // (re-) compute the models
  Compute a new model M = $\{C_1, C_2, ..., C_k\}$ by re-computing $W_i$,
  $\mu C$ and $\sum C$ for each Cluster $C_i$
• Replace M’ by M
  until |E(M) – E(M’)| < ε
• return M
```

First, a large number M, of uniformly distributed random cluster point vectors for the cluster centers are selected. Then any cluster point vectors that are too close to other cluster point vectors are eliminated and M, is reduced accordingly until the clusters produced equals to K, the number of desired clusters. This is done by computing the distances between all the clusters, and eliminating the clusters that their distances are less than e (a value that is selected experimentally).

Assigning each of the feature vectors to the nearest random cluster point vector, is the next step, and it can be achieved by computing the distance between each feature vector and all other cluster point vectors. Then the feature vector will be assigned to the cluster point vector such that the distance between them is the shortest.

Also, each time an assignment happens the number of feature vectors assigned to that cluster point vector will be incremented. All cluster point vectors that are the centers of empty clusters, or have fewer vectors than selected *p* vectors, are eliminated and K is reduced.

Each cluster is then given a new prototype with the current K, and that would be the weighted fuzzy average (WA) of each class, by initially taking the sample mean $\mu^{(0)}$ and variance σ^2 to start the process. Then a Gaussian is centered over the current approximate WA $\mu^{(r)}$ and iterated as follows:

$$w_p^{(r)} = \frac{\exp\left[-(x_p - \mu^{(r)})/2\sigma^2\right]}{\sum_{m=1.P} \exp\left[-(x_m - \mu^{(r)})/2\sigma^2\right]} \tag{16}$$

$$\mu^{(r+1)} = \sum_{p=1,P} w_p^{(r)} x_p \quad , \quad r = 0,1,2,... \tag{17}$$

In this step, the WA for each cluster is calculated to be the new class prototype. The next step is to compute the Maximum Likelihood estimation for the current K clusters using EM as described above and get new centers for each of the clusters. Then each of the feature vectors would be assigned to the class with the nearest weighted fuzzy average. After that, every two clusters whose prototypes are closest are merged, the average of the two prototypes will be used as the new prototype (cluster point) and K will be reduced accordingly. Next, empty clusters are eliminated, and the number of clusters K is reduced. This process is repeated until we reach the desired number of clusters.

2.3.1 Derivation of KEM Performance Factor

Let

$$\mathrm{Perf}_{EM}(X,M) = -\log\left(\prod_{t=1}^{N}\sum_{l=1}^{K} p_l \cdot \frac{1}{\sqrt{(2\pi)^D}} \cdot \mathrm{EXP}(-\|x-m\|^2)\right) \tag{18}$$

be a linear mixing of *K* centroidal functions. *EM* algorithm is a recursive algorithm with the following two steps:

2.3.1.1 E-Step:

$$p(m_l|x_i) = \frac{p(x_i|m_l)\cdot p(m_l)}{\sum_{i=1,\dots,N} p(x_i|m_l)\cdot p(m_l)} \tag{19}$$

where *p (x|m)* is the prior probability with Gaussian distribution, *p(m$_l$)* is the mixing probability.

2.3.1.2 M-Step:

$$m_l = \frac{\sum_{i=1}^{N} p(m_l|x_i)\cdot x_i}{\sum_{i=1}^{N} p(m_l|x_i)} \tag{20}$$

$$p(m_l) = \frac{1}{N}\sum_{i=1}^{N} p(m_l|x_i) \tag{21}$$

where *N* is the size of the whole data set.

Then, we obtain a unified view of the two performance functions as follow: Without introducing any change, applying the identity mapping *-log* (*EXP* (-( ))) to the performance functions of *K-Means (KM)* and *EM*, the equations below are obtained;

$$\mathrm{Perf}_{KM}(X,M) = -\log\left(\prod_{t=1}^{N} \mathrm{EXP}\ (-\mathrm{MIN}\{\|x-m\|^2\ |m \in M\})\right) \tag{22}$$

$$\mathrm{Perf}_{\mathrm{EM}}(\mathrm{X},\mathrm{M}) = -\log\left(\prod_{t=1}^{N}\sum_{l=1}^{K}\mathrm{p}_l \cdot \frac{1}{\sqrt{(2\pi)^{D}}} \cdot \mathrm{EXP}(-\|\mathrm{x}-\mathrm{m}\|^2)\right) \tag{23}$$

The quantity inside the brackets "[]" in (26) is the linear mixing of the dataset functions – the

EXP()'s. Comparing K-Means and EM's performance function, the extra factor, 1/sqrt[(2pi)D], does not have any real impact because it only changes the performance function by adding a constant to it, which does not change the locations of the optima of the function.

$$\begin{aligned} &-\log\left(\prod_{t=1}^{N}\mathrm{EXP}\ (-\mathrm{MIN}\{\|\mathrm{x}-\mathrm{m}\|^2\ \ |\mathrm{m}\in\mathrm{M}\})\right) \\ &\qquad = -\log\left(\prod_{t=1}^{N}\left[\sum_{l=1}^{K}\mathrm{p}_l \cdot \frac{1}{\sqrt{(2\pi)^{D}}}\mathrm{EXP}(-\|\mathrm{x}-\mathrm{m}\|^2)\right]\right) \end{aligned} \tag{24}$$

Then, the extra factor = 1/sqrt[(2pi) D].

2.4 Time/Space complexities

The storage required at each node is O (*dim·* (*N/L* + *K)*)—the data set *S* is partitioned across the processors and the list of centers *M* is replicated at each processor. In contrast, the algorithm partitions the centers across the processors. Utilization is determined by the percentage of time each unit works on its own data to adjust the centers and collect cluster statistics, vs. the time waiting between sending out the local metrics to the Integrator and receiving the new global parameters. When the data size on each unit is far greater than the number of computing units (*N>>L*, which is true in general), the amount of work each unit has to do, *O(K·dim·N/L),* is far greater than the amount of work during the integration process, *O(K·dim·L).*

3 Results and discussions of findings

3.1 Performance evaluation

The programming tool used to implement the algorithms is MATLAB. This is because MATLAB is a very powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB stands for MATrix LABoratory, because the system was designed to make matrix computations particularly easy.

With MATLAB, computational and graphical tools to solve relatively complex science and engineering problems can be designed, developed and implemented.

Dataset Used: The students' biodata and academic records used for evaluating the performance of the algorithms come from Agric. Engr., CSE, FSE, Chemical Engr., EEE and CVE departments of LAUTECH FET. A number of fields in a University database table can be isolated as case study for data mining to discover hidden data but this study is concerned with discovering the correlations between the Student's mode of admission and their final CGPA at graduation only.

3.2 Performance metrics

The performance metrics considered include the number of iterations, computation time and system memory usage.

Number of iterations

Table1: Table showing number of iterations before convergence for the algorithms

Algorithm	Number of Iterations at convergence
K-Means	No distinct convergence (∞)
Expectation-maximization	1
Hybrid KEM	4

Computation time

Table 2: Table showing computation time at convergence for the algorithms

Algorithm	Computation time
K-Means	Undefined (∞)
Expectation-maximization	3.2s
Hybrid KEM	Approx. 0.26s

System memory usage

Table 3: Table showing space requirements at convergence for the algorithms

Algorithm	System memory usage
K-Means	∞
Expectation-maximization	1.1MB
Hybrid KEM	0.74MB

From the results obtained as summarized in the tables above, it can be observed that the hybrid KEM improves over K-Means by converging after some iterations unlike K-Means that does not guarantee convergence as well as the Expectation-Maximization algorithm which is very rigid and always generated a single iteration. Expectation Maximization has been used to optimize the performance of K-means as an improved version of K-means.

Memory usage/iteration

Given by System memory usage / number of iterations (at convergence)

K-Means: ∞/∞ = not defined

Expectation-Maximization: 1.1MB/1 = 1.1MB/iteration

Hybrid KEM: 0.74MB/4 ≈ 0.185 MB/iteration

3.3 Discussions

3.3.1 Expectation Maximization

The expectation-maximization algorithm converged in a single run giving three clusters. Figure EM1 shows the clusters with their three centroids. Two of the clusters for students admitted through PDS with the eclipse demarcating the clusters and the third cluster for students admitted through JAMB.

From virtual observation of the clusters, the cluster for students admitted through JAMB has its centroid on approximately 3.25 (CGPA). However, the clusters for students admitted through PDS have their centroids on 3.1 (CGPA) and approximately 4.1 (CGPA), so it can be concluded that students that were admitted through PDS on the average performed better than those admitted through JAMB. A possible explanation being that the PDS students would have undergone a thorough pre-university academic training for a year immediately before admission and on the other hand, it is not so for their JAMB counterparts, some of which would have stayed at home for sometime expecting a better JAMB result for their admission into the University.

However, another probable explanation is that the PDS programme affords the students the opportunity of being taught some of the 100 Level courses curriculum so that most of what they are taught in 100 Level becomes more of revision, which gives them a better edge over their JAMB counterparts.

The EM algorithm automatically produced three clusters and on any number of re-runs, it produced exactly the same clusters which make it a very rigid algorithm [7].

3.3.2 K-Means

The K-Means algorithm accepts the number of desired clusters as its input and in this case, five (5) was supplied to the algorithm because it is the number of the classes of degree in the University. The K-Means algorithm produced five (5) clusters for each re-run of the algorithm.

The K-Means algorithm was run over a hundred times (a hundred and two times precisely). It was observed that the algorithm did not show a distinct convergence even though similar pattern clusters were repeating themselves at irregular intervals. Interestingly, some particular patterns of clusters were persistent and the first could be observed from the APPENDIX in figures KMEAN1, KMEAN4, KMEAN10, KMEAN12, KMEAN14, KMEAN17, KMEAN20, KMEAN24, KMEAN26, KMEAN29, KMEAN30, KMEAN31, KMEAN33, KMEAN35, KMEAN38, KMEAN41, KMEAN45, KMEAN46, KMEAN49, KMEAN51, KMEAN56, KMEAN60, KMEAN67, KMEAN69, KMEAN74, KMEAN80, KMEAN81, KMEAN84, KMEAN85, KMEAN88, KMEAN92, KMEAN93, KMEAN94, KMEAN96, KMEAN99 and KMEN102 (36 instances in a 102 (35.3%) re-runs of the K-Means algorithm). A sample of this cluster pattern is shown in Figure 1 below:

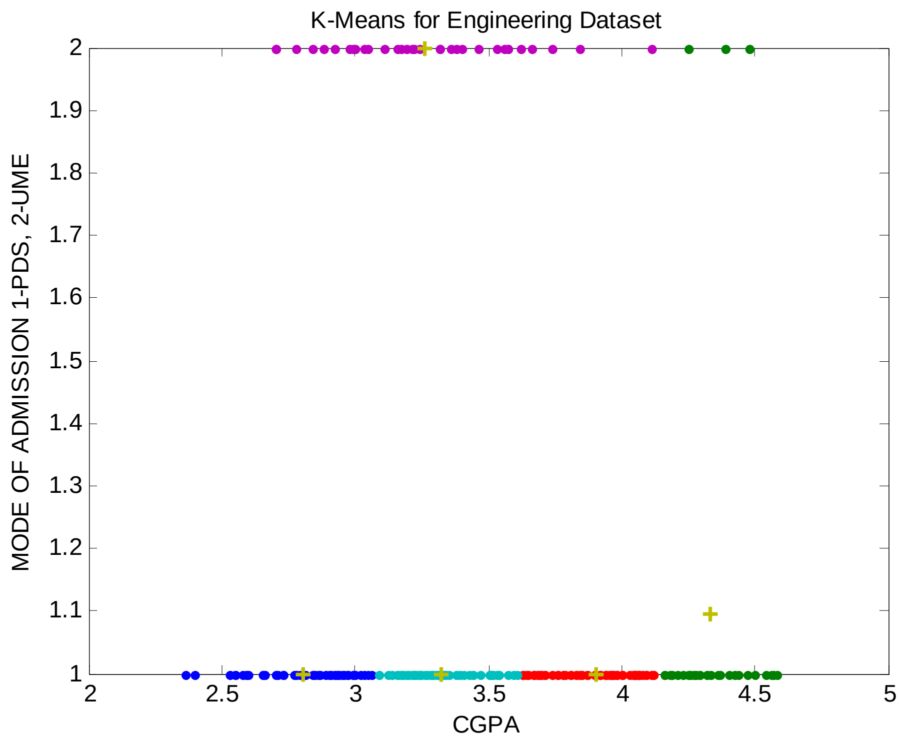

Figure 1: Mode of admission against CGPA

Another pattern of clusters with a closer persistence to the one observed above occurred in figures KMEAN11, KMEAN18, KMEAN19, KMEAN23, KMEA34, KMEAN36, KMEAN43, KMEAN48, KMEAN55, KMEAN61, KMEAN63, KMEAN65, KMEAN70, KMEAN71, KMEAN72, KMEAN78, KMEAN79, KMEAN82, KMEAN83, KMEAN86, KMEAN89 and KMEAN91 (22 instances in a 102 (21.6%) re-runs of the KMeans algorithm. A sample of this cluster pattern is shown in Figure 2 below:

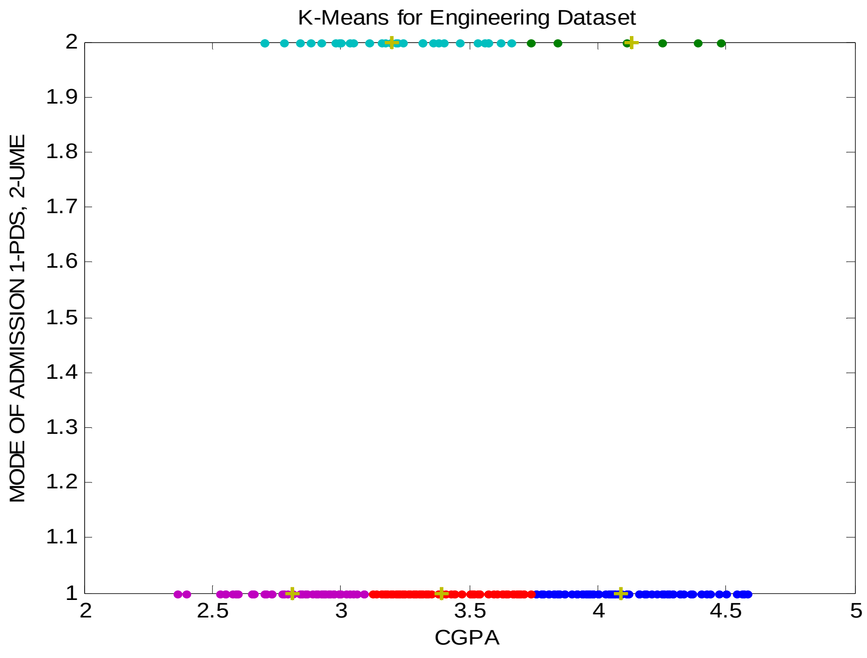

Figure 2: Mode of admission against CGPA

Picking the cluster pattern with the highest number of occurrence, it could be assumed to be the optimum set of clusters even though there is no distinct convergence of the cluster patterns.

3.3.3 Kmeans-Expectation-Maximization (KEM)

Like the K-Means algorithm, the KEM algorithm accepts as input the desired number of clusters and in this case also, five (5) was supplied to it for the same reason. On many re-runs of the KEM algorithm, several cluster patterns were produced.

Interestingly, a particular cluster pattern occurred for the first four (4) re-runs of the algorithm i.e. figures KEM1, KEM2, KEM3 and KEM4 in the APPENDIX. Furthermore, after the twelfth (12th) re-run of the algorithm, the same cluster pattern occurred at the thirteenth, fourteenth, fifteenth and sixteenth re-run i.e. figures KEM13, KEM14, KEM15and KEM16. With these observations, it can be rightly concluded that the KEM algorithm converged with the said cluster pattern shown in Figure 3 below:

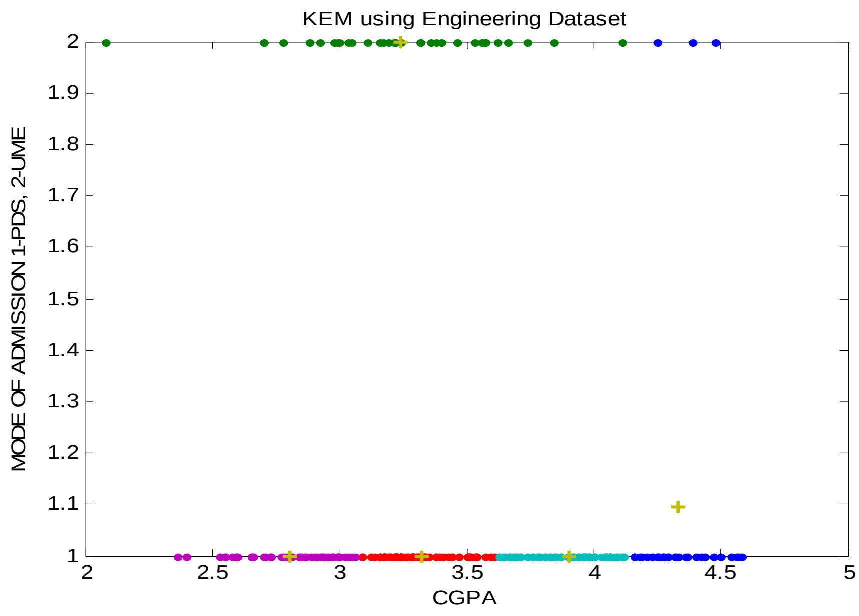

Figure 3: Mode of admission against CGPA

However, with more re-runs (up to 61 re-runs) of the KEM algorithm, similar cluster patterns occurred at close intervals shown from the APPENDIX in figures KEM23, KEM28, KEM32, KEM33, KEM43, KEM45, KEM51, KEM52, KEM55, KEM56 and KEM57.

More interestingly, it is observed that the cluster pattern obtained at convergence of the KEM algorithm is exactly the same cluster pattern with the highest percentage of occurrence produced by the K-Means algorithm.
Analyzing the cluster pattern obtained at convergence, it could be observed that there were exactly three clusters for students admitted through PDS with their centroids on approximately 2.8 (CGPA), 3.35 (CGPA) and 3.9 (CGPA), one cluster that included entries from both modes of admission with the centroid (at 4.35 CGPA) very close to the PDS admission mode showing that the cluster is made up of more entries from students admitted through PDS than those admitted through JAMB. There is only one cluster for students admitted through JAMB with its centroid on 3.25 (CGPA).

In conclusion, from the positions of all the five (5) centroids, it would be observed that students admitted through PDS performed better than those admitted through JAMB [7].

4 Conclusion

Two clustering algorithms considered in this study are K-means and Expectation-Maximization algorithms. After the evaluation of the two algorithms, K-means was found not to guarantee convergence while Expectation-Maximization's quick convergence doesn't guarantee optimality of results. As such, both algorithms are not efficient enough for any clustering problem, hence there arises a need for an algorithm that could both guarantee convergence and optimality of results. A hybrid of K-means and Expectation-Maximization (KEM) was developed for this purpose.

First, the hybrid KEM had the least computational time at convergence (≈0.26s), followed by that of the Expectation-Maximization (3.2s) while the K-

Means algorithm did not exhibit convergence at all giving an infinity (∞) computational time.

Secondly, the KEM algorithm converged with 4 iterations; the Expectation-Maximization algorithm converged with a single iteration while K-Means algorithm did not show any distinct convergence. Thirdly, the memory usage for the hybrid KEM algorithm was estimated at 0.74MB, that of the Expectation-Maximization algorithm was estimated at 1.1MB and that of the K-Means algorithm was not defined (∞). Also, with the calculation of memory usage per iteration, the Hybrid KEM exhibited the least ratio of memory usage to iteration (≈ 0.185 MB/iteration), followed by that of the Expectation-Maximization algorithm (1.1MB/iteration) while the K-Means algorithm seemed to exhibit the highest ratio of memory usage per iteration with a very high value (∞).

Finally, the results of the evaluation showed that the hybrid KEM developed guarantees both optimality and convergence. It was also found to perform better than the other two algorithms in terms of the computational time, memory usage at convergence and the ratio of memory usage to iteration.

Furthermore, the optimum clusters obtained at convergence from the datasets obtained from the database of Faculty of Engineering and Technology, LAUTECH, it was observed that students admitted through the Predegree Science (PDS) programme actually performed better than their counterparts admitted through Joint admission and matriculation board (JAMB). This could have been due to a number of reasons, some of which are probably being that (i), the PDS students would have undergone a thorough pre-university academic training for a year immediately before admission and on the other hand, it is not so for their JAMB counterparts, some of which would have stayed at home for sometime expecting a better JAMB result for their admission into the University. (ii) The PDS programme affords the students the opportunity of being taught some of the 100 Level courses curriculum so that most of what they are taught in 100 Level becomes more of revision, which gives them a better edge over their JAMB counterparts.

In view of the above, the algorithm developed (Hybrid KEM) has demonstrated a sharp and improved performance over the other two (K-Means and Expectation-Maximization).

References

[1] O. Benli and A. Botsali, An Optimization-Based Decision Support System for a University Timetabling Problem: An Integrated Constraint and Binary Integer Programming Approach, (2004), Available from: http://www.csulb.edu/~obenli/research/benli-botsali.pdf [Oct 2005].

[2] Z. Huang, Extensions to the K-means Algorithm for Clustering Large Datasets with Categorical Values, *Data Mining and Knowledge Discovery*, **2**, (1998), 283-304.

[3] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in L.M. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, CA, 1, (1967), 281-297.

[4] N. Sara, A. Rawa and V. Gregory, A modified Fuzzy k-means clustering using Expectation Maximization, *IEEE International Conference on Fuzzy Systems,* Vancouver BC, (2006), 231-235.

[5] B. Zhang, M. Hsu and U. Dayal, *K-Harmonic Means – A Data Clustering Algorithm*, Software Technology Laboratory, HP Laboratories, Palo Alto, 1999.

[6] Yanfeng Zhang, Xutao Li, Yunming Ye, Xiaofei Xu and Shengchun Deng, Information-theoretic Agglomerative K-means, *Information Technology Journal*, **10**, (2011), 2420-2426.

[7] A.A. Adigun, E.O. Omidiora and S.O. Olabiyisi, An Exploratory Study of K-Means and Expectation Maximization Algorithms, *British Journal of Mathematics & Computer Science,* UK, **2**(2), (2012), 62-71.