TRUST-TECH based Expectation Maximization for Learning Finite Mixture Models

Chandan K. Reddy, *Member, IEEE,* Hsiao-Dong Chiang, *Fellow, IEEE,* and Bala Rajaratnam,

Abstract

The Expectation Maximization (EM) algorithm is widely used for learning finite mixture models despite its greedy nature. Most popular model-based clustering techniques might yield poor clusters if the parameters are not initialized properly. To reduce the sensitivity of initial points, a novel algorithm for learning mixture models from multivariate data is introduced in this paper. The proposed algorithm takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibria CHaracterization) to compute neighborhood local maxima on the likelihood surface using stability regions. Basically, our method coalesces the advantages of the traditional EM with that of the dynamic and geometric characteristics of the stability regions of the corresponding nonlinear dynamical system of the log-likelihood function. Two phases namely, the EM phase and the stability region phase, are repeated alternatively in the parameter space to achieve local maxima with improved likelihood values. The EM phase obtains the local maximum of the likelihood function and the stability region phase helps to escape out of the local maximum by moving towards the neighboring stability regions. Though applied to Gaussian mixtures in this paper, our technique can be easily generalized to any other parametric finite mixture model. The algorithm has been tested on both synthetic and real datasets and the improvements in the performance compared to other approaches are demonstrated. The robustness with respect to initialization is also illustrated experimentally.

Keywords - expectation maximization, unsupervised learning, finite mixture models, dynamical systems, stability regions, model-based clustering.

I. Introduction

In the field of statistical pattern recognition, finite mixtures allow a probabilistic model-based approach to unsupervised learning [20]. One of the most popular methods used for fitting mixture models to the

Dr. Chandan Reddy is with the Department of Computer Science at Wayne State University. Corresponding author : reddy@cs.wayne.edu

Dr. Hsiao-Dong Chiang is with the School of Electrical and Computer Engineering at Cornell University.

Dr. Bala Rajaratnam is with the Department of Statistics at Stanford University.

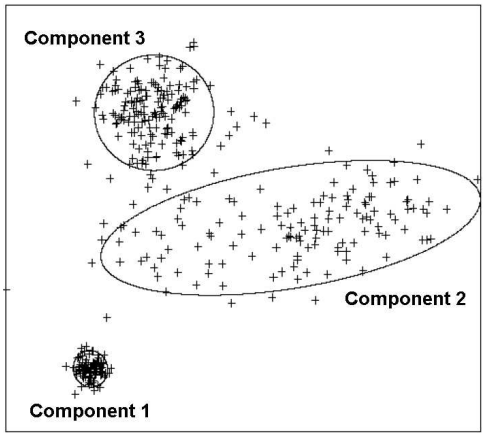


Fig. 1. Data generated by three Gaussian components. The problem of learning mixture models is to obtain the parameters of these Gaussian components and the membership probabilities of each datapoint.

observed data is the *Expectation Maximization* (EM) algorithm which converges to the maximum likelihood estimate of the mixture parameters locally [7], [27]. The traditional steepest descent, conjugate gradient, or Newton-Raphson methods are too complicated for use in solving this problem [39]. EM has become a popular method since it takes advantage of some problem specific properties. EM based methods have been successfully applied to solve a wide range of problems that arise in pattern recognition [2], [3], clustering [1], information retrieval [22], computer vision [5], data mining [32] etc..

In this paper, we consider the problem of learning parameters of Gaussian Mixture Models (GMM). Fig 1 shows data generated by three Gaussian components with different means and variances. Note that every data point has a probabilistic (or soft) membership that gives the probability with which it belongs to each of the components. The data points that belong to component 1 will have high probability of membership for component 1. On the other hand, data points belonging to components 2 and 3 are not well separated. The problem of learning mixture models involves not only estimating the parameters of these components but also finding the probabilities with which each data point belongs to these components. Given the number of components and an initial set of parameters, the EM algorithm can be applied to compute the optimal estimates of the parameters that maximize the likelihood function. However, the main problem with the EM algorithm is that it is a '*greedy*' method which converges to a local maxima on the log-likelihood surface. Hence, the final solution that the EM algorithm converges to, will be very sensitive to the given initial set of parameters.

This local maxima problem (popularly known as the initialization problem) is one of the well studied issues in the context of the EM algorithm. Several algorithms have been proposed in the literature to solve

this issue. To overcome this problem, we propose a novel three-phase algorithm based on stability region analysis [24]. The main research concerns that motivated the new algorithm presented in this paper are:

- EM algorithm for mixture modeling converges to a local maximum of the likelihood function very quickly.
- There are many other promising local optimal solutions in the close vicinity of the solutions obtained from the methods that provide good initial guesses of the solution.
- Model selection criteria usually assumes that the global optimal solution of the log-likelihood function can be obtained. However, achieving this is computationally intractable.
- Some regions in the search space do not contain any promising solutions. The promising and non-promising regions co-exist and it becomes challenging to avoid wasting computational resources to search in non-promising regions.

Of all the concerns mentioned above, the fact that most of the local maxima are not distributed uniformly [35] makes it important for us to develop algorithms that not only help us to avoid searching in the low-likelihood regions but also emphasize the importance of exploring promising subspaces more thoroughly. This subspace search will also be useful for making the solution less sensitive to the initial set of parameters. In this paper, we propose a novel stability region based algorithm for estimating the parameters of mixture models. Using concepts from the dynamical systems literature and the EM algorithm simultaneously to exploit the problem specific features of mixture models, our algorithm tries to obtain the optimal set of parameters by systematically searching multiple local optimal solutions on the likelihood surface.

The rest of this paper is organized as follows: Section II gives some relevant background about various methods proposed in the literature for solving the problem of learning mixture models. Section III discusses some preliminaries about mixture models, EM algorithm and stability regions. Section IV describes our new framework along with the details of our implementation and the complexity. Section V shows the experimental results of our algorithm on synthetic and real datasets. Finally, Section VI concludes the work and discusses future research directions.

II. Relevant Background

Although EM and its variants have been extensively used for learning mixture models, several researchers have approached the problem by identifying new techniques that give good initialization. More generic techniques like deterministic annealing [31], [35], genetic algorithms [23], [17] have been applied

to obtain a good set of parameters. Though, these techniques have asymptotic guarantees, they are very time consuming and hence cannot be used in most practical applications. Some problem specific algorithms like split and merge EM [36], component-wise EM [9], greedy learning [37], incremental version for sparse representations[21], parameter space grid [15] have also been proposed in the literature. Some of these algorithms are either computationally very expensive or infeasible when learning mixture models in high dimensional spaces [15]. In spite of the high computational cost associated with these methods, very little effort has been taken to explore promising subspaces within the larger parameter space. Most of these algorithms eventually apply the EM algorithm to move to a locally maximal set of parameters on the likelihood surface. Simpler practical approaches like running EM from several random initializations, and then choosing the final estimate that leads to the local maximum with the highest value of the likelihood have also been successful to a certain extent [12], [30].

Though some of these methods apply other additional mechanisms (like perturbations [8]) to escape out of the local optimal solutions, systematic methods are yet to be developed for searching the subspace. The dynamical system of the log-likelihood function reveals more information on the neighborhood stability regions and their corresponding local maxima [6]. Hence, the difficulties of finding good solutions when the error surface is very rugged can be overcome by adding stability region based mechanisms to escape out of the convergence zone of the local maxima. Though this method might introduce some additional cost, one has to realize that existing approaches are much more expensive due to their stochastic nature. There appears to be a minor similarity between our method and some of the approaches available in the literature ([36], [9], [37]), in terms of moving to different basins of attraction in the neighborhood of the current local maximum. In fact, some of these methods ([36], [37]) change the number of components in this process of searching for different regions. The main distinction of our work is the use of the theory of stability regions to develop an efficient computational method that can search for neighborhood local maxima.

For a problem of this nature, where there is a non-uniform distribution of local maxima, it is difficult for most of the methods to search neighboring regions [40]. For this reason, it is more desirable to apply TRUST-TECH based Expectation Maximization (TT-EM) algorithm after obtaining some point in a promising region. The main advantages of the proposed algorithm are that it:

- Explores most of the neighborhood local optimal solutions unlike traditional stochastic algorithms.
- Acts as a flexible interface between the EM algorithm and other global methods.
- Allows the user to work with existing clusters obtained from other standard approaches and improves

the quality of the solutions based on the maximum likelihood criteria.

- Helps in truncating some of the expensive global methods during their early stages.
- Exploits the fact that promising solutions are obtained by faster convergence of the EM algorithm.

III. PRELIMINARIES

We now introduce some necessary preliminaries on mixture models, EM algorithm and stability regions. First, we describe the notation used in the rest of the paper:

TABLE I

DESCRIPTION OF THE NOTATIONS USED IN THE PAPER

Notation	Description
d	number of features
n	number of data points
k	number of components
s	total number of parameters
Θ	parameter set
θ_i	parameters of i^{th} component
α_i	mixing weights for i^{th} component
$\mathcal{X}$	observed data
$\mathcal{Z}$	missing data
$\mathcal{Y}$	complete data
t	timestep for the estimates

A. Mixture Models

Let us assume that there are k Gaussians in the mixture model. The form of the probability density function is as follows:

$$p(\boldsymbol{x}|\Theta) = \sum_{i=1}^{k} \alpha_i p(\boldsymbol{x}|\boldsymbol{\theta}_i) \tag{1}$$

where $\boldsymbol{x} = [x_1, x_2, ..., x_d]^T$ is the feature vector of d dimensions. Though out this paper, bold notation indicates that it is a vector quantity unless otherwise explicitly stated. The α_i's represent the *mixing weights*. The symbol Θ represents the parameter set $(\alpha_1, \alpha_2, ...\alpha_k, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, .., \boldsymbol{\theta}_k)$ and p is a d-variate Gaussian density parameterized by $\boldsymbol{\theta}_i$ (i.e. $\boldsymbol{\mu}_i$ and Σ_i):

$$p(\boldsymbol{x}|\boldsymbol{\theta}_i) = \frac{|\Sigma_i|^{-\frac{1}{2}}}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_i)} \tag{2}$$

Also, it should be noticed that being probabilities α_i must satisfy

$$0 \leq \alpha_i \leq 1 \ , \ \forall i = 1, .., k, \ and \ \ \sum_{i=1}^{k} \alpha_i = 1 \tag{3}$$

Given a set of n i.i.d samples $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, .., \boldsymbol{x}^{(n)}\}$, the log-likelihood corresponding to a mixture is

$$\begin{aligned} log \ p(\mathcal{X}|\Theta) &= log \prod_{j=1}^{n} \ p(\boldsymbol{x}^{(j)}|\Theta) \\ &= \sum_{j=1}^{n} log \sum_{i=1}^{k} \alpha_i \ p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta}_i) \end{aligned} \tag{4}$$

The goal of learning mixture models is to obtain the parameters $\widehat{\Theta}$ from a set of n data points which are samples from a distribution with density given by (1). The *Maximum Likelihood Estimate* (MLE) is given by :

$$\widehat{\Theta}_{MLE} = arg \max_{\tilde{\Theta}} \ \{ \ log \ p(\mathcal{X}|\Theta) \ \} \tag{5}$$

where $\tilde{\Theta}$ indicates the entire parameter space. Since, this MLE cannot be found analytically for mixture models, one has to rely on iterative procedures that can find the global maximum of $log \ p(\mathcal{X}|\Theta)$. The EM algorithm described in the next section has been used successfully to find the local maximum of such a function [18].

B. Expectation Maximization

The EM algorithm considers $\mathcal{X}$ to be $observed$ data. The missing part, termed as $hidden$ data, is a set of *n* labels $\mathcal{Z} = \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, .., \mathbf{z}^{(n)}\}$ associated with n samples, indicating that component that produced each data point [18]. Each label $\mathbf{z}^{(j)} = [z_1^{(j)}, z_2^{(j)}, .., z_k^{(j)}]$ is a binary vector where $z_i^{(j)} = 1$ and $z_m^{(j)} = 0$ $\forall m \neq i$, means the sample $\boldsymbol{x}^{(j)}$ was produced by the i^{th} component. Now, the complete log-likelihood (i.e. the one from which we would estimate Θ if the *complete data* $\mathcal{Y} = \ \{ \ \mathcal{X}, \mathcal{Z} \ \}$ is

$$log \ p(\mathcal{X}, \mathcal{Z}|\Theta) = \sum_{j=1}^{n} \ log \prod_{i=1}^{k} \ [\ \alpha_i \ p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta}_i) \ ]^{z_i^{(j)}} \tag{6}$$

$$log \ p(\mathcal{Y}|\Theta) = \sum_{j=1}^{n} \sum_{i=1}^{k} z_i^{(j)} \ log \ [\ \alpha_i \ p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta}_i) \ ] \tag{7}$$

The EM algorithm produces a sequence of estimates $\{\widehat{\Theta}(t), t = 0, 1, 2, ...\}$ by alternately applying the following two steps until convergence:

- **E-Step :** Compute the conditional expectation of the hidden data, given $\mathcal{X}$ and the current estimate $\widehat{\Theta}(t)$. Since $log\ p(\mathcal{X}, \mathcal{Z}|\Theta)$ is linear with respect to the missing data $\mathcal{Z}$, we simply have to compute the conditional expectation $\mathcal{W} \equiv E[\mathcal{Z}|\mathcal{X}, \widehat{\Theta}(t)]$, and plug it into $log\ p(\mathcal{X}, \mathcal{Z}|\Theta)$. This gives the Q-function as follows:

$$Q(\Theta|\widehat{\Theta}(t)) \equiv E_Z[log\ p(\mathcal{X}, \mathcal{Z})|\mathcal{X}, \widehat{\Theta}(t)] \tag{8}$$

Since $\mathcal{Z}$ is a binary vector, its conditional expectation is given by :

$$\begin{aligned} w_i^{(j)} &\equiv E\ [\ z_i^{(j)}|\mathcal{X}, \widehat{\Theta}(t)\ ] \\ &= Pr\ [\ z_i^{(j)} = 1|\mathbf{x}^{(j)}, \widehat{\Theta}(t)\ ] \\ &= \frac{\widehat{\alpha}_i(t)p(\mathbf{x}^{(j)}|\widehat{\theta}_i(t))}{\sum_{i=1}^{k}\widehat{\alpha}_i(t)p(\mathbf{x}^{(j)}|\widehat{\theta}_i(t))} \end{aligned} \tag{9}$$

where the last equality follows from Bayes law (α_i is the a priori probability that $z_i^{(j)} = 1$), while $w_i^{(j)}$ is the a posteriori probability that $z_i^{(j)} = 1$ given the observation $\boldsymbol{x}^{(j)}$.

- **M-Step :** The estimates of the new parameters are updated using the following equation :

$$\widehat{\Theta}(t+1) = arg\max_{\Theta}\{Q(\Theta, \widehat{\Theta}(t))\} \tag{10}$$

C. EM for GMMs

Several variants of the EM algorithm have been extensively used to solve this problem. The convergence properties of the EM algorithm for Gaussian mixtures are thoroughly discussed in [39]. The $Q - function$ for GMM is given by:

$$\begin{aligned} Q(\Theta|\widehat{\Theta}(t)) = \sum_{j=1}^{n}\sum_{i=1}^{k} w_i^{(j)}[log\frac{|\Sigma|^{-\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}} \\ - \frac{1}{2}(\boldsymbol{x}^{(j)} - \boldsymbol{\mu}_i)^T\Sigma^{-1}(\boldsymbol{x}^{(j)} - \boldsymbol{\mu}_i) + log\ \alpha_i] \end{aligned} \tag{11}$$

where

$$w_i^{(j)} = \frac{\widehat{\alpha}_i(t)|\widehat{\Sigma}_i(t)|^{-\frac{1}{2}}e^{-\frac{1}{2}(\mathbf{x^{(j)}} - \widehat{\boldsymbol{\mu}_i}(\mathbf{t}))^{\mathbf{T}}|\widehat{\boldsymbol{\Sigma}_i}(\mathbf{t})|^{-\mathbf{1}}(\mathbf{x^{(j)}} - \widehat{\boldsymbol{\mu}_i}(\mathbf{t}))}}{\sum_{i=1}^{k}\widehat{\alpha}_i(t)|\widehat{\Sigma}_i(t)|^{-\frac{1}{2}}e^{-\frac{1}{2}(\boldsymbol{x}^{(j)} - \widehat{\boldsymbol{\mu}_i}(t))^T|\widehat{\Sigma}_i(t)|^{-1}(\boldsymbol{x}^{(j)} - \widehat{\boldsymbol{\mu}_i}(t))}} \tag{12}$$

The maximization step is given by the following equation :

$$\frac{\partial}{\partial \Theta_k} Q(\Theta|\widehat{\Theta}(t)) = 0 \tag{13}$$

where Θ_k is the parameters for the k^{th} component. Because the posterior probabilities in the E-step now appear in the Q-function as given constants, and therefore resembling the Gaussian likelihood when the components are pre-specified, maximizing this function in the M-step becomes trivial. The updates for the maximization step in the case of GMMs are given as follows:

$$\begin{aligned}
\boldsymbol{\mu}_i(t+1) &= \frac{\sum_{j=1}^{n} w_i^{(j)} \boldsymbol{x}^{(j)}}{\sum_{j=1}^{n} w_i^{(j)}} \\
\Sigma_i(t+1) &= \frac{\sum_{j=1}^{n} w_i^{(j)} (\boldsymbol{x}^{(j)} - \boldsymbol{\mu}_i(t+1))(\boldsymbol{x}^{(j)} - \boldsymbol{\mu}_i(t+1))^T}{\sum_{j=1}^{n} w_i^{(j)}} \\
\alpha_i(t+1) &= \frac{1}{n} \sum_{j=1}^{n} w_i^{(j)}
\end{aligned} \tag{14}$$

D. Stability Regions

This section mainly deals with the transformation of the original log-likelihood function into its corresponding nonlinear dynamical system and introduces some terminology pertinent to comprehend our algorithm. Let us denote the number of unknown parameters as s. This transformation gives the correspondence between all the critical points of the s-dimensional likelihood surface and that of its dynamical system. For the case of a simple spherical Gaussian mixture with k components, we have $s = 3k - 1$. It should be noted that only (k-1) α values are considered in the gradient system because of the unity constraint. The dependent variable α_k is written as follows:

$$\alpha_k = 1 - \sum_{j=1}^{k-1} \alpha_j \tag{15}$$

For the case of a general full covariance d-variate Gaussian mixture with k components, contribution to s stems from the kd parameters through the component means $\boldsymbol{\mu}_i$, $\frac{1}{2}kd(d+1)$ parameters through the k covariances Σ_i and the $k-1$ mixing weights α_i. Altogether s is therefore given by $s = kd + \frac{1}{2}kd(d+1) + (k-1)$=$k\left[\frac{1}{2}(d+1)(d+2)\right] - 1$. For convenience, the maximization problem is transformed into a minimization problem as follows :

$$\max_{\Theta} \{ \, log \; p(\mathcal{X}|\Theta) \, \} = \min_{\Theta} \{ \, - \, log \; p(\mathcal{X}|\Theta) \, \} = \min_{\Theta} f(\Theta) \tag{16}$$

In our construction of the dynamical system corresponding to the s-dimensional likelihood surface we require that $f(\Theta)$ be twice continuously differentiable. The following simple lemma asserts this result.

Lemma 1: $f(\Theta)$ is $C^2(\Re^s, \Re)$.

Proof:

Note from Eq.(4), we have

$$\begin{aligned} f(\Theta) = -log\ p(\mathcal{X}|\Theta) &= -\sum_{j=1}^{n} log \sum_{i=1}^{k} \alpha_i\ p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta}_i) \\ &= -\sum_{j=1}^{n} log \sum_{i=1}^{k} \alpha_i\ \frac{|\Sigma_i|^{-\frac{1}{2}}}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)} \end{aligned} \tag{17}$$

Each of the simple functions which appear in Eq.(17) are twice differentiable and continuous in the interior of the domain over which $f(\Theta)$ is defined (in fact, they are infinitely differentiable). The function $f(\Theta)$ is composed of arithmetic operations of these simple functions and from basic results in analysis, we can conclude that $f(\Theta)$ is twice continuously differentiable. ■

Definition 1: $\bar{\Theta}$ is said to be a *critical point* of (16) if it satisfies the following condition

$$\nabla f(\bar{\Theta}) = 0 \tag{18}$$

A critical point is said to be *nondegenerate* if at the critical point $\bar{\Theta} \in \Re^s$, $\boldsymbol{d}^T\nabla^2 f(\bar{\Theta})\boldsymbol{d} \neq 0\ \ (\forall \boldsymbol{d} \neq 0)$. We construct the following *gradient system* in order to locate critical points of the objective function (16):

$$\dot{\Theta}(t) = \ F(\Theta) = \ \ -\nabla f(\Theta) \tag{19}$$

where the state vector Θ belongs to the Euclidean space $\Re^s$, and the vector field $F : \Re^s \rightarrow \Re^s$ satisfies the sufficient condition for the existence and uniqueness of the solutions. The solution curve of Eq. (19) starting from Θ at time $t = 0$ is called a *trajectory* and it is denoted by $\Phi(\Theta, \cdot) : \Re \rightarrow \Re^s$. A state vector Θ is called an *equilibrium point* of Eq. (19) if $F(\Theta) = 0$. An equilibrium point is said to be *hyperbolic* if the Jacobian of F at point $\bar{\Theta}$ has no eigenvalues with zero real part.

Lemma 1 and the preceeding arguments guarantee the existence of the gradient system associated with $f(\Theta)$ and allows us to construct the following dynamical system:

$$\begin{aligned} \dot{\Theta}(t) &= [\dot{\boldsymbol{\mu}}_1(t)\ ..\ \dot{\boldsymbol{\mu}}_k(t)\ \dot{\sigma}_1(t)\ ..\ \dot{\sigma}_k(t)\ \dot{\alpha}_1(t)\ ..\ \dot{\alpha}_{k-1}(t)]^T = -\nabla f(\Theta) \\ &= -\left[\frac{\partial f}{\partial \boldsymbol{\mu}_1}\ ..\ \frac{\partial f}{\partial \boldsymbol{\mu}_k}\ \frac{\partial f}{\partial \sigma_{111}}\ ..\ \frac{\partial f}{\partial \sigma_{kd(d+1)/2}}\ \frac{\partial f}{\partial \alpha_1}\ ..\ \frac{\partial f}{\partial \alpha_{k-1}}\right]^T \end{aligned} \tag{20}$$

From the implementation point of view, it is not required to define this gradient system. However, to understand the details of our method and its theoretical foundations, it is necessary to obtain this gradient

system. This will enable us to define concepts like stable equilibrium point, stability region and stability boundary, all of which will be described here briefly. In this paper, we introduce a novel computational technique which uses these concepts and build a few approximations to make this method work well in practice.

Definition 2: A hyperbolic equilibrium point is called a (asymptotically) *stable equilibrium point* (SEP) if all the eigenvalues of its corresponding Jacobian have negative real part. Conversely, it is an *unstable equilibrium point* if some eigenvalues have a positive real part.

An equilibrium point is called a *type-k equilibrium point* if its corresponding Jacobian has exact k eigenvalues with positive real part. The *stable* ($W^s(\tilde{x})$) and *unstable* ($W^u(\tilde{x})$) manifolds of an equilibrium point, say $\tilde{x}$, is defined as:

$$W^s(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\to\infty} \Phi(x,t) = \tilde{x}\} \tag{21}$$

$$W^u(\tilde{x}) = \{x \in \Re^s \ : \ \lim_{t\to-\infty} \Phi(x,t) = \tilde{x}\} \tag{22}$$

The task of finding multiple local maxima on the log-likelihood surface is transformed into the task of finding multiple stable equilibrium points on its corresponding gradient system. The advantage of our approach is that this transformation into the corresponding dynamical system will yield more knowledge about the various dynamic and geometric characteristics of the original surface and leads to the development a powerful method for finding improved solutions. In this paper, we are particularly interested in the properties of the local maxima and their one-to-one correspondence to the stable equilibrium points. To comprehend the transformation, we need to define the concept of an *energy function*. A smooth function $V(\cdot) : \Re^s \rightarrow \Re$ satisfying $\dot{V}(\Phi(\Theta,t)) \ < \ \mathbf{0}$, [a vector of zeros] , $\forall \ x \notin$ {set of equilibrium points (E)} and t $\in \Re^+$ is termed as the energy function.

Theorem 3.1: [6]: $f(\Theta)$ is a energy function for the gradient system (20).

Definition 3: A type-1 equilibrium point x_d (k=1) on the practical stability boundary of a stable equilibrium point x_s is called a *decomposition point*.

Our approach takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibria CHaracterization) to compute neighborhood local maxima on likelihood surface using stability regions. Originally, the basic idea of our algorithm was to find decomposition points on the practical stability boundary. Since, each decomposition point connects two local maxima uniquely, it is important to obtain the saddle points from the given local maximum and then move to the next local maximum through

this decomposition point [25]. Though, this procedure gives a guarantee that the local maximum is not revisited, the computational expense for tracing the stability boundary and identifying the decomposition point is high compared to the cost of applying the EM algorithm directly using the exit point without considering the decomposition point. In a particular direction away from the local maximum, the point where the value of the energy function reaches the maximum is called an *exit point* (x_1, x_2 and x_3 in Fig. 4(a)). One can use the saddle point tracing procedure described in [25] for applications where the local methods like EM are more expensive.

Definition 4: Tier-1 local maximum is a neighborhood local maximum obtained by integrating the gradient system after a small perturbation along the eigenvector direction corresponding to the positive eigenvalue of the Jacobian at the decomposition point on the stability boundary of any given local maximum.

Definition 5: The *practical stability region* of a stable equilibrium point x_s of a nonlinear dynamical system (19), denoted by $A_p(x_s)$ and is the interior of closure of the stability region $A(x_s)$ which is given by :

$$A(x_s) = \{x \in \Re^s \ : \ \lim_{t\to\infty} \Phi(x,t) = x_s\} \tag{23}$$

The boundary of the practical stability region is called the *practical stability boundary* of x_s and will be denoted by $\partial A_p(x_s)$. Theorem 3.2 asserts that the practical stability boundary is contained in the union of the closure of the stable manifolds of all the decomposition points on the practical stability boundary. Hence, if the decomposition points can be identified, then an explicit characterization of the practical stability boundary can be established using (24). This theorem gives an explicit description of the geometrical and dynamical structure of the practical stability boundary.

Theorem 3.2: (Characterization of practical stability boundary)[14]: Consider a negative gradient system described by (19). Let $\gamma_i \quad i = 1, 2, ...$ be the decomposition points on the practical stability boundary $\partial A_p(x_s)$ of a stable equilibrium point, say x_s. Then

$$\partial A_p(x_s) = \bigcup_{\gamma_i \in \partial A_p} \overline{W^s(\gamma_i)}. \tag{24}$$

Fig. 2 gives an example of a Tier-1 local maximum. The EM algorithm converges to a local maximum (LM_i). By tracing the stability boundary, one can find a decomposition point (γ_1) using the procedure described in [25]. The eigenvector direction corresponding to the positive eigenvalue of the Jacobian evaluated at γ_1 will connect two local maxima. Hence, a new local maximum (new_j) can be obtained by following the gradient after perturbing in that eigenvector direction towards the opposite side of the existing local maximum (LM_i).

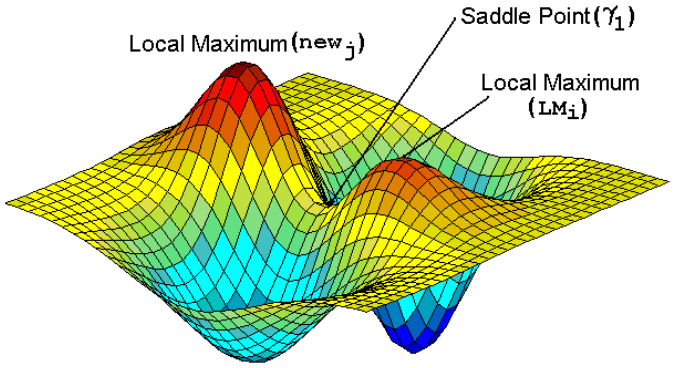


Fig. 2. Demonstration of Tier-1 local maximum. The original local maximum is LM_i. γ_1 is a decomposition point on the stability boundary of LM_i. Another local maximum (new_j) can be obtained through the saddle point.

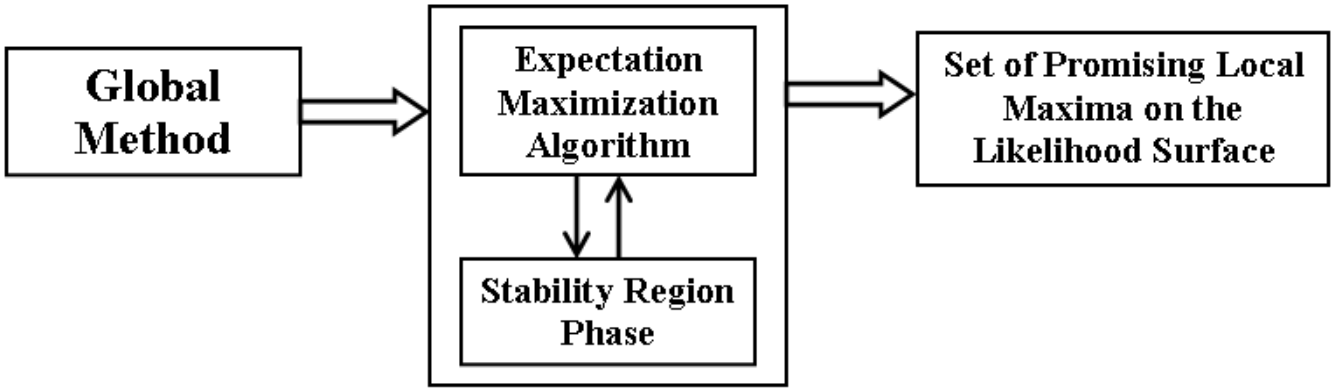


Fig. 3. Block diagram of our algorithm. The global method gives some initial promising subspaces. The EM algorithm along with the stability region phase can obtain a set of promising neighborhood local maxima on the likelihood surface.

E. Model Selection Criterion

Another important issue in mixture modeling is the selection of the number of components which is usually unknown in real-world problems. A mixture with too many components will over-fit the data, while a mixture with too few components will be too simple to approximate the underlying distribution. Most of the methods available in the literature for model selection add a penalty term in the objective

function that can potentially penalize more complex models. A detailed review of different criteria used to penalize complex models is given in [19]. Bayesian approaches are also successful in identifying the number of components [30]. The methods of model selection can be divided into two families. One is based on a random sampling mechanism, e.g. Markov Chain Monte Carlo (MCMC) methods [11], [28], [29]. The main disadvantage of these methods is the computational expense in attaining the global optimal solution. The other one is based on some deterministic criteria similar to the minimum message length (MML) criterion [9]. A review of cross validation schemes for mixture modeling can be seen in [33].

We will be able to incorporate any model selection criterion in our method by merely modifying the likelihood function. The proposed method works in the same way as before, except that the likelihood function will have some additional terms. Our method deals with the nonlinearity of the likelihood surface directly and hence it is independent of the number of components chosen in the model. In fact, the improvements in the likelihood will be bigger in this case because of the additional terms that create more nonlinearity of the surface. We would like to emphasize that the main focus of our work is to reduce or solve the initialization problem associated with the learning of mixture models and not the automatic selection of the number of components. However, we justify that using our methodology, we can exploit the maximum potential of the given number of mixture components by trying to find better estimates of the likelihood. In other words, solving the initialization issue will indirectly impact the choice of the number of components.

IV. TRUST-TECH BASED ALGORITHM

Our framework consists of three phases. The first phase is the global phase, in which the promising solutions in the entire search space are obtained. The second phase is the local phase (or the EM phase), where the promising solutions obtained from the previous phase are refined to the corresponding locally optimal parameter set. The third phase, which is the main contribution of this paper, is the stability region phase. The exit points are computed and the neighborhood solutions are systematically explored through these exit points in this phase. Fig. 3 shows the block diagram of our algorithm. The EM phase and the stability region phase are repeated alternatively in the promising regions of the parameter search space.

This approach can be treated as a hybrid between global methods for initialization and the EM algorithm which gives the local maxima. One of the main advantages of our approach is that it searches the parameter space more systematically. This approach differs from traditional local methods by computing multiple local solutions in the neighborhood region. This also enhances user flexibility by allowing the users to

choose between different sets of good clusterings. Though global methods give promising subspaces, it is important to explore this subspace more thoroughly especially in problems like this one. Algorithm 1 describes our approach.

Algorithm 1 TT_EM Algorithm

Input: Parameters Θ, Data $\mathcal{X}$, tolerance τ, Step S_p

Output: $\widehat{\Theta}_{MLE}$

Algorithm:

Apply global method and store the q promising solutions $\Theta_{init} = \{\Theta_1, \Theta_2, .., \Theta_q\}$

Initialize E= ϕ

while $\Theta_{init} \neq \phi$ **do**

Choose $\Theta_i \in \Theta_{init}$, set $\Theta_{init} = \Theta_{init} \backslash \{\Theta_i\}$

$LM_i = EM(\Theta_i, \mathcal{X}, \tau)$ $\quad E = E \cup \{LM_i\}$

Generate promising direction vectors d_j from LM_i

for each d_j **do**

Compute Exit Point (X_j) along d_j starting from LM_i by evaluating the log-likelihood function given by (4)

$New_j = EM(X_j + \epsilon \cdot d_j, \mathcal{X}, \tau)$

if $new_j \notin E$ **then**

$E = E \cup New_j$

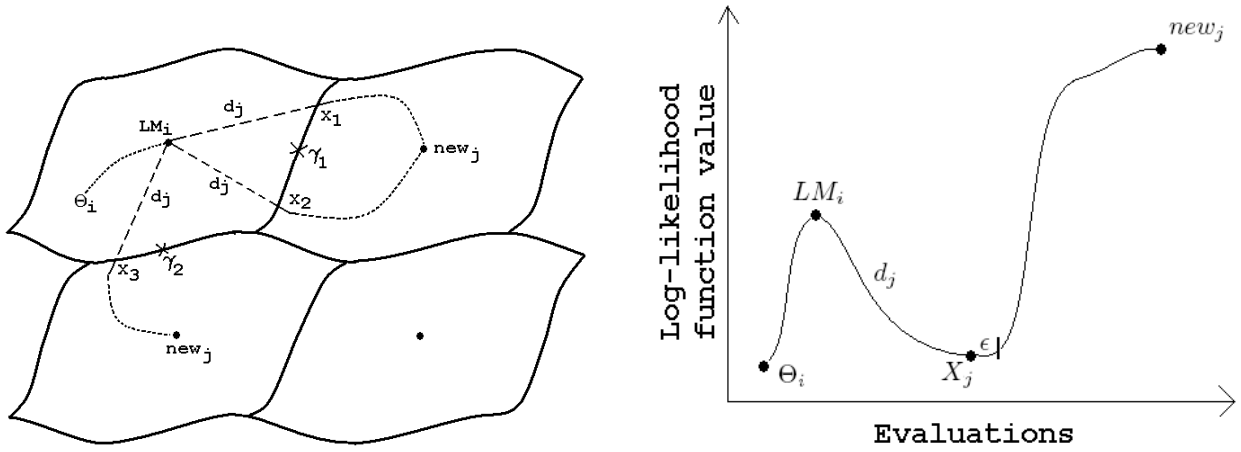
end if

end for

end while

$\widehat{\Theta}_{MLE} = max\{val(E_i)\}$

Fig. 4 shows the different steps of our algorithm both in (a) the parameter space and (b) the function space. In order to escape out of this local maximum, our method needs to compute certain promising directions based on the local behavior of the function. One can realize that generating these promising directions is one of the important aspects of our algorithm. Surprisingly, choosing random directions to move out of the local maximum works well for this problem. One might also use other directions like eigenvectors of the Hessian or incorporate some domain-specific knowledge (like information about priors, approximate location of cluster means, user preferences on the final clusters etc..) depending on the



(a) Parameter Space (b) Function Space

Fig. 4. Various stages of our algorithm in (a) Parameter space - the solid lines indicate the practical stability boundary. Points highlighted on the stability boundary (γ_1, γ_2) are the decomposition points. The dotted lines indicate the convergence of the EM algorithm. The promising directions, labeled as d_j, are generated at the local maximum LM_i. The dashed lines indicate the stability region phase. The points x_1, x_2 and x_3 are the exit points on the practical stability boundary (b) Different variables in the function space and their corresponding log-likelihood values.

application that they are working on and the level of computational expense that they can afford. We used random directions in our work because they are very cheap to compute. Once the promising directions are generated, exit points are computed along these directions. *Exit points* are points of intersection between any given direction and the practical stability boundary of that local maximum along that particular direction. If the stability boundary is not encountered along a given direction, it is very likely that one might not find any new local maximum in that direction. With a new initial guess in the vicinity of the exit points, EM algorithm is applied again to obtain a new local maximum.

A. Implementation Details

Our program is implemented in MATLAB and runs on a Pentium IV 2.8 GHz machine. The main procedure implemented is TT_EM described in Algorithm 2. The algorithm takes the mixture data and the initial set of parameters as input, along with step size for moving out, and tolerance for convergence of the EM algorithm. It returns the set of parameters that correspond to the $Tier-1$ neighboring local optimal solutions. The procedure $eval$ returns the log-likelihood score given by (4). The Gen_Dir procedure generates promising directions from the local maxima. Exit points are obtained along these generated directions. The procedure $update$ moves the current parameter to the next parameter set along a given k^{th} direction $Dir[k]$. Some of the directions might have one of the following two problems: (i) Exit points might not be obtained in these directions. (ii) Even if the exit point is obtained, it might converge to a

less promising solution. If the exit points are not found along these directions, search will be terminated after $Eval\_MAX$ number of evaluations. For all exit points that are successfully found, EM procedure is applied and all the corresponding neighborhood set of parameters are stored in the $Params[\ ]$. To ensure that the new initial points are in different stability regions, one should move along the directions 'ϵ' away from the exit points. Because the parameters will be of different ranges, care must be taken while computing the step sizes. It is important to use the current estimates to get an approximation of the step size with which one should move out along each parameter in the search space. Finally, the solution with the highest log-likelihood score amongst the original set of parameters and the Tier-1 solutions is returned.

B. Complexity of the Algorithm

We will now analyze the time complexity of the TRUST-TECH-EM algorithm presented in the previous section. An extensive discussion on the rate of convergence of the EM algorithm is available in [7]. This rate depends on the proportion of information available in the observed data. Consider the case of a full covariance d-variate Gaussian mixture model with k components and n data points. Let us summarize the cost of each EM step. The E-step in Eq.(9) requires operations of order $O(nk\lambda)$ where λ is the cost of computing the density of a multivariate normal at a given point $p(\boldsymbol{x}^{(j)}|\boldsymbol{\theta}_i)$. The M-step from Eq.(14) requires operations of order $O(nkd)$ due to the calculation of the component means, of order $O(nkd^2)$ from calculations of the component covariance matrices and of order $O(k)$ from calculation of the mixing weights. The computational cost of the EM step for each iteration is therefore of the order $O(nk\lambda+nkd^2)$. For higher dimensional problems, λ is negligible compared to d^2 and $d^2 \approx n$. Hence the cost of EM will be $O(n^2)$ which is quadratic in n.

We will now quantify the additional cost incurred due to the stability region phase of the Trust-Tech method. This phase allows us to escape out of the local maximum and it basically requires the evaluation of the log-likelihood function at several points along a particular direction. The evaluation of the log-likelihood function value during the stability region phase requires $nk\lambda+2nk+2n$ operations (see Eq.(4)). Assuming there are a maximum of c number of function evaluations ($Eval\_MAX$) along a particular direction, the total complexity of the stability region phase will be $O(2ckn)$ (since $k << n$). This is linear in terms of n. Hence, the stability region phase requires a negligible additional cost compared to the EM phase. Finally, the overall cost of the TRUST-TECH method will be equal to $O(bn^2)$ where b is the number of directions chosen. This quantity is the same as the cost of using b random starts and

Algorithm 2 Params[ ] $TT\_EM(Pset, Data, Tol, Step)$

$Val = eval(Pset)$

$Dir[\,] = Gen\_Dir(Pset)$

$Eval\_MAX = 500$

for $k = 1$ to $size(Dir)$ **do**

$Params[k] = Pset \qquad ExtPt = OFF$

$Prev\_Val = Val \qquad Cnt = 0$

while (! $ExtPt$) && ($Cnt < Eval\_MAX$) **do**

$Params[k] = update(Params[k], Dir[k], Step)$

$Cnt = Cnt + 1$

$Next\_Val = eval(Params[k])$

if ($Next\_Val > Prev\_Val$) **then**

$ExtPt = ON$

end if

$Prev\_Val = Next\_Val$

end while

if $count < Eval\_MAX$ **then**

$Params[k] = update(Params[k], Dir[k], ASC)$

$Params[k] = EM(Params[k], Data, Tol)$

else

$Params[k] = NULL$

end if

end for

Return $max(eval(Params[\,]))$

applying EM algorithm for each of the starts.

V. Results and Discussion

Our algorithm has been tested on both synthetic and real datasets. The initial values for the centers and the covariances were chosen to be uniformly random. Uniform priors were chosen for initializing the components. For real datasets, the centers were chosen randomly from the sample points.

A. Results on Synthetic and Real-World Datasets

A simple synthetic data with 40 samples and 5 spherical Gaussian components was generated and tested with our algorithm. Priors were uniform and the standard deviation was 0.01. The centers for the five components are given as follows: $\mu_1 = [0.3\ 0.3]^T$, $\mu_2 = [0.5\ 0.5]^T$, $\mu_3 = [0.7\ 0.7]^T$, $\mu_4 = [0.3\ 0.7]^T$ and $\mu_5 = [0.7\ 0.3]^T$.

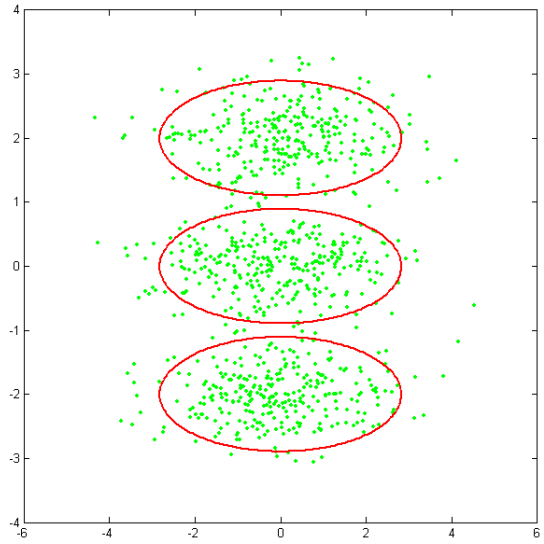
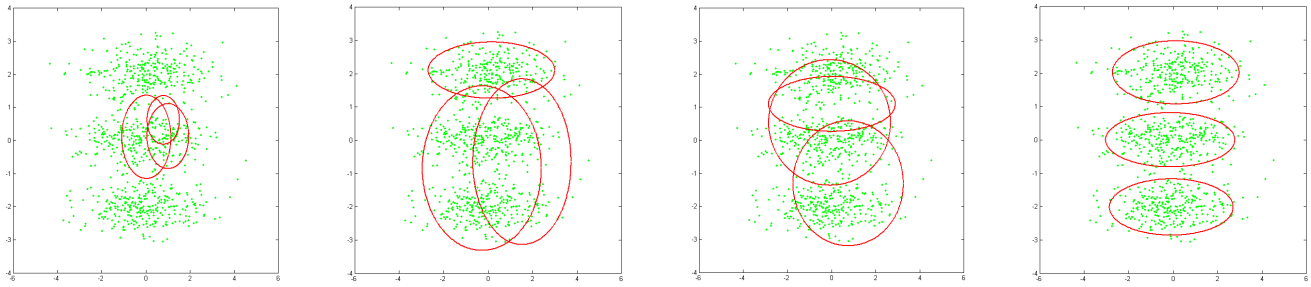


Fig. 5. True mixture of the three Gaussian components with 900 samples.



(a) (b) (c) (d)

Fig. 6. Parameter estimates at various stages of our algorithm on the three component Gaussian mixture model (a) Poor random initial guess (b) Local maximum obtained after applying EM algorithm with the poor initial guess (c) Exit point obtained by our algorithm (d) The final solution obtained by applying the EM algorithm to the initial point in the neighboring stability region.

The second dataset was that of a diagonal covariance case containing $n = 900$ data points. The data generated from a two-dimensional, three-component Gaussian mixture distribution with mean vectors at $[0\ \ -2]^T, [0\ 0]^T, [0\ 2]^T$ and same diagonal covariance matrix with values 2 and 0.2 along the diagonal [35]. All the three mixtures have uniform priors. The true mixtures with data generated from these three components are shown in Fig. 5. The data points appear to be in three well separated Gaussian clusters.

Fig. 6 shows various stages of our algorithm and demonstrates how the clusters obtained from existing algorithms are improved using our algorithm. The initial clusters obtained are of low quality because of the poor initial set of parameters. Our algorithm takes these clusters and applies the stability region step and the EM step simultaneously to obtain the final result. Fig. 7 shows the value of the log-likelihood during the stability region phase and the EM iterations.

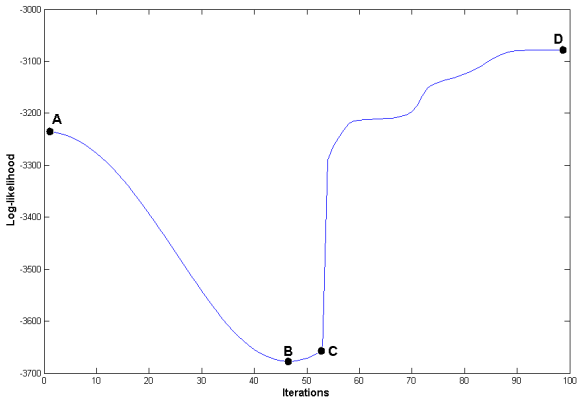


Fig. 7. Graph showing likelihood vs Evaluations. A corresponds to the original local maximum (L=-3235.0). B corresponds to the exit point (L=-3676.1). C corresponds to the new initial point in the neighboring stability region (L=-3657.3) after moving out by 'ϵ'. D corresponding to the new local maximum (L=-3078.7).

To demonstrate the performance of our algorithm on high dimensional problems, we extended the elliptical dataset to higher dimensions (50, 100 and 200). The centers were generated by simultaneously repeating the following 3-by-2 matrix - [0 rand; rand 0; 0 rand] and the covariance matrix is obtained by repeating the following 3-by-2 matrix - [200*rand 20*rand; 200*rand 20*rand; 200*rand 20*rand]. Here, rand indicates a uniform random variable between 0 and 1. To improve the running time, we used only 90 datapoints with uniform mixing weights. The number of search directions (=20) is a constant in all these three cases.

In the third synthetic dataset, a more complicated overlapping Gaussian mixtures are considered [9]. The parameters are as follows: $\mu_1 = \mu_2 = [-4\ -4]^T$, $\mu_3 = [2\ 2]^T$ and $\mu_4 = [-1\ -6]^T$. $\alpha_1 = \alpha_2 = \alpha_3 = 0.3$ and $\alpha_4 = 0.1$.

$$\Sigma_1 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \qquad \Sigma_2 = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \qquad \Sigma_4 = \begin{bmatrix} 0.125 & 0 \\ 0 & 0.125 \end{bmatrix}$$

Originally, there were 1000 samples generated and is shown in Fig. 8. However, we have conducted experiments to show the performance improvements of our algorithm by increasing the number of data points.

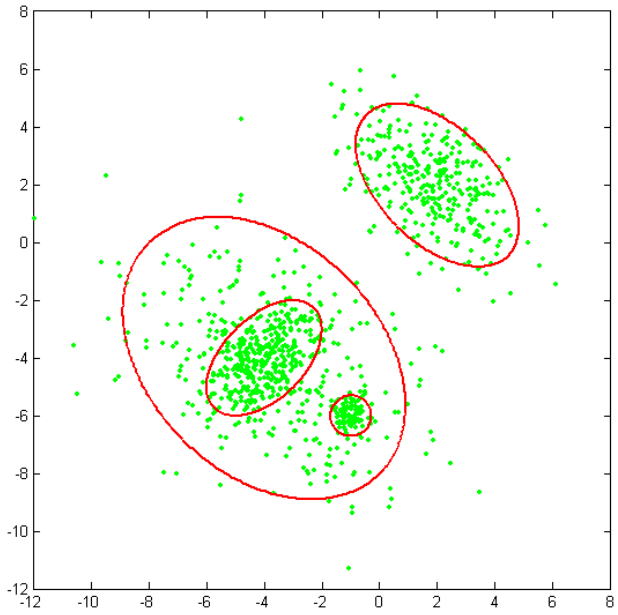


Fig. 8. True mixtures of the more complicated overlapping Gaussian case with 1000 samples. This dataset was used to show the improvements in the performance by varying the number of data points.

Two real datasets obtained from the UCI Machine Learning repository [4] were also used for testing the performance of our algorithm. The widely used Iris data with 150 samples, 3 classes and 4 features was used. Wine data set with 178 samples was also used for testing. Wine data had 3 classes and 13 features. For these real data sets, the class labels were deleted thus treating it as unsupervised learning problem. Table II summarizes our results over 100 runs. The mean and the standard deviations of the log-likelihood values are reported. The traditional EM algorithm with random starts is compared against our algorithm on both the synthetic and real data sets. Our algorithm not only obtains higher likelihood value but also produces it with higher confidence. The low standard deviation of our results indicates the robustness of obtaining higher likelihood values. In the case of the wine data, the improvements with our algorithm are not much more significant compared to the other datasets. This might be due to the fact that the dataset might not have Gaussian components. Our method assumes that the underlying distribution of the data is a mixture of Gaussians. Finally, the performance of our algorithm on high dimensional data is also shown. With a fixed number of search directions, the dimensionality of the data is increased and the improvements in the average MLE are shown.

TABLE II

PERFORMANCE OF OUR ALGORITHM ON AN AVERAGE OF 100 RUNS ON VARIOUS SYNTHETIC AND REAL DATASETS

Dataset	Samples	Clusters	Features	EM (mean ± std)	MRS+EM (mean ± std)	TRUST-TECH-EM (mean ± std)
Spherical	40	5	2	38.07±2.12	41.48 ± 0.93	43.55±0.6
Elliptical	900	3	2	-3235±0.34	-3128.84 ± 0.2	-3078.7±0.03
Elliptical-50	90	3	50	-13449±142	-13298 ± 19	-13264±12
Elliptical-100	90	3	100	-26843±89	-26603 ± 87	-26512±44
Elliptical-200	90	3	200	-53184±578	-52716 ± 14	-52609±29
Full covariance 1	500	4	2	-2345.5 ±175.13	-2135.73± 24.23	-2121.9± 21.16
Full covariance 2	2000	4	2	-9309.9 ±694.74	-8883.4± 132.56	-8609.7 ±37.02
Iris	150	3	4	-198.13±27.25	-179.45± 7.89	-173.63±11.72
Wine	178	3	13	-1652.7±1342.1	-1619.4± 1313.2	-1618.3±1349.9

Table III gives the results of TRUST-TECH compared with other methods proposed in the literature like split and merge EM and k-means+EM [41]. MRS+EM indicates the multiple random starts experiment which includes the same number of starts as the number of valid directions in the case of the TRUST-TECH method. The multiple starts are made only in the promising regions. RS+EM is just a single random start and is given here to illustrate the lower bound on the performance of our method empirically.

TABLE III

COMPARISON OF TRUST-TECH-EM WITH OTHER METHODS

Method	Elliptical	Iris
RS+EM	-3235 ± 14.2	-198 ± 27
MRS+EM	-3128.84 ± 0.2	-179.45± 7.89
K-Means+EM	-3195 ± 54	-186 ± 10
SMEM	-3123 ± 54	-178.5 ± 6
TRUST-TECH-EM	-3079 ± 0.03	-173.6 ± 11

B. Discussion

It will be effective to use our algorithm for those solutions that appear to be promising. Due to the nature of the problem, it is very likely that the nearby solutions surrounding the existing solution will be more promising. One of the primary advantages of our method is that it can be used along with other popular methods already available and thus improve the quality of existing solutions. In clustering problems, it is

an added advantage to perform refinement of the final clusters obtained. As shown in fig. 6, our algorithm can help in improving the quality of existing clusters. Even in such a simple case, the initial algorithm identified only one of the three clusters correctly. Our algorithm used the existing solution and identified three distinct clusters that are identical to the true mixtures. Most of the focus in the literature has been on new methods for initialization or new clustering techniques which often do not take advantage of the existing results and completely start the clustering procedure "*from scratch*". Though shown only for the case of multivariate Gaussian mixtures, our technique can be effectively applied to any parametric finite mixture model.

A closer examination of table II indicates that for the high-dimensional examples(Elliptical 50,100,200) the variance does not necessarily go up with increasing dimensionality. This might be due to the limited number of initial starts and the associated inherent inability to explore vast higher dimensional spaces with these relatively fewer number of starts. This could possibly lead to the same likelihood values from different initial starts and thus reducing the variance. A detailed investigation of this is beyond the scope of this paper but could shed further light on the properties of trust-tech-EM and other methods in high-dimensional examples.

Table IV summarizes the average number of iterations taken by the EM algorithm for convergence to the local optimal solution. We can see that the most promising solution produced by our TRUST-TECH methodology converges much faster. In other words, our method can effectively take advantage of the fact that the convergence of the EM algorithm is much faster for high quality solutions. This is an inherent property of the EM algorithm when applied to the mixture modeling problem. We exploit this property of the EM for improving the efficiency of our algorithm. Hence, for obtaining Tier-1 solutions using our algorithm, the threshold for the number of iterations can be significantly lowered.

TABLE IV

NUMBER OF ITERATIONS TAKEN FOR THE CONVERGENCE OF THE BEST SOLUTION.

Dataset	Avg. no. of iterations	No. of iterations for the best solution
Spherical	126	73
Elliptical	174	86
Full covariance	292	173

Using stability regions, we have a systematic approach to obtain neighboring local maxima as opposed to widely used stochastic methods (like data perturbation, annealing, mutations in genetic algorithms

etc.). These stochastic methods never guarantee the presence of a new stability region. Our algorithm not only guarantees that a new local maximum obtained is different from the original solution but also confirms that we will not miss a solution in any particular direction. Also, this algorithm provides the flexibility in choosing these directions and thus avoiding some standard problems of using EM algorithm like the boundary space problem. The standard EM algorithm might sometimes converge to the boundary of the parameter space. The boundary space problem (popularly known as the singularity problem) occurs when one of the unconstrained covariance matrices approaches zero. This can solved by using some soft constraints over the covariance matrices. In addition to that, we can take an extra step to generate directions that will not lead the parameters of these covariances matrices towards zero.

Our algorithm can be easily extended to the popularly used k-means clustering technique [20]. The proposed algorithm works not only for mixture models but also for more general nonlinear likelihood surfaces. The TRUST-TECH based EM algorithm has been successfully applied to a real-world example in bioinformatics, namely, the motif finding problem where EM is one of the popularly used algorithms for motif refinement [26].

VI. Conclusion and Future Work

A novel TRUST-TECH based EM algorithm has been introduced for estimating the parameters of mixture models. The EM phase and the stability region phase are applied alternatively in the context of the well-studied problem of learning mixture models. The concept of a stability region helps us to understand the topology of the original log-likelihood surface. Our method computes the neighborhood local maxima of likelihood surfaces using stability regions of the corresponding nonlinear dynamical system. The algorithm has been tested successfully on various synthetic and real-world datasets and the improvements in the performance are clearly manifested.

Our algorithm can be easily extended to the popularly used k-means clustering technique. In the future, we plan to work on applying these stability region based methods for other widely used EM related parameter estimation problems like training Hidden Markov Models [38], Mixtures of Factor Analyzers [10], Probabilistic Principal Component Analysis [34], Bayesian Networks [13] etc. We also plan to extend our technique to Markov Chain Monte Carlo strategies like Gibbs sampling for the estimation of mixture models. Constraints might be added based on some prior information about the samples and a TRUST-TECH based constrained EM algorithm can be developed [16].

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their useful comments and the associate editor for suggesting improvements in both the content and presentation.

REFERENCES

[1] J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49(3):803– 821, 1993.

[2] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[3] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, U. C. Berkeley, April 1998.

[4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. *http://www.ics.uci.edu/mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences*, 1998.

[5] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[6] H.D. Chiang and C.C. Chu. A systematic search method for obtaining multiple local optimal solutions of nonlinear programming problems. *IEEE Transactions on Circuits and Systems: I Fundamental Theory and Applications*, 43(2):99–109, 1996.

[7] A. P. Demspter, N. A. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[8] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. *In Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 132 – 139, 2002.

[9] M. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[10] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. *Technical Report*, CRG-TR-96-1:University of Toronto, 1996.

[11] P. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[12] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of the Royal Statistical Society series B*, 58:158–176, 1996.

[13] D. Heckerman. A tutorial on learning with bayesian networks. *Microsoft Research Technical Report*, MSR-TR-95-06, 1995.

[14] J. Lee and H.D. Chiang. A dynamical trajectory-based methodology for systematically computing multiple optimal solutions of general nonlinear programming problems. *IEEE Transactions on Automatic Control*, 49(6):888 – 899, 2004.

[15] J. Q. Li. *Estimation of Mixture Models*. PhD thesis, Department of Statistics,Yale University, 1999.

[16] Z. Lu and T. Leen. Semi-supervised learning with penalized probabilistic clustering. *proceedings of Neural Information Processing Systems*, 2005.

[17] A. M. Martnez and J. Vitri. Learning mixture models using a genetic version of the EM algorithm. *Pattern Recognition Letters*, 21(8):759–769, 2000.

[18] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley and Sons, New York, 1997.

[19] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley and Sons, 2000.

[20] G. J. McLachlan and K. E. Basford. *Mixture models: Inference and applications to clustering*. Marcel Dekker, New York, 1988.

[21] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

[22] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103 – 134, 2000.

[23] F. Pernkopf and D. Bouchaffra. Genetic-based EM algorithm for learning gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1344–1348, 2005.

[24] C. K. Reddy. *TRUST-TECH based Methods for Optimization and Learning*. PhD thesis, Cornell University, 2007.

[25] C. K. Reddy and H.D. Chiang. A stability boundary based method for finding saddle points on potential energy surfaces. *Journal of Computational Biology*, 13(3):745–766, 2006.

[26] C. K. Reddy, Y. C. Weng, and H. D. Chiang. Refining motifs by improving information content scores using neighborhood profile search. *BMC Algorithms for Molecular Biology*, 1(23):1–14, 2006.

[27] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195–239, 1984.

[28] S. Richardson and P. Green. On bayesian analysis of mixture models with unknown number of components. *J. Royal Statistical Soc., Series B*, 59(4):731–792, 1997.

[29] S. Roberts, C. Holmes, and D. Denison. Minimum-entropy data partitioning using reversible jump markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):909–914, 2001.

[30] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian approaches to gaussian mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1133 – 1142, 1998.

[31] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[32] R.H. Shumway and D.S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[33] P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing*, 10(1):63–72, 2002.

[34] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B*, 61(3):611–622, 1999.

[35] N. Ueda and R. Nakano. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282, 1998.

[36] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[37] J. J. Verbeek, N. Vlassis, and B. Krose. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.

[38] L. R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4), 2003.

[39] L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[40] B. Zhang, C. Zhang, and X. Yi. Competitive EM algorithm for finite mixture models. *Pattern Recognition*, 37(1):131–144, 2004.

[41] Z. Zivkovic and F. V. Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651– 656, 2004.