Performance Comparison of K-Means and Expectation Maximization with Gaussian Mixture Models for Clustering
EE6540 Final Project

Devin Cornell & Sushruth Sastry

May 2015

Abstract

In this article, we explore the theoretical aspects of the expectation maximization algorithm and how it can be applied to estimation of data as a Gaussian mixture model. We form comparisons and show how by the simplification of some parameters we can form the heuristic k-means algorithm. We then demonstrate through the authors' code several situations where the EM-GMM algorithm performs significantly better than k-means and touch upon potential implications of these findings for arbitrary distribution approximation. Finally, we form some conclusions about scenarios where EM-GMM should be used over k-means and how to use them for classification.

Contents

1 Introduction

Because of high costs or time restrictions, many important machine learning problems lack the large set of labelled data required to train some types of classifiers. As such, unsupervised or semi-supervised learning algorithms such as k-means or estimation of Gaussian mixture models have become very important tools in data analysis. This article will explore some of the advantages of using one of these algorithms over the other, and how they are related from a theoretical standpoint. First we start with a theoretical background of the algorithms.

2 Background

Now we will explore some of the theoretical aspects of these algorithms.

2.1 Gaussian Mixture Models

The gaussian mixture model (GMM) is a finite mixture probability model describing a random variable composed of a number of ***mixture components***, each of which resembles a weighted Gaussian distribution [1]. The pdf for this mixture can be given by Equation 1 provided in [2], where there are K *mixture components*, $\Theta = [\theta_1\ \theta_2\ ...\ \theta_K]^T$, and $\theta_k = \{\alpha_k, \mu_k, \Sigma_k\}$. Let there also be a designation of $\mathbf{z}_k \in \mathscr{Z}$ which describes the *mixture component* from which a given observation $\mathbf{x} \in \mathscr{X}$ was generated. z_k can be thought of as a K dimensional binary vector where exactly one component is a 1 [2].

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|\mathbf{z}_k, \mu_k, \Sigma_k) \tag{1}$$

Note that this description should dictate that $\alpha_k = p(z_k)$, and this observation will be useful as we derive the background to compute an estimate for the GMM later. In the case where the pdf of each *mixture component* is assumed to be Gaussian, equation 2 gives $p_k(\mathbf{x}|\mu_k, \Sigma_k)$ where the $d \times 1$ vector μ_k and $d \times d$ matrix Σ_k are the mean and covariance of the mixture component k, respectively [2].

$$p_k(\mathbf{x}|\mathbf{z}_k, \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^d|\Sigma_k|}} \exp(-\frac{1}{2}(\mathbf{x} - \mu_\mathbf{k})^T\Sigma^{-1}(\mathbf{x} - \mu_\mathbf{k})) \tag{2}$$

Given that the *mixture weights* α_k are needed to ensure that $\int_{-\infty}^{\infty} p(\mathbf{x}|\Theta)d\mathbf{x} = 1$, it follows that $\sum_{k=1}^{K} \alpha_k = 1$ [2]. In this way, the *mixture weights* α_k can be seen as the probability that an observation of $\mathbf{x}$ was generated by a particular *mixture component*. Figure 1 shows an example of a $p(\mathbf{x}|\Theta)$.

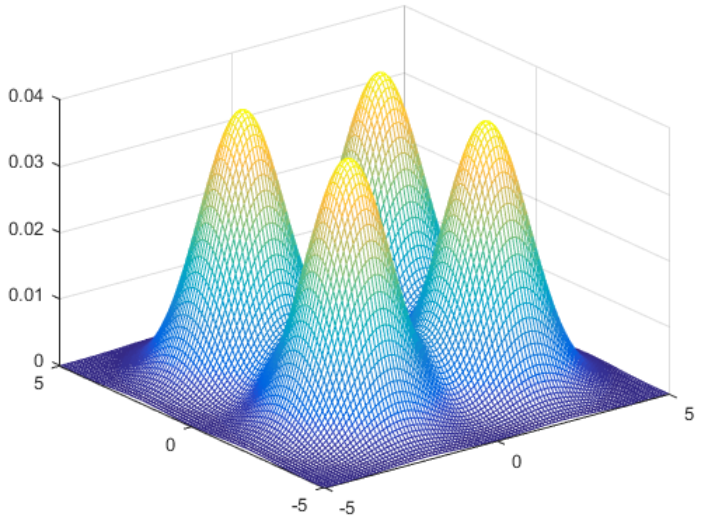

Figure 1: PDF for a gaussian mixture model with four *mixture components*, where $\mu_1 = [2; 2],\ \mu_2 = [2; -2],\ \mu_3 = [-2; 2],\ \mu_4 = [-2; -2]$, each with a $\Sigma_k = I_2$ and $\alpha_k = 0.25$.

Suppose we have N observations of the random variable modelled by $p(\mathbf{x}|\Theta)$, and each observation can be given by $\mathbf{x}_i \in \Re^d$ and appears in $X = [\mathbf{x}_1\ \mathbf{x}_2\ ...\ \mathbf{x}_N]^T \in \Re^{d\times N}$. Then given knowledge about the number of *mixture components* K, the parameters Θ can be recovered by using estimator theory.

Having an estimate for the generative model of observations can have various applications, including the ability to assign an observation $\mathbf{x}_i$ back to the originating *mixture component* as a way of vector quantization. As we will see later, this provides several advantages over heuristic algorithms for clustering such as k-means. While this model tries to reconcile the data in X as having originated from from a GMM, we don't necessarily need to assume the data was generated using this model in order to find it useful.

2.2 Expectation Maximization Algorithm

The expectation maximization (EM) algorithm is a method for obtaining a maximum likelihood (ML) estimate for observations when it is assumed that there is some information missing completely at random for each observation [3]. The theory behind this algorithm provides for two distinct vector spaces: one is the observation space $\mathscr{V}$ from which observations will come and the other is the hidden space $\mathscr{H}$ where the unknown information resides. Each observation v is considered to be generated from a random variable following $p(v|\theta)$ in $\mathscr{V}$ and also corresponds to a hidden observation $h \in \mathscr{H}$ from a random variable following $p(h|\theta)$. Then the relationship between these two random variables can be given by equation 3, where any observation $v \in \mathscr{V}$ is assumed to

be composed of a set of hidden observations $\hat{H} \in \mathscr{H}$ determined by the underlying relationship between the two variables [3].

$$p(v|\theta) = \int_{\hat{H}} p(h|\theta)dh \tag{3}$$

The approach for providing a ML estimate is to find the θ that maximizes the log-likelihood of $p(h|\theta)$, which can also be expressed as a pdf parametrized on observations v because this information is known. This pdf can also be expressed using bayes rule and the priori distribution, as is convenient in many cases. The end result is a ML estimator that provides the h with the highest probability given the observed data and the parameters. In this case, the ML estimate is given from equation 4 [3].

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log p(h|v,\theta) = \arg\max_{\theta} \log(\frac{p(v|h,\theta)p(h|\theta)}{p(v|\theta)}) \tag{4}$$

However because the hidden parameters h are unknown, neither the log likelihood of the hidden observation pdf or its Bayesian equivalent can be computed explicitly [3]. The EM algorithm draws utility from the fact that it can compute the ML estimate of the parameters using only the estimate of the hidden parameter pdf, defined as in equation 5. In this equation, θ' represents an estimate for θ and further explanation of the EM algorithm will explore how to compute these estimates given $Q(\theta|theta')$.

$$Q(\theta|\theta') = E_v[\log p(h|v,\theta)|v,\theta'] \tag{5}$$

Note that Q is a conditional expectation where v and θ' are fixed, and the expectation is taken with respect to observations v. In this sense, Q can be thought of as the log-likelihood of the hidden parameter h averaged across each observation v. Next we will show how Q can be used to compute an ML estimate of the parameters.

For practical purposes, it may also be convenient to develop the notion of a data set within the context of EM. Say we have some observations V from observation space $\mathscr{V}$ which correspond to some hidden observations $H \subset \mathscr{H}$. Note that in order to perform EM on an observed data set, an assumption for the space $\mathscr{H}$ must be made. This determination will be made based on a specific problem and is determined by $p(h|\theta)$, but for the sake of generality we will not use a specific pdf for the hidden random variable.

Because the EM algorithm gives an iterative estimate for θ, we will denote θ^t to be the tth estimate of θ provided by the tth iteration of the algorithm. Assuming that some random or heuristic approach can be used to generate an initial estimate θ^0, the EM algorithm is given in algorithm 1.

Algorithm 1 Generalized EM algorithm

Require: $V \subset \mathscr{V}$ are samples from observation space
Require: ϵ is stop criteria
  function EM(V, ϵ)
    initialize θ^0 randomly, $L_{old} := \infty$, $dist := \infty$
    while $dist > \epsilon$ **do** ▷ while update has significant effect
      $Q(\theta|\theta^t) := E_v[\log p(h|v, \theta)|v, \theta']$ ▷ E-Step: Compute $Q(\theta|\theta^t)$
      $\theta^t := \arg\max_{\theta} Q(\theta|\theta^t)$ ▷ M-Step: Update estimate for θ
      $L_{new} := \sum_{v \in V} \log p(v|\theta^t)$ ▷ calculate log-likelihood
      $dist := ||L_{new} - L_{old}||$ ▷ determine improvement
      $L_{old} := L_{new}$
    end while
  return θ^t
  end function

Although under some constraints the EM algorithm is guaranteed to converge to a local optimum, there is no guarantee of a global optimum convergence [2]. To get around this caveat, it may be appropriate to apply the EM algorithm with different initial conditions and compare the log-likelihood parameter to determine the best of the estimates.

2.3 Expectation Maximization for GMM

In the case of providing a ML estimate for the parameters of a GMM, the hidden information for each observation is the *mixture component* from which an observation is generated. In order to correspond to the theoretical background for the EM algorithm, consider $\mathbf{x}$ to be the observation v generated from observation space $\mathscr{X}$, $\mathbf{z}$ to be the hidden observations from hidden space $\mathscr{Z}$, and let the pdf $p(v|\theta)$ take the form of $p(\mathbf{x}|\Theta)$ from equation 1. This means that the hidden space $\mathscr{Z} = \{1,\ 2,\ ...\ K\}$, and each $z_k \in \mathscr{Z}$ will take one value which is an identifier from which distribution a given observation was generated.

Because the EM algorithm requires us to optimize $p(\mathbf{z}|\Theta)$, we must first describe a way to compute this from our known observations and our assumed GMM equations. Using the bayesian equivalent given by Equation 6.

$$p(\mathbf{z}|\mathbf{x}, \Theta) = \frac{p(\mathbf{x}|\mathbf{z}, \Theta)p(\mathbf{z}|\Theta)}{p(\mathbf{x}|\Theta)} \tag{6}$$

Algorithm 2 for EM-GMM is a reprint from [1].

Algorithm 2 EM-GMM algorithm (reprint from [1])

Initialise the centres $\mathbf{m}_i$, covariances $\mathbf{S}_i$ and weights $p(i) > 0$, $\sum_i p(i) = 1$ $i = 1, \ldots, H$.
while Likelihood not converged or termination criterion not reached **do**
  for $n = 1, \ldots N$ **do**
    for $i = 1, \ldots, H$ **do**
      $p(i|\mathbf{x}^n) = p(i) \exp\left\{-\frac{1}{2}(\mathbf{x}^n - \mathbf{m}_i)^\mathsf{T} \mathbf{S}_i^{-1} (\mathbf{x}^n - \mathbf{m}_i)\right\} \det(\mathbf{S}_i)^{-\frac{1}{2}}$ ▷ responsibility
    end for
    Normalise $p(i|\mathbf{x}^n)$ so that $\sum_i p(i|\mathbf{x}^n) = 1$
    $p(n|i) = p(i|\mathbf{x}^n)$ ▷ membership
  end for
  Normalise $p(n|i)$ so that $\sum_n p(n|i) = 1$ for each i.
  for $i = 1, \ldots, H$ **do**
    $\mathbf{m}_i = \sum_{n=1}^N p(n|i)\mathbf{x}^n$ ▷ M-step for means
    $\mathbf{S}_i = \sum_{n=1}^N p(n|i) (\mathbf{x}^n - \mathbf{m}_i) (\mathbf{x}^n - \mathbf{m}_i)^\mathsf{T}$ ▷ M-step for covariances
    $p(i) = \frac{1}{N} \sum_{n=1}^N p(i|\mathbf{x}^n)$ ▷ M-step for weights
  end for
  $L = \sum_{n=1}^N \log \sum_{i=1}^H p(i) \frac{1}{\sqrt{\det(2\pi\mathbf{S}_i)}} \exp\left\{-\frac{1}{2} (\mathbf{x}^n - \mathbf{m}_i)^\mathsf{T} \mathbf{S}_i^{-1} (\mathbf{x}^n - \mathbf{m}_i)\right\}$ ▷ Log likelihood
end while

2.4 EM-GMM Clustering Algorithm

The EM-GMM clustering algorithm is similar to the normal EM-GMM algorithm, except that a mixture component is assigned to each x_i based on the probabilities of each Gaussian mixture after the GMM model has been fit. This aspect is simple in theory but worth noting to avoid confusion.

2.5 K-Means Algorithm

The k-means algorithm given in Algorithm 3 is a special case of the EM-GMM algorithm where the generating pdf is not estimated. The EM-GMM algorithm can be modified to perform k-means if the prior probabilities α_k are all set to be equal and sum to one, the $\Sigma_k = I_d$ and don't change on each iteration, and the membership weights are not calculated and instead each observation is assumed to belong only to the closest mixture component [2].

3 Methodology

Several experiments were performed to analyze performance between the EM-GMM and k-means clustering algorithms. The general flow of these experiments is that random data is generated from models with multiple components, and the clustering algorithms are used to try and determine the generating mixture of each sample. For the first three tests, the number of fitted mixtures is the same number as the number of generating mixtures, but for the last test, the number of generating

Algorithm 3 Generalized K-means algorithm

Require: K number of expected classes
Require: $\mathbf{X}$ observed data
  function KMEANS(K, $\mathbf{X}$)
    initialize K means denoted by u_k, $\mathbf{u} := \{u_1, u_2 \cdots u_K\}$, observations $x \in \mathbf{X}$
    while no change **do** ▷ while no difference in class since previous iteration
      for each x **do**
        assign class k to data x if u_k is the nearest.
      end for
      for each k **do**
        u_k:= mean of x in class k.
      end for
    end while
  return u_k
  end function

mixtures was considered to be unknown and the number of fitted mixtures is varied. For the first three experiments, scores were generated based on the number of samples correctly labelled, but the last test involves a more difficult metric that will be explained later.

The code used for the EM-GMM algorithm and data generation was developed exclusively by the authors of this paper but also used some basic built-in Matlab functions. The k-means algorithm implementation used here for comparison is the default Matlab implementation.

3.1 Separate GMM Data

As a baseline test, a typical clustering problem was presented to both algorithms. The test data consists of 1000 points generated by a GMM with four mixture components and $\mu_1 = [2; 2]$, $\mu_2 = [2; -2]$, $\mu_3 = [-2; 2]$, $\mu_4 = [-2; -2]$, $\Sigma_k = I_2$ and $\alpha_k = 0.25$ for $k \in [1..K]$. Figure 2 shows the generated data for the two dimensional case.

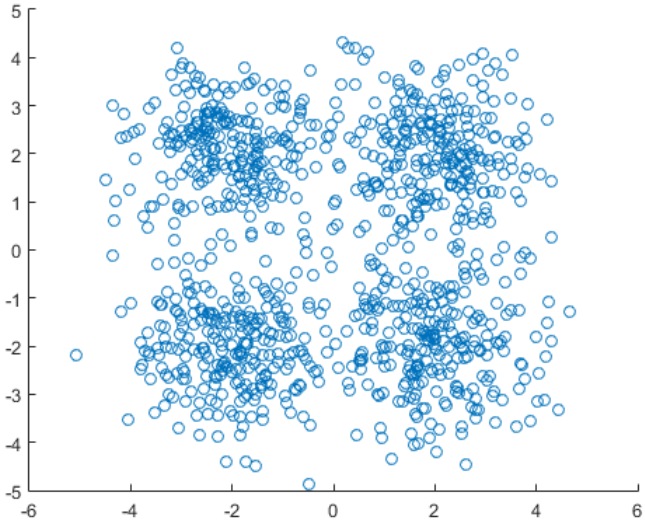

Figure 2: Data generated by a Gaussian mixture model with four *mixture components*, where $\mu_1 = [2; 2],\ \mu_2 = [2; -2],\ \mu_3 = [-2; 2],\ \mu_4 = [-2; -2]$, each with a $\Sigma_k = I_2$ and $\alpha_k = 0.25$.

Due to the circular shape of the Gaussian mixture components, this dataset was designed to perform equally well using both k-means and EM-GMM. As such, this data set will be used as a baseline comparison of the two clustering algorithms.

3.2 Concentric GMM Data

The next data set was designed to be a more complicated GMM where some of the *mixture components* are concentric, in some cases with different weights. Figure 3 shows the result of a Gaussian mixture where different parameters were adjusted for each mixture component. There are 6 mixture components in total, but they will be considered as 3 pairs of concentric mixture components. The top right two mixture components each have the same weights and the same means so that we may analyze the performance of two intersecting identical Gaussians. The top left two mixture components each have the same probability, but have offset centers so that we can analyze the clustering performance for each mixture. The bottom two mixtures each have different weights so we can compare the clustering performance when there are fewer samples from one of the mixtures.

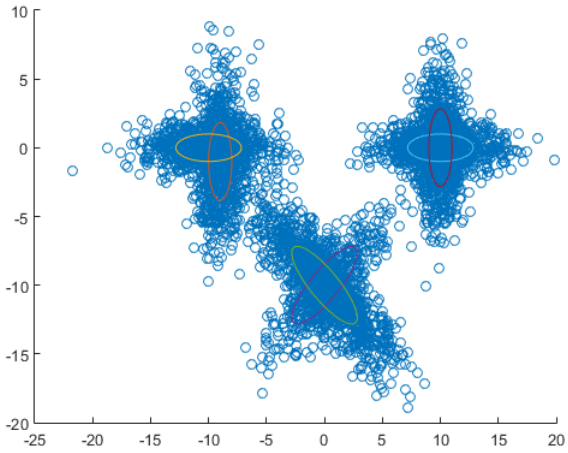

Figure 3: Data generated by a Gaussian mixture model with six *mixture components*, where $\alpha = [1/6; 1/6; 1/12; 3/12; 1/6; 1/6]$, $\mu_1 = [-0.9; -0.1]$, $\mu_2 = [-1; 0]$, $\mu_3 = [0; -1]$, $\mu_4 = [0; -1]$, $\mu_5 = [1; 0]$, $\mu_6 = [1; 0]$, $\Sigma_1 = [1, 0; 0, 8], \Sigma_2 = [8, 0; 0, 1], \Sigma_3 = [8, 8/1.2; 8/1.2, 8], \Sigma_4 = [8, -8/1.2; -8/1.2, 8], \Sigma_5 = [8, 0; 0, 1], \Sigma_6[1, 0; 0, 8]$. The gaussian elipses were also plotted for convenience.

Note that with so many mixture components, the parameter space can become very high dimensional, and thus the EM algorithm may be more prone to initial conditions causing a local optimum to be reached. This is something the EM-GMM algorithm will need to take into account.

3.3 Intermixed Gaussian with Large Covariance Differences

The next data model was designed to show the difference between algorithms when presented with a data set where the *mixture components* have very different covariances. Figure 4 shows a sampled GMM with two mixture components that model this situation.

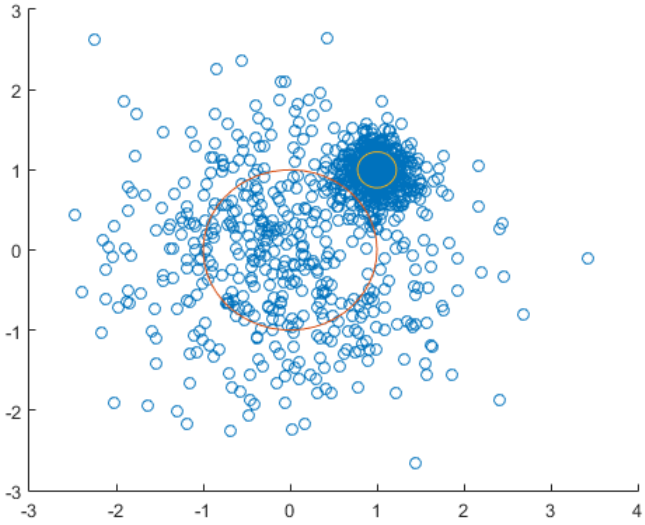

Figure 4: Data generated by a gaussian mixture model with two *mixture components*, where $\alpha = [1/2, 1/2]$, $\mu_1 = [0; 0]$, $\mu_2 = [1; 1]$, $\Sigma_1 = [1, 0; 0, 1], \Sigma_2 = [.05, 0; 0, .05]$. The gaussian elipses were also plotted for convenience.

3.4 Radial Poisson Distributions with Different Means

The final test case that will be analyzed in this document is a non-GMM random set. Instead, a radial Poisson distribution is used where each *mixture component* uses a different mean. While this is not a traditional cluster set up, this can be a practical issue that may come up in real-world analysis although in high dimensional space it may be difficult to detect. The radial Poisson distribution is given in Figure 5.

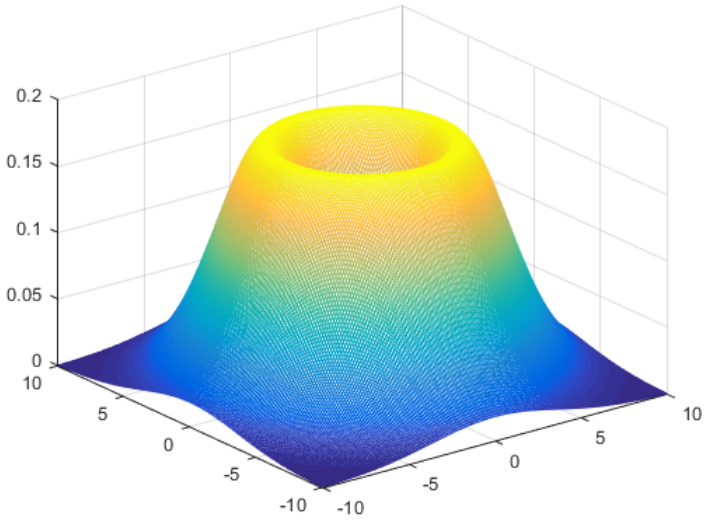

Figure 5: An example of a radial Poisson distribution with a mean value of 5 and a θ range of $[-\pi, \pi]$.

In order to use this distribution with a clustering algorithm, multiple mixtures of this radial Poisson distribution should be used. Figure 6 shows a realization of a multiple-mixture radial Poisson distribution. Given this observation, we can't expect the EM-GMM model or the k-means algorithm to detect exactly the distributions used to generate the data set, but using more than two clusters could allow the algorithms to model the generating distributions using a series of Gaussian distributions.

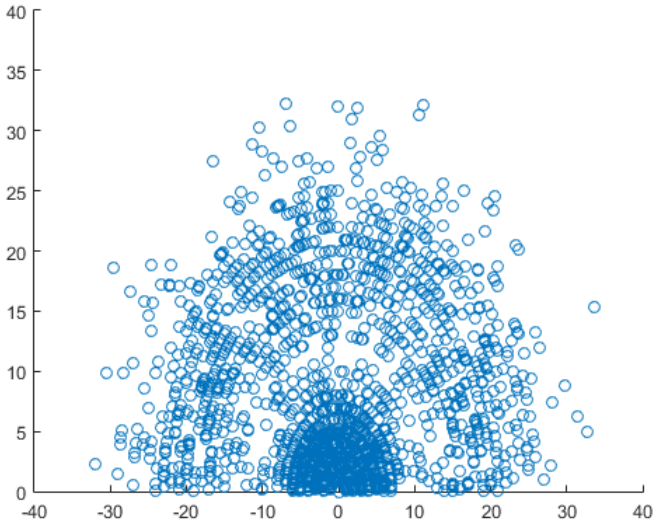

Figure 6: Data generated by a radial Poisson distribution where $\lambda_1 = 5$, and $\lambda_2 = 20$.

The performance of the clustering algorithms will be shown in the next section.

4 Results

The results of the four described tests are given here. Note that in the first three tests, the number of means K was selected to be the same as the number of *mixture components* of the generating pdf. In practice, this may require some knowledge about the structure of the data, or it may be determined by trying multiple values for K to see which perform the best. While this is an important practical consideration, it will be outside the scope of this document.

4.1 Separate GMM Data

The data shown in Figure 7 displays the result of clustering on the data generated by a GMM with four mixture components, all separated. Note the performance of both K-Means and EM-GMM are very good and also comparable. This test was used as a baseline test to ensure the comparability of the two algorithms.

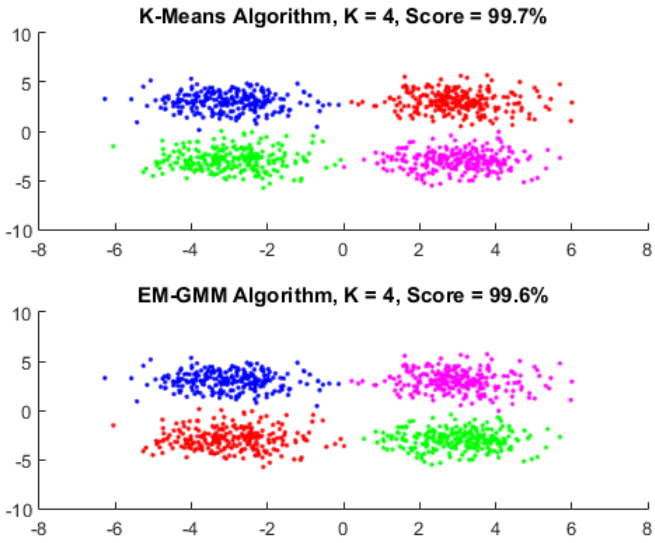

Figure 7: The resulting labels and clustering performance of K-Means vs EM-GMM on the test data generated by four GMM mixtures with identical variances.

Given that both k-means and EM-GMM can fit data with symmetric covariance properties, this result is somewhat expected.

4.2 Intermixed GMM Data

The data given in Figure 8 shows the result from using both k-means and EM-GMM on the data set generated by a model of six intersecting gaussian mixtures.

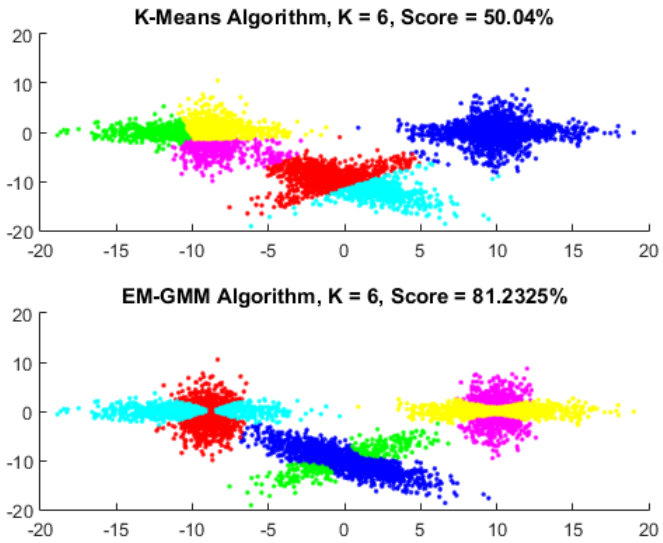

Figure 8: The resulting labels and clustering performance of k-means vs EM-GMM on data generated by six GMM mixtures, each with a different mean and covariance, selected so that the distributions will overlap in different ways. In this example, EM-GMM performs much better than k-means.

The poor performance of the k-means algorithm is likely due to the assumption that all the components will have spherical models determined by the covariances.

4.3 Concentric Gaussian with Large Covariance Differences

The test results shown in Figure 9 demonstrate the ability to classify when two clusters are combined where one has a much larger covariance than the other. Because the k-means algorithm fails to take into account the difference in sizes of gaussians, it ends up creating a linear decision boundary in the classification process.

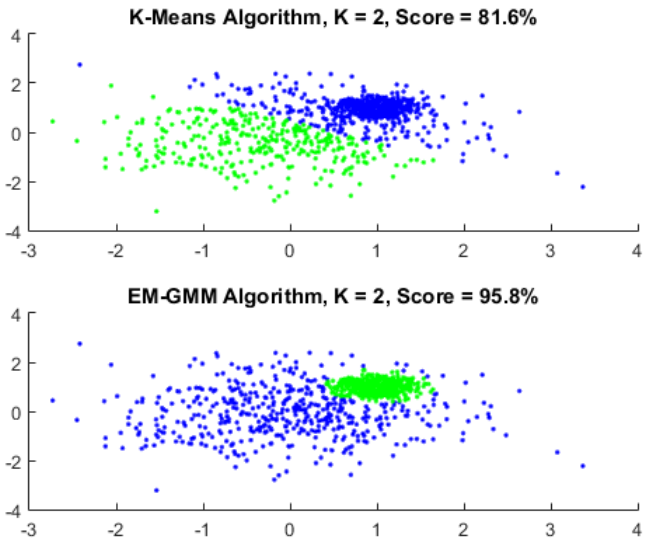

Figure 9: The resulting labels and clustering performance of K-Means vs EM-GMM on the test data generated by two GMM components where one has a large covariance and the other has a smaller covariance.

4.4 Radial Poisson Distributions with Different Means

While all of the previous experiments use data generated by a GMM, this data is generated by two radial Poisson distributions. Although this data is decidedly not gaussian, it still may be reasonable to estimate the distribution with a number of gaussian mixtures. As such, multiple values of K were analyzed, and the results are shown in Figures 10, 11, and 12.

Note that in the cases where the number of fitted mixtures is greater than the number of generating mixtures, the score is calculated upon the assumption that an algorithm exclusively mapping a set of fitted mixtures to each generating mixture can be applied. Although this may not be a straightforward procedure, it is practical in a case where some arbitrary data is modelled by a GMM or set of clusters.

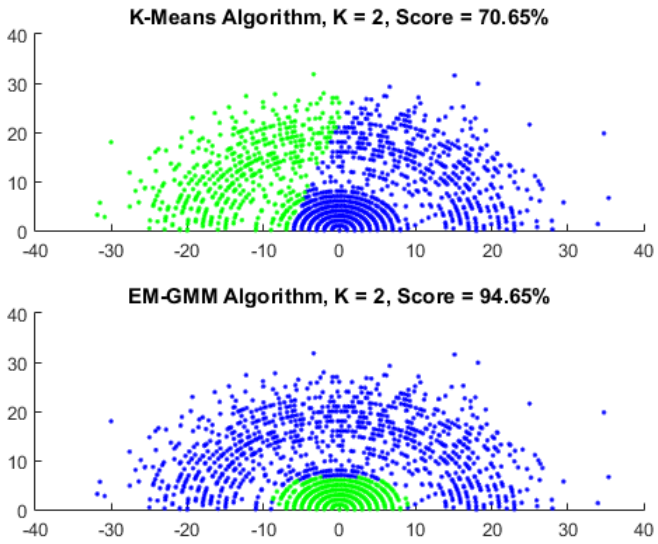

Figure 10: The resulting labels and clustering performance of K-Means vs EM-GMM on the test data generated by two radial poisson mixtures where $\lambda_1 = 5$ and $\lambda_2 = 15$ and where the number of fitted mixtures was $K = 2$. EM-GMM seemed to significantly outperform k-means in this experiment.

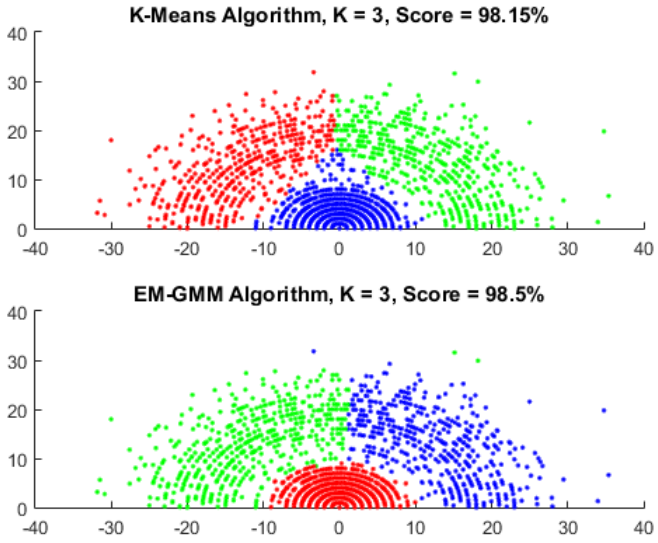

Figure 11: The resulting labels and clustering performance of K-Means vs EM-GMM on the test data generated by two radial poisson mixtures where $\lambda_1 = 5$ and $\lambda_2 = 15$ and where the number of fitted mixtures was $K = 3$. The two algorithms had comparable performance in this experiment.

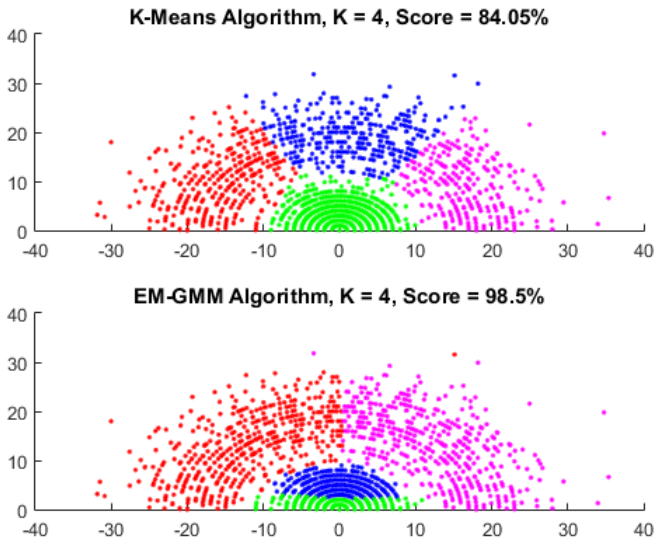

Figure 12: The resulting labels and clustering performance of K-Means vs EM-GMM on the test data generated by two radial poisson mixtures where $\lambda_1 = 5$ and $\lambda_2 = 15$ and where the number of fitted mixtures was $K = 4$. EM-GMM seemed to outperform k-means by a decent margin in this experiment.

4.5 Results Summary

The classification accuracy of both algorithms for each experiment can be found in Table 4.5. In almost all cases, the EM-GMM algorithm will outperform the k-means algorithm, but the trade off comes in the computation time. Because he k-means is not updating a new Σ on each iteration it is significantly faster. Although k-means is expected to be faster, we were not able to collect algorithmic complexity data for these experiments due to the fact that the Matlab k-means algorithm is implemented by professionals and the EM-GMM was implemented by the authors, so the results should not be comparable.

Table 1: This table summarizes the cluster accuracy for each experiment.

Test	k-means (%)	EM-GMM (%)
Separated GMM	99.7	99.6
Intermixed GMM	50.0	81.2
Concentric Gaussian	81.6	95.8
Radial Poisson,$K = 2$	70.7	94.7
Radial Poisson,$K = 3$	98.15	98.5
Radial Poisson,$K = 4$	84.1	98.5
Radial Poisson,$K = 5$	88.15	98.6

5 Conclusions

Based on the data published in the Results section as well as the experience gained from running the experiments, several observations and conclusions can be made.

- The EM-GMM is significantly slower than the k-means algorithm because it maintains partial membership information for each data point and it needs to update the estimate for the covariance matrix on each iteration.
- The EM-GMM outperformed k-means for all of the experiments performed here, but in some cases the results were comparable for each algorithm, particularly in the case when the clustering algorithms were being used to estimate non-gaussian variables with higher numbers of K. It is hypothesized that with enough fitted mixtures and a way to map the fitted mixtures to the a set of classes, it would be possible to model many arbitrary distributions using these clustering algorithms.
- In many practical situations where the generating distribution is unknown and it is desirable to cluster a much higher dimensional observation space, the value of K should either be adjusted, or a more complex algorithm should be used where the number of fitted mixtures can be considered to be infinite.

References

[1] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[2] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.