Stability Region based Methods for Learning and Discovery

Chandan K. Reddy and Hsiao-Dong Chiang

Department of Electrical and Computer Engineering
Cornell University, Ithaca, NY-14853.

Abstract

Many problems that arise in machine learning and data mining domains deal with nonlinearity and quite often demand users to obtain global optimal solutions rather than local optimal ones. Several algorithms had been proposed in the optimization literature and inherited by the machine learning community. Popularly known as the *initialization problem*, the ideal set of parameters required will significantly depend on the initial values given by the user. In this paper, we propose stability region based methods for systematically exploring the subspace of the parameters to obtain the neighborhood local optimal solutions. The proposed algorithm takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibria CHaracterization) to compute neighborhood local optimal solutions on the nonlinear surface in a systematic manner using stability regions. Our method explores the dynamic and geometric characteristics of stability boundaries of a nonlinear dynamical system corresponding to the nonlinear function of interest. Basically, our method coalesces the advantages of the traditional local optimizers 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 local phase and the stability region phase, are repeated alternatively in the parameter space to achieve improvements in the quality of the solutions. The local phase obtains the local maximum of the nonlinear function and the stability region phase helps to escape out of the local maximum by moving towards the neighboring stability regions. The stability region based algorithms are applied to three important machine learning problems in: (1) Unsupervised learning - model-based clustering, (2) Pattern discovery - motif finding problem and (3) Supervised learning - training artificial neural networks. Our algorithms were tested on both synthetic and real datasets and the advantages of using this stability region based framework are clearly manifested. This framework not only reduces the sensitivity to initialization, but also allows the flexibility for the practitioners to use various global and local methods that work well for a particular problem of interest.

Key words: stability regions, model-based clustering, neural networks, pattern discovery, dynamical systems, global optimization.

1 Introduction

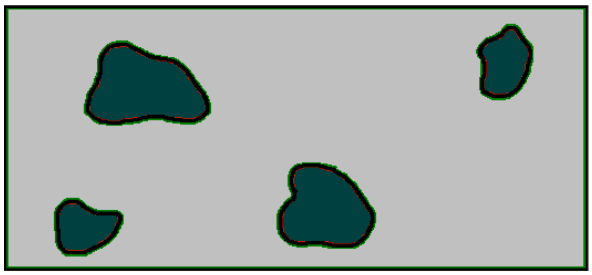
The problem of finding global optimal solutions arise in many disciplines ranging from science to engineering. The task of finding such solutions is quite complex and increases rapidly with the dimensionality of the problems. Identifying some promising regions in a solution space is relatively easier using certain global methods available in the literature. However, the fine tuning capability of these global methods is very poor and most of the times we end up with a poor solution even though the surrounding region is promising. Hence, there is an absolute necessity for exploring this surrounding to get better solution. Due to the nature of the problem, it is very likely that the nearby solutions surrounding the existing solution will be more promising.

Fig. 1 clearly shows the problems with nonlinear surfaces. Global methods can be used to obtain promising subspaces in the parameter space. These are indicated by drak shaded regions in (a). However, these promising regions are not convex in nature. i.e. they will have multiple local optimal solutions. (b) gives the top view of the nonlinear surface in the promising region. The dots indicate the local optimal solutions. 'S' is the initial point obtained from the global methods. Applying local method, it converges to 'A'. There are other stochastic methods that can search the neighborhood regions e.g. mutations in genetic algorithms, low temperature annealing in simulated annealing method.

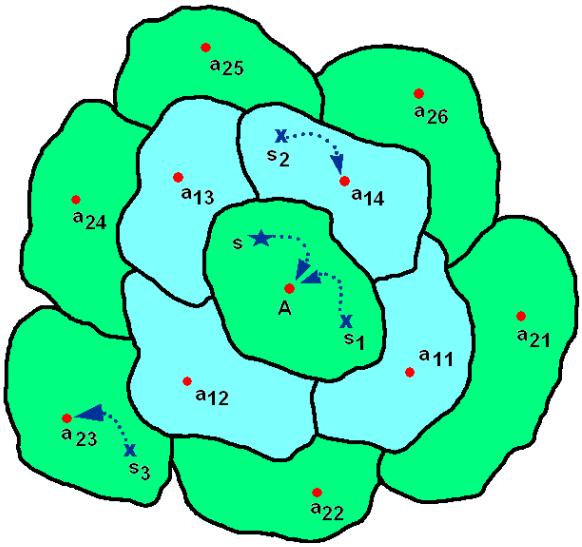
From the view of the parameter space, these methods randomly perturb the given point without much knowledge of the topology of the nonlinear surface. Using stability regions, we have a deterministic approach to obtain neighboring local maxima as opposed to widely used stochastic methods. These stochastic methods never guarantee the presence of a new stability region. Our approach 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. As shown in Fig. 1(b), the given local optimal solution 'A' is randomly perturbed to obtain new initial points (s_1, s_2, s_3). Applying local method using these initial points again, one can obtain the local optimal solutions A, a_{14} and a_{23} respectively. It can be observed that the solutions might appear again or might miss the neighborhood solutions.

In this work, we present a new stability region based method for finding such neighborhood solutions in a systematic manner. This method is more reliable when compared to other stochastic approaches which merely use random moves to find new solutions. Though some methods proposed in the literature apply other additional mechanisms (like perturbations [1]) to escape out of the local optimal solutions, systematic methods are yet to be developed for

Email address: ckr6@cornell.edu (Chandan K. Reddy and Hsiao-Dong Chiang).



(a) Entire Parameter Space



(b) Promising Subspace

Fig. 1. Various stages of our algorithm in (a) Entire Parameter space - the dark regions indicate the promising solution subspaces. (b) The dots indicate the local optimal solutions and the regions surrounding are the stability regions of the corresponding local optimal solutions.

searching the subspace. In this paper, the stability region based algorithms are applied to three important machine learning problems namely: (1) Unsupervised learning - model-based clustering, (2) Pattern discovery - motif finding problem and (3) Supervised learning - training artificial neural networks.

2 Problem Formulation and Transformations

Now, we will introduce some of the terminology that is required to understand the theoretical insights of our approach. Most importantly, this section deals with the transformation of the nonlinear function into a dynamical system. It also gives the correspondence between all the critical points of a n-dimensional nonlinear surface and that of its corresponding dynamical system. Our method is based on some of the fundamental results on stability regions of nonlinear dynamical systems [2–4].

2.1 Mathematical Preliminaries

Before presenting the details of our method, we review some fundamental concepts of nonlinear dynamical systems. Let us consider an unconstrained search problem on an energy surface defined by the objective function

$$f(x) \tag{1}$$

where $f(x)$ is assumed to be in $C^2(\Re^n, \Re)$.

Lemma 1 *$\bar{x}$ is said to be a critical point of (23) if it satisfies the following condition*

$$\nabla f(x) = 0 \tag{2}$$

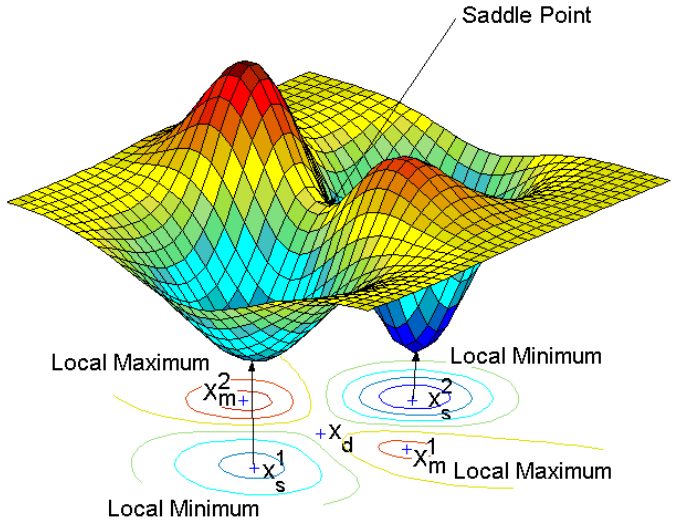


Fig. 2. The surface and contour plots of a two-dimensional energy function.A saddle point (x_d) is located between two local minima (x_s^1 and x_s^2). x_m^1 and x_m^2 are two local minima located in the orthogonal direction.

Now, we will define *Saddle points* which are the vital elements for understanding our methodology. The saddle points are critical points whose gradient is zero and Hessian of the nonlinear function has only one negative eigenvalue [5]. Intuitively, this means that a saddle point is a maximum along one direction but a minimum along all other orthogonal directions [6]. Fig. 1 shows a saddle point (x_d) located between two local minima (x_s^1 and x_s^2) and two local maxima (x_m^1 and x_m^2). As shown in the figure, the saddle point is a maximum along the direction of the vector joining the two local minima and a minimum along its orthogonal direction (or the direction of the vector joining the two local maxima). The direction in which the saddle point is the maximum is usually unknown in most of the practical problems and is the direction of interest.

A critical point is said to be *nondegenerate* if at the critical point $\bar{x} \in \Re^n$, $d^T \nabla^2_{xx} f(\bar{x}) d \neq 0 \ \ (\forall d \neq 0)$. Now, we construct the following *negative gradient system* in order to locate critical points of the objective function (23):

$$\frac{dx}{dt} = -\nabla f(x) \tag{3}$$

where the state vector x belongs to the Euclidean space $\Re^n$, and the vector field $f : \Re^n \rightarrow \Re^n$ satisfies the sufficient condition for the existence and uniqueness of the solutions. The solution curve of Eq. (30) starting from x at time $t = 0$ is called a *trajectory* and it is denoted by $\Phi(x, \cdot) : \Re \rightarrow \Re^n$. A state vector x is called an *equilibrium point* of Eq. (30) if $f(x) = 0$.

Lemma 2 *An equilibrium point is said to be hyperbolic if the Jacobian of f at point x has no eigenvalues with zero real part. 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. When $k = 0$, the equilibrium point is (asymptotically) stable and it is called a *sink* (or *attractor*). If $k = n$, then the equilibrium point is called a *source* (or *repeller*). A dynamical system is completely *stable* if every trajectory of the system leads to one of its stable equilibrium points. The *stable* ($W^s(\tilde{x})$) and *unstable* ($W^u(\tilde{x})$) manifolds of an equilibrium point, say $\tilde{x}$, is defined as:

$$W^s(\tilde{x}) = \{x \in \Re^n \ : \ \lim_{t \rightarrow \infty} \Phi(x, t) = \tilde{x}\} \tag{4}$$

$$W^u(\tilde{x}) = \{x \in \Re^n \ : \ \lim_{t \rightarrow -\infty} \Phi(x, t) = \tilde{x}\} \tag{5}$$

The stable and unstable manifolds of an equilibrium point are said to satisfy the *transversality* condition if either they do not intersect at all, or at every intersection point x_0 between these two manifolds, the tangent spaces of $W^s(x_0)$ and $W^u(x_0)$ span $\Re^n$. This is shown in Eq. (6)

$$T(W^s(x_0)) \oplus T(W^u(x_0)) = \Re^n \tag{6}$$

Lemma 3 *The stability region (also called region of attraction) of a stable equilibrium point x_s of a dynamical system (30) is denoted by $A(X_s)$ and is*

$$A(x_s) = \{x \in \Re^n \ : \ \lim_{t\to\infty} \Phi(x,t) = x_s\} \tag{7}$$

The boundary of stability region is called the *stability boundary* of x_s and will be denoted by $\partial A(x_s)$. It has been shown that the stability region is an open, invariant and connected set [2]. From the topological viewpoint, the stability boundary is a (n-1) dimensional closed and invariant set. A new concept related to the stability regions namely the *quasi-stability region* (or *practical stability region*), was developed in [3].

Lemma 4 *The practical stability region of a stable equilibrium point x_s of a nonlinear dynamical system (30), denoted by $A_p(x_s)$ and is*

$$A_p(x_s) = int\ \overline{A(x_s)} \tag{8}$$

where $\bar{A}$ denotes the closure of A and int $\bar{A}$ denotes the interior of $\bar{A}$. int $\overline{A(x_s)}$ is an open set. The boundary of practical stability region is called the practical stability boundary of x_s and will be denoted by $\partial A_p(x_s)$.

It has been shown that the practical stability boundary $\partial A_p(x_s)$ is equal to $\partial \bar{A}(x_s)$. The practical stability boundary is a subset of its stability boundary. It eliminates the complex portion of the stability boundary which has no "contact" with the complement of the closure of the stability region. A complete characterization of the practical stability boundary for a large class of nonlinear dynamical systems can be found in [2].

Lemma 5 *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.*

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

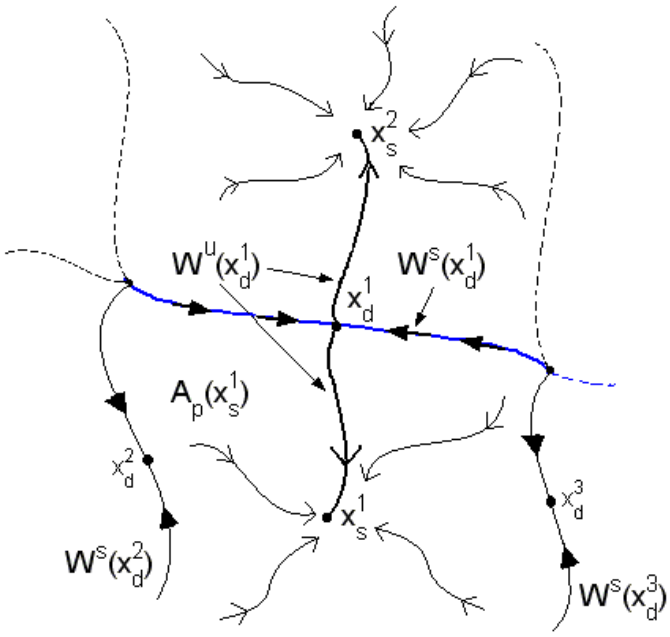


Fig. 3. Phase potrait of the gradient system corresponding to Fig. 2. The solid lines represent the basin boundary which shows that $\partial A_p(x_s^1) = \bigcup_{i=1}^{3} \overline{W^s(x_d^i)}$. The local minima x_s^1 and x_s^2 correspond to the stable equilibrium points of the gradient system. The saddle point (x_d^1) corresponds to the decomposition point of the gradient system.

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 optimal solutions and their one-to-one correspondence to the stable equilibrium points.

2.2 Theoretical Background

To comprehend the transformation, we need to define *energy function*. A smooth function $V(\cdot) : \Re^n \rightarrow \Re^n$ satisfying $\dot{V}(\Phi(x,t)) < 0$, $\forall\ x \notin$ {set of equilibrium points (E)} and t $\in \Re^+$ is termed as energy function.

Theorem 6 *[7]: $f(x)$ is a lyapunov function for the negative quasi-gradient system (30).*

The stability region of a stable equilibrium point can be completely characterized for a fairly large class of nonlinear systems.

Theorem 7 *(Characterization of stability boundary): Consider a nonlinear dynamical system described by (30) which satisfy assumptions (A1)-(A3). Let*

σ_i, i=1,2,... be the equilibrium points on the stability boundary $\partial A(x_s)$ of a stable equilibrium point, say x_s. Then

$$\partial A(x_s) = \bigcup_{\sigma_i \in \partial A} W^s(\sigma_i). \tag{9}$$

Theorem 7 completely characterizes the stability boundary for nonlinear dynamical systems satisfying assumptions (A1)-(A3) by asserting that the stability boundary is the union of the stable manifolds of all critical elements on the stability boundary. This theorem gives an explicit description of the geometrical and dynamical structure of the stability boundary. This theorem can be extended to the characterization of the practical stability boundary in terms of the stable manifold of the decomposition point.

Theorem 8 *(Characterization of practical stability boundary)[3]: Consider a nonlinear dynamical system described by (30) which satisfy assumptions (A1)-(A3). Let σ_i , 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*

$$\partial A_p(x_s) = \bigcup_{\sigma_i \in \partial A_p} \overline{W^s(\sigma_i)}. \tag{10}$$

Theorem 8 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 (10).

Theorem 9 *(Unstable manifold of type-1 equilibrium point)[8]: Let x_s^1 be a stable equilibrium point of the gradient system (30) and x_d be a type-1 equilibrium point on the practical stability boundary $\partial A_p(x_s)$. Assume that there exist ϵ and δ such that $\|\nabla f(x)\| > \epsilon$ unless $x \in B_\delta(\hat{x}), \hat{x} \in \{x : \nabla f(x) = 0\}$. If assumptions (A1)-(A3) are satisfied, then there exists another stable equilibrium point x_s^2 to which the one dimensional unstable manifold of x_d converges. Conversely, if $\overline{A_p(x_s^1)} \bigcap \overline{A_p(x_s^2)} \neq \emptyset$, then there exists a decomposition point x_d on $\partial A_p(x_s^1)$.*

Theorem 9 is imperative to understand some of the underlying concepts behind the development of our method. It associates the notion of stable equilibrium points, practical stability regions ($A_p(x_s)$), practical stability boundaries ($\partial A_p(x_s)$) and type-1 equilibrium points. As shown in fig. 3, The unstable manifold (W^u) of the decomposition point x_d^1 converges to the two stable equilibrium points x_s^1 and x_s^2. Also, it should be noted that x_d^1 is present on the stability boundary of x_s^1 and x_s^2.

We also need to show that under the transformation from (23) to (30), the properties of the critical points remain unchanged. Theorem 10 illustrates the correspondence of the critical points of the original system.

Theorem 10 *(Critical Points and their correspondence): An equilibrium point of (30) is hyperbolic if, and only if, the corresponding critical point f is nondegenerate. Moreover, if $\bar{x}$ is a hyperbolic equilibrium point of (30), then*

(1) $\bar{x}$ is a stable equilibrium point of (30) if and only if $\bar{x}$ is an isolated local minimum for (23)
(2) $\bar{x}$ is a source of (30) if and only if $\bar{x}$ is an isolated local maximum for (23)
(3) $\bar{x}$ is a decomposition point of (30) if and only if $\bar{x}$ is a saddle point for (23)

Our approach takes advantage of TRUST-TECH (TRansformation Under STability-reTaining Equilibria CHaracterization) to compute neighborhood local optimal solutions on the 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 optima uniquely, it is important to obtain the saddle points from the given solution and then move to the next solution through this decomposition point. The details of a stability boundary tracing algorithm to obtain decomposition points on the stability boundary is presented in [6]. 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 local solver directly using the exit point without considering the decomposition point. However, it is beneficial to use the saddle point tracing procedure developed in [6] for applications where the local methods are very expensive. For most of the applications that arise in machine learning domain, it is not necessary to obtain the saddle point because of the fact that the local methods are highly optimized for the specific problem in hand and in general, these methods are much cheaper than tracing the boundary using gradient computation.

3 Unsupervised Learning - Model-based Clustering

In this section, we will develop a stability region based algorithm for solving the problem of mixture modeling. Finite mixtures allow a probabilistic model-based approach to unsupervised learning [9] which plays an important role in predictive data mining applications. One of the most popular methods used for fitting mixture models to the observed data is the *Expectation-Maximization* (EM) algorithm which converges to the maximum likelihood estimate of the mixture parameters locally [10,11]. The usual steepest descent, conjugate gradient, or Newton-Raphson methods are too complicated for use in solving this problem [12]. EM has become a popular method since it takes advantages of problem specific properties. EM based approaches have been successfully used to solve problems that arise in various other applications [13,14].

Without loss of generality, we will consider the problem of learning parameters of Gaussian Mixture Models (GMM). Fig 4 shows data generated by three Gaussian components with different mean and variance. Note that every data point has a probabilistic (or soft) membership that gives the probability with which it belongs to each of the components. 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, EM algorithm can be applied to compute the optimal estimates of the parameters that maximize the likelihood of the data given the estimates of these components. However, the main problem with the EM algorithm is that it is a '*greedy*' method which is very sensitive to the given initial set of parameters [15]. To overcome this problem, a novel two phase algorithm based on stability region analysis is proposed. 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 coexist and it becomes challenging to avoid wasting computational resources to search in non-promising regions.

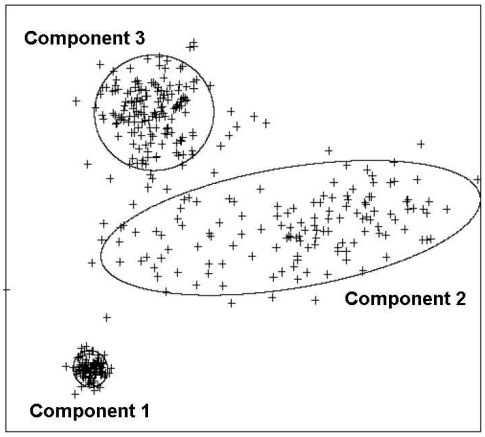


Fig. 4. Data generated by three Gaussian components. The problem of learning mixture models is to obtain the parameters of these Gaussian components and the membership probabilities of each datapoint.

Of all the concerns mentioned above, the fact that most of the local maxima are not distributed uniformly [16] 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. Here, we propose a novel two phase stability region based expectation maximization (SREM) algorithm for estimating the parameters of mixture models. Using concepts of dynamical systems and EM algorithm simultaneously to exploit the problem specific features of the mixture models, our algorithm obtains the optimal set of parameters by searching for the global maximum on the likelihood surface in a systematic manner.

3.1 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 [16], genetic algorithms [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 for most of the practical applications. Some problem specific algorithms like split and merge EM [18], component-wise EM [11], greedy learning [19], incremental version for sparse representations[20], parameter space grid [21] are also proposed in the literature. Some of these algorithms are either computationally very expensive or infeasible when learning mixtures in high dimensional spaces [21]. Inspite of all the expense in 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 optimal

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 higher value of the likelihood are also successful to certain extent [22].

The dynamical system of the log-likelihood function reveals more information on the neighborhood stability regions and their corresponding local maxima [7]. 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. Specifically, for a problem in this context, where there is a non-uniform distribution of local maxima, it is difficult for most of the methods to search neighboring regions [23].

3.2 Preliminaries

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

Table 1
Description of the Notations used

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

3.2.1 Mixture Models

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

$$p(x|\Theta) = \sum_{i=1}^{k} \alpha_i p(x|\theta_i) \tag{11}$$

where $x = [x_1, x_2, ..., x_d]^T$ is the feature vector of d dimensions. The α_k's represent the *mixing weights*. Θ represents the parameter set $(\alpha_1, \alpha_2, ...\alpha_k, \theta_1, \theta_2, ...\theta_k)$ and p is a univariate Gaussian density parameterized by θ_i(i.e. μ_i and σ_i):

$$p(x|\theta_i) = \frac{1}{\sqrt{(2\pi)}\sigma_i} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \tag{12}$$

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

$$0 \leq \alpha_i \leq 1 \ , \ \forall i = 1, .., k, \ and \ \sum_{i=1}^{k} \alpha_i = 1 \tag{13}$$

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

$$\begin{aligned} log\ p(\mathcal{X}|\Theta) &= log \prod_{j=1}^{n}\ p(x^{(j)}|\Theta) \\ &= \sum_{j=1}^{n} log \sum_{i=1}^{k} \alpha_i\ p(x^{(j)}|\theta_i) \end{aligned} \tag{14}$$

The goal of learning mixture models is to obtain the parameters $\widehat{\Theta}$ from a set of n data points which are the samples of a distribution with density given by (11). The *Maximum Likelihood Estimate* (MLE) is given by :

$$\widehat{\Theta}_{MLE} = arg \max_{\tilde{\Theta}}\ \{\ log\ p(\mathcal{X}|\Theta)\ \} \tag{15}$$

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 [24].

3.2.2 Expectation Maximization

The EM algorithm assumes $\mathcal{X}$ 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 which component produced each sample [24]. 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 $x^{(j)}$ 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

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

$$log\ p(\mathcal{Y}|\Theta) = \sum_{j=1}^{n} \sum_{i=1}^{k} z_i^{(j)}\ log\ [\ \alpha_i\ p(x^{(j)}|\theta_i)\ ] \tag{16}$$

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:

$$Q(\Theta|\widehat{\Theta}(t)) \equiv E_Z[log\ p(\mathcal{X}, \mathcal{Z})|\mathcal{X}, \widehat{\Theta}(t)] \tag{17}$$

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

$$\begin{aligned} w_i^{(j)} &\equiv E\ [\ z_i^{(j)}|\mathcal{X}, \widehat{\Theta}(t)\ ] \\ &= Pr\ [\ z_i^{(j)} = 1|x^{(j)}, \widehat{\Theta}(t)\ ] \\ &= \frac{\widehat{\alpha}_i(t) p(x^{(j)}|\widehat{\theta}_i(t))}{\sum_{i=1}^{k} \widehat{\alpha}_i(t) p(x^{(j)}|\widehat{\theta}_i(t))} \end{aligned} \tag{18}$$

  where the last equality is simply the 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 $x^{(j)}$.
- **M-Step :** The estimates of the new parameters are updated using the following equation :

$$\widehat{\Theta}(t+1) = arg \max_{\Theta} \{Q(\Theta, \widehat{\Theta}(t))\} \tag{19}$$

3.2.3 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 [12]. The $Q-function$ for GMM is given by:

$$Q(\Theta|\widehat{\Theta}(t)) = \sum_{j=1}^{n}\sum_{i=1}^{k} w_i^{(j)}[log\frac{1}{\sigma_i\sqrt{2\pi}} - \frac{(x^{(j)}-\mu_i)^2}{2\sigma_i^2} + log\ \alpha_i] \tag{20}$$

where

$$w_i^{(j)} = \frac{\frac{\alpha_i(t)}{\sigma_i(t)}e^{-\frac{1}{2\sigma_i(t)^2}(x^{(j)}-\mu_i(t))^2}}{\sum_{i=1}^{k}\frac{\alpha_i(t)}{\sigma_i(t)}e^{-\frac{1}{2\sigma_i(t)^2}(x^{(j)}-\mu_i(t))^2}} \tag{21}$$

The maximization step is given by the following equation :

$$\frac{\partial}{\partial\Theta_k}Q(\Theta|\widehat{\Theta}(t)) = 0 \tag{22}$$

where Θ_k is the parameters for the k^{th} component. Because of the assumption made that each data point comes from a single component, solving the above equation becomes trivial. The updates for the maximization step in the case of GMMs are given as follows:

$$\mu_i(t+1) = \frac{\sum_{j=1}^{n} w_i^{(j)}x^{(j)}}{\sum_{j=1}^{n} w_i^{(j)}}$$

$$\sigma_i^2(t+1) = \frac{\sum_{j=1}^{n} w_i^{(j)}(x^{(j)}-\mu_i(t+1))^2}{\sum_{j=1}^{n} w_i^{(j)}}$$

$$\alpha_i(t+1) = \frac{1}{n}\sum_{j=1}^{n} w_i^{(j)}$$

3.2.4 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. 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 spherical Gaussian mixtures with k components, we have the number of unknown parameters $s = 3k-1$. For convenience, the maximization problem is transformed into a minimization problem defined by the following objective function :

$$\begin{aligned} \max_{\Theta} \{ \, log \; p(\mathcal{Y}|\Theta) \, \} = & \min_{\Theta} \{ \; - \; log \; p(\mathcal{Y}|\Theta) \, \} \\ & = \min_{\Theta} f(\Theta) \end{aligned} \tag{23}$$

where $f(\Theta)$ is assumed to be in $C^2(\Re^s, \Re)$.

The gradient system for the log-likelihood function in the case of spherical Gaussians is constructed as follows :

$$[\dot{\mu}_1(t) \; .. \; \dot{\mu}_k(t) \; \dot{\sigma}_1(t) \; .. \; \dot{\sigma}_k(t) \; \dot{\alpha}_1(t) \; .. \; \dot{\alpha}_{k-1}(t)]^T$$

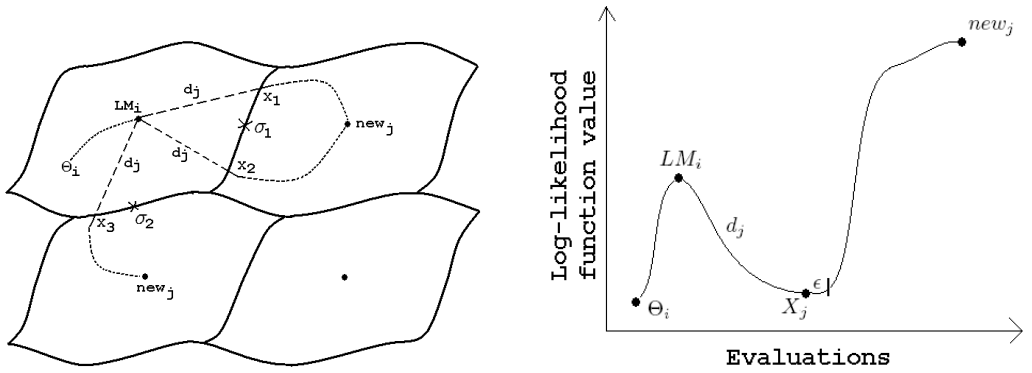
$$= \; - \; \left[\frac{\partial f}{\partial \mu_1} \; .. \; \frac{\partial f}{\partial \mu_k} \; \frac{\partial f}{\partial \sigma_1} \; .. \; \frac{\partial f}{\partial \sigma_k} \; \frac{\partial f}{\partial \alpha_1} \; .. \; \frac{\partial f}{\partial \alpha_{k-1}} \right]^T$$

where

$$\begin{aligned} \frac{\partial f}{\partial \mu_i} &= \sum_{j=1}^{n} w_i^{(j)} \frac{(x^{(j)} - \mu_i)}{2\sigma_i^2} && \forall i = 1, .., k \\ \frac{\partial f}{\partial \sigma_i} &= \sum_{j=1}^{n} w_i^{(j)} \left[-\frac{1}{\sigma_i} + \frac{(x^{(j)} - \mu_i)^2}{\sigma_i^3} \right] && \forall i = 1, .., k \\ \frac{\partial f}{\partial \alpha_i} &= \frac{1}{\alpha_i} \sum_{j=1}^{n} w_i^{(j)} && \forall i = 1, .., k-1 \end{aligned} \tag{24}$$

For simplicity, we show the construction of the gradient system for the case of spherical Gaussians. It can be easily extended to the full covariance Gaussian mixture case. 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:

$$\alpha_k = 1 - \sum_{j=1}^{k-1} \alpha_j \tag{25}$$



(a) Parameter Space (b) Function Space

Fig. 5. Various stages of our algorithm in (a) Parameter space - the solid lines indicate the practical stability boundary. Points highlighted on the stability boundary are the decomposition points. The dotted arrows indicate the convergence of the EM algorithm. The dashed lines indicate the stability region phase. x_1 and x_2 are the exit points on the practical stability boundary (b) Different variables in the function space.

3.3 Our Algorithm

Our framework consists mainly of two phases which are repeated in the promising subspaces of the parameter search space. It is more effective to use our algorithm at only these promising subspaces which are usually obtained by stochastic global methods. The first phase is the local phase (or the EM phase) where the promising solutions are refined to the corresponding locally optimal parameter set. The second phase which is the main contribution of this paper, is the stability region phase, where the exit points are computed and the neighborhood solutions are systematically explored through these exit points. Fig. 5 shows the different steps of our algorithm both in (a) the parameter space and (b) the function 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 deterministically. 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 parameter estimation. Algorithm 1 describes our approach.

In order to escape out of this local maximum, our method needs to compute certain promising directions based on the local behaviour of the function.

Algorithm 1 Stability Region based EM Algorithm

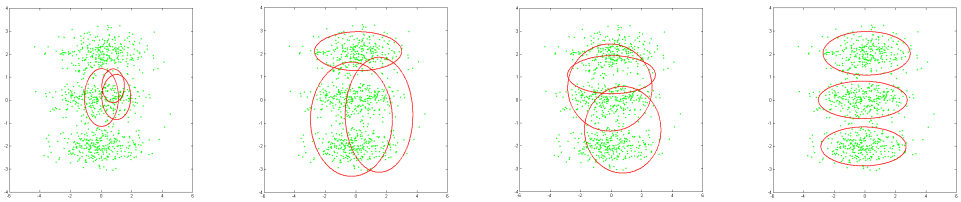
Input: Parameters Θ, Data $\mathcal{X}$, tolerance τ, Step S_p
Output: $\widehat{\Theta}_{MLE}$
Algorithm:
Apply global method and store the q promising solutions $\Theta_{init} = \{\Theta_1, \Theta_2, .., \Theta_q\}$ Initialize E= ϕ
while $\Theta_{init} \neq \phi$ **do**
  Choose $\Theta_i \in \Theta_{init}$, set $\Theta_{init} = \Theta_{init} \backslash \{\Theta_i\}$
  $LM_i = EM(\Theta_i, \mathcal{X}, \tau)$ $E = E \cup \{LM_i\}$
  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 (14)
    $New_j = EM(X_j + \epsilon \cdot d_j, \mathcal{X}, \tau)$
    if $new_j \notin E$ **then**
      $E = E \cup New_j$
    end if
  end for
end while
$\widehat{\Theta}_{MLE} = max\{val(E_i)\}$

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) depending on the 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.

3.4 Implementation Details

Our program is implemented in MATLAB and runs on Pentium IV 2.8 GHz machine. The main procedure implemented is TT_EM described in Algorithm 4. 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 in 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 (14). 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 $Params[\,]$ [1] . Since, different parameters will be of different range, care must be taken while multiplying with 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 likelihood score amongst the original set of parameters and the Tier-1 solutions is returned.



(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.

3.5 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 uniformly random. Uniform priors were chosen for initializing the components. For real datasets, the centers were chosen randomly from the sample points.

[1] To ensure that the new initial points are in the different stability regions, one should move along the directions 'ϵ' away from the exit points.

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

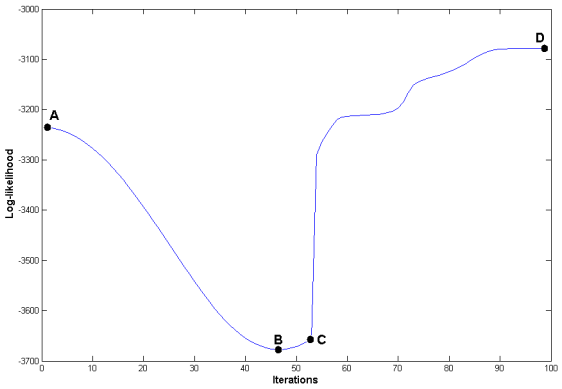


Fig. 7. Graph showing likelihood vs Evaluations. A corresponds to the original local maximum (L=-3235.0). B corresponds to the exit point (L=-3676.1). C corresponds to the new initial point in the neighboring stability region (L=-3657.3) after moving out by 'ϵ'. D corresponds to the new local maximum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

3.5.1 Synthetic 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 $\mu_5 = [0.7\ 0.3]^T$.

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 [16]. All the three mixtures have uniform priors. 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.

In the third synthetic dataset, a more complicated overlapping Gaussian mixtures are considered [11]. 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 $\alpha_4 = 0.1$.

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

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

Table 2
Performance of our algorithm on an average of 100 runs on various synthetic and real datasets

Dataset	Samples	Clusters	Features	EM (mean ± std)	SREM (mean ± std)
Spherical	40	5	2	38.07±2.12	43.55±0.6
Elliptical	900	3	2	-3235±0.34	-3078.7±0.03
Full covariance 1	500	4	2	-2345.5 ±175.13	-2121.9± 21.16
Full covariance 2	2000	4	2	-9309.9 ±694.74	-8609.7 ±37.02
Iris	150	3	4	-198.13±27.25	-173.63±11.72
Wine	178	3	13	-1652.7±1342.1	-1618.3±1349.9

3.5.2 Real Datasets

Two real datasets obtained from the UCI Machine Learning repository [25] were also used for testing the performance of our algorithm. Most 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 5.4.2 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 synthetic and real data sets. Our algorithm not only obtains higher likelihood value but also produces it with high confidence. The low standard deviation of our results indicates the robustness of obtaining the global maximum. In the case of the wine data, the improvements with our algorithm are not much 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 mixture of Gaussians. Table 3 gives the results of SREM compared with other methods like split and merge EM and k-means+EM proposed in the literature.

Table 3
Comparison of SREM with other methods

Method	Elliptical	Iris
RS+EM	-3235 ± 14.2	-198 ± 27
K-Means+EM	-3195 ± 54	-186 ± 10
SMEM	-3123 ± 54	-178.5 ± 6
SREM	-3079 ± 0.03	-173.6 ± 11

3.5.3 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 available and improve the quality of the existing solutions. In clustering problems, it is an added advantage to perform refinement of the final clusters obtained. Most of the focus in the literature was 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.

Table 4 summarizes the average number of iterations taken by the EM algorithm for the convergence to the local optimal solution. We can see that the most promising solution produced by our SREM algorithm 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 the Tier-1 solutions using our algorithm, the threshold for the number of iterations can be significantly lowered.

Table 4
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

Our algorithm can be easily extended to other widely used EM related problems like k-means clustering,training Hidden Markov Models, Mixture of Factor Analyzers, Probabilistic Principal Component Analysis, Bayesian Networks etc. We would also like to extend these techniques to Markov Chain Monte Carlo methods like Gibbs sampling for the estimation of mixture models.

4 Pattern Discovery - Motif Finding Problem

As a case study of the algorithm developed in the previous section, we describe its application to the motif finding problem in bioinformatics. In fact, we do not have to restrict the proposed algorithm to the case of mixture models. Here, we will demonstrate its capability in obtaining promising solutions on general likelihood surfaces. The main goal of the motif finding problem is to detect novel, over-represented unknown signals in a set of sequences (e.g. transcription factor binding sites in a genome). The most widely used algorithms for finding motifs obtain a generative probabilistic representation of these over-represented signals and try to discover profiles that maximize the information content score. Although these profiles form a very powerful representation of the signals, the major difficulty arises from the fact that the best motif corresponds to the global maximum of a non-convex continuous function. Popular algorithms like Expectation Maximization (EM) and Gibbs sampling tend to be very sensitive to the initial guesses and are known to converge to the nearest local maximum very quickly [26]. In order to improve the quality of the results, EM is used with multiple random starts or any other powerful stochastic global methods that might yield promising initial guesses (like projection algorithms). Global methods do not necessarily give initial guesses in the convergence region of the best local maximum but rather suggest that a promising solution is in the neighborhood region. In this section, we apply the SREM algorithm proposed in the previous section to this motif finding problem. It has the capability to search the neighborhood regions of the initial alignment in a systematic manner to explore the multiple local optimal solutions. This effective search is achieved by transforming the original optimization problem into its corresponding dynamical system and estimating the practical stability boundary of the local maximum.

GAATTCATACCAGATCACCGGATTCCCGACTCCAAATGTGTCCCCCTCACAC

TCCCCCGATTACCGTCTTCTGCTCTTAGACCACTCTACCCTATTCCCCACACT

CACCGGAGCCAAAGCCGCGGCCCTTCCGTTCCGATTACCGAAAAGACCCCA

CCCGTAGGTGGCAAGCTAGCTTAAGTAACGCCACTTCGATTAACGAGGAAA

AATACATAACTGACCTATTATCGAGTTCAGATCAAGGTCAGGAACAAAGAA

ACACCGATTACCGTAACCGTAAGATARTGGTATCGATACGTAGACAGTTTA

Fig. 8. Synthetic DNA sequences containing some instance of the pattern 'CCGATTACCGA' with a maximum number of 2 mutations. The motifs in each sequence are highlighted in the box. We have a (11,2) motif where 11 is the length of the motif and 2 is the number of mutations allowed.

Recent developments in DNA sequencing have allowed biologists to obtain complete genomes for several species. However, knowledge of the sequence does not imply the understanding of how genes interact and regulate one an-

other within the genome. Many transcription factor binding sites are highly conserved throughout the sequences and the discovery of the location of such binding sites plays an important role in understanding gene interaction and gene regulation. Although there are several variations of the motif finding algorithms, the problem discussed here is defined as follows: without any previous knowledge of the consensus pattern, discover all the occurrences of the motifs and then recover a pattern for which all of these instances are within a given number of mutations (or substitutions). Despite the significant amount of literature available on the motif finding problem, many do not exploit the probabilistic models used for motif refinement [27,28]. In this section, we consider a precise version of the motif discovery problem in computational biology as discussed in [29,30]. The planted (l,d) motif problem [30] considered here is described as follows: Suppose there is a fixed but unknown nucleotide sequence M(the *motif*) of length l. The problem is to determine M, given t sequences with t_i being the length of the i^{th} sequence and each containing a planted variant of M. More precisely, each such planted variant is a substring that is M with exactly d point substitutions (see fig. 8). More details about the complexity of the motif finding problem is given in [31]. A detailed assessment of different motif finding algorithms was published recently in [32].

4.1 Motif Finding Literature

Existing approaches used to solve the motif finding problem can be classified into two main categories [33]. The first group of algorithms utilizes a generative probabilistic representation of the nucleotide positions to discover a consensus DNA pattern that maximizes the information content score. In this approach, the original problem of finding the best consensus pattern is formulated as finding the global maximum of a continuous non-convex function. The main advantage of this approach is that the generated profiles are highly representative of the signals being determined [34]. The disadvantage, however, is that the determination of the "best" motif cannot be guaranteed and is often a very difficult problem since finding global maximum of any continuous non-convex function is a challenging problem. Current algorithms converge to the nearest local optimum instead of the global solution. Gibbs sampling [27], MEME [28], greedy CONSENSUS algorithm [35] and HMM based methods [36] belong to this category.

The second group uses patterns with 'mismatch representation' which define a signal to be a consensus pattern and allow up to a certain number of mismatches to occur in each instance of the pattern. The goal of these algorithms is to recover the consensus pattern with the highest number of instances. These methods view the representation of the signals as discrete and the main advantage of these algorithms is that they can guarantee that the highest scoring

pattern will be the global optimum for any scoring function. The disadvantage, however, is that consensus patterns are not as expressive of the DNA signal as profile representations. Recent approaches within this framework include Projection methods [29,37], string based methods [30], Pattern-Branching [38], MULTIPROFILER [39] and other branch and bound approaches [40,33].

A hybrid approach could potentially combine the expressiveness of the profile representation with convergence guarantees of the consensus pattern. An example of a hybrid approach is the Random Projection [29] algorithm followed by EM algorithm [28]. It uses a global solver to obtain promising alignments in the discrete pattern space followed by further local solver refinements in continuous space[41,42]. Currently, only few algorithms take advantage of a combined discrete and continuous space search [29,33,37]. We consider the profile representation of the motif and a new hybrid algorithm is developed to escape out of the local maxima of the likelihood surface.

Some motivations to develop the new SREM algorithm are :

- A motif refinement stage is vital and popularly used by many pattern based algorithms (like PROJECTION, MITRA etc) which try to find optimal motifs.
- The traditional EM algorithm used in the context of motif finding converges very quickly to the nearest local optimal solution (within 5-8 iterations).
- There are many other promising local optimal solutions in the close vicinity of the profiles obtained from the global methods.

In spite of the importance placed on obtaining a global optimal solution in the context of motif finding, little work has been done in the direction of finding such solutions [43]. There are several proposed methods to escape out of the local optimal solution to find better solutions in machine learning [1] and optimization [44] related problems. Most of them are stochastic in nature and usually rely on perturbing either the data or the hypothesis. These stochastic perturbation algorithms are inefficient because they will sometimes miss a neighborhood solution or obtain an already existing solution. To avoid these problems, we introduce a novel optimization framework that has a better chance of avoiding sub-optimal solutions. It systematically escapes out of the convergence region of a local maximum to explore the existence of other nearby local maxima. Our method is primarily based on some fundamental principles of finding exit points on the stability boundary of a nonlinear continuous function. The underlying theoretical details of our method are described in [7,4].

4.2 Preliminaries

We will first describe our problem formulation and the details of the EM algorithm in the context of motif finding problem. We will then describe some details of the dynamical system of the log-likelihood function which enables us to search for the nearby local optimal solutions.

4.2.1 Problem Formulation

Some promising initial alignments are obtained by applying projection methods or random starts on the entire dataset. Typically, random starts are used because they are not expensive. Most promising set of alignments are considered for further processing. These initial alignments are then converted into profile representation.

Table 5
A count of nucleotides A, T, G, C at each position $K = 1..l$ in all the sequences of the data set. $K = 0$ denotes the background count.

j	$k=0$	$k=1$	$k=2$	$K=3$	$k=4$	...	$k=l$
A	$C_{0,1}$	$C_{1,1}$	$C_{2,1}$	$C_{3,1}$	$C_{4,1}$	...	$C_{l,1}$
T	$C_{0,2}$	$C_{1,2}$	$C_{2,2}$	$C_{3,2}$	$C_{4,2}$	...	$C_{l,2}$
G	$C_{0,3}$	$C_{1,3}$	$C_{2,3}$	$C_{3,3}$	$C_{4,3}$	...	$C_{l,3}$
C	$C_{0,4}$	$C_{1,4}$	$C_{2,4}$	$C_{3,4}$	$C_{4,4}$	...	$C_{l,4}$

Let t be the total number of sequences and n be the average length of the sequences. Let $S = \{S_1, S_2...S_t\}$ be the set of t sequences. Let P be a single alignment containing the set of segments $\{P_1, P_2, ..., P_t\}$. l is the length of the consensus pattern. For further discussion, we use the following variables

$$\begin{aligned} i &= 1 \ldots t && -- \text{ for } t \text{ sequences} \\ k &= 1 \ldots l && -- \text{ for positions within an } l\text{-mer} \\ j &\in \{A, T, G, C\} && -- \text{ for each nucleotide} \end{aligned}$$

The count matrix can be constructed from the given alignments as shown in Table 5. We define $C_{0,j}$ to be the non-position specific background count of each nucleotide in all of the sequences where $j \in \{A, T, C, G\}$ is the running total of nucleotides occurring in each of the l positions. Similarly, $C_{k,j}$ is the count of each nucleotide in the k^{th} position (of the $l-mer$) in all the segments in P.

$$Q_{0,j} = \frac{C_{0,j}}{\sum_{J \in \{A,T,G,C\}} C_{0,J}} \tag{26}$$

$$Q_{k,j} = \frac{C_{k,j} + b_j}{t + \sum_{J \in \{A,T,G,C\}} b_J} \tag{27}$$

Eq. (26) shows the background frequency of each nucleotide where b_j is known as the Laplacian or Bayesian correction and is equal to $d * Q_{0,j}$ where d is some constant usually set to unity. Eq. (27) gives the weight assigned to the type of nucleotide at the k^{th} position of the motif.

A Position Specific Scoring Matrix (PSSM) can be constructed from one set of instances in a given set of t sequences. From (26) and (27), it is obvious that the following relationship holds:

$$\sum_{j \in \{A,T,G,C\}} Q_{k,j} = 1 \qquad \forall k = 0, 1, 2, ...l \tag{28}$$

For a given k value in (28), each Q can be represented in terms of the other three variables. Since the length of the motif is l, the final objective function (i.e. the information content score) would contain $3l$ independent variables [2].

To obtain the score, every possible $l - mer$ in each of the t sequences must be examined. This is done so by multiplying the respective $Q_{i,j}/Q_{0,j}$ dictated by the nucleotides and their respective positions within the $l - mer$. Only the highest scoring $l - mer$ in each sequence is noted and kept as part of the alignment. The total score is the sum of all the best scores in each sequence.

$$\begin{aligned} A(Q) &= \sum_{i=1}^{t} log(A)_i = \sum_{i=1}^{t} log \left(\prod_{k=1}^{l} \frac{Q_{k,j}}{Q_b} \right)_i \\ &= \sum_{i=1}^{t} \sum_{k=1}^{l} log(Q'_{k,j})_i \end{aligned} \tag{29}$$

$Q'_{k,j}$ is the ratio of the nucleotide probability to the corresponding background probability, i.e. $Q_{k,j}/Q_b$. $Log(A)_i$ is the score at each individual i^{th} sequence where t is the total number of sequences. In equation (29), we see that A is composed of the product of the weights for each individual position k.

[2] Although, there are $4l$ variables in total, because of the constraints obtained from (28), the parameter space will contain only $3l$ independent variables. Thus, the constraints help in reducing the dimensionality of the search problem.

We consider this to be the Information Content (IC) score which we would like to maximize. $A(Q)$ is the non-convex $3l$ dimensional continuous function for which the global maximum corresponds to the best possible motif in the dataset. EM refinement performed at the end of a combinatorial approach has the disadvantage of converging to a local optimal solution [45]. Our method improves the procedure for refining motif by understanding the details of the stability boundaries and by trying to escape out of the convergence region of the EM algorithm.

4.2.2 Hessian Computation and Dynamical System for the Scoring Function

In order to present our algorithm, we have defined the dynamical system corresponding to the log-likelihood function and the PSSM. The key contribution here is the development of this nonlinear dynamical system which will enable us to realize the geometric and dynamic nature of the likelihood surface. We construct the following *gradient system* in order to locate critical points of the objective function (29):

$$\dot{Q}(t) = -\nabla A(Q) \tag{30}$$

One can realize that this transformation preserves all of the critical points [7]. Now, we will describe the construction of the gradient system and the Hessian in detail. In order to reduce the dominance of one variable over the other, the values of the each of the nucleotides that belong to the consensus pattern at the position k will be represented in terms of the other three nucleotides in that particular column. Let P_{ik} denote the k^{th} position in the segment P_i. This will also minimize the dominance of the eigenvector directions when the Hessian is obtained. The variables in the scoring function are transformed into new variables described in Table 6.

$$A(Q) = \sum_{i=1}^{t} \sum_{k=1}^{l} log\ f_{ik}(w_{3k-2}, w_{3k-1}, w_{3k})_i \tag{31}$$

where f_{ik} can take the values $\{w_{3k-2},\ w_{3k-1},\ w_{3k},\ 1-(w_{3k-2}+w_{3k-1}+w_{3k})\ \}$ depending on the P_{ik} value.

The first derivative of the scoring function is a one dimensional vector with $3l$ elements.

$$\nabla A = \left[\begin{array}{cccccc} \frac{\partial A}{\partial w_1} & \frac{\partial A}{\partial w_2} & \frac{\partial A}{\partial w_3} & \cdots & \frac{\partial A}{\partial w_{3l}} \end{array} \right]^T \tag{32}$$

Table 6
A count of nucleotides $j \in \{A, T, G, C\}$ at each position $k = 1..l$ in all the sequences of the data set. C_k is the k^{th} nucleotide of the consensus pattern which represents the nucleotide with the highest value in that column. Let the consensus pattern be GACT...G and b_j be the background.

j	$k = b$	$k = 1$	$k = 2$	$K = 3$	$k = 4$	...	$k = l$
A	b_A	w_1	C_2	w_7	w_{10}	...	w_{3l-2}
T	b_T	w_2	w_4	w_8	C_4	...	w_{3l-1}
G	b_G	C_1	w_5	w_9	w_{11}	...	C_l
C	b_C	w_3	w_6	C_3	w_{12}	...	w_{3l}

and each partial derivative is given by

$$\frac{\partial A}{\partial w_p} = \sum_{i=1}^{t} \frac{\frac{\partial f_{ip}}{\partial w_p}}{f_{ik}(w_{3k-2}, w_{3k-1}, w_{3k})} \tag{33}$$

$$\forall p = 1, 2 \ ... \ 3l \quad and \quad k = round(p/3) + 1$$

The Hessian $\nabla^2 A$ is a block diagonal matrix of block size 3X3. For a given sequence, the entries of the 3X3 block will be the same if that nucleotide belongs to the consensus pattern (C_k). The gradient system is mainly obtained for enabling us to identify the stability boundaries and stability regions on the likelihood surface. The theoretical details of these concepts are published in [7]. The stability region of each local maximum is an approximate convergence zone of the EM algorithm. If we can identify all the saddle points on the stability boundary of a given local maximum, then we will be able to find all the tier-1 local maxima. However, finding all of the saddle points is computationally intractable and hence we have adopted a heuristic by generating the eigenvector directions of the PSSM at the local maximum. The next section details our approach and explains the different phases of our algorithm. In this problem, we will not trace the stability boundary for obtaining saddle points and hence, we call this as "Exit-point" framework.

4.3 Stability Region based Exit-point Framework

Our framework consists of the following three phases:

- *Global phase* in which the promising solutions in the entire search space are obtained.
- *Refinement phase* where a local method is applied to the solutions obtained in the previous phase in order to refine the profiles.

- *Exit phase* where the exit points are computed and the Tier-1 and Tier-2 solutions are explored systematically.

In the global phase, a branch and bound search is performed on the entire dataset. All of the profiles that do not meet a certain threshold (in terms of a given scoring function) are eliminated in this phase. The promising patterns obtained are transformed into profiles and local improvements are made to these profiles in the refinement phase. The consensus pattern is obtained from each nucleotide that corresponds to the largest value in each column of the PSSM. The $3l$ variables chosen are the nucleotides that correspond to those that are not present in the consensus pattern. Because of the probability constraints discussed in the previous section, the largest weight can be represented in terms of the other three variables.

To solve (29), current algorithms begin at random initial alignment positions and attempt to converge to an alignment of $l-mer$s in all of the sequences that maximize the objective function. In other words, the $l-mer$ whose $log(A)_i$ is the highest (with a given PSSM) is noted in every sequence as part of the current alignment. During the maximization of $A(Q)$ function, the probability weight matrix and hence the corresponding alignments of $l-mer$s are updated. This occurs iteratively until the PSSM converges to the local optimal solution. The consensus pattern is obtained from the nucleotide with the largest weight in each position (column) of the PSSM. This converged PSSM and the set of alignments correspond to a local optimal solution. The exit phase where the neighborhood of the original solution is explored in a systematic manner is shown below:

Input: Local Maximum (A).
Output: Best Local Maximum in the neighborhood region.
Algorithm:
Step 1: Construct the PSSM for the alignments corresponding to the local maximum (A) using Eqs.(26) and (27).
Step 2: Calculate the eigenvectors of the Hessian matrix for this PSSM.
Step 3: Find exit points (e_{1i}) on the practical stability boundary along each eigenvector direction.
Step 4: For each of the exit points, the corresponding Tier-1 local maxima (a_{1i}) are obtained by applying the EM algorithm after the ascent step.
Step 5: Repeat this process for promising Tier-1 solutions to obtain Tier-2 (a_{2j}) local maxima.
Step 6: Return the solution that gives the maximum information content score of $\{A,\ a_{1i},\ a_{2j}\}$.

To escape out of this local optimal solution, our approach requires the computation of a Hessian matrix (i.e. the matrix of second derivatives) of dimension

$(3l)^2$ and the $3l$ eigenvectors of the Hessian. The main reasons for choosing the eigenvectors of the Hessian as search directions are:

- Computing the eigenvectors of the Hessian is related to finding the directions with extreme values of the second derivatives, i.e., directions of extreme normal-to-isosurface change.
- The eigenvectors of the Hessian will form the basis vectors for the search directions. Any other search direction can be obtained by a linear combination of these directions.
- This will make our algorithm deterministic since the eigenvector directions are always unique.

The value of the objective function is evaluated along these eigenvector directions with some small step size increments. Since the starting position is a local optimal solution, one will see a steady decline in the function value during the initial steps; we call this the *descent stage*. Since the Hessian is obtained only once during the entire procedure, it is more efficient compared to Newton's method where an approximate Hessian is obtained for every iteration. After a certain number of evaluations, there may be an increase in the value indicating that the current point is out of the convergence region of the local maximum. The point along this direction where $A(Q)$ has the lowest value is called the *exit point*. Once the exit point has been reached, few more evaluations are made in the direction of the same eigenvector to ensure that one has left the original region of convergence. This procedure is clearly shown in Fig 9. Applying the local method directly from the exit point may give the original local maximum. The ascent stage is used to ensure that the new guess is in a different convergence zone. Hence, given the best local maximum obtained using any current local methods, this framework allows us to systematically escape out of the local maximum to explore surrounding local maxima. The complete algorithm is shown below :

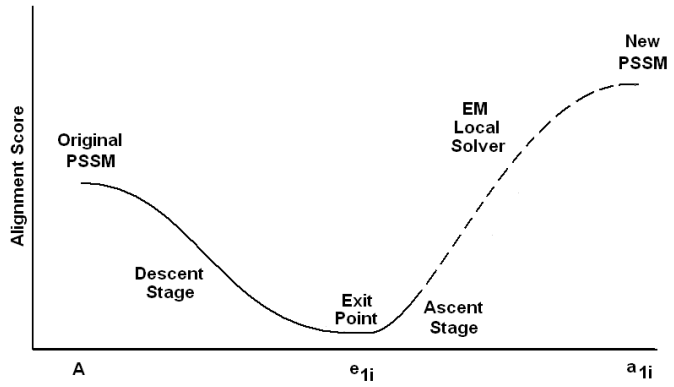


Fig. 9. A summary of escaping out of the local optimum to the neighborhood local optimum. Observe the corresponding trend of $A(Q)$ at each step.

Input: The DNA sequences, length of the motif (l), Maximum Number of Mutations (d)
Output: Motif (s)
Algorithm:
Step 1: Given the sequences, apply Random Projection algorithm to obtain different set of alignments.
Step 2: Choose the promising buckets and apply EM algorithm to refine these alignments.
Step 3: Apply the exit point method to obtain nearby promising local optimal solutions.
Step 4: Report the consensus pattern that corresponds to the best alignments and their corresponding PSSM.

The new framework can be treated as a hybrid approach between global and local methods. It differs from traditional local methods by computing multiple local solutions in the neighborhood region in a systematic manner. It differs from global methods by working completely in the profile space and searching a subspace efficiently in a deterministic manner. For a given non-convex function, there is a massive number of convergence regions that are very close to each other and are separated from one another in the form of different basins of attraction. These basins are effectively modeled by the concept of stability regions.

4.4 Implementation Details

Our program was implemented on Red Hat Linux version 9 and runs on a Pentium IV 2.8 GHz machine. The core algorithm that we have implemented is XP_EM described in Algorithm 3. XP_EM obtains the initial alignments and the original data sequences along with the length of the motif. It returns the best motif that is obtained in the neighboring region of the sequences. This procedure constructs the PSSM, performs EM refinement, and then computes the Tier-1 and Tier-2 solutions by calling the procedure $Next\_Tier$. The eigenvectors of the Hessian were computed using the source code obtained from [46]. $Next\_Tier$ takes a PSSM as an input and computes an array of PSSMs corresponding to the next tier local maxima using the exit point methodology.

Given a set of initial alignments, Algorithm 3 will find the best possible motif in the neighborhood space of the profiles. Initially, a PSSM is computed using $construct\_PSSM$ from the given alignments. The procedure $Apply\_EM$ will return a new PSSM that corresponds to the alignments obtained after the EM algorithm has been applied to the initial PSSM. The details of the procedure $Next\_Tier$ are given in Algorithm 4. From a given local solution (or PSSM),

Algorithm 3 Motif $XP\_EM(init\_aligns, seqs, l)$

$PSSM = Construct\_PSSM(init\_aligns)$
$New\_PSSM = Apply\_EM(PSSM, seqs)$
$TIER1 = Next\_Tier(seqs, New\_PSSM, l)$
for $i = 1$ to $3l$ **do**
  if $TIER1[i] <> zeros(4l)$ **then**
    $TIER2[i][\,] = Next\_Tier(seqs, TIER1[i], l)$
  end if
end for
Return $best(PSSM, TIER1, TIER2)$

$Next\_Tier$ will compute all the $3l$ new $PSSM$s in the neighborhood of the given local optimal solution. The second tier patterns are obtained by calling the $Next\_Tier$ from the first tier solutions [3]. Finally, the pattern with the highest score amongst all the PSSMs is returned.

Algorithm 4 PSSMs[ ] $Next\_Tier(seqs, PSSM, l)$

$Score = eval(PSSM)$
$Hess = Construct\_Hessian(PSSM)$
$Eig[\,] = Compute\_EigVec(Hess)$
$MAX\_Iter = 100$
for $k = 1$ to $3l$ **do**
  $PSSMs[k] = PSSM$ $Count = 0$
  $Old\_Score = Score$ $ep\_reached = FALSE$
  while (! $ep\_reached$) && ($Count < MAX\_Iter$) **do**
    $PSSMs[k] = update(PSSMs[k], Eig[k], step)$
    $Count = Count + 1$
    $New\_Score = eval(PSSMs[k])$
    if ($New\_Score > Old\_Score$) **then**
      $ep\_reached = TRUE$
    end if
    $Old\_Score = New\_Score$
  end while
  if $count < MAX\_Iter$ **then**
    $PSSMs[k] = update(PSSMs[k], Eig[k], ASC)$
    $PSSMs[k] = Apply\_EM(PSSMs[k], Seqs)$
  else
    $PSSMs[k] = zeros(4l)$
  end if
end for
Return $PSSMs[\,]$

[3] New PSSMs might not be obtained for certain search directions. In those cases, a zero vector of length $4l$ is returned. Only those new PSSMs which do not have this value will be used for any further processing.

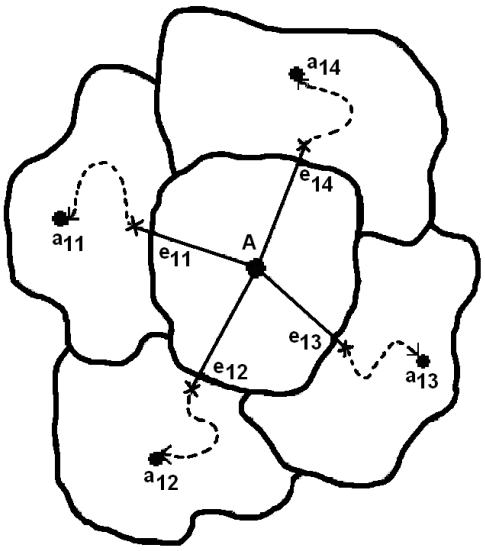


Fig. 10. Diagram illustrates the exit point method of escaping from the original solution (A) to the neighborhood local optimal solutions (a_{1i}) through the corresponding exit points (e_{1i}). The dotted lines indicate the local convergence of the EM algorithm.

The procedure $Next\_Tier$ takes a PSSM, applies the Exit-point method and computes an array of PSSMs that corresponds to the next tier local optimal solutions. The procedure *eval* evaluates the scoring function for the PSSM using (29). The procedures $Construct\_Hessian$ and $Compute\_EigVec$ compute the Hessian matrix and the eigenvectors respectively. MAX_iter indicates the maximum number of uphill evaluations that are required along each of the eigenvector directions. The neighborhood PSSMs will be stored in an array variable $PSSMs[\,]$. The original PSSM is updated with a small step until an exit point is reached or the number of iterations exceeds the MAX_Iter value. If the exit point is reached along a particular direction, some more iterations are run to guarantee that the PSSM has exited the original stability region and has entered a new one. The EM algorithm is then used during this ascent stage to obtain a new PSSM [4].

The initial alignments are converted into the profile space and a PSSM is constructed. The PSSM is updated (using the EM algorithm) until the alignments converge to a local optimal solution. The Exit-point methodology is then employed to escape out of this local optimal solution to compute nearby first tier local optimal solutions. This process is then repeated on promising first tier solutions to obtain second tier solutions. As shown in Fig. 11, from

[4] For completeness, the entire algorithm has been shown in this section. However, during the implementation, several heuristics have been applied to reduce the running time of the algorithm. For example, if the first tier solution is not very promising, it will not be considered for obtaining the corresponding second tier solutions.

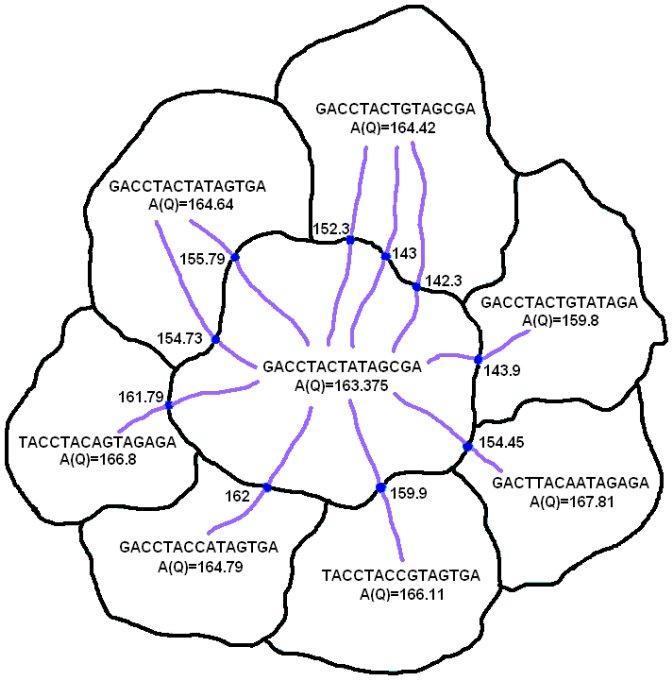


Fig. 11. 2-D illustration of first tier improvements in a $3l$ dimensional objective function. The original local maximum has a score of 163.375. The various Tier-1 solutions are plotted and the one with highest score (167.81) is chosen.

the original local optimal solution, various exit points and their corresponding new local optimal solutions are computed along each eigenvector direction. Sometimes two directions may yield the same local optimal solution. This can be avoided by computing the saddle point corresponding to the exit point on the stability boundary [6]. There can be many exit points, but there will only be a unique saddle point corresponding to the new local minimum. However, in high dimensional problems, this is not very efficient. Hence, we have chosen to compute the exit points. For computational efficiency, the Exit-point approach is only applied to promising initial alignments (i.e. random starts with higher Information Content score). Therefore, a threshold $A(Q)$ score is determined by the average of the three best first tier scores after 10-15 random starts; any current and future first tier solution with scores greater than the threshold is considered for further analysis. Additional random starts are carried out in order to aggregate at least ten first tier solutions. The Exit-point method is repeated on all first tier solutions above a certain threshold to obtain second tier solutions.

4.5 Experimental Results

Experiments were performed on both synthetic data and real data. Two different methods were used in the global phase: random start and random projec-

tion. The main purpose of our work is not to demonstrate that our algorithm can outperform the existing motif finding algorithms. Rather, the main work here focuses on improving the results that are obtained from other efficient algorithms. We have chosen to demonstrate the performance of our algorithm on the results obtained from the random projection method which is a powerful global method that has outperformed other traditional motif finding approaches like MEME, Gibbs sampling, WINNOWER, SP-STAR, etc. [29]. Since the comparison was already published, we mainly focus on the performance improvements of our algorithm as compared to the random projection algorithm. For the random start experiment, a total of N random numbers between 1 and $(t - l + 1)$ corresponding to initial set of alignments are generated. We then proceeded to evaluate our Exit-point methodology from these alignments.

4.5.1 Synthetic Datasets

The synthetic datasets were generated by implanting some motif instances into $t = 20$ sequences each of length 600. Let m correspond to one full random projection + EM cycle. We have set $m = 1$ to demonstrate the efficiency of our approach. We compared the performance coefficient (PC) which gives a measure of the average performance of our implementation compared to that of Random Projection. The PC is given by :

$$PC = \frac{|K \cap P|}{|K \cup P|} \tag{34}$$

where K is the set of the residue positions of the planted motif instances, and P is the corresponding set of positions predicted by the algorithm. Table 8 gives an overview of the performance of our method compared to the random projection algorithm on the (l,d) motif problem for different l and d values.

Our results show that by branching out and discovering multiple local optimal solutions, higher m values are not needed. A higher m value corresponds to more computational time because projecting the l-mers into k-sized buckets is a time consuming task. Using our approach, we can replace the need for randomly projecting l-mers repeatedly in an effort to converge to a global optimum by deterministically and systematically searching the solution space modeled by our dynamical system and improving the quality of the existing solutions. The improvements of our algorithm are clearly shown in Table 8. We can see that for higher length motifs, the improvements are more significant.

Fig. 11 shows the Tier-1 solutions obtained from a given consensus pattern. Since the exit points are being used instead of saddle points, our method might sometimes find the same local optimal solution obtained before. As seen from

the figure, the Tier-1 solutions can differ from the original pattern by more than just one nucleotide position. Also, the function value at the exit points is much higher than the original value.

Table 7
The consensus patterns and their corresponding scores of the original local optimal solution obtained from multiple random starts on the synthetic data. The best first tier and second tier optimal patterns and their corresponding scores are also reported.

(l,d)	Initial Pattern	Score	First Tier Pattern	Score	Second Tier Pattern	Score
(11,2)	AACGGTCGCAG	125.1	CCCGGTCGCTG	147.1	CCCGGGAGCTG	153.3
(11,2)	ATACCAGTTAC	145.7	ATACCAGTTTC	151.3	ATACCAGGGTC	153.6
(13,3)	CTACGGTCGTCTT	142.6	CCACGGTTGTCTC	157.8	CCTCGGGTTTGTC	158.7
(13,3)	GACGCTAGGGGGT	158.3	GAGGCTGGGCAGT	161.7	GACCTTGGGTATT	165.8
(15,4)	CCGAAAAGAGTCCGA	147.5	CCGCAATGACTGGGT	169.1	CCGAAAGGACTGCGT	176.2
(15,4)	TGGGTGATGCCTATG	164.6	TGGGTGATGCCTATG	166.7	TGAGAGATGCCTATG	170.4
(17,5)	TTGTAGCAAAGGCTAAA	143.3	CAGTAGCAAAGACTACC	173.3	CAGTAGCAAAGACTTCC	175.8
(17,5)	ATCGCGAAAGGTTGTGG	174.1	ATCGCGAAAGGATGTGG	176.7	ATTGCGAAAGAATGTGG	178.3
(20,6)	CTGGTGATTGAGATCATCAT	165.9	CAGATGGTTGAGATCACCTT	186.9	CATTTAGCTGAGTTCACCTT	194.9
(20,6)	GGTCACTTAGTGGCGCCATG	216.3	GGTCACTTAGTGGCGCCATG	218.8	CGTCACTTAGTCGCGCCATG	219.7

As opposed to stochastic processes like mutations in genetic algorithms, our approach reduces the stochastic nature and obtains the nearby local optimal solutions systematically. Fig. 12 shows the performance of the Exit-point approach on synthetic data for different (l,d) motifs. The average scores of the ten best solutions obtained from random starts and their corresponding improvements in Tier-1 and Tier-2 are reported. One can see that the improvements become more prominent as the length of the motif is increased. Table 7 shows the best and worst of these top ten random starts along with the consensus pattern and the alignment scores.

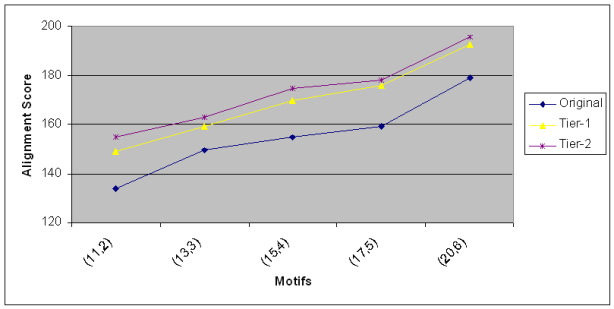


Fig. 12. The average scores with the corresponding first tier and second tier improvements on synthetic data using the random starts with Exit-point approach with different (l,d) motifs.

With a few modifications, more experiments were conducted using the Random Projection method. The Random Projection method will eliminate non-promising regions in the search space and gives a number of promising sets of initial patterns. EM refinement is applied to only the promising initial patterns. Due to the robustness of the results, the Exit-point method is employed only on the top five local optima. The Exit-point method is again repeated on

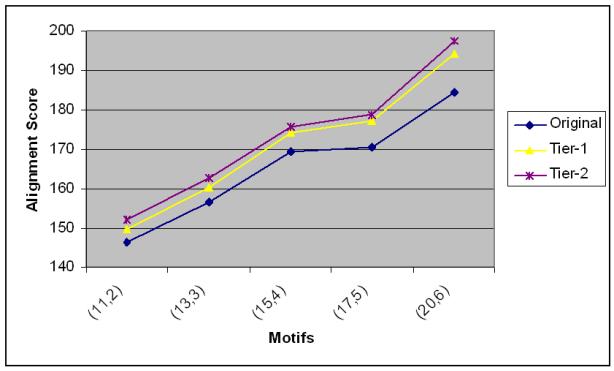


Fig. 13. The average scores with the corresponding first tier and second tier improvements on synthetic data using the Random Projection with Exit-point approach with different (l,d) motifs.

Table 8
The results of performance coefficient with $m = 1$ on synthetically generated sequences. The IC scores are not normalized and the perfect score is 20 since there are 20 sequences.

Motif (l,d)	PC obtained using Random Projection	PC obtained using Exit-point method
(11,2)	20	20
(15,4)	14.875	17
(20,6)	12.667	18

the top scoring first tier solutions to arrive at the second tier solutions. Fig. 13 shows the average alignment scores of the best random projection alignments and their corresponding improvements in tier-1 and tier-2 are reported. In general, the improvement in the first tier solution is more significant than the improvements in the second tier solutions.

4.5.2 Real Datasets

Table 9 shows the results of the Exit-point methodology on real biological sequences. We have chosen $l = 20$ and $d = 2$. 't' indicates the number of sequences in the real data. For the biological samples taken from [29,38], the value m once again is the average number of random projection + EM cycles required to discover the motif. All other parameter values (like projection size $k = 7$ and threshold s=4) are chosen to be the same as those used in the Random projection paper [29]. All of the motifs were recovered with $m = 1$ using the Exit-point strategy. Without the exit point strategy, the Random Projection algorithm needed *multiple cycles* (m=8 in some cases and m=15 in others) in order to retrieve the correct motif. This elucidates the fact that global methods can only be used to a certain extent and should be combined

with refined local heuristics in order to obtain better efficiency. Since the random projection algorithm has outperformed other prominent motif finding algorithms like SP-STAR, WINNOWER, Gibbs sampling etc., we did not repeat the same experiments that were conducted in [29]. Running one cycle of random projection + EM is much more expensive computationally. The main advantage of our strategy comes from the deterministic nature of our algorithm in refining motifs.

Table 9
Results of Exit-point method on biological samples. The real motifs were obtained in all the six cases using the Exit-point framework.

Sequence	Sample Size	t	Best (20,2) Motif	Reference Motif
E. coli CRP	1890	18	TGTGAAATAGATCACATTTT	TGTGANNNNGNTCACA
preproinsulin	7689	4	GGAAATTGCAGCCTCAGCCC	CCTCAGCCC
DHFR	800	4	CTGCAATTTCGCGCCAAACT	ATTTCNNGCCA
metallothionein	6823	4	CCCTCTGCGCCCGGACCGGT	TGCRCYCGG
c-fos	3695	5	CCATATTAGGACATCTGCGT	CCATATTAGAGACTCT
yeast ECB	5000	5	GTATTTCCCGTTTAGGAAAA	TTTCCCNNTNAGGAAA

The Exit-point framework broadens the search region in order to obtain an improved solution which may potentially correspond to a better motif. In most of the profile based algorithms, EM is used to obtain the nearest local optimum from a given starting point. In our approach, we consider the boundaries of these convergence regions and find the surrounding local optimal solutions based on the theory of stability regions. We have shown on both real and synthetic data sets that beginning from the EM converged solution, the Exit-point approach is capable of searching in the neighborhood regions for another solution with an improved information content score. This will often translate into finding a pattern with less hamming distance from the resulting alignments in each sequence. Our approach has demonstrated an improvement in the score on all datasets that it was tested on. One of the primary advantages of the Exit-point methodology is that it can be used with different global and local methods. The main contribution of our work is to demonstrate the capability of this hybrid EM algorithm in the context of the motif finding problem. Our algorithm can potentially use any global method and improve its results efficiently. From our results, we see that motif refinement stage plays a vital role and can yield accurate results deterministically. We would like to continue our work by combining other global methods available in the literature with existing local solvers like EM or GibbsDNA that work in continuous space. By following the example of [32], we may improve the chances of finding more promising patterns by combining our algorithm with different global and local methods.

5 Supervised Learning - Training Neural Networks

In this section, we demonstrate the ability of the stability region based methods to obtain multiple local minima on the error surface during the training procedure of artificial neural networks (ANN) [47]. ANNs are powerful statistical machine learning tools that are widely used in several domains of science and engineering for problems like function approximation, timeseries prediction, medical diagnosis, character recognition, load forecasting, speaker identification, risk management etc. They were developed in analogy to the human brain for the purpose of improving conventional learning capabilities. These networks serve as excellent approximators of nonlinear continuous functions [48]. However, using an artificial neural network to model a system involves in dealing with certain difficulties in achieving the best representation of the classification problem. The two challenging tasks in the process of learning using ANNs are network architecture selection and optimal training. In deciding the architecture, a larger network will always provide better prediction accuracy for the data available [49]. However, such a large network that is too complicated to model and also it might be customized to some given problem and hence will lose its generalization capability for the unseen data. Also, every additional node into the network translates to increased hardware cost.

Hence, it is vital to develop algorithms that can exploit the power of a given network architecture and this can be achieved by obtaining the global minimum of the error on the training data along with the cross validation procedure. The main goal of optimal training of the network is to find a set of weights that achieves the global minimum of the mean square error (MSE) [50]. We will consider a simple one hidden layer neural network with n (number of features) input nodes, k hidden nodes and 1 output node. Thus, each network has nk weights and k biases to the hidden layer, and k weights and one bias to the output node. Hence, training a neural network effectively is necessarily a search problem of dimension $s = (n + 2)k + 1$. The network is trained to deliver the output value (Y_i) of the the i^{th} sample at the output node which will be compared to the actual target value (t_i). The local minima problem is one of the thoroughly studied aspects of neural networks [51]. To avoid this problem, some global methods like multiple random starts, genetic algorithms, simulated annealing etc. can identify promising regions of the weight space, but are time-consuming and are essentially stochastic in nature. Expecting such stochastic algorithms to fine-tune the training weights will be even more time consuming. Thus, there is a necessity to efficiently search for good solutions in promising regions of the solution space. The main focus of this is to develop a robust training algorithm for obtaining the optimal set of weights of an artificial neural network by searching the parameter subspace in a systematic manner using the concepts of stability regions that were presented earlier.

The performance of a network is usually gauged by measuring the mean square error (MSE) of its outputs from the expected targets. The goal of optimal training is to find a set of parameters that achieves the global minimum of the MSE [52,53,48]. For a n-dimensional dataset, the MSE over Q samples in the training set is given by:

$$C(W) = \frac{1}{Q} \sum_{i=1}^{Q} [t(i) - y(X, W)]^2 \tag{35}$$

where t is the target output, X is the input vector and W is the weight vector. The MSE as a function of the parameters will adopt a complex topology with several local optimal solutions. The network's weights and thresholds must be set so as to minimize the prediction error made by the network. Since it is not possible to analytically determine the global minimum of the error surface, the neural network training is essentially an exploration of the error surface for an optimal set of parameters that attain this global minimum.

Training algorithms can be broadly classified into '*local*' and '*global*' methods. Local methods begin at some initial guesses and deterministically lead to a nearby local minimum. From an initial random configuration of weights and thresholds, the local training methods incrementally seek for improved solution until they reach local minima. Typically, some form of the gradient information at the current point on the error surface is calculated and used to make a downhill move. Based on the movement towards improved solutions, local methods can be subdivided into two categories: (1) Line search methods that minimize the function repeatedly along some descent directions until the local minimum is reached (e.g. Newton's method, BFGS method and conjugate gradient methods). (2) Trust region methods where the surface is assumed to be a simple model (like a parabola) such that the minimum can be located directly if the model assumption is good (e.g. Levenberg-Marquardt(LM) method). More details on these methods are described in [50].

All these local methods discussed so far assume that they already have an initial guess to begin with. Due to the high non-linearity of the error surface, the quality of the final solution depends significantly on the initial set of parameters available. In practice, most of these methods are typically combined with stochastic global methods which yield a promising set of parameters in the weight space. These global methods explore the entire error surface and thus the chance of attaining a near-global optimal solution is high. Advanced techniques like Genetic algorithms and simulated annealing are applied in combination with standard BP inorder to allow for more promising solutions

and avoid being stuck at local minima [54]. Inspite of their (asymptotic) guarantees in convergence to the global minimum, even for simple learning tasks, these usually exhibits are very slow. Also, these methods can explore the entire solution space effectively and obtain promising local optimal solutions, but they lack fine-tuning capability to obtain a precise final solution and require a local methods to be employed. From both of the methods mentioned above, one can realize that there is a clear gap between these methods and our approach tries to communicate with both local and global methods.

Probably, the algorithms that resemble our methodology are TRUST [44] and dynamic tunneling [55]. These methods attempt to move out of the local minimum in a stochastic manner. The training algorithm proposed here differs from these methods by deterministically escaping out of the local minimum and systematically exploring multiple local minima on the error surface in a tier-by-tier manner in order to advance towards the global minimum. This approach is based on the fundamental concepts of stability regions that were established in [7,4]. Basically, a global method yields initial points in certain promising regions of the search space. These promising initial points are used to search the neighborhood subspace in a systematic manner. TRUST-TECH relies on a robust, fast local method to obtain a local optimal solution. It explores the subspace in a tier-by-tier manner by transforming the function into its corresponding dynamical system and exploring the neighboring stability regions. Thus, it gives a set of promising local optimal solutions from which a global minimum is selected. In this manner, TRUST-TECH can be treated as an effective interface between the global and local methods, which enables the communication between these two methods. It also allows the flexibility of choosing different global and local methods depending on their availability and performance for certain specific classification tasks. Also, using our procedure, we will be able to truncate global methods at early stages and not use them to fine-tune the solutions and thus saving a lot of computational effort.

5.2 Problem Formulation

Table 10 gives the important notations used here. The final nonlinear mapping of the network model is given by:

$$y(W,X) = \phi_2 \left(\sum_{j=1}^{k} w_{0j} \phi_1 \left(\sum_{i=1}^{n} w_{ij} x_i + b_j \right) + b_0 \right)$$

where ϕ_1 and ϕ_2 are the activation functions of the hidden nodes and the output nodes respectively. ϕ_1 and ϕ_2 can be same functions or can be different functions. We have chosen to use ϕ_2 to be sigmoidal and ϕ_1 to be linear. Results in the literature [56], suggest that this set of activation functions yield the best

Table 10
Description of the notations used

Notation	Description
Q	Number of training samples
X	Input vector
W	Weight vector
n	Number of features
k	Number of hidden nodes
w_{0j}	weight for j^{th} hidden and the output node
w_{ij}	weight for i^{th} input and j^{th} hidden node
b_0	bias of the output node
b_j	bias of the j^{th} hidden node
t_i	target value of the i^{th} input sample
y	output of the network
e_i	Error for the i^{th} input sample

results for feedforward neural networks. The task of the network is to learn associations between the input-output pairs $(X_1, t_1), (X_2, t_2), ..., (X_Q, t_Q)$. Without loss of generality, lets construct the following weight vector :

$$w{=}(w_{01}, w_{02}, .., w_{0k}, .., w_{n1}, w_{n2}, .., w_{nk}, b_0, b_1, b_2.., b_k)^T$$

which includes all the weights and biases that are to be computed. Hence, the problem of training neural networks is s-dimensional unconstrained minimization problem where $s = (n + 2)k + 1$. The mean squared error which is to be minimized can be written as

$$C(w) = \frac{1}{Q} \sum_{i=1}^{Q} e_i^2(w) \tag{36}$$

where the error $e_i(w) = t_i - y(w, x_i)$. The error cost function $C(\cdot)$ averaged over all training data is a highly nonlinear function of the synaptic vector w Ignoring the constant for simplicity, it can be shown that

$$\nabla C(w) = J^T(w)e(w) \tag{37}$$

$$\nabla^2 C(w) = J^T(w)J(w) + S(w) \tag{38}$$

where $J(w)$ is the Jacobian matrix

$$J(w) = \begin{bmatrix} \frac{\partial e_1}{\partial W_1} & \frac{\partial e_1}{\partial W_2} & . & . & \frac{\partial e_1}{\partial W_s} \\ \frac{\partial e_2}{\partial W_1} & \frac{\partial e_2}{\partial W_2} & . & . & \frac{\partial e_2}{\partial W_s} \\ . & . & . & . & \\ . & . & . & . & \\ \frac{\partial e_Q}{\partial W_1} & \frac{\partial e_Q}{\partial W_2} & . & . & \frac{\partial e_Q}{\partial W_s} \end{bmatrix}$$

and

$$S(w) = \sum_{i=1}^{Q} e_i(w) \nabla^2 e_i(w) \tag{39}$$

Generally, if we would like to minimize $J(w)$ with respect to the parameter vector w, any variation of Newton's method can be written as

$$\begin{aligned} \Delta w &= -\left[\nabla^2 C(w)\right]^{-1} \nabla C(w) \\ &= -\left[J^T(w) J(w) + S(w)\right]^{-1} J^T(w) e(w) \end{aligned} \tag{40}$$

5.3 Stability Region based Approach

In this work, we exploit the geometric and topological structure of the error surface to explore multiple local optimal solutions in a systematic manner. Firstly, we describe the transformation of the original problem into its corresponding nonlinear dynamical system and then propose a new algorithm for finding multiple local optimal solutions.

5.3.1 Problem Transformation

This section mainly deals with the transformation of the original likelihood function into its corresponding nonlinear dynamical system and introduces some terminology pertinent to comprehend our algorithm. This transformation gives the correspondence between all the critical points of the error surface and that of its corresponding gradient system. To analyze the geometric structure of the error surface, we build a *generalized gradient system* described by

$$\frac{dw}{dt} = -grad_R\, C(w) = -R(w)^{-1} \nabla C(w) \tag{41}$$

where the error function C is assumed to be twice differentiable to guarantee unique solution for each initial condition $w(0)$ and $R(w)$ is a positive definite symmetric matrix (also known as *Reimannian metric*) for all $w \in \Re^s$.

It is interesting to note the relationship between (41) and (40) and obtain different local solving methods used to find the nearest local optimal solution with guaranteed convergence. For example, if $R(w) = I$, then it is a naive error back-propagation algorithm. If $R(w) = [J(w)^T J(w)]$ then it is the Gauss-Newton method and if $R(w) = [J(w)^T J(w) + \mu I]$ then it is the Levenberg-Marquardt method.

5.3.2 Algorithm

The proposed algorithm uses a promising starting point (A^*) as input and outputs the best local minimum of the neighborhood in the weight space.

Input: Initial guess(A^*), Tolerance (τ), Step size (s)
Output: Best neighborhood local minimum (A_{ij})
Algorithm:
Step 1: Obtaining good initial guess (A^*): The initial guess for the algorithm can be obtained from other global search methods or from a purely random start. Some domain knowledge about the specific dataset that is being trained on, might help in eliminating non-promising set of initial weights.
Step 2: Moving to the local minimum (m_i): Using an appropriate local solver, the local optimum m_i is obtained using the A^* as initial guess. The starting point will be x_i for the later stages.
Step 3: Determining the search direction (d_j): The eigenvectors d_i of the Jacobian are computed at m_i. These eigenvector directions might lead to promising regions of the subspace. Other search directions can also be chosen based on the specific problem that is being dealt.
Step 4: Escaping from the local minimum: Taking small step sizes away from m_i along d_i directions increases the objective function value till it hits the stability boundary. However, the objective function value then decreases after the search trajectory passes the exit point on the stability boundary. x_i is used as initial guess and local solver is applied again (go to Step 2).
Step 5: Finding Tier-1 local minima (A_{1i}): Exploring the neighborhood of the optimum solution local to the initial guess leads to tier 1 local optima. Exploring from tier k local optima leads to tier $k+1$ local optima.
Step 6: Exploring Tier-k local minima (A_{kj}): Explore all other tiers in a similar manner described above. Selecting the best solution from all these tiers, the global optimum of desired quality can be obtained.
Step 7: Termination Criteria: The procedure can be terminated when the best solution obtained so far is satisfactory (lesser than sol_{req}) or a predefined maximum number of tiers is explored.

Fig. 14 illustrates two tier TRUST-TECH methodology. The '*' represents

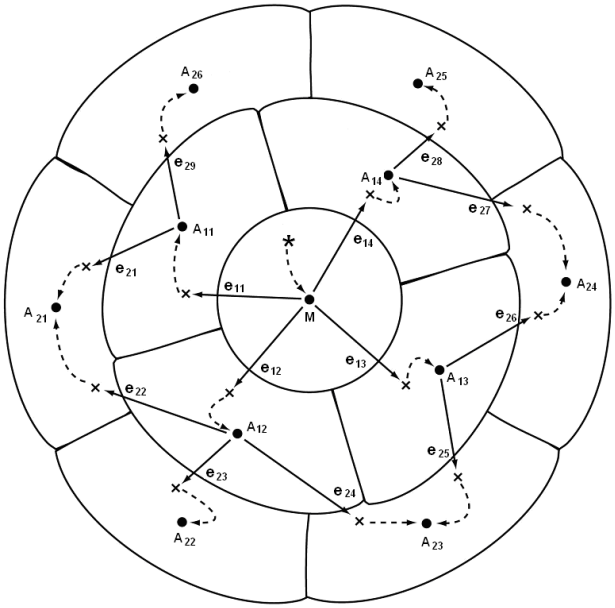


Fig. 14. Illustration of Tier-2 TRUST-TECH procedure for obtaining neighborhood local minima.

the initial guess. Dotted arrows represent the convergence of the local solver. Solid arrows represent the gradient ascent linear searches along eigenvector directions. 'X' indicates a new initial condition in the neighborhood stability region. M represents the local minimum obtained by applying local methods from 'X'. A_{1i} indicates Tier-1 local minima. e_{1i} are the exit points between M and A_{1i}. Similarly, A_{2j} and e_{2j} are the second-tier local minima and their corresponding exit points respectively.

5.4 Implementation Details

All programs were implemented in MATLAB v6.5 and run on Pentium IV 2.8 GHz machines. We will now describe some implementation issues of our algorithm.

5.4.1 Network Architecture and Training

As described in the introduction section, we have chosen to demonstrate the capability of our newly proposed algorithm on a network with single hidden layer and an output layer with only one output node. This architecture is not complicated and has the capability to precisely demonstrate the problems with the existing approaches. Each hidden node has a tangent-sigmoid transfer function and the output node has a pure linear transfer function. The number of nodes in the hidden layer is determined by increasing number of nodes, and selecting the number of nodes that achieves a compromise between

minimal error value and minimal number of nodes. The trust region based Levenberg-Marquardt algorithm is chosen because of efficiency in terms of time complexity and space requirements. It utilizes the approximation of the Jacobian in its iterative gradient descent, which will be used for generating promising directions in TRUST-TECH. Two different initialization schemes were implemented. The most basic global method which is multiple random starts with initial set of parameters between -1 and 1. More effective global method namely Nguyen-Widrow (NW) algorithm [57] has also been used to test the performance of our algorithm. The NW algorithm is implemented as the standard initialization procedure in MATLAB. In both cases, the best initial set of parameters in terms of training error is chosen and improved with our TRUST-TECH algorithm.

5.4.2 TRUST-TECH

Algorithm 5 $TRUST\_TECH(NET, Wts, ss, \tau)$

$Wts = Train(NET, Wts, \tau)$
$Error = Estimate(NET, Wts)$
$Thresh = c * Error$
$Wts1[\,] = Neighbors(NET, Wts, ss, \tau)$
for $k = 1$ to $size(Wts1)$ **do**
  if $Estimate(NET, Wts1[k]) < Thresh$ **then**
    $Wts2[k][\,] = Neighbors(NET, Wts1, ss, \tau)$
  end if
end for
Return $best(Wts, Wts1, Wts2)$

It is effective to use TRUST-TECH methodology for those promising solutions obtained from stochastic global methods. Hence, our algorithm assumes that it is being invoked from a promising set of initial parameters. Algorithm 5 describes the two-tier TRUST-TECH algorithm. NET assumes to have a fixed architecture with a single output node. ss is the step size required for moving out of the stability region to obtain the exit point. τ is the tolerance of error used for the convergence of the local method. $Weights$ give the initial set of weight parameter values. $Train$ function implements the Levenberg-Marquardt method that obtains the local optimal solution from the initial condition. The procedure $Estimate$ computes the mean square error (MSE) value of the network model. A threshold value ($Thresh$) is set based on this MSE value. The procedure $Neighbors$ returns all the next tier local optimal solutions from a given local solution. After obtaining all the tier-1 solutions, Neighbors is again invoked (only for promising solutions) to obtain the second-tier solutions. The algorithm finally compares the initial solution, tier-1 and tier-2 solutions and returns the network corresponding to the lowest error among all these solutions.

Algorithm 6 $Wts[\,]\ Neighbors\ (NET, Wts,\ ss,\ \tau)$

$[Wts, Hess] = Train(NET, Wts, \tau)$
$evec = Eig\_Vec(Hess)$
$Wts[\,] = NULL$
for $k = 1$ to $size(evec)$ **do**
  $Old\_Wts = Wts$
  $ext\_Pt = Find\_Ext(NET, Old\_Wts, ss, evec[k])$
  if (ext_Pt) **then**
    $New\_Wts = Move(NET, Old\_Wts, evec[k])$
    $New\_Wts = Train(NET, New\_Wts, \tau)$
    $Errors = Estimate(NET, New\_Wts)$
    $Wts[\,] = Append(Wts[\,], New\_Wts, Errors)$
  end if
end for
Return $Wts[\,]$

The approximate Hessian matrix used for updation in the Levenberg-Marquardt method is utilized to obtain the search direction. Since there is no optimal way of obtaining promising search directions, the Eigen vectors of this Hessian matrix are used as search directions. Along each search direction, the exit point is obtained by evaluating the function value along that particular direction. The step size for evaluation is chosen to be the average step size taken during the convergence of the local procedure. The function value increases initially and then starts to reduce indicating the presence of exit point on the stability boundary. *Move* function ensures that a new point (obtained from the exit point) is located in a different (neighboring) stability region. From this new initial guess, the local method is applied again to obtain the local optimal solution of the neighborhood stability region. The search for exit points along these directions will be stopped if exit points are not found after certain number of function evaluations.

Table 11
Summary of Benchmark Datasets. Dataset (D), Sample size (Q), Input features (n), Output classes (p), No. of Hidden nodes (k), No. of search variables ((n+2)k+1)

D	Q	n	p	k	(n+2)k+1
Cancer	683	9	2	5	56
Diabetes	178	8	3	4	61
Image	2310	19	7	8	169
Ionosphere	351	34	2	9	325
Iris	150	4	3	3	19
Sonar	208	60	2	8	497
Wine	178	13	3	4	61

Table 12
Percentage improvements in the classification accuracies over the training and test data using TRUST-TECH with multiple random restarts.

Dataset	Train Error			Test Error		
	MRS+BP	TRUST-TECH	Improvement	MRS+BP	TRUST-TECH	Improvement
Cancer	2.21	1.74	27.01	3.95	2.63	50.19
Image	9.37	8.04	16.54	11.08	9.74	13.76
Ionosphere	2.35	0.57	312.28	10.25	7.96	28.77
Iris	1.25	1.00	25.00	3.33	2.67	24.72
Diabetes	22.04	20.69	6.52	23.83	20.58	15.79
Sonar	1.56	0.72	116.67	19.17	12.98	47.69
Wine	4.56	3.58	27.37	14.94	6.73	121.99

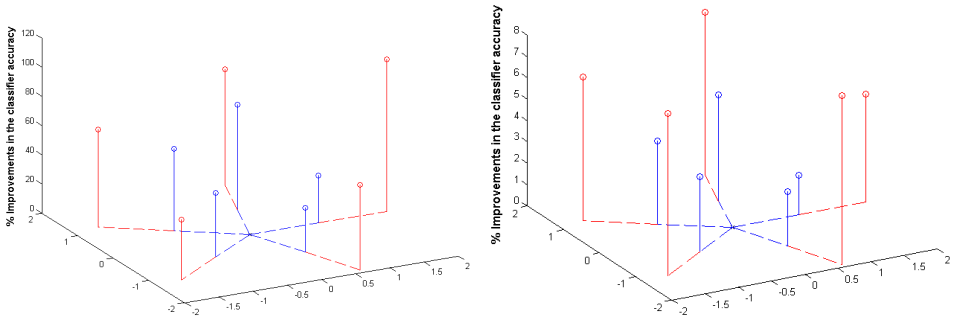
Table 13
Percentage improvements in the classification accuracies over the training and test data using TRUST-TECH with MATLAB initialization.

Dataset	Train Error			Test Error		
	NW+BP	TRUST-TECH	Improvement	NW+BP	TRUST-TECH	Improvement
Cancer	2.25	1.57	42.99	3.65	3.06	19.06
Image	7.48	5.17	44.82	9.39	7.40	26.90
Ionosphere	1.56	0.92	69.57	8.67	6.54	32.57
Iris	1.33	0.67	100.00	3.33	2.67	25.00
Diabetes	21.41	19.55	9.53	23.70	21.09	12.37
Sonar	2.35	0.42	456.96	17.26	14.38	20.03
Wine	7.60	1.62	370.06	14.54	4.48	224.82

5.5 Experimental Results

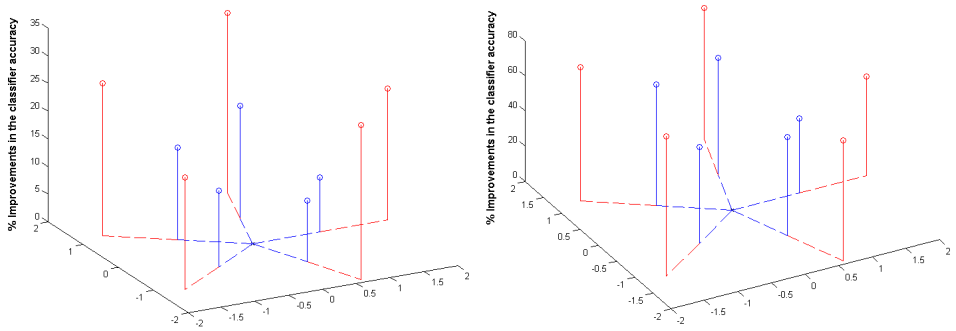
Our algorithm is evaluated using seven benchmark datasets in the UCI machine learning repository [25]. Optimal architectures have been fixed for the networks based on some heuristic training. The main focus of our algorithm is to demonstrate the capability of obtaining optimal weight parameters and improvements in the generalization ability of the networks. Table 5.4.2 summarizes the datasets. It gives the number of samples, input features, output classes along with the number of hidden nodes of the optimal architecture. These datasets have varying degrees of complexity in terms of sample size, output classes and the class overlaps. To demonstrate the generalization capability (and hence the robustness) of the training algorithm, 10-fold cross validation is performed on each dataset. The criteria of evaluation is given by the classification accuracy of the network model in terms of the percentage of misclassified samples in the test cases. Tables 5.4.2 and 5.4.2 shows the improvements in the train error and the test error using TRUST-TECH methodology when multiple random starts (MRS) and matlab initialization is used. Five tier-1 and corresponding tier-2 solutions were obtained using the TRUST-TECH strategy. For some of the datasets, there had been considerable improvements in the classifier performance. Spiderweb diagram shown in Fig. 15 a pretentious way to demonstrate the improvements in a tier-by-tier manner.

Fig.15 shows spider web diagrams showing the tier-1 and tier-2 improvements



(a) Wine Dataset

(b) Diabetes Dataset



(c) Cancer Dataset

(d) Image Dataset

Fig. 15. Spider web diagrams showing the tier-1 and tier-2 improvements using TRUST-TECH method on various benchmark datasets.

using TRUST-TECH method on various benchmark datasets. The circle in the middle of the plot represents the starting local optimal solution. The basic two dimensions are chosen arbitrarily for effective visualization and the vertical axis represents the percentage improvement in the classification accuracy. The two basic axes are chosen arbitrarily and the vertical axis represents the improvements in the classifier accuracy. The distances between each tier are normalized to unity and the improvements are averaged out across 10 folds. The five blue vertical lines surrounding the center circle are the best five local minima obtained from a tier 1 search across all folds. The five best second-tier improvements are plotted using red lines. It should be noted that the best first tier solution need not give the best second tier solution. Also, the second tier solution is as good as or better than the first tier solution.

This method provides an optimal set of training parameters for a neural network model thus allowing improved classification accuracies. Because of its non-probabilistic nature, multiple runs of our algorithm from any given initial guess will provide exactly the same result. The proposed method also allows the user to have the flexibility of choosing different global and local techniques for training.

The main advantages of the proposed TRUST-TECH framework are that it:

- Explores most of the neighborhood local optimal solutions unlike the 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 the traditional approaches and improves the quality of the solutions based on the maximum likelihood criteria.
- Helps the expensive global methods to truncate early.
- Exploits the fact that promising solutions are obtained by faster convergence of the EM algorithm.

6 Conclusion and Future Work

Global methods are powerful and stochastically search the entire parameter space and obtain promising regions. Local methods, on the other hand, are deterministic and usually converge to a locally optimal solution that is nearest to a given initial condition. Careful investigation reveals that there is a significant gap between these two methods and in some applications, the objective function that these methods try to optimize are slightly different. In other words, the objective scores for these methods are not well calibrated and hence, it is important to develop methods that can search a subspace. In this paper, we proposed stability region based methods for systematically exploring the parameter subspace to obtain the neighborhood local optimal solutions. Our method explores the dynamic and geometric characteristics of stability boundaries of a nonlinear dynamical system corresponding to the nonlinear function of interest. Basically, our method coalesces the advantages of the traditional local optimizers with that of the dynamic and geometric characteristics of the stability regions of the corresponding nonlinear dynamical system of the corresponding nonlinear function. These methods have been successfully used for machine learning and pattern discovery problems. Our algorithm has been tested on both synthetic and real datasets and the advantages of using this stability region based framework are clearly manifested. This framework not only reduces the sensitivity to initialization, but also allows the flexibility for the practitioners to use various global and local methods that work well for a particular problem of interest.

The stability region based Expectation-Maximization algorithm can be easily extended to other widely used EM related problems like k-means clustering,training Hidden Markov Models, Mixture of Factor Analyzers, Probabilistic Principal Component Analysis, Bayesian Networks etc. We would

also like to extend these techniques to Markov Chain Monte Carlo methods like Gibbs sampling for the estimation of mixture models. Several real-world applications such as image segmentation, speech processing and text classification can benefit heavily from these methods. Different global methods and local solvers will be used along with the stability region based framework to understand the flexibility. Frameworks for combining these methods with other hierarchical stochastic algorithms such as evolutionary algorithms and smoothing approaches are also proposed. The performance of such hierarchical algorithms will be tested on large scale applications. Extensions to constrained optimization problems also appears to be a very promising direction. Recently, there had been a lot of interest in handling constraints in the data. Clustering problems with Constraints might be added based on the prior information about the samples and a stability region based constrained EM algorithm can be developed [58].

Acknowledgements

We would like to thank Bala Rajaratnam for some valuable discussions on the EM algorithm. Thanks to undergraduates Yao-Chung Weng and Jay Huang for their help during the initial phase of this work.

References

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

[2] H. Chiang, M. Hirsch, F. F. Wu, Stability regions of nonlinear autonomous dynamical systems, IEEE Transactions on Automatic Control 33 (1988) 16–27.

[3] H. Chiang, A. Fekih-Ahmed, Quasi-stability regions of nonlinear dynamical systems:Theory, IEEE Transactions on Circuits and Systems 43 (1996) 627–635.

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

[5] D. Heidrich, W. Kliesch, W. Quapp, Properties of Chemical Interesting Potential Energy Surfaces, Springer, Lecture Notes in Chemistry 56, Berlin, 1991.

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

[7] H. Chiang, 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) (1996) 99–109.

[8] L. Lastras, Continuation methods and saddle points : a new framework, Master's thesis, Cornell University (1998).

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

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

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

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

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

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

[15] C. K. Reddy, H. D. Chiang, B. Rajaratnam, Stability region based expectation maximization for model-based clustering, in proceedings of sixth IEEE International Conference on Data Mining (ICDM) 2006 (2006) 522–531.

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

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

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

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

[20] R. M. Neal, G. E. Hinton, A new view of the EM algorithm that justifies incremental, sparse and other variants, in: M. I. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Publishers, 1998, pp. 355–368.
URL `citeseer.ist.psu.edu/neal93new.html`

[21] J. Q. Li, Estimation of mixture models, Ph.D. thesis, Department of Statistics,Yale University (1999).

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

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

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

[25] C. Blake, C. 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.

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

[27] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, J. Wootton, Detecting subtule sequence signals: a gibbs sampling strategy for multiple alignment, Science 262 (1993) 208–214.

[28] T. Bailey, C. Elkan, Fitting a mixture model by expectation maximization to discover motifs in biopolymers, The First International Conference on Intelligent Systems for Molecular Biology (1994) 28–36.

[29] J. Buhler, M. Tompa, Finding motifs using random projections, Proceedings of the fifth annual international conference on Research in computational molecular biology (2001) 69–76.

[30] P. Pevzner, S.-H. Sze, Combinatorial approaches to finding subtle signals in DNA sequences, The Eighth International Conference on Intelligent Systems for Molecular Biology (2000) 269–278.

[31] P. Pevzner, Computational Molecular Biology - an algorithmic approach, MIT Press, 2000, Ch. Finding Signals in DNA, pp. 133–152.

[32] M. Tompa, N. L. et al, Assessing computational tools for the discovery of transcription factor binding sites, Nature Biotechnology 23 (1) (2005) 137 – 144.

[33] E. Eskin, From profiles to patterns and back again: A branch and bound algorithm for finding near optimal motif profiles, Proceedings of the eighth annual international conference on Research in computational molecular biology (2004) 115–124.

[34] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1999.

[35] G. Hertz, G. Stormo, Spelling approximate or repeated motifs using a suffix tree, Lecture Notes in Computer Science 1380 (1999) 111–127.

[36] S. R. Eddy, Profile hidden markov models, Bioinformatics 14 (9) (1998) 755–763.

[37] B. Raphael, L. Liu, G. Varghese, A uniform projection method for motif discovery in DNA sequences, IEEE Transactions on Computational biology and Bioinformatics 1 (2) (2004) 91–94.

[38] A. Price, S. Ramabhadran, P. Pevzner, Finding subtle motifs by branching from sample strings, BIOINFORMATICS 1 (1) (2003) 1–7.

[39] U. Keich, P. Pevzner, Finding motifs in the twilight zone, Bioinformatics 18 (2002) 1374–1381.

[40] E. Eskin, P. Pevzner, Finding composite regulatory patterns in DNA sequences, Tenth International Conference on Intelligent Systems for Molecular Biology (2002) 354–363.

[41] Y. Barash, G. Bejerano, N. Friedman, A simple hyper-geometric approach for discovering putative transcription factor binding sites, Proc. of First International Workshop on Algorithms in Bioinformatics.

[42] E. Segal, Y. Barash, I. Simon, N. Friedman, D. Koller, From promoter sequence to expression: a probabilistic framework, in: Proceedings of the sixth annual international conference on Computational biology, Washington, DC, USA, 2002, pp. 263 – 272.

[43] E. Xing, W. Wu, M. Jordan, R. Karp, LOGOS: A modular bayesian model for de novo motif detection, Journal of Bioinformatics and Computational Biology 2 (1) (2004) 127–154.

[44] B. C. Cetin, J. Barhen, J. W. Burdick, Terminal repeller unconstrained subenergy tunneling (TRUST) for fast global optimization, Journal of Optimization Theory and Applications 77.

[45] K. Blekas, D. Fotiadis, A. Likas, Greedy mixture learning for multiple motif discovery in biological sequences, Bioinformatics 19 (5) (2003) 607–617.

[46] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992.

[47] H. D. Chiang, C. K. Reddy, Trust-tech based training for neural networks, Tech. Rep. 2058, Cornell Univerisity (November 2006).

[48] F. Leung, H. Lam, S. Ling, P. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks 14 (1) (2003) 79–88.

[49] J. Branke, Evolutionary algorithms for neural network design and training, Proceedings of the 1st Nordic Workshop on Genetic Algorithms and its Applications 3 (1995) 145–163.

[50] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999.

[51] M. Gori, A. Tesi, On the problem of local minima in backpropagation, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1) (1992) 76–86.

[52] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[53] Y. Shang, B. W. Wah, Global optimization for neural network training, IEEE Computer 29 (3) (1996) 45–54.

[54] C. Reeves, Modern Heuristic Techniques for Combinatorial Problems, John Wiley and Sons, New York, NY, 1993.

[55] P. RoyChowdhury, Y. P. Singh, R. A. Chansarkar, Dynamic tunneling technique for efficient training of multi-layer perceptrons, IEEE transactions on Neural Networks 10 (1) (1999) 48–55.

[56] D. Gorse, A. Shepherd, J. Taylor, The new era in supervised learning, Neural Networks 10 (2) (1997) 343–352.

[57] D. Nguyen, B. Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, Proceedings of the International Joint Conference on Neural Networks.

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