A New Gradient-Assent Method for Learning Mixture Distributions

Duncan Herring and David P. Helmbold

UCSC-CRL-98-01
January 10, 1998

Board of Studies in Computer and Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

This work investigates some refinements of the exponentiated gradient algorithm, a recent mixture-solution method. The EG_η algorithm uses a relative-entropy distance function as a penalty term inside a gradient-assent framework to learn the values of the parameters of the log likelihood of a given data sample. These parameters include a mixture vector and the mean vectors and covariance matrices of the axis-parallel or spherical Gaussian distributions that comprise the mixture model.

1 Introduction

This paper describes a new gradient-assent algorithm for learning the parameters of mixture distributions. The new method resembles EG_η, the *exponentiated gradient* algorithm, but it contains some important additions and refinements. We will discuss these changes in section four of this paper, but first let us take a quick look at traditional gradient-assent theory.

In unsupervised mixture estimation, we process data that we assume consists of a mixture of observations of results of several different processes or of the members of several different populations. The object of our efforts is to model the data with a group of distributions whose parameters we can guess or determine.

Our data sets consist of P *observations* with D *dimensions* each; thus, the data has P rows of D numbers. We want software that runs quickly and smoothly when P equals thousands of observations and D is on the order of one hundred dimensions.

Early in the process of developing our method, we decided to simplify the solution along parametric lines by assuming that we could model the data using N axis-parallel or spherical Gaussian distributions. Because, the covariance matrices of these special Gaussians are diagonal; we need to learn only $2D$ parameters for each distribution: a D-dimensional mean vector and the D entries on the diagonal of the covariance matrix. Additionally, we need to learn the value of an N-dimensional probability vector called the mixture vector. An N-dimensional *probability vector*, $\mathbf{v}$, is a vector such that $\forall i : (1 \leq i \leq N), v_i \geq 0$ and $\sum_{i=1}^{N} v_i = 1$. The *mixture vector*, $\mathbf{w}$, is a probability vector such that w_i equals the probability that an arbitrary observation from the data belongs in the i^{th} distribution of the solution, $\forall i : 1 \leq i \leq N$.

In our unsupervised learning setting, we do not know which data points belong wto which distribution; thus, we cannot use analytical methods to arrive at a mixture solution. Instead we must resort to using iterative methods for learning the parameters that we seek. A popular Bayesian iterative method of mixture estimation is *Expectation Maximization*, or *EM* [2]. Many researchers use EM; because, they believe that it converges predictably (the likelihood is non-decreasing from iteration to iteration) and that it produces reliable solutions [8]. The present popularity of EM makes its performance an ideal benchmark for testing our algorithm.

2 Gradient Assent

This paper describes a new gradient-assent solution that compares with *EM*. Like many other gradient-assent algorithms, our method tempers the gradient of the log-likelihood function with a learning coefficient, η. Let us begin by representing the current set of parameters using Θ_t and the parameters of the developing iteration by Θ_{t+1}. Remember that

P = the number of observations in the data set;
D = the number of dimensions of each observation;
N = the number of Gaussian distributions in our model;
$\mathbf{w}$ = an N-dimensional probability vector called the mixture vector.

Let
χ = the PxD matrix that holds the raw data set;

X = the PxN likelihood matrix where
$x_{p,n} = P(\boldsymbol{\chi}_p|\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n)$, and
$\mathbf{x}_p = (x_{p,1}, \ldots, x_{p,N})$.

Axis-parallel Gaussians are D dimensional normal distributions with diagonal covariance matrices: all of the covariance-matrix entries off of the diagonal are zero. The entries on the diagonal may take any value. *Spherical Gaussians* are axis-parallel normal distributions where all of the values on the diagonal of the covariance matrix are equal.

Because, the multivariate distributions of our model are axis-parallel or spherical Gaussians; their D univariate components are all aligned with vectors of the orthogonal basis of the data space; thus they are independent, and the multivariate distributions that they compose have diagonal covariance matrices. The multivariate densities equal the products of the densities of the D independent univariate Gaussian distributions that compose them. For the axis-parallel case we have

$$x_{p,n} = \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2} \right] ,$$

and for the spherical case we have

$$x_{p,n} = \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n}}\right]^2} \right] .$$

Now,
$\mathbf{w}$ = an N-dimensional probability vector called the mixture vector, and
$\mathbf{x}_p = (x_{p,1}, \ldots, x_{p,N})$;

therefore,

$$\text{LogLike}(\chi|\Theta) = \frac{1}{P}\sum_{p=1}^{P} \ln\left(\sum_{i=1}^{N} x_{p,i} w_i\right) = \frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{x}_p \cdot \mathbf{w}) \ .$$

Let $\nabla\mathcal{L}(\Theta_t)$ stand for the gradient of the log likelihood of our data matrix, χ, given parameter set Θ and evaluated at $\Theta = \Theta_t$. If M equals the dimensionality of Θ,

$$\nabla\mathcal{L}(\Theta_t) \stackrel{\text{def}}{=} \left(\frac{\partial \text{LogLike}(\chi|\Theta)}{\partial\Theta_1}\,|_{\Theta=\Theta_t}, \ldots, \frac{\partial \text{LogLike}(\chi|\Theta)}{\partial\Theta_M}\,|_{\Theta=\Theta_t}\right) \ ; \qquad (2.1)$$

thus,

$$\begin{aligned} \nabla\mathcal{L}(\mathbf{w}_t) &= \left(\frac{\partial \text{LogLike}(\chi|\Theta)}{\partial w_1}\,|_{\mathbf{w}=\mathbf{w}_t}, \ldots, \frac{\partial \text{LogLike}(\chi|\Theta)}{\partial w_N}\,|_{\mathbf{w}=\mathbf{w}_t}\right) \\ &= \left(\frac{1}{P}\sum_{p=1}^{P}\frac{x_{p,1}}{\mathbf{x}_p\cdot\mathbf{w}_t}, \ldots, \frac{1}{P}\sum_{p=1}^{P}\frac{x_{p,N}}{\mathbf{x}_p\cdot\mathbf{w}_t}\right) . \end{aligned}$$

To formulate the updates, we wish to attempt to maximize

$$F(\Theta_{t+1}) \quad = \quad \eta \text{LogLike}(\chi|\Theta_{t+1}), \;\; \eta > 0 \;\; . \tag{2.2}$$

Unfortunately, to maximize F we must find the gradient of the unknown log likelihood of Θ_{t+1}, $\nabla\mathcal{L}(\Theta_{t+1})$. To avoid this difficulty, we will approximate $\nabla\mathcal{L}(\Theta_{t+1})$ using the first-order Taylor approximation of the new mixture vector. Now we are maximizing

$$\tilde{F}(\Theta_{t+1}) \quad = \quad \eta \left(\text{LogLike}(\chi|\Theta_t) + \nabla\mathcal{L}(\Theta_t) \cdot (\Theta_{t+1} - \Theta_t)\right) \;\; . \tag{2.3}$$

Because, $\text{LogLike}(\Theta_t) - \nabla\mathcal{L}(\Theta_t) \cdot \Theta_t$ is independent of Θ_{t+1}; it is a constant in $\tilde{F}(\Theta_{t+1})$, and we are really maximizing

$$\ddot{F}(\Theta_{t+1}) \quad = \quad \eta \nabla\mathcal{L}(\Theta_t) \cdot \Theta_{t+1} \; . \tag{2.4}$$

Using the Taylor approximation causes inaccuracies when $(\Theta_{t+1} - \Theta_t)$ is large. This observation makes sense; we would not expect $\nabla\mathcal{L}(\Theta_{t+1})$ to point in the same direction as $\nabla\mathcal{L}(\Theta_t)$ points when Θ_{t+1} is far from Θ_t. To minimize this effect, we subtract a penalty term, $d(\Theta_{t+1}||\Theta_t)$, from $\ddot{F}(\Theta_{t+1})$. This penalty term represents a notion of distance between Θ_{t+1} and Θ_t and limits the difference between $\nabla\mathcal{L}(\Theta_{t+1})$ and $\nabla\mathcal{L}(\Theta_t)$. Another name for the penalty term is the *distance function*, $d(\Theta_{t+1}||\Theta_t)$. The new formula that we are maximizing is

$$\hat{F}(\Theta_{t+1}) \quad = \quad \eta \nabla\mathcal{L}(\Theta_t) \cdot \Theta_{t+1} - d(\Theta_{t+1}||\Theta_t) \; . \tag{2.5}$$

The learning coefficient, η, has an expanded role in this new formula. The original learning coefficient controlled how far we moved in the direction of $\nabla\mathcal{L}(\Theta_t)$; a large value for the positive coefficient, η, produced a large jump from Θ_t to Θ_{t+1}. In the new formula, η controls the relative potency of $\nabla\mathcal{L}(\Theta_t)$ and $d(\Theta_{t+1}||\Theta_t)$ in developing Θ_{t+1} given Θ_t.

Several distance functions appear in the paper that motivated our current work [5]. These functions take values that represent "distances" between Θ_{t+1} and Θ_t. The functions can assume only non-negative values. They take the value zero if and only if $\Theta_{t+1} = \Theta_t$. Two distance functions are especially important in gradient-assent methods. We used one of these distance functions in our work. The first important function appears in many standard gradient-assent algorithms. Its value is half of the square of the Euclidean length of $(\Theta_{t+1} - \Theta_t)$;

$$\mathrm{d}_{eu^2}(\mathbf{u}||\mathbf{v}) \quad \stackrel{\text{def}}{=} \quad \tfrac{1}{2}||\mathbf{u} - \mathbf{v}||_2^2 = \tfrac{1}{2}\sum_{i=1}^{N}(u_i - v_i)^2 \; . \tag{2.6}$$

The second function represents the *relative entropy*;

$$\mathrm{d}_{re}(\mathbf{u}||\mathbf{v}) \quad \stackrel{\text{def}}{=} \quad \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i} \; . \tag{2.7}$$

This function plays a fundamental role in our implementation of the EG_η algorithm: we use it in the form with a summation that appears in the definition of equation (2.7) to update the mixture vector. We use an approximation of another form of the relative-entropy distance function–the general form that contains an integral–to update the mean vectors and covariance matrices of the Gaussian distributions of our model. The use of a relative-entropy distance function in a gradient-assent framework was investigated by Manfred Warmuth and Jyrki Kivinen [6].

Choosing a suitable distance formula and maximizing the log likelihood by setting the derivative of equation (2.5) with respect to Θ_{t+1} equal to zero and solving for Θ_{t+1} motivates a family of updates for the parameters of our solution. Ideally, we choose some initial values for these parameters and apply the updates iteratively until we reach convergence. Eventually, we must prove that our updates converge and that the values at convergence parameterize a good mixture solution. Such a proof is beyond the scope of our current work, but section six of this paper contains empirical evidence that our model does converge quickly to substantially the same solution set that EM provides.

3 The Algorithm

3.1 Initialization

It is extremely important to choose good initial values for the mixture vector and for the parameters of the axis-parallel and spherical Gaussian distributions of the model. Good initial values must be close to the final values. Otherwise the software could easily model a local maximum that happens to be close to the initial settings but has a smaller log likelihood than the global solution.

Finding a starting value for the mixture vector, $\mathbf{w}$ is easy. We simply set $\mathbf{w}$ equal to the *uniform probability vector* in N dimensions, the N-vector, $\mathbf{w}$, such that $\forall i : (1 \leq i \leq N), w_i = 1/N$.

We devised a new method for initializing values for the Gaussian parameters. If N is the number of distributions that we will consider, let M signify the power of two such that

$$M = 2^{\alpha}$$

for some integer, α, and

$$\frac{M}{2} < N \leq M \ .$$

Initially, we divide the raw data into M groups of observations. We pick one raw-data dimension at random and find the mean of the the values that the observations take in the chosen dimension. We split the raw data into two groups; one group has values in the chosen dimension that are less than the mean; the other group has values in the chosen dimension that are greater than or equal to the mean. We call the splitting routine recursively on the observations in each group until we have produced exactly M groups of raw data observations.

We reduce the number of observation groups from M to N by combining neighboring groups with the fewest total number of elements. The result is one group for each of the distributions that we believe to be in our mixture.

We can use analytical methods to find the mean vectors and the covariance matrices of the N distributions of points that result from the initialization process described above [3].

3.2 The Learning Coefficient, η

Our algorithm has privatized learning coefficients for the mixture vector and for each of the means and covariance matrices of the N Gaussian distributions contained in the model; thus, there are $2N+1$ different private learning coefficients in our updates. We update each of these η values with every iteration.

When a parameter is far from the area of maximum log likelihood, there is an advantage to keeping its private value of η large: a large η value means quick progress toward the solution.

When we are close to the area of maximum log likelihood, there is an advantage to keeping the value of η small: a small η value reduces the probability of jumping over the top of the maximum. If η is too large near the solution, the algorithm will fail to converge.

We vary the value of each of the $2N + 1$ learning coefficients from one iteration to the next, maintaining a data structure that contains arrays for storing the values of the various coefficients of the log-likelihood parameter updates. Each parameter has its own individual array in the learning-coefficient data structure.

During each iteration we update the values of the parameters of the log likelihood: the mixture vector, $\mathbf{w}$, and the mean vectors and covariance matrices of the N Gaussian distributions. Because, the Gaussian distributions are axis parallel, their covariance matrices are diagonal–that is all of the values off of the diagonal are zero. Clearly, we only need to update the D values on the diagonal when we update a covariance matrix. Notice that now we are updating only vectors; the N-dimensional mixture vector, the N D-dimensional mean vectors, and the N D-dimensional vectors whose components are the values that fall on the diagonals of the covariance matrices. I have coined the term *standard-deviation vector* to label these last vectors; please do not confuse this new usage of the term with the scalar standard deviation of a single univariate Gaussian distribution. The connection between the standard scalar usage of the term "standard deviation" and this new vector usage will soon be obvious.

To make the implementation easy, the vector that we actually update when we are dealing with a covariance matrix is the standard-deviation vector, $\boldsymbol{\sigma}$, whose components are the square roots of the values that fall on the diagonal of the matrix. The components of $\boldsymbol{\sigma}$ are the standard deviations of the univariate Gaussian distributions whose product equals one of the multivariate Gaussians of our model.

Now we are updating $2N + 1$ vectors: the mixture vector, $\mathbf{w}$; N mean vectors, $\boldsymbol{\mu}_n : (1 \leq n \leq N)$; and N standard-deviation vectors, $\boldsymbol{\sigma}_n : (1 \leq n \leq N)$. Each of these vectors has its own private learning coefficient.

We use a simple method for adjusting the $2N + 1$ η values. At the end of each iteration, we tabulate the changes that we made to each of the $2N+1$ parameter vectors. For example, during iteration $t + 1$, the change in the mean vector of a given distribution, $\boldsymbol{\mu}_\Delta$, is equal to $\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t$. Notice that $\boldsymbol{\mu}_\Delta$ is a vector. Using a well-known property of the dot product, we can find the cosine of the angle, ϕ, between any change vector from the current iteration and the corresponding change vector from the previous iteration, for example $\boldsymbol{\mu}_{\Delta,t+1}$ and $\boldsymbol{\mu}_{\Delta,t}$. When we know the value of $\cos(\phi)$, we can calculate the new value of the learning coefficient, η_{t+1}, using the formula

$$\eta_{t+1} = \eta_t \left(\alpha + \frac{\cos(\phi)}{\beta} \right) \quad . \tag{3.1}$$

We have determined typical values for α and β by experiment. For the typical spherical Gaussian model, $\alpha = 0.70$ and $\beta = 3$. For the typical axis-parallel Gaussian model, $\alpha = 0.75$ and $\beta = 3$. The values for β are stable for most types of data; however, the correct value for α depends upon both the model type and the data. The typical α values above work for many data sets. In section six we discuss fine tuning the α value.

Whenever we jump over the point of maximum log likelihood, the gradient changes direction drastically causing $\cos(\phi)$ to assume a negative value and reducing or tempering the value of η. Otherwise the direction of the gradient is comparable from one iteration to the next, and the resulting positive value for $\cos(\phi)$ causes η to increase with each iteration. Increasing η in this way is called the *bold-driver strategy* [10].

3.3 Approximation of The Relative-Entropy Distance Function

The distance functions that we use to form the exponentiated-gradient updates all spring from the relative-entropy distance function. Let $\boldsymbol{\xi} \in \mathbb{R}^D$. The general form of the relative-entropy distance function is

$$\mathcal{D}_{\mathcal{RE}}\left(\times_{\sqcup+\infty}||\times_{\sqcup}\right) \quad = \quad \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{t+1}(\boldsymbol{\xi}|\Theta_{t+1}) \ln\left(\frac{P_{t+1}(\boldsymbol{\xi}|\Theta_{t+1})}{P_t(\boldsymbol{\xi}|\Theta_t)}\right) d\boldsymbol{\xi} \ .$$

Unfortunately, evaluating this function and its derivatives is difficult. The probabilities behind the logarithms are really sums of products of more basic probabilities;

$$P(\boldsymbol{\xi}|\Theta) \quad = \quad \sum_{n=1}^{N} w_n \prod_{d=1}^{D} P(\boldsymbol{\xi}|\Theta_{n,d}) \ .$$

The derivations of the updates for the all of the exponentiated-gradient algorithms that we have seen depend upon approximations to circumvent the difficulties in evaluating the derivatives of the general form of the relative-entropy distance function.

We avoided the sums behind the logs entirely while deriving the updates of our algorithm. To make the update of parameters of distribution n, we approximated $\mathcal{D}_{\mathcal{RE}}\left(\times_{\sqcup+\infty}||\times_{\sqcup}\right)$ with $\mathrm{d}_{RE}\left(\Theta_{n,t+1}||\Theta_{n,t}\right)$; thus while working with the parameters of distribution n, we used

$$\mathrm{d}_{RE}\left(\Theta_{n,t+1}||\Theta_{n,t}\right) \quad = \quad \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi}$$

as the distance function for the μ and σ updates.

Our approximation will clearly work well when the distances between the means of the model distributions are large compared to their standard deviations. When the Gaussians are widely spread, an arbitrary data point will tend to have a significant density in only one of the model distributions; thus, our approximation will be a natural simplification of the general relative-entropy equation.

Presently, we will compare our method with an exponentiated-gradient algorithm due to Manfred Warmuth. Mr. Warmuth's derivations are not sensitive to the distance between the means of the model distributions.

In a yet unpublished manuscript [11], Manfred Warmuth introduces an exponentiated-gradient algorithm that uses an approximation of the general relative-entropy distance function to form the updates. In his approximation, $\mathrm{D}_{RE}(\Theta_{t+1}, \Theta_t)$, Mr. Warmuth begins with the general form itself and moves the summations over N outside of the log for greater convexity and easier calculation. Appendix E contains a derivation of an axis-parallel version of Mr. Warmuth's algorithm. Assume $\boldsymbol{\xi} \in \mathbb{R}^D$ and

$$\mathrm{d}_{RE}\left(\Theta_{n,t+1}||\Theta_{n,t}\right) \quad = \quad \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi} \ ;$$

then

$$\begin{aligned}
\mathrm{D}_{RE}(\Theta_{t+1},\Theta_t) &= \int_{\boldsymbol{\xi}\in\mathbb{R}^D}\sum_{n=1}^{N} w_{n,t+1}P(\boldsymbol{\xi}|\Theta_{t+1})\ln(\frac{w_{n,t+1}P(\boldsymbol{\xi}|\Theta_{n,t+1})}{w_{n,t}P(\boldsymbol{\xi}|\Theta_{n,t})})d\boldsymbol{\xi} \\
&= \sum_{n=1}^{N} w_{n,t+1}\ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1}\int_{\boldsymbol{\xi}\in\mathbb{R}^D} P(\boldsymbol{\xi}|\Theta_{n,t+1})\ln(\frac{P(\boldsymbol{\xi}|\Theta_{n,t+1})}{P(\boldsymbol{\xi}|\Theta_{n,t})}) \\
&= \sum_{n=1}^{N} w_{n,t+1}\ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1}\mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t}) \ .
\end{aligned}$$

3.4 The Updates

As we explained in the learning-rate section, during each iteration, we update three types of vectors, the mixture vector, $\mathbf{w}$; the mean vectors, $\boldsymbol{\mu}$; and the standard-deviation vectors, $\boldsymbol{\sigma}$. For those readers who are interested, we derive the updates for axis-parallel Gaussians in Appendix B and the updates for the spherical Gaussian model in Appendix C.

The Axis-Parallel Update for the Mixture Vector

We use the relative-entropy distance function,

$$\mathrm{d}_{re}(\mathbf{w}_{t+1}||\mathbf{w}_t) = \sum_{i=1}^{N}\mathbf{w}_{t+1}\ln\frac{\mathbf{w}_{t+1}}{\mathbf{w}_t}\ , \tag{3.2}$$

to produce the exponentiated gradient update for the mixture vector, $\mathbf{w}$,

$$w_{n,t+1} = \frac{w_{n,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_n}}{\sum_{i=1}^{N} w_{i,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_i}} \tag{3.3}$$

$$= \frac{w_{n,t}e^{\left[\frac{\eta}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right)\right]}}{\sum_{i=1}^{N} w_{i,t}e^{\left[\frac{\eta}{P}\sum_{p=1}^{P}\left(\frac{x_{p,i}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right)\right]}} \ . \tag{3.4}$$

The Axis-Parallel Update for the Mean Vector

Let $\boldsymbol{\xi}\in\mathbb{R}^D$. We will use an approximation of the general form of the relative-entropy distance function. In the update of $\mu_{n,d}$, we will approximate $\mathcal{D}_{\mathcal{RE}}(\times_{\sqcup+\infty}||\times_{\sqcup})$ with $\mathrm{d}_{RE}(\Theta_{n,t+1}||\Theta_{n,t})$, where

$$\mathrm{d}_{RE}\left(\Theta_{t+1}||\Theta_t\right) = \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{t+1})\ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_t)}\right)d\boldsymbol{\xi} \ .$$

The exponentiated-gradient update for the mean vector, $\boldsymbol{\mu}$, is

$$\mu_{n,d,t+1} = \eta_n \sigma_{n,d}^2 \nabla \mathcal{L}(\mu_{n,t})_d + \mu_{n,d,t} \tag{3.5}$$

$$= \frac{\eta_n \sigma_{n,d}^2}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}^2} \right) + \mu_{n,d,t} \tag{3.6}$$

$$= \frac{\eta_n}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} (\chi_{p,d} - \mu_{n,d}) \right) + \mu_{n,d,t} \ . \tag{3.7}$$

The σ^2 co-factor limits the size of a proposed change in μ whenever σ is small. When σ is small, we have a sharply peaked distribution, and we certainly cannot tolerate large changes in the mean. A large change could take us past the desired convergence value; after a really large change we could actually be farther from the desired value than we were before the change. The result is divergence. Alternately, the co-factor increases the size of changes to the mean when σ is large. When σ is large, we have a wide distribution; thus, we desire large changes in the mean; they are necessary for quick convergence when σ is large.

The Axis-Parallel Update for the Covariance Matrix

In updating the covariance matrix, we actually update the vector whose components are the square roots of the elements on the matrix diagonal, the standard deviation vector, $\boldsymbol{\sigma}$. We will use an approximation of the the general form of the relative-entropy distance function. In the update of $\sigma_{n,d}$, we will approximate $\mathcal{D}_{\mathcal{RE}}(\times_{\sqcup+\infty} || \times_{\sqcup})$ with $\mathrm{d}_{RE}(\Theta_{n,t+1} || \Theta_{n,t})$ to arrive at the update,

$$\sigma_{n,d,t+1} = \frac{\sqrt{\eta_n^2 \nabla \mathcal{L}(\sigma_{n,t})_d{}^2 \sigma_{n,d,t}^4 + 4\sigma_{n,d,t}^2} + \eta_n \nabla \mathcal{L}(\sigma_{n,t})_d \sigma_{n,d,t}^2}{2}$$

$$= \sqrt{\left(\frac{\eta_n}{2P} \sum_{p=1}^{P} \frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d} \right] \right)^2 + \sigma_{n,d,t}^2}$$
$$+ \frac{\eta_n}{2P} \sum_{p=1}^{P} \frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d} \right] \ .$$

Updates for the Spherical Gaussian Model

The updates in the previous sections are based on modeling the data using N axis-parallel Gaussian distributions. The axis-parallel model is one important special case of the general Gaussian solution. Another special case assumes a model of spherical Gaussian distributions–normal distributions with diagonal covariance matrices such that all of the entries on the diagonal equal some single value, σ^2.

The mixture-vector and mean-vector updates for the spherical-Gaussian solution are the same as the corresponding updates for the axis-parallel model; one simply uses the formulas that appear above. The covariance-matrix update for the spherical-Gaussian model differs

from the axis-parallel case. One can specify a spherical-model covariance matrix with a single value, σ; because, the only non-zero values in the matrix are on the diagonal, and they all equal σ^2. A derivation for the σ update appears in appendix C. The update is

$$\sigma_{n,t+1} = \frac{\sqrt{\eta_n^2 \nabla\mathcal{L}(\sigma_{n,t})^2 \sigma_{n,t}^4 + 4D^2\sigma_{n,t}^2} \; + \; \eta_n \nabla\mathcal{L}(\sigma_{n,t})\sigma_{n,t}^2}{2D}$$

$$= \sqrt{\left(\frac{\eta_n}{2DP}\sum_{p=1}^{P}\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d}\right]\right)^2 + \sigma_{n,d,t}^2}$$
$$+\frac{\eta_n}{2DP}\sum_{p=1}^{P}\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d}\right] \; .$$

3.5 The Magic of EG_η

For the mixture vector, the EG_η algorithm produces a multiplicative update that is greater than or equal to zero. The mean, however, must take on negative values, and the EG_η algorithm updates it additively. It is not necessary to produce these appropriate updates by manipulation, the algorithm produces them automatically.

Another strange feature of the EG_η update is the resemblance of the mixture-vector update to the equation in Bayes' theorem. Perhaps this fact will remind us that when we do updates, we are developing posterior values from the prior values that enter each iteration.

4 Producing the Test Data

We performed most of the tests of the engine using data that we generated with two modules that produce test files consisting of points on an annulus. The first module, *testdisc*, produces random data on the annulus, the second module, *testgrid*, produces uniform data on the annulus. The module, "testdisc", selects a random probability, P, from the interval $[0, 1]$ and selects a radius, r, for each test point according to the following algorithm. Let

$$\begin{aligned} M &= \textit{the outer radius of the annulus}; \\ m &= \textit{the inner radius of the annulus}; \\ \textit{calculate} & \\ r &= \sqrt{P\,(M-m)} + m \; . \end{aligned}$$

The module then selects an angle, θ randomly and exports the rectangular coordinates corresponding to r and θ to a file.

The module, "testgrid", merely selects all of the test points on a rectangular mesh that also fall inside the annulus and exports these values to a file.

A third module, *generate*, produces D univariate Gaussian distributions of P/D points each using random parameters. The program then combines these distributions into a single P-point multivariate sample with D dimensions. This module is useful whenever we need data with more than the two dimensions inherent in all of the annulus samples.

The final module, *testbox*, generates a uniform, three-dimensional, rectilinear distribution centered on the origin. We used "generate" and "testbox" together to produce files with both uniform and Gaussian Components.

5 Testing the Algorithms

An enormous quantity of data is available from sources who collect it during their own scientific endeavors. We used artificial means to build most of the files that we will use during our tests, but the lure of using our software to analyze "live" data is too tempting to ignore entirely. Consequently, we tested our program using data that B.S Everitt and D.J Hand [4] claim has known parameters for two Gaussian distributions. One test consisted of the ash-content figures for 430 peat samples; another test involved the lengths of 1000 trypanosome protozoon from two species. Our results for the first test were substantially the same as those that Everitt and Hand calculated. Our results for the second test varied slightly from those of the Everitt and Hand; however, we determined that the original calculations were for grouped data. Our engine does not deal directly with grouped data, and we undoubtedly introduced some inaccuracy in the sample when we converted the data for use with our program.

All of the remaining tests utalize data that we produced artificially using the special built-in modules that we described in the previous section of this paper. These special data-generation units include the "testdisc", "testgrid", "testbox", and "generate" modules. The first series of tests involves running our software engine on three data files that we produced using "testdisc". This module makes two-dimensional data on an annulus. The files that we obtained contain one thousand two-dimensional points each. During testing, we invoke the "C" "times" command just before our software terminates; thus, it is possible to print out a record of the amount of user time that the program needs for learning the parameters of the model.

Our first test is for machine independence. We ran our software and a specially prepared version of the EM program on two different machines, a Solaris 2.x and a Sun-4. We used the first of the three "testdisc" data files as input and varied the number of axis-parallel Gaussian distributions of the model from two to fourteen. The results of the tests that we ran on the Solaris 2.x machine appear in graphical form in Figure 1. The results of the tests that we ran on the Sun-4 machine appear in Figure 2. The Solaris 2.x is a much faster machine than the Sun-4, but the graphs of the results have an almost identical shape My conclusion is that despite the compiler optimization that we did while fine tuning the program, our software runs equally well on the two machines that we used for testing.

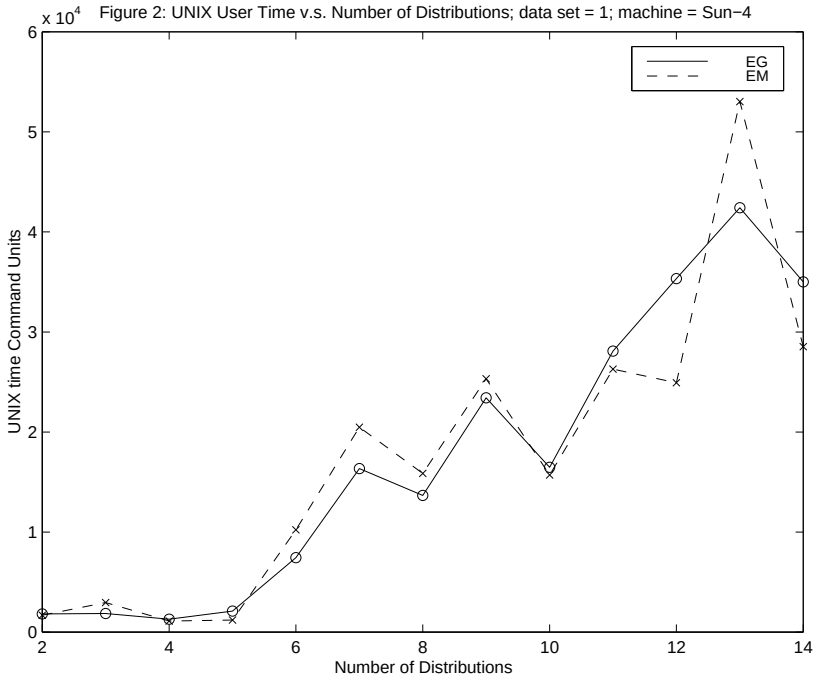


Figure 3 gives more information about the same tests, showing the number of iterations required to arrive at a solution when the number of axis-parallel Gaussians of the model varies from two to fourteen. We used the first "testdisc" file as input and ran the software on the Solaris 2.x machine.

Figure 4 depicts the log likelihood of the same solutions. We used the first "testdisc" file as input and varied the number of axis-parallel Gaussians of the model from two to fourteen. We ran the software on the Solaris 2.x machine.

Figure 3: Number of Iterations v.s. Number of Distributions; data set = 1; machine = Solaris 2.x

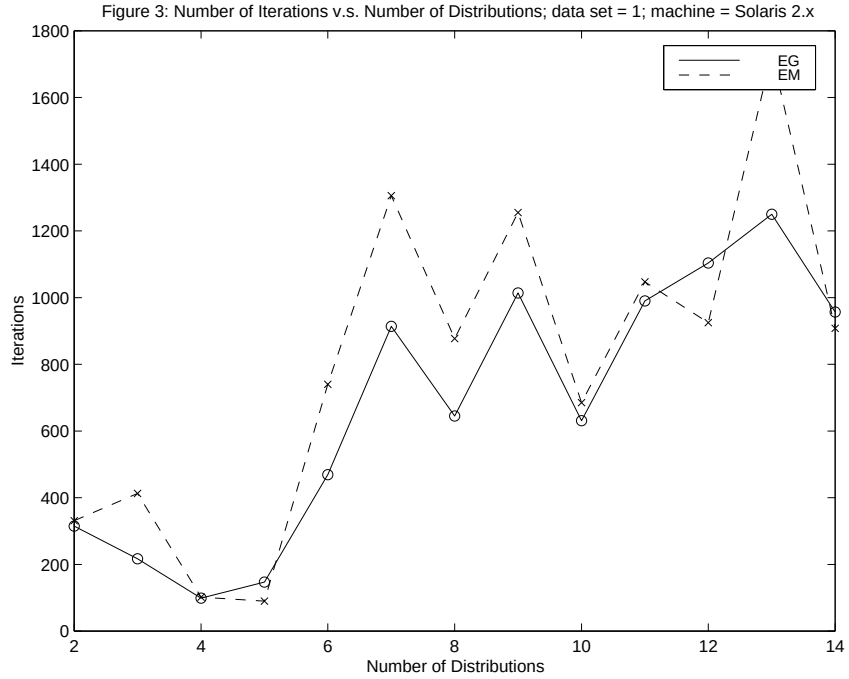
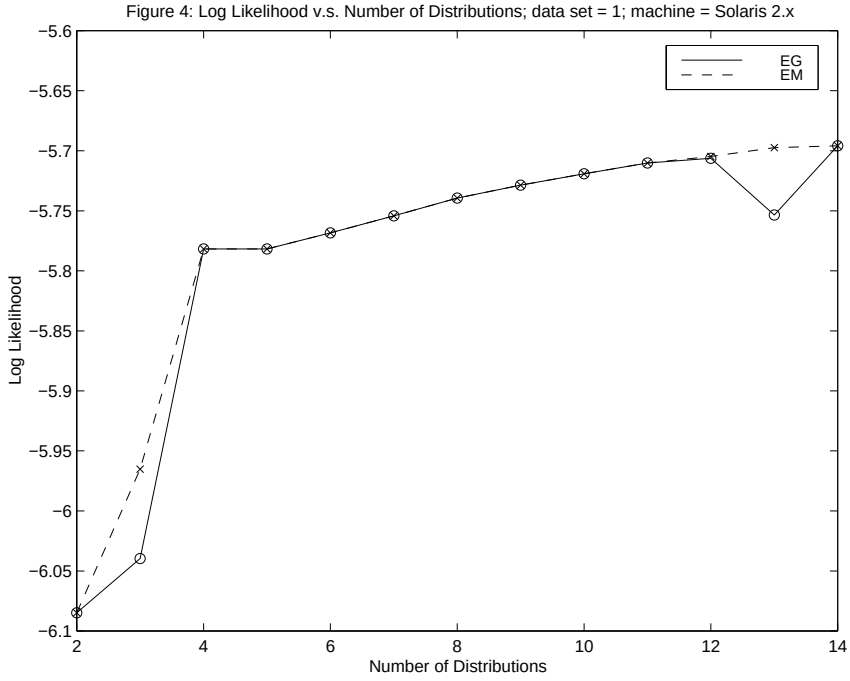
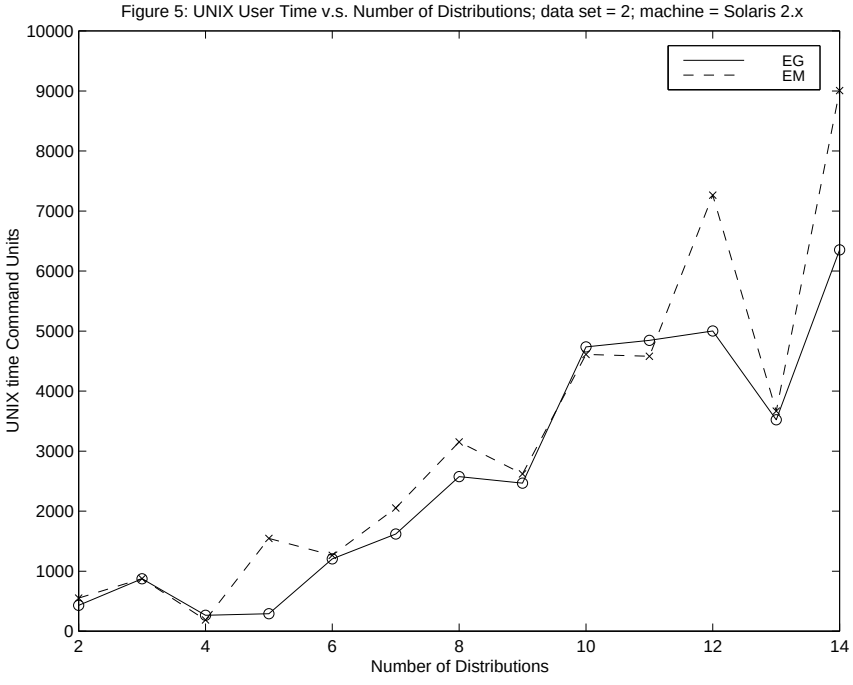


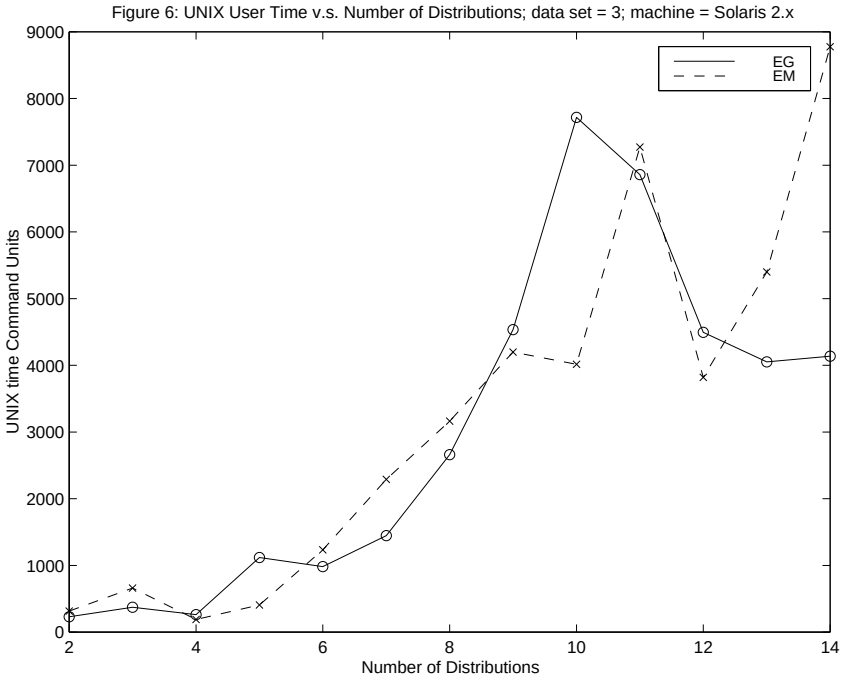
Figure 4: Log Likelihood v.s. Number of Distributions; data set = 1; machine = Solaris 2.x



To discover how running time relates to the number of distributions of the model, we did execution-time tests using the second and third "testdisc" files, running the software on the Solaris 2.x machine. Figure 5 shows the time required for a solution using the second "testdisc" file as input, and Figure 6 shows the time needed for a solution using the third "testdisc" file. In these tests and the first "testdisc" trial, our algorithm performs comparably with EM. Neither method has a clear-cut advantage in regard to the running time.

Figure 5: UNIX User Time v.s. Number of Distributions; data set = 2; machine = Solaris 2.x





The complexity of the algorithms that we used to make the program suggests that the running time per iteration should vary linearly with the number of Gaussians in the model. Figures 6a and 6b show the time per iteration versus the number of Gaussian distributions in the model when the input files were the first and second "testdisc" files respectively. These graphs show monotonic increasing functions, and it is easy to imagine that the motivation behind these curves is, indeed, linear.

Figure 6a: UNIX User Time / Iteration v.s. Number of Distributions; data set = 1

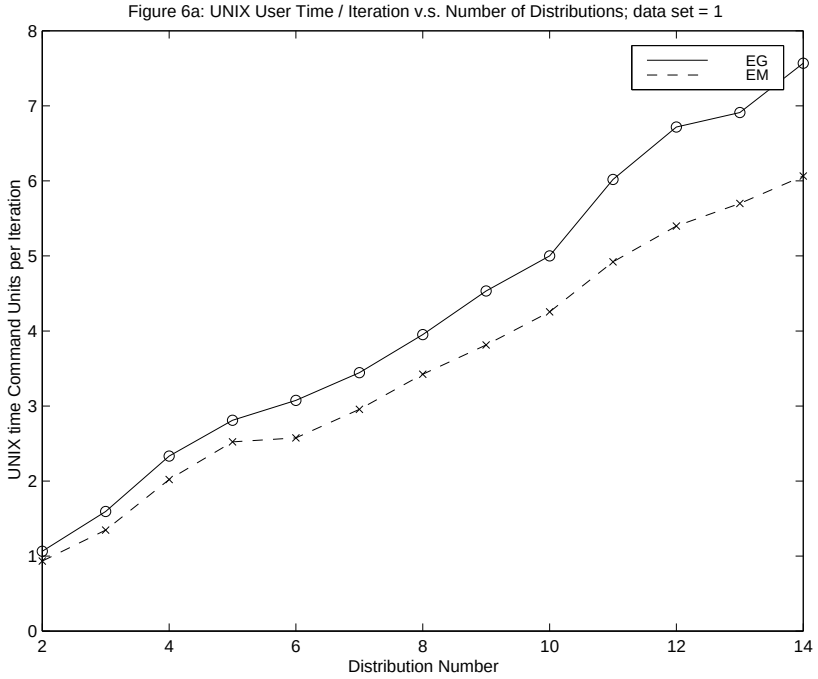
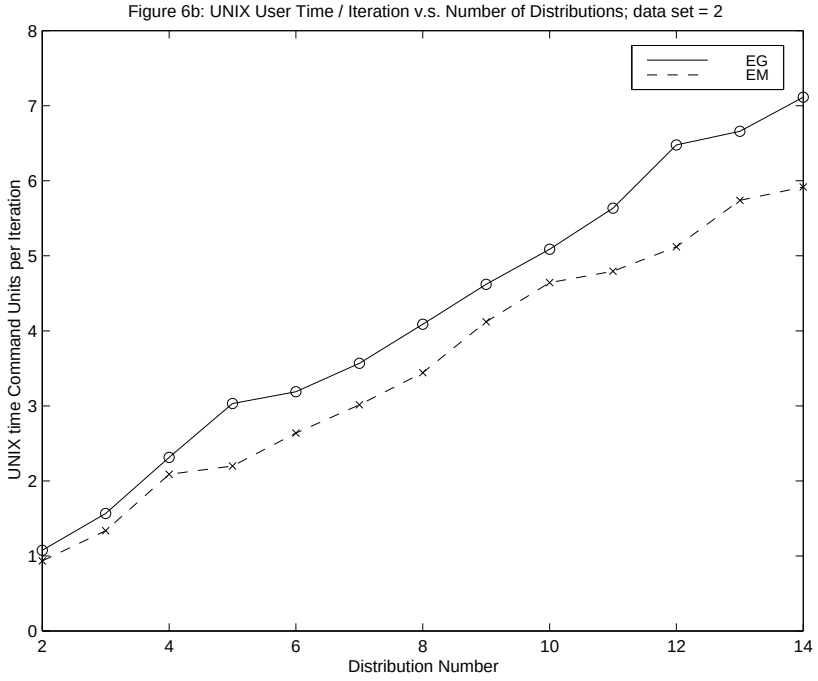


Figure 6b: UNIX User Time / Iteration v.s. Number of Distributions; data set = 2



Clearly, increasing the number of distributions in our model causes a corresponding increase in the amount of time that it takes the program to do an average iteration of the solution. Why is it that the total time to find a solution is not a monotonic increasing function of the number of distributions in the model? To find out if our initialization process is causing the jaggedness of the initial graphs, we repeated the tests using several random initializations of the distribution parameters for each model size instead of employing the standard initialization module. These new tests would determine if the jagged quality of the first graphs resulted from our standard initialization process. As before, we varied the model size from two to fourteen distributions. We used five random initializations for each model size. When we made the graphs that depict the results of these tests, we drew a special curve to represent the average of the running times associated with the five random initializations. We used the first "testdisc" data file to make the graph in Figure 7. Figure 8 shows the results when the input file was the second "testdisc" file.

Figure 7: UNIX User Time v.s. Number of Distributions; data set = 1; machine = Solaris 2.x; Random Initialization

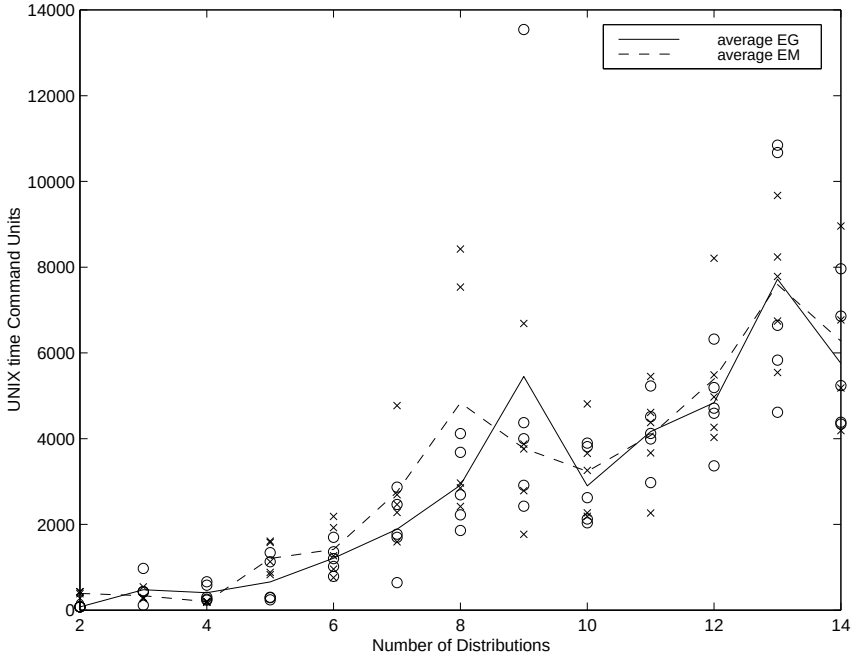
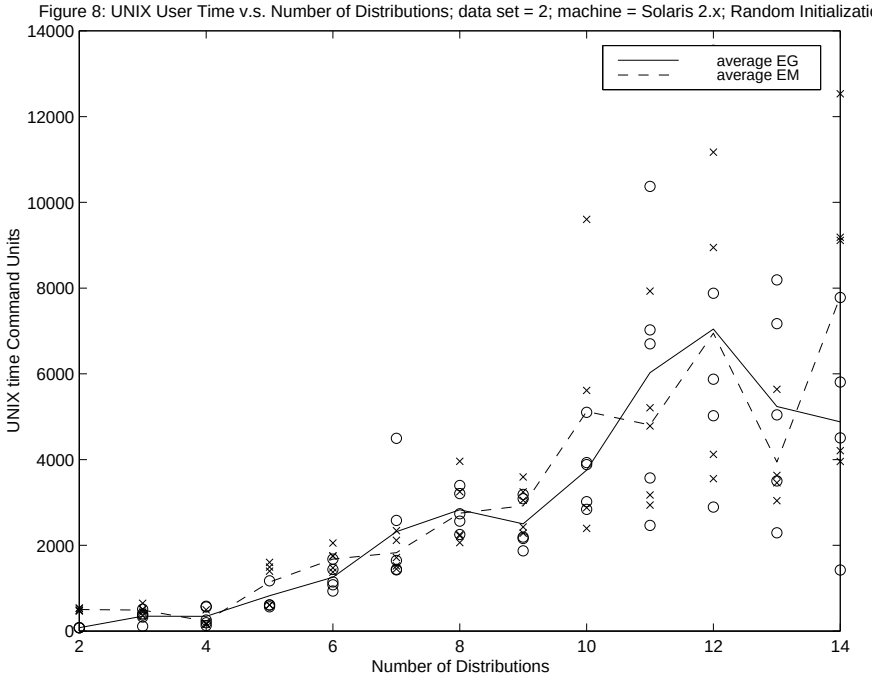


Figure 8: UNIX User Time v.s. Number of Distributions; data set = 2; machine = Solaris 2.x; Random Initialization



These graphs are certainly less jagged than the ones in Figure 1 and Figure 5, but the new curves are still not monotonic increasing. I conclude that the initialization process has little effect on the jaggedness of the graphs. That Figure 7 and Figure 8 are smoother than Figure 1 and Figure 5 appears to be an effect of the averaging process. Running time is proportional to the number of iterations needed for a solution. The number of iterations is not proportional to the number of distributions in the model.

The dashed lines in all of the graphs represent the performance of the EM algorithm. Note that Figure 6a and Figure 6b both indicate that our solution spends more time per iteration than the EM solution spends. EM, however, often requires more iterations to learn the parameters of the model; thus, the running times for our program compare with the EM running times when the input files for both engines result from the "testdisc" module.

The "testgrid" module produces a different sort of data set from the "testdisc" data files; "testgrid" makes uniform distributions. Figure 9 and Figure 10 show how our program and EM compare when the input is uniform data. As before, we have plotted the running times against the number of distributions in the model. We used the standard initialization module in preparing Figure 9 and random initialization for Figure 10. The curves in Figure 10 represent an average of the data points.

Figure 9: UNIX User Time v.s. Number of Distributions; data set = 4; machine = Solaris 2.x

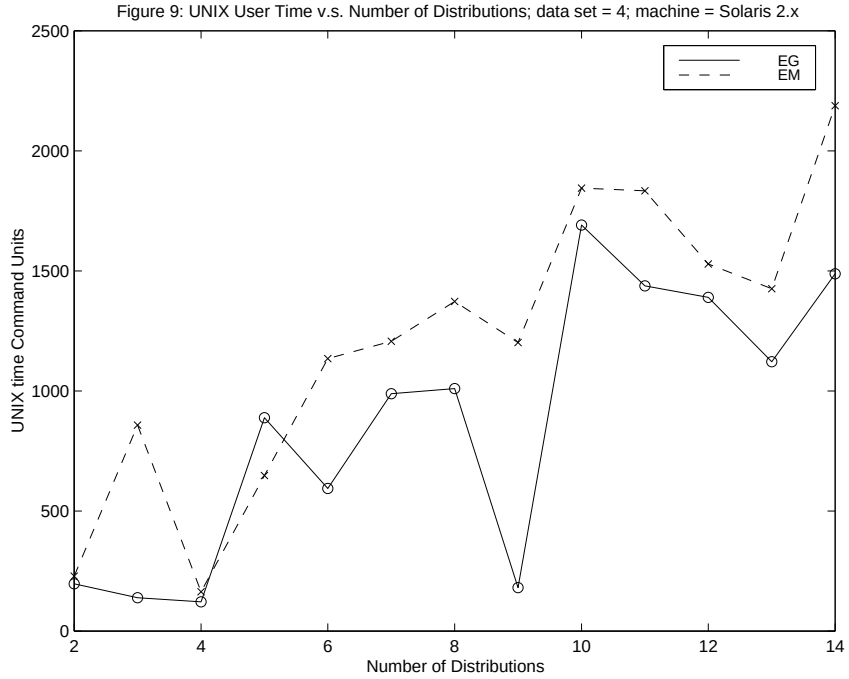
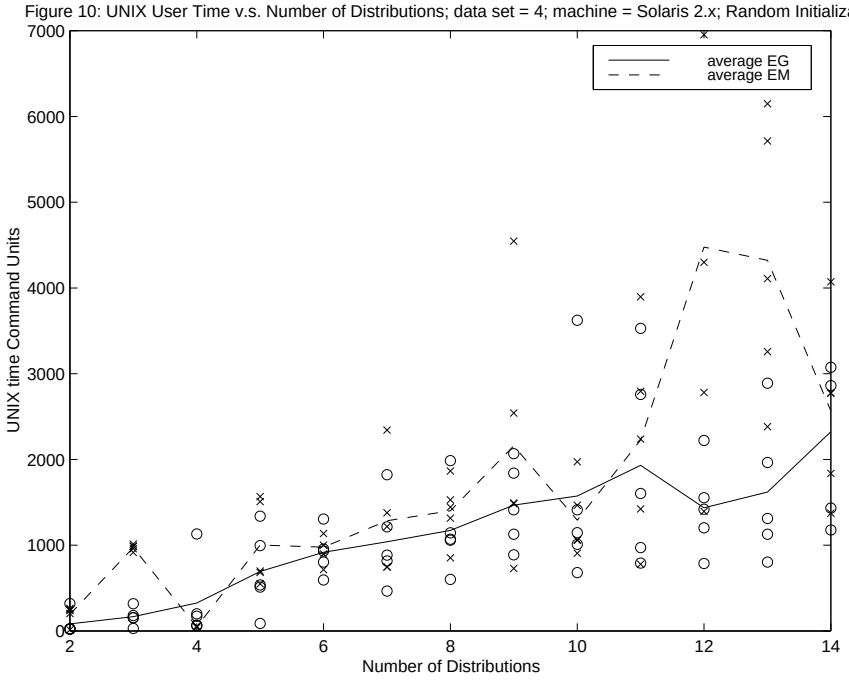


Figure 10: UNIX User Time v.s. Number of Distributions; data set = 4; machine = Solaris 2.x; Random Initialization



Our algorithm seems to be slightly faster than EM when the input is uniform data from "testgrid".

Figure 11 depicts the total running time in terms of the number of observations in the data sample. We used the first "testdisc" file and fixed the number of distributions of the model at four. The result is not linear; sometimes the new points that we add to the file enhance an existing clump of points, and sometimes they set up a competing clump.

The surprising graph is Figure 12. Here the plot of running time versus the number of data dimensions seems almost linear. We used the "testgrid" file and four distributions in preparing this chart. It is possible that the uniform distribution "testgrid" foils the "clumping effect" that caused the non-linearity in the graph of running time v.s. the number of observations that appears in Figure 11.

Figure 11: UNIX User Time v.s. Size of Data Set; machine = Solaris 2.x

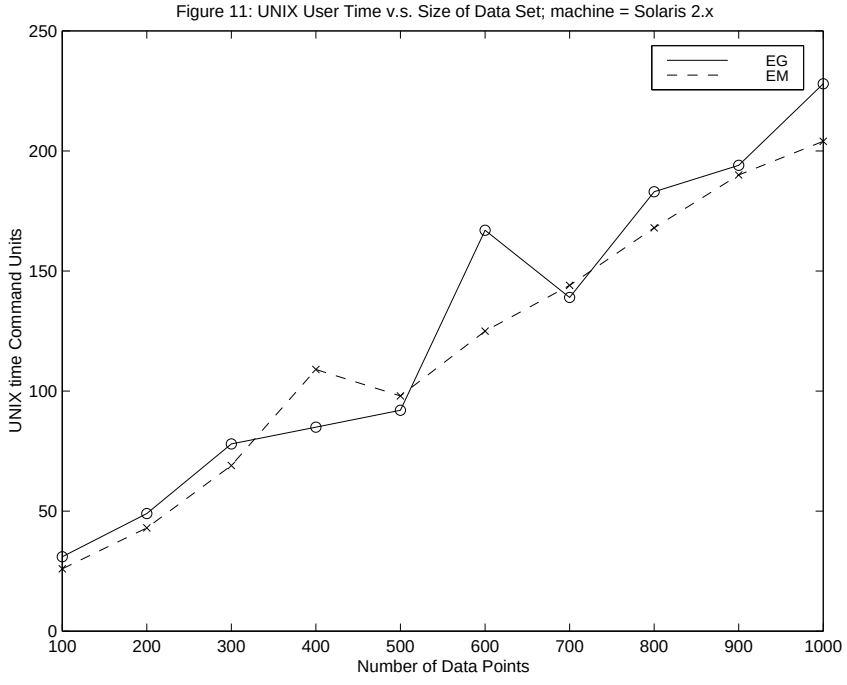
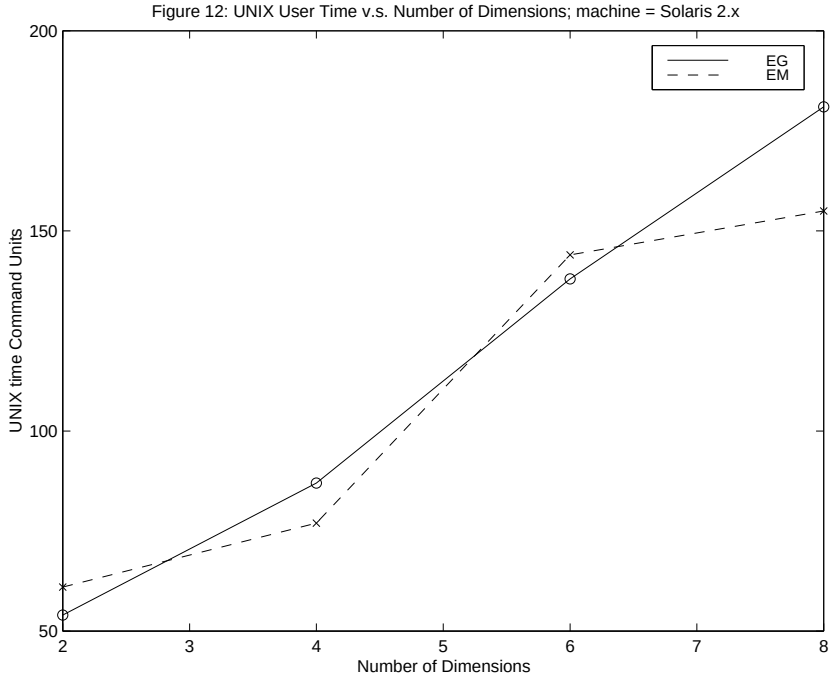


Figure 12: UNIX User Time v.s. Number of Dimensions; machine = Solaris 2.x



Our program increases the privatized learning rate for a parameter vector whenever the present value of that vector is quite different from the value at the time of the solution. This strategy is called the "bold driver" [10]. Unfortunately, it is not possible to make the rate of increase large; because, its cumulative effect is exponential. We use a default value of 1.08333334 for the bold-driver increase rate. As shown in Figure 13 and Figure 14, when the data set is the first "testdisc" file, setting the bold-driver rate too low results in large total execution times for the program. Fixing the rate too high degrades the solutions and increases the running time. The default rate works well with the first "testdisc" input file.

Figure 13: UNIX User Time v.s. Value of Bold Driver; data set = 1; machine = Solaris 2.x

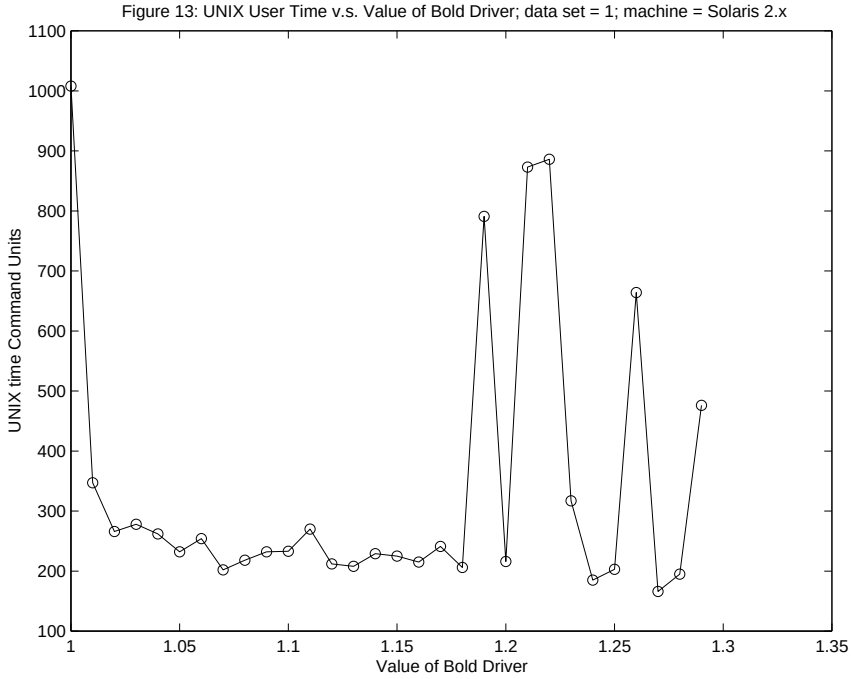


Figure 14: Log Likelihood v.s. Value of Bold Driver; data set = 1; machine = Solaris 2.x

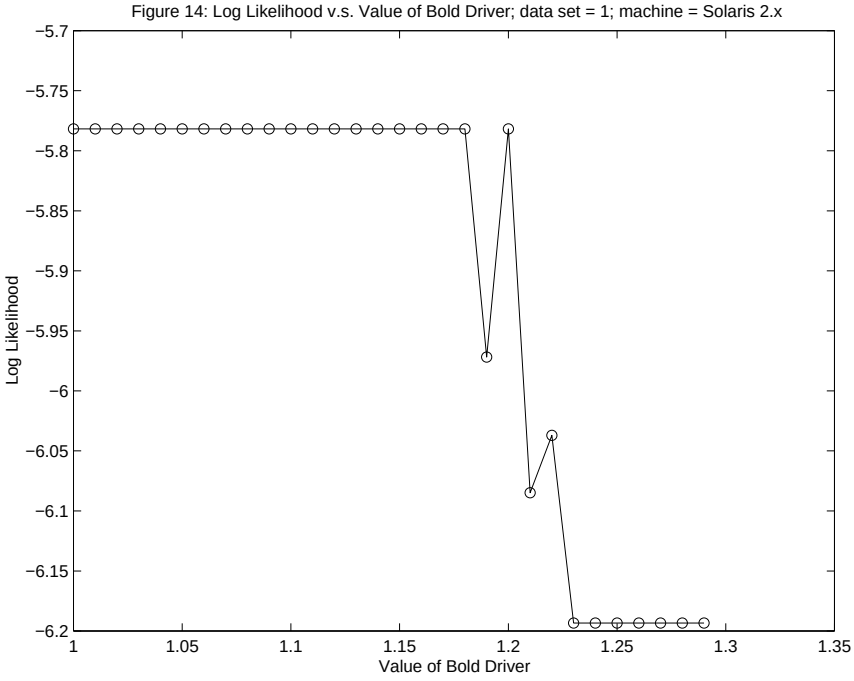
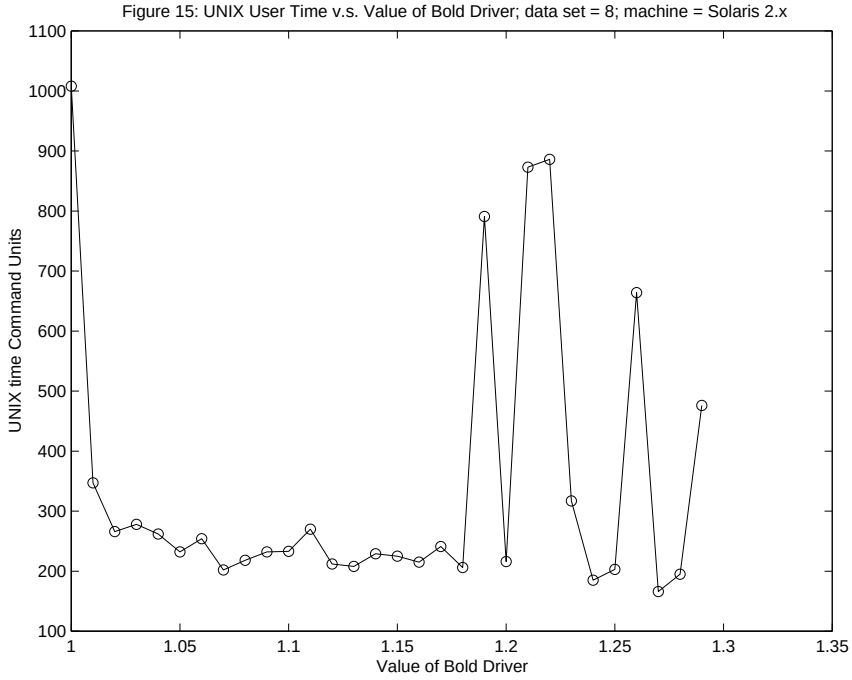


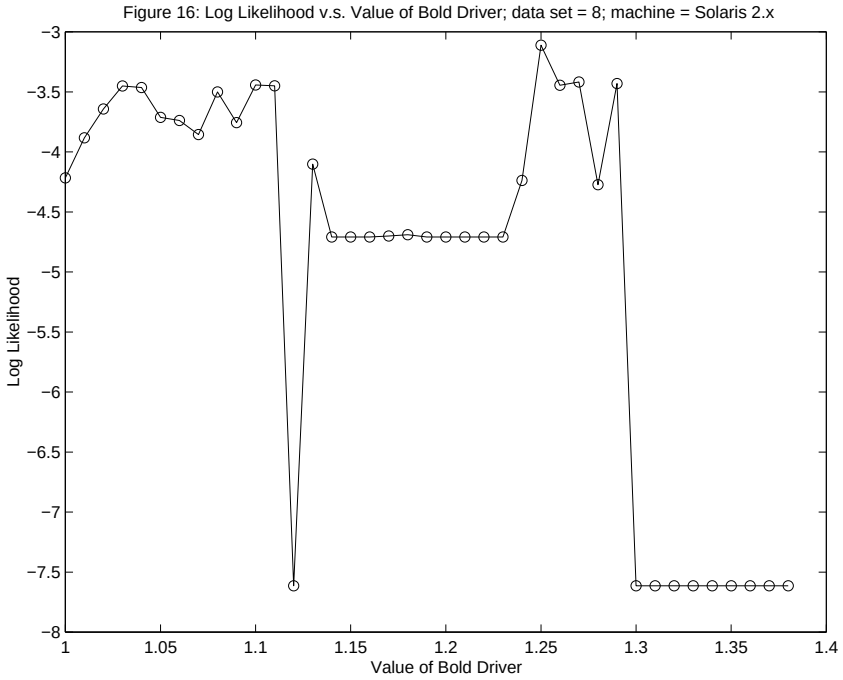
Figure 15 and Figure 16 show similar experiments with a very different data set. Data set eight has 784 observations generated by the "generate" module and 216 observations from "testbox". We adjusted the "generate" to produce an equal number of points from each of four axis-parallel Gaussian distributions with random means and covariance matrices. The "testbox" module gives a three-dimensional, uniform distribution that is rectilinear and centered at the origin.

Changing the bold-driver rate with data set eight seems to effect the running time in a way that is similar to what we saw with the first "testdisc" input. The log-likelihood is, however, very unstable. A good value for the bold-driver rate for data set eight might be 1.05.

Clearly, we need to learn the proper bold-driver rate for our input data by *training* the software: running it repeatedly on a representative portion of the data to locate the area of the bold-driver rate curve where the solutions are reliable and the running times are quick.

Figure 15: UNIX User Time v.s. Value of Bold Driver; data set = 8; machine = Solaris 2.x





Data sets five through ten contain points from "generate", "testbox", or both. The sizes of the contributions of these modules appear below in table 1.

Table 1

file name	*generate*	*testbox*
input5	0	1000
input6	272	729
input7	488	512
input8	656	343
input9	784	216
input10	1000	0

Interestingly enough, both our program and EM seem to have long execution times when between a quarter and a half of the points are from the Gaussian distributions produced by "generate". At that frequency, the Gaussian points do not occur often enough to be easy to locate, and the execution times can be large. The uniform distribution of the "input5" file is sometimes easier; because, it has no clumps to attract the model distributions. The pure Gaussian distribution of the "input10" file is almost always easier; because, it lacks the distraction of the uniform distribution points.

Figure 17 and Figure 18 show running time versus input file number. Figure 17 uses the standard initialization module, and Figure 18 utalizes random initialization and averaging.

In the results from Figure 18, our method seems to have a slight advantage over EM when the data is uniform or almost uniform.

Figure 17: UNIX User Time v.s. Data Set Number; machine = Solaris 2.x

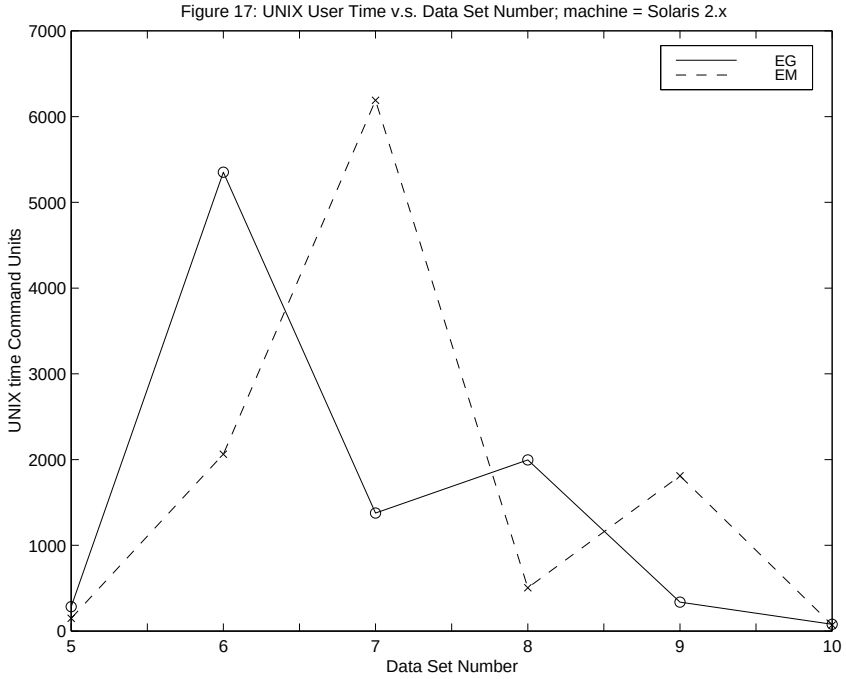
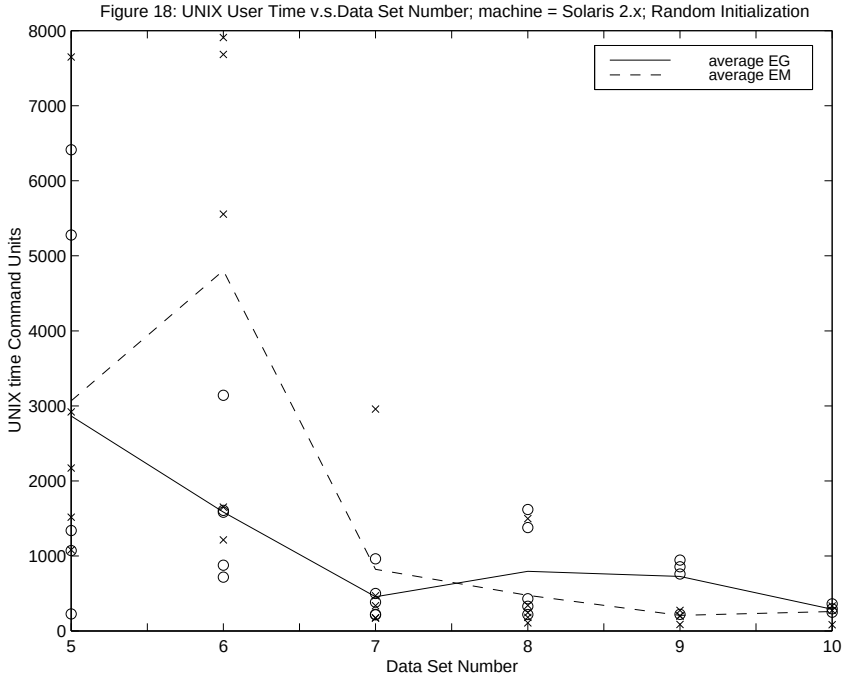


Figure 18: UNIX User Time v.s.Data Set Number; machine = Solaris 2.x; Random Initialization



All of the previous results came from modeling the input data using axis-parallel Gaussian distributions. Figure 19 and Figure 20 give running time and log-likelihood results using spherical Gaussian distributions in the model.

Spherical Gaussians are not as powerful as axis-parallel Gaussians for modeling arbitrary data sets; however, when the data occurs in isolated spherical clumps, they will obviously offer an economical solution. Other data configurations often work well with the spherical model if the number of distributions is limited to a certain range. We are able to model the

first "testdisk" data quickly and accurately using either nine or ten spherical Gaussians. The log likelihoods of the results from these spherical modelings compare with the axis-parallel log likelihoods, and the running times for the spherical model are thirty to fifty percent smaller than the corresponding axis-parallel running times.

Figure 19: UNIX User Time v.s. Number of Distributions; data set = 1; machine = Solaris 2.x; Spherical Gaussians

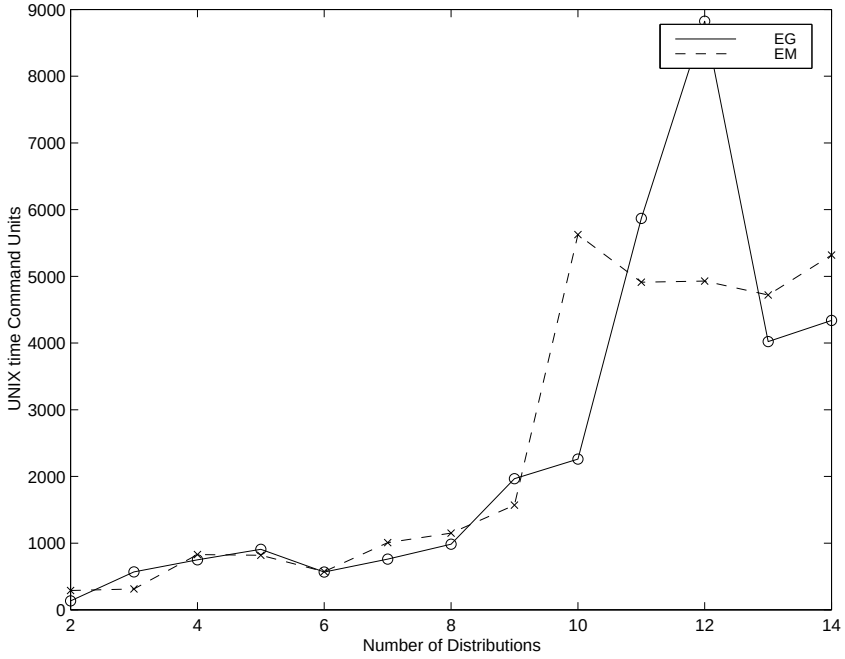
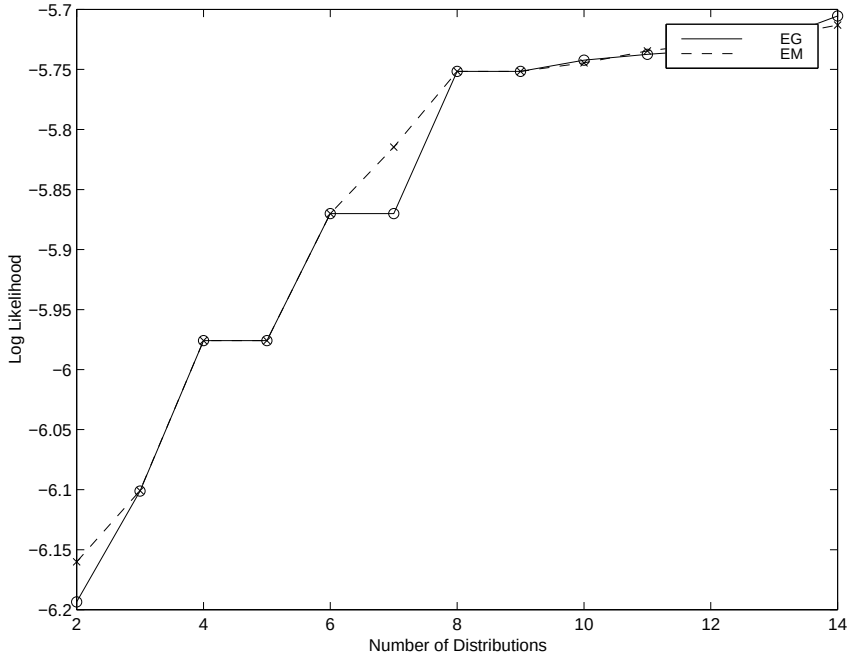


Figure 20: Log Likelihood v.s. Number of Distributions; data set = 1; machine = Solaris 2.x; Spherical Gaussians



We made a MATLAB engine to model mixture solutions with axis-parallel Gaussian distributions. Our software contains routines that implement axis-parallel EM, an axis-parallel version of an EG algorithm due to Manfred Warmuth [11], and the EG algorithm that we developed. We borrowed much of the MATLAB code from Yoram Singer of AT&T Bell Laboratories.

Mr. Warmuth's algorithm is very efficient; because, it works with a single, fixed learning rate. Our program has higher overhead due to the work of updating the privatized learning coefficients. Our method compares well with both EM and Mr. Warmuth's EG under some conditions, but Mr. Warmuth's program is often the fastest. I suspect that low overhead is the reason that his method succeeds.

We wondered if difficulty in breaking symmetry of the distributions of the model was causing our method to run slowly. Because, our approximation of the relative entropy distance function express the distance in terms of the relative entropy of an update of a single Gaussian distribution; we are at our theoretical best during times when the model distributions do not overlap significantly. We might, therefore, have problems breaking model symmetry. Figure 21 shows the results of running the MATLAB engine on a tight round cloud of random points surrounding the origin. The initialization is a set of axis-parallel Gaussians that are all practically identical.

Figure 21: MATLAB User Time v.s. Number of Distributions; data set = 3

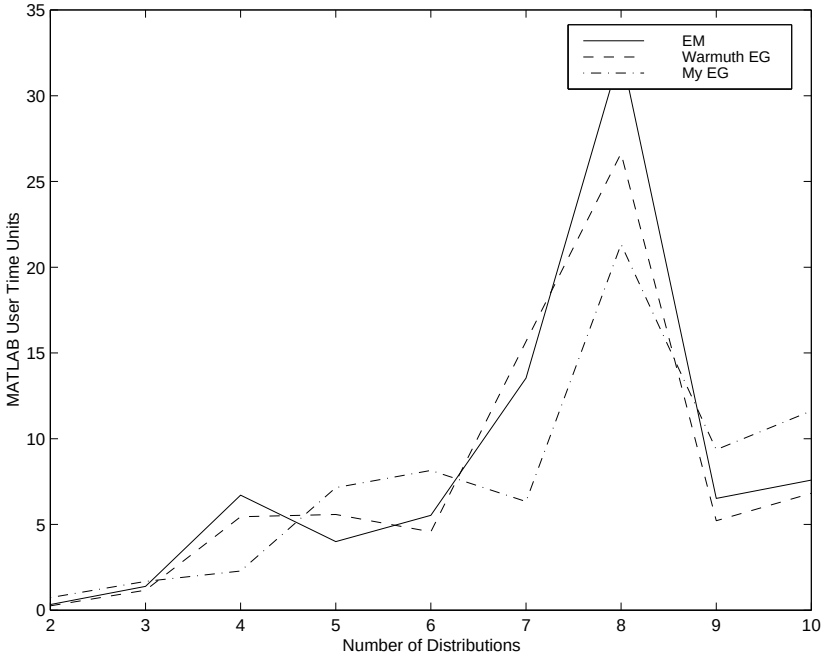
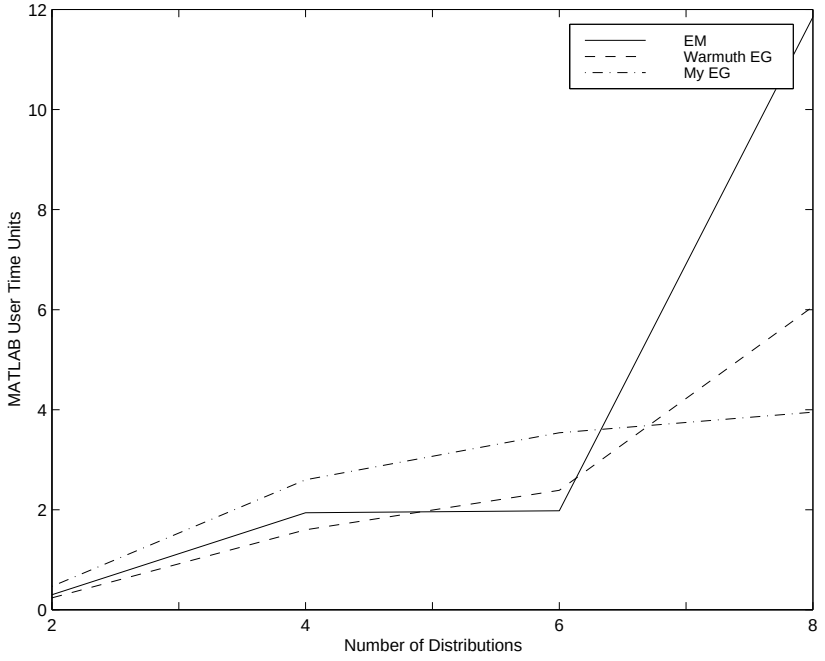


Figure 22 shows results with the same cloud of points, but the initialization here is a set of Gaussians with widespread mean vectors. These initial distributions are quite far apart in contrast to the starting distributions in Figure 21.

We do not know if the distributions overlap significantly at some time after initialization; however, we do know that they do not overlap much at the point of initialization.

All of the algorithms seem to perform better with the new initial conditions, but our method shows the same improvement as the other two algorithms show. We conclude that poor symmetry breaking is not our most significant performance problem.

Figure 22: MATLAB User Time v.s. Number of Distributions; data set = 11



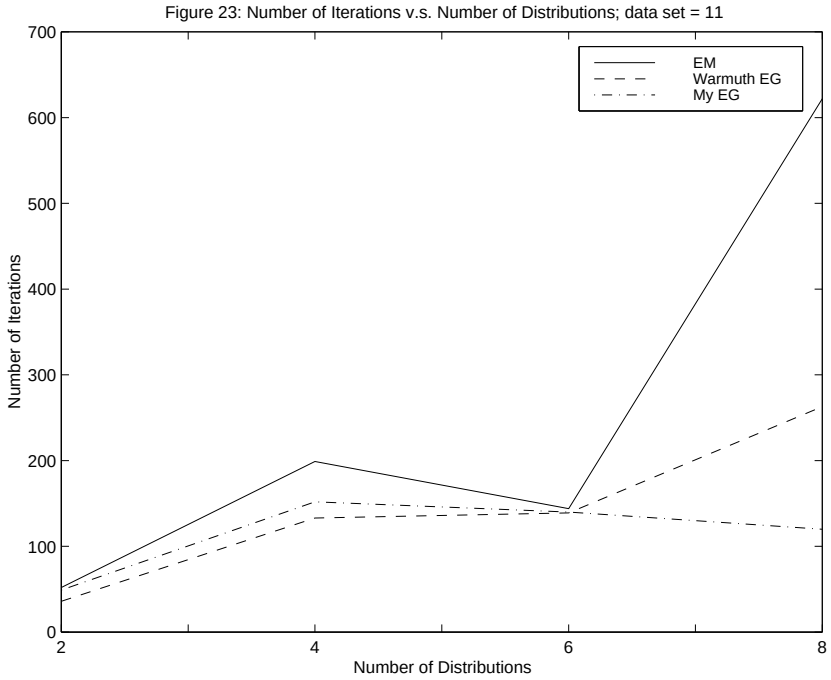


Figure 23 shows the number of iterations that the three methods required for convergence. In this frame, our algorithm performs quite well compared to the other two, but these results do not include the overhead for updating the many privatized learning rates. EM has no learning coefficient, and Manfred Warmuth uses a single fixed learning rate. Neither of these methods spends time changing the value of η. The graph of these results with the overhead included is Figure 22.

In Figures 24 and 25 we repeat the experiment using the familiar first "testdisc" data set. The annulus configuration of these points automatically assures that we have a minimum of overlapping whenever the distributions are evenly spaced around the ring. Tests using the visualization functions of MATLAB convince us that this experiment contains only slight Gaussian intersection.

Figure 24: MATLAB User Time v.s. Number of Distributions; data set = 1

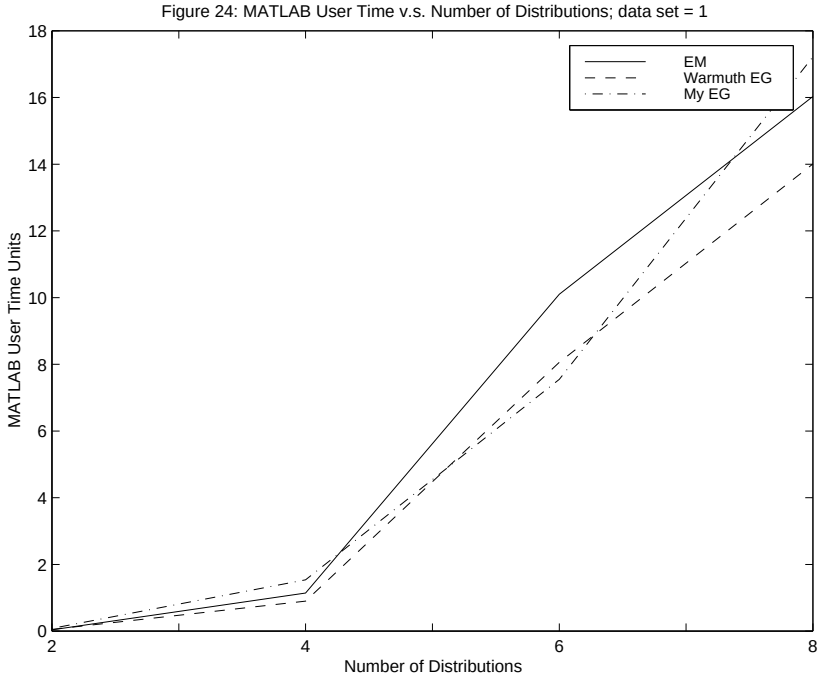
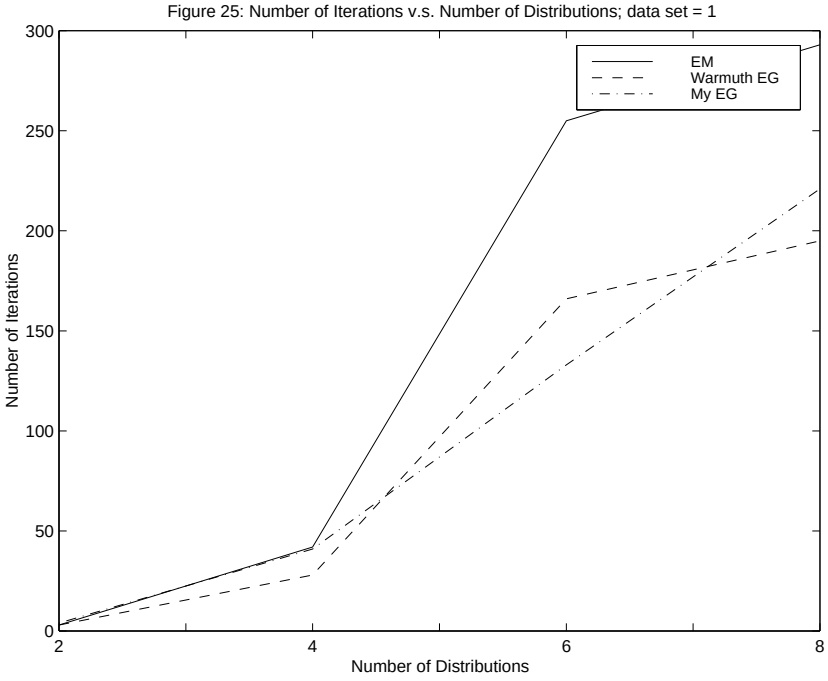


Figure 25: Number of Iterations v.s. Number of Distributions; data set = 1



6 Conclusion

Using relative-entropy distance functions, we produced a variant of the exponentiated-gradient algorithm for a mixture solution that features especially formulated updates for the parameters of the model of axis-parallel or spherical Gaussian distributions and a privatized and articulated learning rate. The private learning rate for a given parameter vector is small when that vector is nearly the same as a potential solution vector and large when the given vector is not part of a potential solution.

The performance of our new algorithm rivals that of EM in tests using uniform or random quasi-uniform data on an annulus, and three-dimensional data that is a mixture of Gaussian and uniform points.

Possible future work involves adding a training mechanism to our program for discovering the data-dependent optimum rate of increase of the privatized learning rates when the parameter vectors are far from the solution.

References

[1] Christopher M. Bishop; *Neural Networks for Pattern Recognition*; Clarendon Press, Oxford, (1995).

[2] A. P. Dempster; N. M. Laird; and D. B. Rubin; "Maximum Likelihood From Incomplete Data via the EM Algorithm (with discussion)"; *Journal of the Royal Statistical Society* B, 39, 1-38 (1977).

[3] Richard O. Duda and Peter E. Hart; *Pattern Classification and Scene Analysis*; John Wiley and Sons, New York (1973).

[4] B.S. Everitt and D.J. Hand; *Finite Mixture Distributions*; Chapman and Hall; London, 1981; p34, p46.

[5] David P. Helmbold; Robert E. Schapire; Yoram Singer; and Manfred K. Warmuth; "A Comparison of New and Old Algorithms for A Mixture Estimation Problem"; *Proceedings of the Eighth Conference on Computational Learning Theory*; July, 1995.

[6] Jyrki Kivinen; Manfred Warmuth; "Additive Versus Exponentiated Gradient Updates for Linear Prediction"; *Journal of Information and Computation*; vol. 132, no. 1, pp1-64; January 1997.

[7] Alberto Leon-Garcia; *Probability and Random Processes for Electrical Engineering*; Addison-Wesley, Reading, MA(1994).

[8] Geoffrey J. McLachlan and Thriyambakam Krishnan; *The EM Algorithm and Extensions*; John Wiley and Sons, New York, (1997).

[9] John A. Rice; *Mathematical Statistics and Data Analysis* (Second Addition); Duxbury Press, Belmont, CA(1995).

[10] Dilip Sarkar; "Methods to Speed Up Error Back-Propagation Learning Algorithm"; *ACM Computing Surveys*; December 1995; v27, n4, pp519-42

[11] Yoram Singer; Manfred Warmuth; "Learning Gaussian Mixtures Based on the Relative Entropy"; yet unpublished.

A Definitions for Deriving the Updates

Let
P = the number of observations in the data set;
D = the number of dimensions of each observation;
N = the number of Gaussian distributions in our model;
$\mathbf{w}$ = an N-dimensional probability vector called the mixture vector.
χ = the PxD matrix that holds the raw data set;
X = the PxN likelihood matrix where
$x_{p,n} = P(\boldsymbol{\chi}_p|\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n)$, and
$\mathbf{x}_p = (x_{p,1}, \ldots, x_{p,N})$.

Because, the distributions are spherical or axis-parallel Gaussians; their covariance matrices are all diagonal; therefore, the entries off of the diagonal are all zero for these matrices, and the univariate densities of the D dimensions of the raw data are independent. The multivariate density under distribution n of observation p of the raw data matrix, $\boldsymbol{\chi}_p$, is the product of the univariate densities of the D dimensions of the raw data, $\chi_{p,1} \ \ldots \ \chi_{p,D}$; thus, for the axis-parallel case

$$x_{p,n} = \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2} \right] ,$$

and for the spherical case

$$x_{p,n} = \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n}}\right]^2} \right] .$$

Now,
$\mathbf{x}_p = (x_{p,1}, \ldots, x_{p,N})$, and
$\mathbf{w}$ = an N-dimensional probability vector called the mixture vector;
therefore,

$$\text{LogLike}(\chi|\Theta) = \frac{1}{P}\sum_{p=1}^{P} \ln\left(\sum_{i=1}^{N} x_{p,i} w_i\right) = \frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{x}_p \cdot \mathbf{w}) \ .$$

To formulate the updates, we wish to maximize the function $\hat{F}$,

$$\hat{F}(\Theta_{t+1}) \quad = \quad \eta \nabla \mathcal{L}(\Theta_t) \cdot \Theta_{t+1} - d(\Theta_{t+1}||\Theta_t) \, ; \tag{A.1}$$

thus, we wish to find the value of Θ_{t+1} that makes the derivative of $\hat{F}$ with respect to Θ and evaluated at Θ_{t+1} vanish.

B The Axis-Parallel Update

B.1 The Likelihood and Log Likelihood of the Raw Data

The likelihood of our raw data matrix, χ, is

$$\begin{aligned} likelihood(\chi|\Theta) &= \prod_{p=1}^{P}\sum_{n=1}^{N}\left[w_n P\left(\boldsymbol{\chi}_p \mid \mu_n, \sigma_n\right)\right] \\ &= \prod_{p=1}^{P}\sum_{n=1}^{N}\left[w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right] . \end{aligned}$$

The log likelihood of our raw data matrix, χ, is

$$LogLike(\chi|\Theta) = \frac{1}{P}\sum_{p=1}^{P}\ln\left[\sum_{n=1}^{N}\left(w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right)\right] .$$

The $1/P$ is a scaling factor.

Because, the Gaussians of our model are axis-parallel; the multivariate density under distribution n of observation p of the raw data matrix, $\boldsymbol{\chi}_p$, is the product of the univariate densities of the D dimensions of the raw data, $\chi_{p,1} \ \ldots \chi_{p,D}$; thus, X is the *likelihood matrix*, if

$$x_{p,n} = \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right] .$$

B.2 Updating the Mixture Vector

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial w_n} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{\displaystyle\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]}{\displaystyle\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right)}\right) \\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\right) ; \\
\nabla\mathcal{L}(\mathbf{w}_t)_n &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right) .
\end{aligned}$$

We wish to update the mixture vector; therefore, equation (A.1) becomes

$$\hat{F}(\mathbf{w}_{t+1}) = \eta\nabla\mathcal{L}(\mathbf{w}_t)\cdot\mathbf{w}_{t+1} - d(\mathbf{w}_{t+1}||\mathbf{w}_t) \ . \tag{B.1}$$

We add a Lagrangian term to enforce the constraint that

$$\sum_{n=1}^{N} w_{n,t+1} = 1 \ , \tag{B.2}$$

and

$$\hat{F}(\mathbf{w}_{t+1},\gamma) \ = \ \eta\nabla\mathcal{L}(\mathbf{w}_t)\cdot\mathbf{w}_{t+1} \ - \ d(\mathbf{w}_{t+1}||\mathbf{w}_t) + \gamma\left(\sum_{n=1}^{N} w_{n,t+1} - 1\right), \quad \eta > 0 \ . \tag{B.3}$$

To maximize $\hat{F}(\mathbf{w}_{t+1},\gamma)$, set the N partial derivatives of $\hat{F}(\mathbf{w}_{t+1},\gamma)$ with respect to the components of $\mathbf{w}$ and evaluated at $\mathbf{w}=\mathbf{w}_{t+1}$ equal to zero and obtain

$$\frac{\partial\hat{F}(\mathbf{w}_{t+1},\gamma)}{\partial w_{n,t+1}} \ = \ \eta\nabla\mathcal{L}(\mathbf{w}_t)_n - \frac{\partial d(\mathbf{w}_{t+1}||\mathbf{w}_t)}{\partial w_{n,t+1}} + \gamma = 0 \ . \tag{B.4}$$

Set the partial derivative with respect to γ of $\hat{F}(\mathbf{w}_{t+1},\gamma)$ equal to zero, and discover that

$$\sum_{n=1}^{N} w_{n,t+1} = 1 \ . \tag{B.5}$$

If we use the summation form of the relative-entropy distance function,

$$\mathrm{d}_{re}(\mathbf{w}_{t+1}||\mathbf{w}_t) = \sum_{n=1}^{N} w_{n,t+1}\ln\frac{w_{n,t+1}}{w_{n,t}} \ ,$$

the N equations (B.4) become

$$\eta \nabla \mathcal{L}(\mathbf{w}_t)_n - (\ln \frac{w_{n,t+1}}{w_{n,t}} + 1) + \gamma = 0 \ .$$

Solving this system of equations for $w_{n,t+1}$ we have

$$\begin{aligned} w_{n,t+1} &= w_{n,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_n + \gamma - 1} && \text{(B.6)} \\ &= w_{n,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_n} e^{\gamma - 1} && \text{(B.7)} \\ &= ?\, w_{n,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_n}, \quad ? = e^{\gamma - 1} \ . && \text{(B.8)} \end{aligned}$$

Summing the N equations we get

$$\sum_{n=1}^{N} w_{n,t+1} = ? \sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i} \ .$$

By equation (B.5),

$$? = \frac{1}{\sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i}} \ ,$$

and we get the *exponentiated gradient* update for the mixture vector,

$$\begin{aligned} w_{n,t+1} &= \frac{w_{n,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_n}}{\sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i}} && \text{(B.9)} \\ &= \frac{w_{n,t} e^{\left[\frac{\eta}{P} \sum_{p=1}^{P} \left(\frac{x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w_t}} \right) \right]}}{\sum_{i=1}^{N} w_{i,t} e^{\left[\frac{\eta}{P} \sum_{p=1}^{P} \left(\frac{x_{p,i}}{\mathbf{x}_p \cdot \mathbf{w_t}} \right) \right]}} \ . && \text{(B.10)} \end{aligned}$$

B.3 Updating the μ Vector

We are updating the means; therefore, equation (A.1) becomes

$$\hat{F}(\mu_{n,d,t+1}) \quad = \quad \eta_n \nabla \mathcal{L}(\mu_{n,t})_d \cdot \mu_{n,d,t+1} - d(\mu_{n,d,t+1} || \mu_{n,d,t}) \; ; \tag{B.11}$$

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial \mu_{n,d}} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]}{\sum_{n=1}^{N}\left(w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right)} \frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_{n,d}^2}\right) \\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_{n,d}^2}\right) \; ; \\
\nabla \mathcal{L}(\mu_{n,t})_d &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_{n,d}^2}\right) \; .
\end{aligned}$$

We use the following derivation for an update for μ based on an approximation of the relative-entropy distance formula. In the update of $\mu_{n,d}$, we will approximate $\mathcal{D}_{\mathcal{RE}}(\times_{\sqcup+\infty} || \times_{\sqcup})$ with $\mathrm{d}_{RE}(\Theta_{n,t+1} || \Theta_{n,t})$.

Let $\boldsymbol{\xi} \in \mathbb{R}^D$. For distribution n,

$$\begin{aligned}
\mathrm{d}_{RE}\left(\Theta_{n,t+1} || \Theta_{n,t}\right) &= \int_{\boldsymbol{\xi} \in \mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi} \; ; \\
&= \int_{\boldsymbol{\xi} \in \mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\prod_{d=1}^{D} \frac{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t+1}\right)}{\sigma_{n,d}}\right]^2}\right]}{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t}\right)}{\sigma_{n,d}}\right]^2}\right]}\right) d\boldsymbol{\xi} \; ;
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln e^{-\frac{1}{2}\left[\frac{(\xi_d-\mu_{n,d,t+1})}{\sigma_{n,d}}\right]^2} d\boldsymbol{\xi} \\
&\quad -\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln e^{-\frac{1}{2}\left[\frac{(\xi_d-\mu_{n,d,t})}{\sigma_{n,d}}\right]^2} d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \left[\frac{(\xi_d-\mu_{n,d,t+1})-(\xi_d-\mu_{n,d,t})}{\sigma_{n,d}}\right] d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \left[\frac{\mu_{n,d,t+1}^2-\mu_{n,d,t}^2-2\xi_d\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)}{\sigma_{n,d}}\right] d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \left[\frac{\mu_{n,d,t+1}^2-\mu_{n,d,t}^2-2\mu_{n,d,t+1}\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)}{\sigma_{n,d}}\right] \ ; \\
&= \frac{1}{2}\sum_{d=1}^{D} \frac{\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)^2}{\sigma_{n,d}} \ .
\end{aligned}
$$

Differentiating with respect to $\mu_{n,d,t+1}$ we get

$$\frac{(\mu_{n,d,t+1}-\mu_{n,d,t})}{\sigma_{n,d}} \ .$$

To maximize $\hat{F}(\mu_{n,d,t+1})$, we wish to solve

$$\eta_n \nabla\mathcal{L}(\mu_{n,t})_d - \frac{(\mu_{n,d,t+1}-\mu_{n,d,t})}{\sigma_{n,d}} = 0 \ ;$$

thus, the new update is

$$
\begin{aligned}
\mu_{n,d,t+1} &= \eta_n \sigma_{n,d}^2 \nabla\mathcal{L}(\mu_{n,t})_d + \mu_{n,d,t} && \text{(B.12)} \\
&= \frac{\eta_n\sigma_{n,d}^2}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_{n,d}^2}\right) + \mu_{n,d,t} && \text{(B.13)} \\
&= \frac{\eta_n}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}(\chi_{p,d}-\mu_{n,d})\right) + \mu_{n,d,t} \ . && \text{(B.14)}
\end{aligned}
$$

The σ^2 co-factor that appears in this update limits the size of a proposed change in μ whenever σ is small. When σ is small, we have a sharply peaked distribution, and we certainly cannot tolerate large changes in the mean. A large change could take us past the desired convergence value; after a really large change we could actually be farther from the desired value than we were before the change. The result is divergence. Alternately, the co-factor increases the size of changes to the mean when σ is large. When σ is large, we have a wide distribution; thus, we desire large changes in the mean; they are necessary for quick convergence when σ is large.

B.4 The σ Update

We are updating σ; therefore, equation (A.1) becomes

$$\hat{F}(\sigma_{n,d,t+1}) = \eta_n \nabla \mathcal{L}(\sigma_{n,t})_d \cdot \sigma_{n,d,t+1} - d(\sigma_{n,d,t+1} || \sigma_{n,d,t}) \; ; \tag{B.15}$$

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial \sigma_{n,d}} &= \frac{1}{P}\sum_{p=1}^{P} \left(\frac{w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2} \right]}{\sum_{n=1}^{N} \left(w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2} \right] \right)} \left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \\
&= \frac{1}{P}\sum_{p=1}^{P} \left(\frac{w_n x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \; ; \\
\nabla \mathcal{L}(\sigma_{n,t})_d &= \frac{1}{P}\sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \; .
\end{aligned}$$

We use the following derivation for an update for σ based on an approximation of the relative-entropy distance formula. In the update of $\sigma_{n,d}$, we will approximate $\mathcal{D}_{\mathcal{RE}}(\times_{\sqcup+\infty} || \times_{\sqcup})$ with $\mathrm{d}_{RE}(\Theta_{n,t+1} || \Theta_{n,t})$.

Let $\boldsymbol{\xi} \in \mathbb{R}^D$. For distribution n,

$$\begin{aligned}
\mathrm{d}_{RE}\left(\Theta_{n,t+1} || \Theta_{n,t}\right) &= \int_{\boldsymbol{\xi} \in \mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln \left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})} \right) d\boldsymbol{\xi} \; ; \\
&= \int_{\boldsymbol{\xi} \in \mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln \left(\prod_{d=1}^{D} \frac{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d,t+1}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d}\right)}{\sigma_{n,d,t+1}}\right]^2} \right]}{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d,t}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d}\right)}{\sigma_{n,d,t}}\right]^2} \right]} \right) d\boldsymbol{\xi} \; ;
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{\sigma_{n,d,t}}{\sigma_{n,d,t+1}}\right) d\boldsymbol{\xi} \\
&\quad + \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})(\xi_d - \mu_{n,d})^2 \frac{1}{2}\left(\frac{1}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}^2}\right) d\boldsymbol{\xi} \ ; \\
&= \sum_{d=1}^{D} \left[\ln\left(\frac{\sigma_{n,d,t}}{\sigma_{t+1}}\right) + \frac{1}{2}\sigma_{n,d,t+1}^2\left(\frac{1}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}^2}\right)\right] \ ; \\
&= \sum_{d=1}^{D} \left[\ln\left(\frac{\sigma_{n,d,t}}{\sigma_{n,d,t+1}}\right) + \frac{1}{2}\left(\frac{\sigma_{n,d,t+1}^2}{\sigma_{n,d,t}^2} - 1\right)\right] \ .
\end{aligned}
$$

Differentiating with respect to $\sigma_{n,d,t+1}$ we get

$$
\frac{\sigma_{n,d,t+1}}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}} \ .
$$

To maximize $\hat{F}(\sigma_{n,d,t+1})$, we wish to solve

$$
\eta_n \nabla\mathcal{L}(\sigma_{n,t})_d + \frac{1}{\sigma_{n,d,t+1}} - \frac{\sigma_{n,t+1}}{\sigma_{n,t}^2} \ ;
$$

thus, the new update is

$$
\begin{aligned}
\sigma_{n,d,t+1} &= \frac{\sqrt{\eta_n^2 \nabla\mathcal{L}(\sigma_{n,t})_d{}^2 \sigma_{n,d,t}^4 + 4\sigma_{n,d,t}^2} + \eta_n \nabla\mathcal{L}(\sigma_{n,t})_d \sigma_{n,d,t}^2}{2} \\
&= \sqrt{\left(\frac{\eta_n}{2P}\sum_{p=1}^{P} \frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d}\right]\right)^2 + \sigma_{n,d,t}^2} \\
&\quad + \frac{\eta_n}{2P}\sum_{p=1}^{P} \frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d}\right] \ .
\end{aligned}
$$

C The Spherical Update

C.1 The Likelihood and Log Likelihood of the Raw Data

The *likelihood* of our raw data matrix, χ, is

$$\begin{aligned} likelihood(\chi) &= \prod_{p=1}^{P}\sum_{n=1}^{N}\left[w_n P\left(\boldsymbol{\chi}_p \mid \mu_n, \sigma_n\right)\right] \\ &= \prod_{p=1}^{P}\sum_{n=1}^{N}\left[w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]\right] . \end{aligned}$$

The *log likelihood* of our raw data matrix, χ, is

$$LogLike(\chi) = \frac{1}{P}\sum_{p=1}^{P}\ln\left[\sum_{n=1}^{N}\left(w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]\right)\right] .$$

The $1/P$ is a scaling factor.
As in the axis-parallel case, because, the Gaussians of our model are spherical; the multivariate density under distribution n of observation p of the raw data matrix, $\boldsymbol{\chi}_p$, is the product of the univariate densities of the D dimensions of the raw data, $\chi_{p,1} \ldots \chi_{p,D}$; thus, X is the *likelihood matrix*, if

$$x_{p,n} = \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right] .$$

C.2 Updating the Mixture Vector

$$
\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial w_n} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{\displaystyle\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]}{\displaystyle\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]\right)}\right) \\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\right); \\
\nabla\mathcal{L}(\mathbf{w}_t)_n &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right).
\end{aligned}
$$

The relative-entropy update, EG_η, uses the distance function

$$
\mathrm{d}_{re}(\mathbf{w}_{t+1}||\mathbf{w}_t)=\sum_{n=1}^{N}w_{n,t+1}\ln(w_{n,t+1}/w_{n,t}) \ .
$$

The derivation for the EG_η update in the spherical-Gaussian case is analogous with the derivation in the axis-parallel case. The EG_η update for a mixture-vector component is

$$
\begin{aligned}
w_{n,t+1} &= \frac{w_{n,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_n}}{\sum_{i=1}^{N}w_{i,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_i}} && \text{(C.1)} \\
&= \frac{w_{n,t}e^{\left[\frac{\eta}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right)\right]}}{\sum_{i=1}^{N}w_{i,t}e^{\left[\frac{\eta}{P}\sum_{p=1}^{P}\left(\frac{x_{p,i}}{\mathbf{x}_p\cdot\mathbf{w_t}}\right)\right]}}. && \text{(C.2)}
\end{aligned}
$$

C.3 Updating the μ Vector

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial \mu_{n,d}} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]}{\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]\right)}\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n^2}\right)\\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n^2}\right)\\
\nabla\mathcal{L}(\mu_{n,t})_d &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n^2}\right)\ .
\end{aligned}$$

Again, the derivation is analogous with the axis-parallel case, and the exponentiated-gradient update for μ is

$$\begin{aligned}
\mu_{n,d,t+1} &= \eta_n\sigma_n^2\nabla\mathcal{L}(\mu_{n,t})_d+\mu_{n,d,t} &\text{(C.3)}\\
&= \frac{\eta_n\sigma_n^2}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n^2}\right)+\mu_{n,d,t} &\text{(C.4)}\\
&= \frac{\eta_n}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}(\chi_{p,d}-\mu_{n,d})\right)+\mu_{n,d,t}\ . &\text{(C.5)}
\end{aligned}$$

C.4 The σ Update

In updating the variance, we actually update its square root, the standard deviation, σ.

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial \sigma_n} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]}{\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2}\right]\right)}\sum_{d=1}^{D}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_n^3}-\frac{1}{\sigma_n}\right]\right)\\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\sum_{d=1}^{D}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_n^3}-\frac{1}{\sigma_n}\right]\right)\ ;\\
\nabla\mathcal{L}(\sigma_{n,t})_d &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\sum_{d=1}^{D}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_n^3}-\frac{1}{\sigma_n}\right]\right)\ .
\end{aligned}$$

We use the following derivation for an update for σ based on an approximation of the relative-entropy distance formula. In the update of $\sigma_{n,d}$, we will approximate $\mathcal{D}_{\mathcal{RE}}(\times_{\sqcup+\infty}||\times_{\sqcup})$ with $\mathrm{d}_{RE}(\Theta_{n,t+1}||\Theta_{n,t})$.

Let $\boldsymbol{\xi} \in \mathbb{R}^D$. For distribution n,

$$\begin{aligned}
\mathrm{d}_{RE}\left(\Theta_{n,t+1}||\Theta_{n,t}\right) &= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi} \ ; \\
&= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}) \ln\left(\prod_{d=1}^{D} \frac{P_{n,d,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,d,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi} \ ; \\
&= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{\left[\frac{1}{\left(\sqrt{2\pi}\sigma_{n,t+1}\right)^D} e^{-\frac{1}{2}\sum_{d=1}^{D}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,t+1}}\right]^2}\right]}{\left[\frac{1}{\left(\sqrt{2\pi}\sigma_{n,t}\right)^D} e^{-\frac{1}{2}\sum_{d=1}^{D}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,t}}\right]^2}\right]}\right) d\boldsymbol{\xi} \ ; \\
&= D\int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{\sigma_{n,t}}{\sigma_{n,t+1}}\right) d\boldsymbol{\xi} \\
&\quad + \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})(\xi_d-\mu_{n,d})^2 \frac{1}{2}\left(\frac{1}{\sigma_{n,t}^2} - \frac{1}{\sigma_{n,t+1}^2}\right) d\boldsymbol{\xi} \ ; \\
&= D\ln\left(\frac{\sigma_{n,t}}{\sigma_{n,t+1}}\right) + \frac{D}{2}\sigma_{n,t+1}^2\left(\frac{1}{\sigma_{n,t}^2} - \frac{1}{\sigma_{n,t+1}^2}\right) \ ; \\
&= D\left[\ln\left(\frac{\sigma_{n,t}}{\sigma_{n,t+1}}\right) + \frac{1}{2}\left(\frac{\sigma_{n,t+1}^2}{\sigma_{n,t}^2} - 1\right)\right] \ .
\end{aligned}$$

Differentiating with respect to $\sigma_{n,t+1}$ we get

$$D\left(\frac{\sigma_{n,t+1}}{\sigma_{n,t}^2} - \frac{1}{\sigma_{n,t+1}}\right) \ .$$

We wish to solve (for $\sigma_{n,t+1}$)

$$\eta_n \nabla\mathcal{L}(\sigma_{n,t}) + D\left(\frac{1}{\sigma_{n,t+1}} - \frac{\sigma_{n,t+1}}{\sigma_{n,t}^2}\right) = 0 \ ;$$

thus, the new update is

$$\begin{aligned}
\sigma_{n,t+1} &= \frac{\sqrt{\eta_n^2\nabla\mathcal{L}(\sigma_{n,t})^2\sigma_{n,t}^4 + 4D^2\sigma_{n,t}^2} \ + \ \eta_n\nabla\mathcal{L}(\sigma_{n,t})\sigma_{n,t}^2}{2D} \\
&= \sqrt{\left(\frac{\eta_n}{2DP}\sum_{p=1}^{P}\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}} - \sigma_{n,d}\right]\right)^2 + \sigma_{n,d,t}^2}
\end{aligned}$$

$$+\frac{\eta_n}{2DP}\sum_{p=1}^{P}\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}}-\sigma_{n,d}\right] \quad .$$

D Axis-Parallel and Spherical Versions of EM

We need standard algorithms to act as controls in our experiments. These special versions of the EM algorithm work with axis-parallel and spherical Gaussian distributions; they will be the benchmarks that we require. To develop axis-parallel and spherical EM, remember that the standard EM updates for Gaussian mixture solutions are [1]

$$\begin{aligned}
\mu_{n,d,t+1} &= \frac{\sum_{p=1}^{P} P_t(n|\chi_p)\chi_p}{\sum_{p=1}^{P} P_t(n|\chi_p)} \\
\sigma^2_{n,d,t+1} &= \frac{\sum_{p=1}^{P} P_t(n|\chi_p)[\chi_p - \mu_{n,d,t+1}]^2}{\sum_{p=1}^{P} P_t(n|\chi_p)} \\
w_{n,t+1} &= \frac{1}{P}\sum_{p=1}^{P} P_t(n|\chi_p) \quad .
\end{aligned}$$

In the axis-parallel case, Bayes' theorem results in

$$\begin{aligned}
P_t(n|\chi_p) &= \frac{P_t(\chi_p|n)P_t(n)}{\sum_{i=1}^{N} P_t(\chi_p|i)P_t(i)} \\
&= \frac{P_t(\chi_p|n)w_{n,t}}{\sum_{i=1}^{N} P_t(\chi_p|i)w_{i,t}} \\
&= \frac{w_{n,t}\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]}{\sum_{i=1}^{N}\left(w_{i,t}\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{i,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{i,d}\right)}{\sigma_{i,d}}\right]^2}\right]\right)} \\
&= \frac{w_{n,t}x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \quad ;
\end{aligned}$$

thus, we have

$$\begin{aligned}
\mu_{n,d,t+1} &= \frac{\sum_{p=1}^{P} \frac{w_{n,t}x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\chi_{p,d}}{\sum_{p=1}^{P} \frac{w_{n,t}x_{p,i}}{\mathbf{x}_p\cdot\mathbf{w}}} \\
\sigma^2_{n,d,t+1} &= \frac{\sum_{p=1}^{P} \frac{w_{n,t}x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}[\chi_{p,d} - \mu_{n,d,t+1}]^2}{\sum_{p=1}^{P} \frac{w_{n,t}x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}} \\
w_{n,t+1} &= \frac{1}{P}\sum_{p=1}^{P} \frac{w_{n,t}x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \quad .
\end{aligned}$$

In the spherical case, Bayes' theorem results in

$$\begin{aligned}
P_t(n|\chi_p) &= \frac{P_t(\chi_p|n)P_t(n)}{\sum_{i=1}^{N} P_t(\chi_p|i)P_t(i)} \\
&= \frac{P_t(\chi_p|n)w_{n,t}}{\sum_{i=1}^{N} P_t(\chi_p|i)w_{i,t}}
\end{aligned}$$

$$
\begin{aligned}
& w_{n,t} \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_n}\right]^2} \right] \\
= & \frac{}{\sum_{n=1}^{N} \left(w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{i,d}\right)}{\sigma_i}\right]^2} \right] \right)} \\
= & \frac{w_{n,t} x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \quad ;
\end{aligned}
$$

thus, we have

$$
\begin{aligned}
\mu_{n,d,t+1} &= \frac{\sum_{p=1}^{P} \frac{w_{n,t} x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \chi_{p,d}}{\sum_{p=1}^{P} \frac{w_{n,t} x_{p,i}}{\mathbf{x}_p \cdot \mathbf{w}}} \\
\sigma^2_{n,d,t+1} &= \sum_{d=1}^{D} \frac{1}{D} \frac{\sum_{p=1}^{P} \frac{w_{n,t} x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \left[\chi_{p,d} - \mu_{n,d,t+1}\right]^2}{\sum_{p=1}^{P} \frac{w_{n,t} x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}}} \\
w_{n,t+1} &= \frac{1}{P} \sum_{p=1}^{P} \frac{w_{n,t} x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \quad .
\end{aligned}
$$

E The Axis-Parallel Updates of Manfred Warmuth's Method

E.1 The Likelihood and Log Likelihood of the Raw Data

The likelihood of our raw data matrix, χ, is

$$\begin{aligned} likelihood(\chi|\Theta) &= \prod_{p=1}^{P}\sum_{n=1}^{N}\Big[w_n P\left(\boldsymbol{\chi}_p \mid \mu_n, \sigma_n\right)\Big] \\ &= \prod_{p=1}^{P}\sum_{n=1}^{N}\left[w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right] . \end{aligned}$$

The log likelihood of our raw data matrix, χ, is

$$LogLike(\chi|\Theta) = \frac{1}{P}\sum_{p=1}^{P}\ln\left[\sum_{n=1}^{N}\left(w_n \prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right)\right] .$$

The $1/P$ is a scaling factor.

E.2 The Algorithm

In a yet unpublished manuscript [11], Manfred Warmuth introduces a new EG_η method built on his special approximation of the general form of the relative-entropy distance formula,

$$\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t) = \sum_{n=1}^{N} w_{n,t+1}\ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1}\mathrm{d}_{RE}(\Theta_{n,t+1}, \Theta_{n,t}) .$$

Mr. Warmuth uses a loss function based on the negative log likelihood, and he wishes to minimize

$$U_1(\Theta_{t+1}) = \mathrm{D}_{RE}(\Theta_{t+1}, \Theta_t) - \frac{\eta}{|\chi|}\ln(P(\chi|\Theta_{t+1})) . \tag{E.1}$$

An easier update, however, minimizes

$$U_2(\Theta_{t+1}) = \mathrm{D}_{RE}(\Theta_{t+1}, \Theta_t) - \frac{\eta}{|\chi|}\left(\ln(P(\chi|\Theta_t)) + (\Theta_{t+1}-\Theta_t)\nabla_{\Theta_t}\ln(P(\chi|\Theta_t))\right) .$$

If we remove the parts of equation $U_2(\Theta_{t+1})$ that do not depend upon Θ_{t+1}, we obtain

$$\hat{U}_2(\Theta_{t+1}) = \mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t) - \eta\nabla\mathcal{L}(\Theta_t)\cdot\Theta_{t+1} . \tag{E.2}$$

E.3 Updating the Mixture Vector

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial w_n} &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{\displaystyle\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]}{\displaystyle\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}}e^{-\frac{1}{2}\left[\frac{\left(\chi_{p,d}-\mu_{n,d}\right)}{\sigma_{n,d}}\right]^2}\right]\right)}\right) \\
&= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\right); \\
\nabla\mathcal{L}(\mathbf{w}_t)_n &= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\right) \ .
\end{aligned}$$

We add a Lagrangian term to equation (E.2) to enforce the constraint that

$$\sum_{n=1}^{N} w_{n,t+1} = 1 \ , \tag{E.3}$$

and

$$\hat{U}_2(\Theta_{t+1},\gamma) = \mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t) - \eta\nabla\mathcal{L}(\mathbf{w}_t)\cdot\mathbf{w}_{t+1} + \gamma\left(\sum_{n=1}^{N} w_{n,t+1} - 1\right), \quad \eta > 0 \ .$$

To minimize $\hat{U}_2(\Theta_{t+1},\gamma)$, set the N partial derivatives of $\hat{U}_2(\Theta_{t+1},\gamma)$ with respect to the components of $\mathbf{w}$ and evaluated at $\mathbf{w} = \mathbf{w}_{t+1}$ equal to zero and obtain

$$\frac{\partial\hat{U}_2(\Theta_{t+1},\gamma)}{\partial w_{n,t+1}} = -\frac{\partial\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t)}{\partial w_{n,t+1}} - \eta\nabla\mathcal{L}(\mathbf{w}_t)_n + \gamma = 0 \ . \tag{E.4}$$

Set the partial derivative with respect to γ of $\hat{U}_2(\Theta_{t+1},\gamma)$ equal to zero, and discover that

$$\sum_{n=1}^{N} w_{n,t+1} = 1 \ . \tag{E.5}$$

The N equations (E.4) become

$$\ln\frac{w_{n,t+1}}{w_{n,t}} + 1 + \mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t}) - \eta\nabla\mathcal{L}(\mathbf{w}_t)_n + \gamma = 0 \ .$$

Solving this system of equations for $w_{n,t+1}$ we have

$$\begin{aligned}
w_{n,t+1} &= w_{n,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_n - \gamma - 1 - \mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t})} \\
&= w_{n,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_n}e^{-\gamma-1-\mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t})} \\
&= ?\, w_{n,t}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_n}, \quad ? = e^{\gamma-1-\mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t})} \ .
\end{aligned}$$

Summing the N equations we get

$$\sum_{n=1}^{N} w_{n,t+1} = ? \sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i} \quad .$$

By equation (E.5),

$$? = \frac{1}{\sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i}} \quad ,$$

and we get the *exponentiated gradient* update for the mixture vector,

$$\begin{aligned} w_{n,t+1} &= \frac{w_{n,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_n}}{\sum_{i=1}^{N} w_{i,t} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i}} \\ &= \frac{w_{n,t} e^{\left[\frac{\eta}{P} \sum_{p=1}^{P} \left(\frac{x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w_t}} \right) \right]}}{\sum_{i=1}^{N} w_{i,t} e^{\left[\frac{\eta}{P} \sum_{p=1}^{P} \left(\frac{x_{p,i}}{\mathbf{x}_p \cdot \mathbf{w_t}} \right) \right]}} \quad . \end{aligned}$$

E.4 Updating the μ Vector

We are updating the mean vectors; therefore, equation (E.2) becomes

$$\hat{U}_2(\Theta_{t+1}) = \mathrm{D}_{RE}(\Theta_{t+1} || \Theta_t) - \eta_n \nabla \mathcal{L}(\mu_{n,t})_d \cdot \mu_{n,d,t+1} \quad ; \tag{E.6}$$

$$\begin{aligned} \frac{\partial LogLike(\chi|\Theta)}{\partial \mu_{n,d}} &= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}} \right]^2} \right]}{\sum_{n=1}^{N} \left(w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}} \right]^2} \right] \right)} \frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}^2} \right) \\ &= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}^2} \right) \quad ; \\ \nabla \mathcal{L}(\mu_{n,t})_d &= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}^2} \right) \quad . \end{aligned}$$

The update for μ is based on Mr. Warmuth's approximation of the relative-entropy distance formula,

$$\mathrm{D}_{RE}(\Theta_{t+1} || \Theta_t) = \sum_{n=1}^{N} w_{n,t+1} \ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1} \mathrm{d}_{RE}(\Theta_{n,t+1}, \Theta_{n,t}) \quad .$$

Let $\boldsymbol{\xi} \in \mathbb{R}^D$. For distribution n,

$$\begin{aligned}
\mathrm{d}_{RE}\left(\Theta_{n,t+1}||\Theta_{n,t}\right) &= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}\right) d\boldsymbol{\xi} \ ; \\
&= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln\left(\prod_{d=1}^{D} \frac{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t+1}\right)}{\sigma_{n,d}}\right]^2}\right]}{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t}\right)}{\sigma_{n,d}}\right]^2}\right]}\right) d\boldsymbol{\xi} \ ;
\end{aligned}$$

$$\begin{aligned}
&= \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t+1}\right)}{\sigma_{n,d}}\right]^2} d\boldsymbol{\xi} \\
&\quad -\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln e^{-\frac{1}{2}\left[\frac{\left(\xi_d-\mu_{n,d,t}\right)}{\sigma_{n,d}}\right]^2} d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \left[\frac{\left(\xi_d-\mu_{n,d,t+1}\right)^2 - \left(\xi_d-\mu_{n,d,t}\right)^2}{\sigma_{n,d}^2}\right] d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \left[\frac{\mu_{n,d,t+1}^2 - \mu_{n,d,t}^2 - 2\xi_d\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)}{\sigma_{n,d}^2}\right] d\boldsymbol{\xi} \ ; \\
&= -\frac{1}{2}\sum_{d=1}^{D} \left[\frac{\mu_{n,d,t+1}^2 - \mu_{n,d,t}^2 - 2\mu_{n,d,t+1}\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)}{\sigma_{n,d}^2}\right] \\
&= \frac{1}{2}\sum_{d=1}^{D} \frac{\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)^2}{\sigma_{n,d}^2} \ .
\end{aligned}$$

Differentiating with respect to $\mu_{n,d,t+1}$ we get

$$\frac{\partial \mathrm{d}_{RE}(\Theta_{t+1},\Theta_t)}{\partial \mu_{n,d,t+1}} = \frac{\left(\mu_{n,d,t+1}-\mu_{n,d,t}\right)}{\sigma_{n,d}^2} \ .$$

Remember

$$\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t) = \sum_{n=1}^{N} w_{n,t+1} \ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1} \mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t}) \ .$$

Differentiating $\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t)$ with respect to $\mu_{n,d,t+1}$ we obtain

$$\frac{\partial \mathrm{D}_{RE}(\Theta_{t+1},\Theta_t)}{\partial \mu_{n,d,t+1}} = w_n \frac{\partial \mathrm{d}_{RE}(\Theta_{t+1},\Theta_t)}{\partial \mu_{n,d,t+1}} \ .$$

To minimize $\hat{U}_2(\mu_{n,d,t+1})$, we wish to solve

$$w_n \frac{(\mu_{n,d,t+1} - \mu_{n,d,t})}{\sigma_{n,d}^2} - \eta_n \nabla \mathcal{L}(\mu_{n,t})_d \quad = \quad 0 \ ;$$

thus, the new update is

$$\begin{aligned}
\mu_{n,d,t+1} &= \frac{\eta_n}{w_n} \sigma_{n,d}^2 \nabla \mathcal{L}(\mu_{n,t})_d + \mu_{n,d,t} && \text{(E.7)} \\
&= \frac{\eta_n \sigma_{n,d}^2}{w_n P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}^2} \right) + \mu_{n,d,t} && \text{(E.8)} \\
&= \frac{\eta_n}{P} \sum_{p=1}^{P} \left(\frac{x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} (\chi_{p,d} - \mu_{n,d}) \right) + \mu_{n,d,t} \ . && \text{(E.9)}
\end{aligned}$$

E.5 The σ Update

We are updating σ; therefore, equation (E.2) becomes

$$\hat{U}_2(\Theta_{t+1}) \quad = \quad \mathrm{D}_{RE}(\Theta_{t+1} || \Theta_t) - \eta_n \nabla \mathcal{L}(\sigma_{n,t})_d \cdot \sigma_{n,d,t+1} \ ; \qquad \text{(E.10)}$$

$$\begin{aligned}
\frac{\partial LogLike(\chi|\Theta)}{\partial \sigma_{n,d}} &= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}} \right]^2} \right]}{\sum_{n=1}^{N} \left(w_n \prod_{d=1}^{D} \left[\frac{1}{\sqrt{2\pi}\sigma_{n,d}} e^{-\frac{1}{2}\left[\frac{(\chi_{p,d} - \mu_{n,d})}{\sigma_{n,d}} \right]^2} \right] \right)} \left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \\
&= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n}}{\mathbf{x}_p \cdot \mathbf{w}} \left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \ ; \\
\nabla \mathcal{L}(\sigma_{n,t})_d &= \frac{1}{P} \sum_{p=1}^{P} \left(\frac{w_n x_{p,n,t}}{\mathbf{x}_{p,t} \cdot \mathbf{w}} \left[\frac{(\chi_{p,d} - \mu_{n,d})^2}{\sigma_{n,d}^3} - \frac{1}{\sigma_{n,d}} \right] \right) \ .
\end{aligned}$$

The update for σ is based on Mr. Warmuth's approximation of the relative-entropy distance formula,

$$\mathrm{D}_{RE}(\Theta_{t+1} || \Theta_t) \quad = \quad \sum_{n=1}^{N} w_{n,t+1} \ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1} \mathrm{d}_{RE}(\Theta_{n,t+1}, \Theta_{n,t}) \ .$$

Let $\boldsymbol{\xi} \in \mathbb{R}^D$. For distribution n,

$$\mathrm{d}_{RE}\left(\Theta_{n,t+1} || \Theta_{n,t}\right) \quad = \quad \int_{\boldsymbol{\xi} \in \mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln \left(\frac{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})} \right) d\boldsymbol{\xi} \ ;$$

$$= \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln \left(\prod_{d=1}^{D} \frac{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d,t+1}} e^{-\frac{1}{2}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,d,t+1}}\right]^2} \right]}{\left[\frac{1}{\sqrt{2\pi}\sigma_{n,d,t}} e^{-\frac{1}{2}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,d,t}}\right]^2} \right]} \right) d\boldsymbol{\xi} \ ;$$

$$\begin{aligned}
&= \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1}) \ln \left(\frac{\sigma_{n,d,t}}{\sigma_{n,d,t+1}} \right) d\boldsymbol{\xi} \\
&\quad + \sum_{d=1}^{D} \int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})(\xi_d-\mu_{n,d})^2 \frac{1}{2} \left(\frac{1}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}^2} \right) d\boldsymbol{\xi} \ ; \\
&= \sum_{d=1}^{D} \left[\ln \left(\frac{\sigma_{n,d,t}}{\sigma_{t+1}} \right) + \frac{1}{2}\sigma_{n,d,t+1}^2 \left(\frac{1}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}^2} \right) \right] \ ; \\
&= \sum_{d=1}^{D} \left[\ln \left(\frac{\sigma_{n,d,t}}{\sigma_{n,d,t+1}} \right) + \frac{1}{2} \left(\frac{\sigma_{n,d,t+1}^2}{\sigma_{n,d,t}^2} - 1 \right) \right] \ .
\end{aligned}$$

Differentiating with respect to $\sigma_{n,d,t+1}$ we get

$$\frac{\partial \mathrm{d}_{RE}(\tilde{\Theta},\Theta)}{\partial \sigma_{n,d,t+1}} = \frac{\sigma_{n,d,t+1}}{\sigma_{n,d,t}^2} - \frac{1}{\sigma_{n,d,t+1}} \ .$$

Remember

$$\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t) = \sum_{n=1}^{N} w_{n,t+1} \ln(\frac{w_{n,t+1}}{w_{n,t}}) + \sum_{n=1}^{N} w_{n,t+1} \mathrm{d}_{RE}(\Theta_{n,t+1},\Theta_{n,t}) \ .$$

Differentiating $\mathrm{D}_{RE}(\Theta_{t+1}||\Theta_t)$ with respect to $\sigma_{n,d,t+1}$ we obtain

$$\frac{\partial \mathrm{D}_{RE}(\Theta_{t+1},\Theta_t)}{\partial \sigma_{n,d,t+1}} = w_n \frac{\partial \mathrm{d}_{RE}(\Theta_{t+1},\Theta_t)}{\partial \sigma_{n,d,t+1}} \ .$$

To minimize $\hat{U}_2(\sigma_{n,d,t+1})$, we wish to solve

$$w_n \left(\frac{1}{\sigma_{n,d,t+1}} - \frac{\sigma_{n,t+1}}{\sigma_{n,t}^2} \right) - \eta_n \nabla \mathcal{L}(\sigma_{n,t})_d = 0 \ ;$$

thus, the new update is

$$\sigma_{n,d,t+1} = \frac{\sqrt{\frac{\eta_n^2}{w_n^2} \nabla \mathcal{L}(\sigma_{n,t})_d^{\,2} \sigma_{n,d,t}^4 + 4\sigma_{n,d,t}^2} + \frac{\eta_n}{w_n} \nabla \mathcal{L}(\sigma_{n,t})_d \sigma_{n,d,t}^2}{2}$$

$$= \sqrt{\left(\frac{\eta_n}{2P}\sum_{p=1}^{P}\frac{x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}}-\sigma_{n,d}\right]\right)^2+\sigma_{n,d,t}^2}$$
$$+\frac{\eta_n}{2P}\sum_{p=1}^{P}\frac{x_{p,n,t}}{\mathbf{x}_{p,t}\cdot\mathbf{w}}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_{n,d}}-\sigma_{n,d}\right] \ .$$

F Reverse-Order Evaluation: $\mathbf{d}_{RE}(\Theta_t||\Theta_{t+1})$

In updating the variance, we actually update its square root, the standard deviation, σ.

$$\frac{\partial LogLike(\chi|\Theta)}{\partial\sigma_n} = \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n}\right]^2}\right]}{\sum_{n=1}^{N}\left(w_n\prod_{d=1}^{D}\left[\frac{1}{\sqrt{2\pi}\sigma_n}e^{-\frac{1}{2}\left[\frac{(\chi_{p,d}-\mu_{n,d})}{\sigma_n}\right]^2}\right]\right)}\sum_{d=1}^{D}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_n^3}-\frac{1}{\sigma_n}\right]\right)$$
$$= \frac{1}{P}\sum_{p=1}^{P}\left(\frac{w_n x_{p,n}}{\mathbf{x}_p\cdot\mathbf{w}}\sum_{d=1}^{D}\left[\frac{(\chi_{p,d}-\mu_{n,d})^2}{\sigma_n^3}-\frac{1}{\sigma_n}\right]\right) \ .$$

Here is the derivation for an update for σ based our approximation of the relative-entropy distance formula.

Let $\boldsymbol{\xi}\in\mathbb{R}^D$. For distribution n,

$$\mathrm{d}_{RE}\left(\Theta_{n,t}||\Theta_{n,t+1}\right) = \int_{\boldsymbol{\xi}\in\mathbb{R}^D}P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})\ln\left(\frac{P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})}{P_{n,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}\right)d\boldsymbol{\xi} \ ;$$
$$= \int_{\boldsymbol{\xi}\in\mathbb{R}^D}P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})\ln\left(\prod_{d=1}^{D}\frac{P_{n,d,t}(\boldsymbol{\xi}|\Theta_t)}{P_{n,d,t+1}(\boldsymbol{\xi}|\Theta_{n,t+1})}\right)d\boldsymbol{\xi} \ ;$$
$$= \int_{\boldsymbol{\xi}\in\mathbb{R}^D}P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})\ln\left(\frac{\left[\frac{1}{\left(\sqrt{2\pi}\sigma_{n,t}\right)^D}e^{-\frac{1}{2}\sum_{d=1}^{D}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,t}}\right]^2}\right]}{\left[\frac{1}{\left(\sqrt{2\pi}\sigma_{n,t+1}\right)^D}e^{-\frac{1}{2}\sum_{d=1}^{D}\left[\frac{(\xi_d-\mu_{n,d})}{\sigma_{n,t+1}}\right]^2}\right]}\right)d\boldsymbol{\xi} \ ;$$
$$= D\int_{\boldsymbol{\xi}\in\mathbb{R}^D}P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})\ln\left(\frac{\sigma_{n,t+1}}{\sigma_{n,t}}\right)d\boldsymbol{\xi}$$

$$
\begin{aligned}
& +D\int_{\boldsymbol{\xi}\in\mathbb{R}^D} P_{n,t}(\boldsymbol{\xi}|\Theta_{n,t})(\xi_d-\mu_{n,d})^2\frac{1}{2}\left(\frac{1}{\sigma^2_{n,t+1}}-\frac{1}{\sigma^2_{n,t}}\right)d\boldsymbol{\xi} \ ; \\
= \ & D\ln\left(\frac{\sigma_{n,t+1}}{\sigma_{n,t}}\right)+\frac{D}{2}\sigma^2_{n,t}\left(\frac{1}{\sigma^2_{n,t+1}}-\frac{1}{\sigma^2_{n,t}}\right) \ ; \\
= \ & D\left[\ln\left(\frac{\sigma_{n,t+1}}{\sigma_{n,t}}\right)+\frac{1}{2}\left(\frac{\sigma^2_{n,t}}{\sigma^2_{n,t+1}}-1\right)\right] \ .
\end{aligned}
$$

Differentiating with respect to $\sigma_{n,t+1}$ we get

$$D\left(\frac{1}{\sigma_{n,t+1}}-\frac{\sigma^2_{n,t}}{\sigma^3_{n,t+1}}\right) \ .$$

We wish to solve (for $\sigma_{n,t+1}$)

$$\eta_n\nabla\mathcal{L}(\sigma_{n,t})_i - D\left(\frac{1}{\sigma_{n,t+1}}-\frac{\sigma^2_{n,t}}{\sigma^3_{n,t+1}}\right) = 0 \ ;$$

thus, the new update is

$$
\begin{aligned}
\sigma_{n,t+1} \ = \ & \\
& \left[-\frac{1}{2}\left(\frac{D\sigma^2_{n,t}}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)+\frac{1}{27}\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^3+\frac{1}{18}\left(81\left(\frac{D\sigma^2_{n,t}}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^2-12\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^4\sigma^2_{n,t}\right)^{\frac{1}{2}}\right]^{\frac{1}{3}} \\
& +\frac{1}{9}\left(\frac{\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^2}{\left[-\frac{1}{2}\left(\frac{D\sigma^2_{n,t}}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)+\frac{1}{27}\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^3+\frac{1}{18}\left(81\left(\frac{D\sigma^2_{n,t}}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^2-12\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right)^4\sigma^2_{n,t}\right)^{\frac{1}{2}}\right]^{\frac{1}{3}}}\right) \\
& +\frac{1}{3}\left(\frac{D}{\eta_n\nabla\mathcal{L}(\sigma_{n,t})}\right) \ .
\end{aligned}
$$