A comparison of new and old algorithms for a mixture estimation problem

David P. Helmbold
Computer and Information Sciences
University of California
Santa Cruz, CA 95064
dph@cse.ucsc.edu

Robert E. Schapire
AT&T Bell Laboratories
600 Mountain Avenue, Room 2A-424
Murray Hill, NJ 07974
schapire@research.att.com

Yoram Singer
Institute of Computer Science
Hebrew University
Jerusalem 91904, Israel
singer@cs.huji.ac.il

Manfred K. Warmuth
Computer and Information Sciences
University of California
Santa Cruz, CA 95064
manfred@cse.ucsc.edu

Abstract

We investigate the problem of estimating the proportion vector which maximizes the likelihood of a given sample for a mixture of given densities. We adapt a framework developed for supervised learning and give simple derivations for many of the standard iterative algorithms like gradient projection and EM. In this framework, the distance between the new and old proportion vectors is used as a penalty term. The square distance leads to the gradient projection update, and the relative entropy to a new update which we call the exponentiated gradient update (EG_η). Curiously, when a second order Taylor expansion of the relative entropy is used, we arrive at an update EM_η which, for $\eta = 1$, gives the usual EM update. Experimentally, both the EM_η-update and the EG_η-update for $\eta > 1$ outperform the EM algorithm and its variants. We also prove a worst-case polynomial bound on the global rate of convergence of the EG_η algorithm.

1 Introduction

The problem of *maximum-likelihood* (ML) estimation of a mixture of densities is an important and well known learning problem [6]. ML estimators are asymptotically unbiased and are a basic tool for other more complicated problems such as clustering and learning hidden Markov models. We investigate the ML-estimation problem when the densities are given and only the mixture proportions are unknown. That is, we assume that we are given a set of distributions $D_1, \ldots, D_N$ over some domain, together with a sample of points from this domain. Our goal is to find the mixture coefficients $v_1, \ldots, v_N$ ($v_i \geq 0$ and $\sum v_i = 1$) which maximize (approximately) the likelihood of the sample under the mixture distribution $\sum v_i D_i$. Most of the common techniques to solve this problem are based on either gradient ascent iterative schemes [11] or on the Expectation Maximization (EM) algorithm for parameter estimation from incomplete data [5, 15].

We derive the standard iterative algorithms for the unsupervised mixture proportions estimation problem by placing them in a common hill-climbing framework. We will see later that this framework is very closely related to one developed by Kivinen and Warmuth [8] for supervised on-line learning. Our framework is very simple. On the tth iteration, we want to update the current mixture-coefficient vector $\mathbf{w}_t$ to an improved vector $\mathbf{w}_{t+1}$ by moving in a direction which increases the likelihood of the observations. On the other hand, the direction increasing the likelihood of the observations changes as we move. Thus, at each iteration, the new vector $\mathbf{w}_{t+1}$ is chosen to maximize the function

$$F(\mathbf{w}_{t+1}) = \eta \nabla \mathrm{LogLike}(\mathbf{w}_t) \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t) - d(\mathbf{w}_{t+1}, \mathbf{w}_t)$$

where $d(\mathbf{w}_{t+1}, \mathbf{w}_t)$ is a non-negative function measuring the distance between the new and old vectors, and η is a positive parameter. The first part of the right-hand-side is a first-order approximation of the increase in likelihood gained by moving to $\mathbf{w}_{t+1}$. The term $-d(\mathbf{w}_{t+1}, \mathbf{w}_t)$ is a penalty term which tends to keep $\mathbf{w}_{t+1}$ close to $\mathbf{w}_t$ (as measured by d). The relative importance between the penalty term and increasing the log-likelihood is governed by the positive parameter η, called the *learning rate*. Since the updates made by iterative methods are based on the local properties (i.e.

COLT'95 Santa Cruz, CA USA© 1995 ACM 0-89723-5/95/0007..$3.50

the gradient) at $\mathbf{w}_t$, the penalty term has the natural motivation as an indication of how rapidly we expect the local conditions to change as we move away from $\mathbf{w}_t$. We want to move as far as possible in a profitable direction but we have no guarantee that the most profitable direction at $\mathbf{w}_t$ remains so as we travel away from $\mathbf{w}_t$.

Maximizing the function F with different distance functions leads to various iterative update rules. Using the square distance gives the update rule of the gradient projection algorithm and the relative entropy distance gives a new update called the *exponentiated gradient* update (EG_η). By using a second order Taylor expansion of the relative entropy we get the χ^2 distance function. When this distance function is used and η is set to one, we get the same update as an iteration of the EM algorithm. Our experimental evidence suggests that setting $\eta > 1$ results in a more effective update. Moreover, we can show mathematically that when the current mixture vector $\mathbf{w}_t$ is close to the ML solution it is better to use a learning rate $\eta > 1$ rather than the rate $\eta = 1$. This shows that the EM algorithm is *not* optimal for this family of update rules.

For the exponentiated gradient algorithm, we are able to prove rigorous polynomial bounds on the number of iterations needed to get an arbitrarily good ML-estimator. However, this result assumes that there is a positive lower bound on the probability of each sample point under each of the given distributions. When no such lower bound exists (i.e., when some point has zero or near-zero probability under one of the distributions), we are able to prove similar but weaker bounds for a modified version of EG_η.

We obtain our global convergence results by viewing the mixture estimation problem as an on-line learning problem. Each iteration becomes a trial where the algorithm is charged a "loss" of $-\mathrm{LogLike}(\mathbf{w}_t)$, so minimizing the loss corresponds to maximizing the log-likelihood. Note that the ML solution will also have a loss on each trial. By bounding the extra loss of the algorithm over the loss incurred by the ML solution $\mathbf{u}$ over a sequence of iterations, we can show that at least one of the $\mathbf{w}_t$ vectors produced by the algorithm is reasonably good. Note that these results show convergence in log-likelihood rather than convergence of the mixture vector to the ML solution. Furthermore, the standard convergence results usually apply only when the algorithm is started with a vector near the ML solution.

The derivations of the learning rules using the above framework are very easy and can readily be applied to other settings. They are similar to previous derivations [15, 12]. Another motivation for the EM algorithm, based on the information geometry approach [2], was recently given by Amari [3]. His motivation seems to be related, although in a complicated way, to the one presented here.

2 Definitions and Problem Statement

Let $\mathbb{R}$ represent the real numbers. Let log denote the base 2 logarithm and let ln denote the natural logarithm. We say a vector $\mathbf{v} = (v_1, ..., v_N) \in \mathbb{R}^N$ is a *probability vector* if, $\forall i : v_i \geq 0$ and $\sum_{i=1}^{n} v_i = 1$. The vector $(1/N, \ldots, 1/N)$ is called the *uniform probability vector*. We use the following distance functions between probability vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathrm{D}_{RE}(\mathbf{u}||\mathbf{v}) \stackrel{\text{def}}{=} \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i}$$

and

$$\mathrm{D}_{\chi^2}(\mathbf{u}||\mathbf{v}) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{i=1}^{N} \frac{(u_i - v_i)^2}{v_i}.$$

Both are non-negative and zero iff $\mathbf{u} = \mathbf{v}$. The first one is the standard *relative entropy* and the second one is a second order Taylor approximation (at $\mathbf{u} = \mathbf{v}$) of the relative entropy called the *χ^2-distance*.

We consider the following maximum-likelihood mixture estimation problem:

Input: A $P \times N$ matrix X of non-negative real numbers with rows $\mathbf{x}_1$ through $\mathbf{x}_P$.
Goal: Find a probability vector $\mathbf{w}$ that maximizes the log-likelihood,

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

where $\mathbf{x}_p$ is the pth row of X.

The maximizers of the log-likelihood are called the *maximum likelihood* (ML) solutions. Clearly there might be more than one solution and throughout the paper $\mathbf{u}$ is used to denote an arbitrary ML solution and we call $\mathbf{u}$ "the ML solution." As there is no straightforward method for computing an ML solution, iterative methods which compute a sequence, $\mathbf{w}_1, \ldots, \mathbf{w}_t, \ldots$, converging to an ML solution are popular.

It is most natural to view each row $\mathbf{x}_p$ of X as representing an observation and the ith column of X as containing the probability of each observation under some known distribution D_i. The entry $x_{p,i}$ is then the probability under distribution D_i of the pth observation, and, for any probability vector $\mathbf{v}$, $\mathbf{x}_p \cdot \mathbf{v}$ is the probability under mixture $\mathbf{v}$ of the pth observation under the mixture distribution $\sum_{i=1}^{N} v_i D_i$. The ML solution $\mathbf{u}$ gives the proportions or weightings of the D_i's that maximize the log-likelihood of the observations.

We use $\nabla \mathcal{L}(\mathbf{w}_t)$ to represent the gradient of the log-likelihood function at probability vector $\mathbf{w}_t$,

$$\begin{aligned} \nabla \mathcal{L}(\mathbf{w}_t) &\stackrel{\text{def}}{=} \left(\frac{\partial \mathrm{LogLike}(\mathbf{w}_t)}{\partial w_{t,1}}, \ldots, \frac{\partial \mathrm{LogLike}(\mathbf{w}_t)}{\partial w_{t,N}} \right) \\ &= \left(\frac{1}{P} \sum_{p=1}^{P} \frac{x_{p,1}}{\mathbf{x}_p \cdot \mathbf{w}_t}, \ldots, \frac{1}{P} \sum_{p=1}^{P} \frac{x_{p,N}}{\mathbf{x}_p \cdot \mathbf{w}_t} \right) . \end{aligned}$$

3 The Updates

Assume that at iteration t we have the current probability vector $\mathbf{w}_t$ and are trying to find a better vector $\mathbf{w}_{t+1}$. Kivinen and Warmuth [8] study the supervised on-line setting where the vector $\mathbf{w}_t$ summarizes the learning done in previous iterations[1] and that learning can be preserved by choosing a $\mathbf{w}_{t+1}$ that is "close" to $\mathbf{w}_t$. Their method finds a new vector $\mathbf{w}_{t+1}$ that (approximately) maximizes the following function:

$$\hat{F}(\mathbf{w}_{t+1}) = \eta \text{LogLike}(\mathbf{w}_{t+1}) - d(\mathbf{w}_{t+1}, \mathbf{w}_t),\ \eta > 0\ . \quad (1)$$

The penalty term, $-d(\mathbf{w}_{t+1}, \mathbf{w}_t)$, tends to keep $\mathbf{w}_{t+1}$ close to $\mathbf{w}_t$ (with respect to the distance measure d) and the relative importance between the penalty term and maximizing the log-likelihood on the current iteration is governed by the positive parameter η, called the *learning rate*. A large learning rate means that maximizing the likelihood for the current row is emphasized while a small learning rate leads to an update which keeps $\mathbf{w}_{t+1}$ close to $\mathbf{w}_t$. Since our iterative updates will be based on the local conditions at the start vector $\mathbf{w}_t$, the penalty term and the learning rate measure how rapidly these local conditions are expected to change as we move away from $\mathbf{w}_t$. Unfortunately, finding a $\mathbf{w}_{t+1}$ maximizing $\hat{F}$ is intractable because $\nabla\mathcal{L}(\mathbf{w}_{t+1})$, the gradient of the log-likelihood at $\mathbf{w}_{t+1}$, is unknown. Kivinen and Warmuth bypass this difficulty by approximating $\nabla\mathcal{L}(\mathbf{w}_{t+1})$ by $\nabla\mathcal{L}(\mathbf{w}_t)$ and thus are really maximizing the function F from the introduction.

To maximize the function F from the introduction we add a Lagrange multiplier for the constraint that the components of $\mathbf{w}_{t+1}$ sum to one and set the N partial derivatives to zero. We also note that $\nabla\mathcal{L}(\mathbf{w}_t) \cdot \mathbf{w}_t$ is independent of $\mathbf{w}_{t+1}$, so maximizing F is equivalent to maximizing

$$\tilde{F}(\mathbf{w}_{t+1}, \gamma) = \eta\nabla\mathcal{L}(\mathbf{w}_t) \cdot \mathbf{w}_{t+1} - d(\mathbf{w}_{t+1}, \mathbf{w}_t) + \gamma\left(\sum_{i=1}^{N} w_{t,i} - 1\right),\quad \eta > 0\ . \quad (2)$$

Hence,

$$\frac{\partial\tilde{F}(\mathbf{w}_{t+1}, \gamma)}{\partial w_{t+1,i}} = \eta\nabla\mathcal{L}(\mathbf{w}_t)_i - \frac{\partial d(\mathbf{w}_{t+1}, \mathbf{w}_t)}{\partial w_{t+1,i}} + \gamma = 0\ .$$

Each partial derivative gives an expression for a component $w_{t+1,i}$ that contains γ but does not depend on $\mathbf{w}_{t+1}$; by setting the sum of these expressions to one we can solve for γ. Note that our choice of F avoids the dependence on the gradient $\nabla\mathcal{L}(\mathbf{w}_{t+1})$ which makes it impossible to solve for $\mathbf{w}_{t+1}$.

Since $\mathbf{w}_{t+1}$ has non-negative components which sum to one, the $\mathbf{w}_{t+1}$ that maximizes $\nabla\mathcal{L}(\mathbf{w}_t) \cdot \mathbf{w}_{t+1}$ is concentrated in the component(s) where $\nabla\mathcal{L}(\mathbf{w}_t)$ is largest. Therefore, this term pushes $\mathbf{w}_{t+1}$ in the direction of the gradient, $\nabla\mathcal{L}(\mathbf{w}_t)$ (as would be expected by its relation to $\nabla\mathcal{L}(\mathbf{w}_t) \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t)$). Again, the penalty term and learning rate have a natural interpretation as how rapidly we expect the local conditions (i.e., the gradient) to change as we move away from $\mathbf{w}_t$.

Using the above framework with the $\tilde{F}$ function we can now motivate all updates used in this paper. We begin by describing the gradient ascent based methods. The constraint that $\mathbf{w}_{t+1}$ must sum to one prevents us from naively following the gradient. However one can project the gradient onto the feasible region and follow this projected gradient. Maximizing $\tilde{F}$ using the distance function $\frac{1}{2}||\mathbf{w}_{t+1} - \mathbf{w}_t||_2^2 = \frac{1}{2}\sum_{i=1}^{N}(w_{t+1,i} - w_{t,i})^2$ leads to the standard *gradient projection update* [11]:

$$w_{t+1,i} = w_{t,i} + \eta\left(\nabla\mathcal{L}(\mathbf{w}_t)_i - \frac{1}{N}\sum_{i=1}^{N}\nabla\mathcal{L}(\mathbf{w}_t)_i\right)\ . \quad (3)$$

If we use the relative entropy $\sum_{i=1}^{n} w_{t+1,i}\log(w_{t+1,i}/w_{t,i})$ as a distance function in Equation (2) then the framework leads to a new update which we call the *exponentiated gradient* (EG_η) update:

$$w_{t+1,i} = \frac{w_{t,i}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_i}}{\sum_{j=1}^{N} w_{t,j}e^{\eta\nabla\mathcal{L}(\mathbf{w}_t)_j}}\ . \quad (4)$$

The framework can also be used to motivate the Expectation Maximization algorithm (EM) which is another algorithm commonly used for maximum likelihood estimation problems. For this we use the χ^2 (Chi-squared) distance measure $\frac{1}{2}\sum_{i=1}^{N}(w_{t+1,i} - w_{t,i})^2/w_{t,i}$ as the distance function in Equation (2) to obtain the update:

$$w_{t+1,i} = w_{t,i}\left(\eta\left(\nabla\mathcal{L}(\mathbf{w}_t)_i - 1\right) + 1\right)\ . \quad (5)$$

We call Equation (5) the EM_η-update because when $\eta = 1$ this gives the standard Expectation-Maximization (EM) update, $w_{t+1,i} = w_{t,i}\nabla\mathcal{L}(\mathbf{w}_t)_i$, for the problem considered in this paper. For the EM update ($\eta = 1$) the non-negativity constraints are automatically satisfied since $\nabla\mathcal{L}(\mathbf{w}_t)_i \geq 0$. However, for the EM_η-Update, the non-negativity constraints imply the condition $\eta < 1/(1 - \nabla\mathcal{L}(\mathbf{w}_t)_i)$ for those components of $\nabla\mathcal{L}(\mathbf{w}_t)_i$ which are less than one.

Since the χ^2 distance approximates the relative entropy it may not be surprising that the EM_η-update (5) also approximates the EG_η-update (4). We first rewrite the exponentiated gradient update by dividing the numerator and denominator by e^η and then replace the exponential function e^z by its first order lower bound $1 + z$:

$$\begin{aligned} w_{t+1,i} &= \frac{w_{t,i}e^{\eta(\nabla\mathcal{L}(\mathbf{w}_t)_i - 1)}}{\sum_{j=1}^{N} w_{t,j}e^{\eta(\nabla\mathcal{L}(\mathbf{w}_t)_j - 1)}} \\ &\approx \frac{w_{t,i}(1 + \eta(\nabla\mathcal{L}(\mathbf{w}_t)_i - 1))}{\sum_{j=1}^{N} w_{t,j}(1 + \eta(\nabla\mathcal{L}(\mathbf{w}_t)_j - 1))} \\ &= w_{t,i}(\eta(\nabla\mathcal{L}(\mathbf{w}_t)_i - 1) + 1)\ . \end{aligned}$$

Thus the EM_η-update can be viewed as a first order approximation of the EG_η-update. The approximation is

[1] In the on-line setting each iteration typically uses only a single observation. It is therefore desirable to preserve information about the previous observations while improving the likelihood of the current observation.

accurate when the exponents $\eta(\nabla\mathcal{L}(\mathbf{w}_t)_j - 1)$ are small. The advantage of the EM_η-update is that it is computationally cheaper as it avoids the exponentiation. However the EG_η-update is easier to analyze. Our experiments indicate that these two update rules tend to approximate each other well.

One can also get an update by re-parameterizing the probability vectors and doing unconstrained gradient ascent in the new parameter space. We use the standard exponential parameterization [13]: $w_i = e^{r_i}/\sum_{j=1}^{N} e^{r_j}$ and maximize the function

$$\mathrm{ParLogLike}(\mathbf{r}) = \mathrm{LogLike}(\mathbf{w}(\mathbf{r})).$$

(Note that the $\mathbf{w}$'s are probability vectors whereas the corresponding vectors $\mathbf{r}$ are unconstrained and lie in $\mathbb{R}^N$.) For this parameterization the updated weight vector becomes

$$\begin{aligned} r_{t+1,i} &= r_{t,i} + \eta \frac{\partial \mathrm{ParLogLike}(\mathbf{r}_t)}{\partial r_{t,i}} \\ &= r_{t,i} + \eta w_{t,i}(\nabla\mathcal{L}(\mathbf{w}_t)_i - 1) \ . \end{aligned}$$

This update is derived in our framework by approximately minimizing a function corresponding to $\hat{F}$ (Equation (1)):

$$\hat{G}(\mathbf{r}_{t+1}) = \eta \mathrm{ParLogLike}(\mathbf{r}_{t+1}) - d(\mathbf{r}_{t+1}, \mathbf{r}_t), \quad \eta > 0 \ .$$

For this minimization we use $d(\mathbf{r}_{t+1}, \mathbf{r}_t) = \frac{1}{2}||\mathbf{r}_{t+1} - \mathbf{r}_t||_2^2$ as a distance function and approximate the gradient at $\mathbf{r}_{t+1}$ with the gradient at $\mathbf{r}_t$.

All of the above update rules can be turned into algorithms by specifying the learning rate η to use in each iteration. The EM algorithm uses a fixed scheduling in which the same learning rate (namely, $\eta = 1$) is used in each iteration. Another possibility is to anneal the learning rate. At first, a high learning rate is used to quickly approach the ML solution. Later iterations use a lower learning rate to aid convergence.

The EM algorithm is in fact a limiting case of a more general approach usually called Generalized EM (GEM). Neal and Hinton [12] considered one variant of GEM which involves examining only a portion of the observation matrix X on each iteration. In general, any subset of the observations could be used, and the algorithm which considers a different row (observation) on each iteration is the natural analogue of on-line algorithms in the supervised case.

4 Convergence and Progress

Using standard methods, as in [11], we can show that all updates described in the previous section converge to an optimal ML solution. We can also show using standard infinitesimal analysis that the EG_η-update and the EM_η-update are contraction mappings, having the property $\| \mathbf{w}_{t+1} - \mathbf{u} \| \leq c \| \mathbf{w}_t - \mathbf{u} \|$ where $\mathbf{u}$ is the ML solution, $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$ are the vectors before and after an update (respectively), and $c < 1$ is a contraction factor. The factor c depends on the *minimal* and *maximal* eigenvalues of the matrix $I - \eta M$ where $M_{i,j} = \frac{w_i}{P}\sum_{p=1}^{P} \frac{x_{p,i}x_{p,j}}{(\mathbf{x}_p \cdot \mathbf{u})^2}$. Moreover, it can be shown that picking $\eta = 1$ in the EM_η-update is almost never the best choice (recall that the EM update uses $\eta = 1$).[2]

If an update is derived with a distance function d then it is interesting to analyze how fast the mixture vector moves towards an (unknown) ML solution $\mathbf{u}$ as measured by this distance function. More precisely, we use the same distance function that motivates the update as a potential function to obtain worst-case cumulative loss bounds over sequences of updates (similar to the methods applied to the supervised case [8]). The natural loss of a mixture vector $\mathbf{w}_t$ for our problem is $-\mathrm{LogLike}(\mathbf{w}_t)$. Note that this loss is unbounded since the likelihood for $\mathbf{w}_t$ is zero when there is some $\mathbf{x}_p$ for which $\mathbf{w}_t \cdot \mathbf{x}_p = 0$. In the supervised case, one can obtain firm worst-case loss bounds with respect to the square loss for various updates by analyzing the progress [8]. But the square loss is bounded and it is not surprising that it is much harder to obtain strong loss bounds for our (unbounded loss) unsupervised setting. Nevertheless this type of analysis can give insight on how an iterative algorithm moves towards the ML solution and on the relationships between different update rules. We obtained some reasonably good bounds for the GP_η and EG_η updates.

We deal with the unboundedness of the loss function by initially assuming that the smallest entry in the matrix is bounded away from zero. Thus, for all p and i we assume $x_{p,i} \geq r > 0$. Below, we give a proof bounding the average additional loss during T trials of the algorithm EG_η over the loss of the ML solution by

$$\frac{1}{r}\sqrt{\frac{\ln N}{2T}} \ .$$

Thus, by picking $T = \ln N / 2\epsilon^2 r^2$ we can guarantee that at least one of the $\mathbf{w}_t$'s computed by algorithm EG_η has loss at most ϵ larger than the ML solution.

This bound can be further refined to

$$\frac{1}{r}\sqrt{\frac{\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1)}{2T}}$$

where $\mathbf{u}$ is the ML solution, and $\mathbf{w}_1$ is the initial mixture vector. (This assumes that $\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1)$ is known a priori.)

In contrast, although omitted from this abstract, we have been able to prove a similar bound for the GP_η update[3] showing that the average additional loss during T trials of the algorithm GP_η above the loss of the ML solution is at most

$$\frac{||\mathbf{u} - \mathbf{w}_1||_2}{r}\sqrt{\frac{N}{T}} \leq \frac{1}{r}\sqrt{\frac{2N}{T}} \ .$$

[2] Peters and Walker [14] give a similar analysis; however, their motivation was based on a stochastic approximation approach and led to an unrealizable negative learning rate.

[3] This algorithm's performance was analyzed in the PAC model in [1].

However, the analysis assumes that the algorithm does not produce mixture vectors with negative components. This assumption may not hold generally since the update of the GP_η algorithm is additive. We have been unable to prove that the η used to obtain the above bound avoids this difficulty.

Even though the above bounds are weak in that they grow with $1/r$, they bring out a crucial difference between the exponentiated gradient and gradient descent family, namely, the logarithmic growth (in terms of N) of the additional loss bound of the former versus the square-root growth of the latter family. Similar observations were made in the supervised setting [8, 9].

We also show below how to obtain bounds when the entries in the matrix have zero-valued components. We essentially average the data matrix with a uniform matrix (this ϵ-Baysian averaging was also used in [1]) and then use the averaged matrix to run our algorithm. One can show that the ML solution for the averaged matrix is not too far (in loss) away from the ML solution of the original matrix, but the averaged matrix has the advantage of having entries bounded away from zero.

4.1 Convergence proofs for exponentiated-gradient algorithms

Recall that the EG_η algorithm receives a (fixed) set of P instances, $\mathbf{x}_1, \ldots, \mathbf{x}_P$, each in $\mathbb{R}^N$ with positive components. At each iteration, the algorithm produces a mixture or probability vector $\mathbf{w}_t \in \mathbb{R}^N$ and suffers a *loss* related to the log-likelihood of the set under the algorithm's mixture. The algorithm then updates $\mathbf{w}_t$.

The loss suffered by the algorithm at time t is

$$-\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p),$$

while the loss of the (unknown) ML solution $\mathbf{u}$ is

$$-\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p).$$

We are interested in bounding the (cumulative) difference between the loss of the algorithm and the loss of the ML solution.

We assume that $\max_i x_{t,i} = 1$ for all p. We make this assumption without loss of generality since multiplying an instance $\mathbf{x}_p$ by some constant simply adds a constant to both losses, leaving their difference unchanged. Put another way, the assumed lower bound r on $x_{p,i}$ used in Theorem 1 can be viewed as a lower bound on the ratio of the smallest to largest component of any instance $\mathbf{x}_p$.

The EG_η algorithm uses the update rule:

$$w_{t+1,i} = \frac{w_{t,i} \exp\left(\frac{\eta}{P}\sum_{p=1}^{P} \frac{x_{p,i}}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)}{Z_t}$$

where $\eta > 0$ is the learning rate, and Z_t is the normalization

$$Z_t = \sum_{i=1}^{N} w_{t,i} \exp\left(\frac{\eta}{P}\sum_{p=1}^{P} \frac{x_{p,i}}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) .$$

Theorem 1 *Let $\mathbf{u} \in \mathbb{R}^N$ be a probability vector, and let $\mathbf{x}_1, \ldots, \mathbf{x}_P$ be a sequence of instances with $x_{p,i} \geq r > 0$ for all i, p, and $\max_i x_{p,i} = 1$ for all p. For $\eta > 0$, EG_η produces a sequence of probability vectors $\mathbf{w}_1, \ldots, \mathbf{w}_T$ such that*

$$-\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p) \leq -\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + \frac{D_{RE}(\mathbf{u}||\mathbf{w}_1)}{\eta} + \frac{\eta T}{8r^2} . \quad (6)$$

Furthermore, if $\mathbf{w}_1$ is chosen to be the uniform probability vector, and we set

$$\eta = 2r\sqrt{\frac{2\ln N}{T}}$$

then

$$-\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p) \leq -\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + \frac{\sqrt{2T\ln N}}{2r} . \quad (7)$$

Proof: We have that

$$\begin{aligned} &\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_{t+1}) - \mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_t) \\ &= -\sum_i u_i \ln(w_{t+1,i}/w_{t,i}) \\ &= -\sum_i u_i \left(-\ln Z_t + \frac{\eta}{P}\sum_{p=1}^{P} \frac{x_{p,i}}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) \\ &= -\frac{\eta}{P}\sum_{p=1}^{P} \frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p} + \ln Z_t . \end{aligned} \quad (8)$$

We now work on bounding Z_t.

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} w_{t,i} \prod_{p=1}^{P} \exp\left(\frac{\eta}{P}\frac{x_{p,i}}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) \\ &= \sum_{i=1}^{N} w_{t,i} \prod_{p=1}^{P} \left(\exp\left(\frac{\eta}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)^{x_{p,i}}\right)^{1/P} \end{aligned}$$

Since $x_{t,i} \in [0,1]$ and since $\beta^x \leq 1-(1-\beta)x$ for $\beta > 0$ and $x \in [0,1]$ we can upper bound the right-hand side by:

$$\sum_{i=1}^{N} w_{t,i} \prod_{p=1}^{P} \left(1 - \left(1 - \exp\left(\frac{\eta}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)\right) x_{p,i}\right)^{1/P}$$

$$= \sum_{i=1}^{N} \prod_{p=1}^{P} \left(w_{t,i} - \left(1 - \exp\left(\frac{\eta}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)\right) w_{t,i} x_{p,i}\right)^{1/P}$$

We will need the following fact: For non-negative numbers $A_{i,p}$,

$$\sum_{i=1}^{N}\prod_{p=1}^{P}A_{i,p} \le \prod_{p=1}^{P}\left(\sum_{i=1}^{N}A_{i,p}^{P}\right)^{1/P}.$$

This fact can be proved by repeated application of Hölder's inequality.[4]

Using this fact with

$$A_{i,p} = \left(w_{t,i} - \left(1-\exp\left(\frac{\eta}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)\right)w_{t,i}x_{p,i}\right)^{1/P}$$

yields an upper bound on Z_t of

$$\begin{aligned}&\prod_{p=1}^{P}\left(\sum_{i=1}^{N}\left(w_{t,i} - \left(1-\exp\left(\frac{\eta}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)\right)w_{t,i}x_{p,i}\right)\right)^{1/P}\\ &= \prod_{p=1}^{P}\left(1-\mathbf{w}_t\cdot\mathbf{x}_p\left(1-\exp\left(\frac{\eta}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)\right)\right)^{1/P}.\end{aligned}$$

To further bound $\ln Z_t$, we apply the following:

Lemma 2 *For all $\alpha\in[0,1]$ and $x\in\mathbb{R}$,*

$$\ln(1-\alpha(1-e^x)) \le \alpha x + x^2/8\,.$$

Proof: Fix $\alpha\in[0,1]$, and let

$$f(x) = \alpha x + x^2/8 - \ln(1-\alpha(1-e^x))\,.$$

We wish to show that $f(x)\ge 0$. We have that

$$f'(x) = \alpha + \frac{x}{4} - g(x)$$

where

$$g(x) = \frac{\alpha e^x}{1-\alpha+\alpha e^x}.$$

Clearly, $f'(0)=0$. Further,

$$f''(x) = \frac{1}{4} - g(x) + (g(x))^2$$

which is non-negative for all x (the minimum is attained when $g(x)=1/2$). Therefore, f is minimized when $x=0$; since $f(0)=0$, this proves the claim. ■

Taking logs of upper bound on Z_t and then applying Lemma 2, we have

$$\begin{aligned}&\ln Z_t\\ &\le \frac{1}{P}\sum_{p=1}^{P}\ln\left(1-\mathbf{w}_t\cdot\mathbf{x}_p\left(1-\exp\left(\frac{\eta}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)\right)\right)\\ &\le \frac{1}{P}\sum_{p=1}^{P}\left[\eta+\frac{1}{8}\left(\frac{\eta}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)^2\right]\\ &\le \eta+\frac{\eta^2}{8r^2}\end{aligned}$$

since r is a lower bound on $\mathbf{w}_t\cdot\mathbf{x}_p$. Plugging into Equation (8) we obtain

$$\begin{aligned}&\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_{t+1}) - \mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_t)\\ &\le -\frac{\eta}{P}\sum_{p=1}^{P}\left(\frac{\mathbf{u}\cdot\mathbf{x}_p}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)+\eta+\frac{\eta^2}{8r^2}\\ &= \frac{\eta}{P}\sum_{p=1}^{P}\left(1-\frac{\mathbf{u}\cdot\mathbf{x}_p}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)+\frac{\eta^2}{8r^2}\\ &\le \frac{\eta}{P}\sum_{p=1}^{P}\left(-\ln\frac{\mathbf{u}\cdot\mathbf{x}_p}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)+\frac{\eta^2}{8r^2}\end{aligned}$$

using the fact that $1-e^x\le -x$ for all real x. By summing over all $t\le T$ we get

$$\begin{aligned}-\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1) &\le \mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_T) - \mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1)\\ &\le \frac{\eta}{P}\sum_{t=1}^{T}\sum_{p=1}^{P}\left(-\ln\frac{\mathbf{u}\cdot\mathbf{x}_p}{\mathbf{w}_t\cdot\mathbf{x}_p}\right)+\frac{T\eta^2}{8r^2}\,,\end{aligned}$$

which implies the first bound (6) stated in the theorem. The second bound (7) follows by straightforward algebra, noting that $\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1)\le\ln N$ when $\mathbf{w}_1$ is the uniform probability vector. ■

Note that if any other upper bound K on $\mathrm{D}_{RE}(\mathbf{u}||\mathbf{w}_1)$ is known a priori (possibly for some other choice of $\mathbf{w}_1$), then the $\ln N$ in the bound (7) of the theorem can be replaced by K.

It follows from Theorem 1 that, if we run for T iterations, then the *average* loss (i.e. minus log-likelihood) of the $\mathbf{w}_t$'s will be at most

$$\sqrt{\frac{\ln N}{2Tr^2}}.$$

greater than the loss of $\mathbf{u}$. Therefore, picking $T=(\ln N)/(2\epsilon^2r^2)$ guarantees that at least one of the $\mathbf{w}_t$'s will have a log-likelihood within ϵ of $\mathbf{u}$.

When some of the components $x_{p,i}$ are zero, or very close to zero, we can use the following algorithm which is parameterized by a real number $\alpha\in[0,1]$. Let

$$\tilde{\mathbf{x}}_p = (1-\alpha/N)\mathbf{x}_p + (\alpha/N)\mathbf{1}$$

where $\mathbf{1}$ is the all 1's vector. As before, we maintain a probability vector $\mathbf{w}_t$ which is updated using $\tilde{\mathbf{x}}_p$ rather than $\mathbf{x}_p$:

$$w_{t+1,i} = \frac{w_{t,i}\exp(\eta\tilde{x}_{p,i}/\mathbf{w}_t\cdot\tilde{\mathbf{x}}_p)}{\sum_i w_{t,i}\exp(\eta\tilde{x}_{p,i}/\mathbf{w}_t\cdot\tilde{\mathbf{x}}_p)}.$$

The vector that we output is also slightly modified. Specifically, the algorithm outputs the mixture

$$\tilde{\mathbf{w}}_t = (1-\alpha)\mathbf{w}_t + (\alpha/N)\mathbf{1}$$

and so suffers loss $-\ln(\tilde{\mathbf{w}}_t\cdot\mathbf{x}_p)$.

We call this modified procedure $\widetilde{\mathrm{EG}}_{\alpha,\eta}$.

[4] In one form, Hölder's inequality states that, for non-negative a_i, b_i, $\sum_i a_ib_i \le \left(\sum_i a_i^p\right)^{1/p}\left(\sum_i b_i^q\right)^{1/q}$ for any positive p,q satisfying $1/p+1/q=1$.

Theorem 3 *Let $\mathbf{u} \in \mathbb{R}^N$ be any probability vector, and let $\mathbf{x}_1, \ldots, \mathbf{x}_P$ be a sequence of instances with $x_{p,i} \geq 0$ for all i, p, and $\max_i x_{t,i} = 1$ for all p. For $\alpha \in (0, 1/2]$ and $\eta > 0$, $\widetilde{EG}_{\alpha,\eta}$ produces a sequence of probability vectors $\tilde{\mathbf{w}}_1, \ldots, \tilde{\mathbf{w}}_T$ such that*

$$-\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P}\ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) \leq -\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P}\ln(\mathbf{u} \cdot \mathbf{x}_p) + 2\alpha T + \frac{D_{RE}(\mathbf{u}||\mathbf{w}_1)}{\eta} + \frac{\eta T N^2}{8\alpha^2}. \quad (9)$$

Furthermore, if $\mathbf{w}_1$ is chosen to be the uniform probability vector, $T \geq 2N^2 \ln N$, and we set

$$\alpha = \left(\frac{N^2 \ln N}{8T}\right)^{1/4}$$

$$\eta = \frac{2\alpha}{N}\sqrt{\frac{2\ln N}{T}}$$

then

$$-\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P}\ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) \leq -\sum_{t=1}^{T}\frac{1}{P}\sum_{p=1}^{P}\ln(\mathbf{u} \cdot \mathbf{x}_p) + 2(2N^2 \ln N)^{1/4}(T)^{3/4}. \quad (10)$$

Proof: From our assumption that $\max_i x_{t,i} = 1$, we have

$$\frac{\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p} \geq \frac{(1-\alpha)\mathbf{w}_t \cdot \mathbf{x}_p + \alpha/N}{(1-\alpha/N)\mathbf{w}_t \cdot \mathbf{x}_p + \alpha/N}.$$

The right hand side of this inequality is decreasing as a function of $\mathbf{w}_t \cdot \mathbf{x}_p$ and so is minimized when $\mathbf{w}_t \cdot \mathbf{x}_p = 1$. Thus,

$$\frac{\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p} \geq (1-\alpha) + \alpha/N,$$

or equivalently,

$$\begin{aligned} -\ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) &\leq -\ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) - \ln(1 - \alpha + \alpha/N) \\ &\leq -\ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) + 2\alpha \quad (11) \end{aligned}$$

(since $\alpha \leq 1/2$).

From Theorem 1 applied to the instances $\tilde{\mathbf{x}}_p$, we have that

$$-\sum_{t=1}^{T}\ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) \leq -\sum_{t=1}^{T}\ln(\mathbf{u} \cdot \tilde{\mathbf{x}}_p) + \frac{D_{RE}(\mathbf{w}_1||\mathbf{u})}{\eta} + \frac{\eta T N^2}{8\alpha^2} \quad (12)$$

where we used the fact that $\tilde{x}_{p,i} \geq \alpha/N$.

Note that

$$\mathbf{u} \cdot \tilde{\mathbf{x}}_p = (1 - \alpha/N)\mathbf{u} \cdot \mathbf{x}_p + \alpha/N \geq \mathbf{u} \cdot \mathbf{x}_p.$$

Combined with equations (11) and (12), and summing over all t, this gives the first bound (9) of the theorem. The second bound follows from the fact that $D_{RE}(\mathbf{u}||\mathbf{w}_1) \leq \ln N$ when $\mathbf{w}_1$ is the uniform probability vector. ■

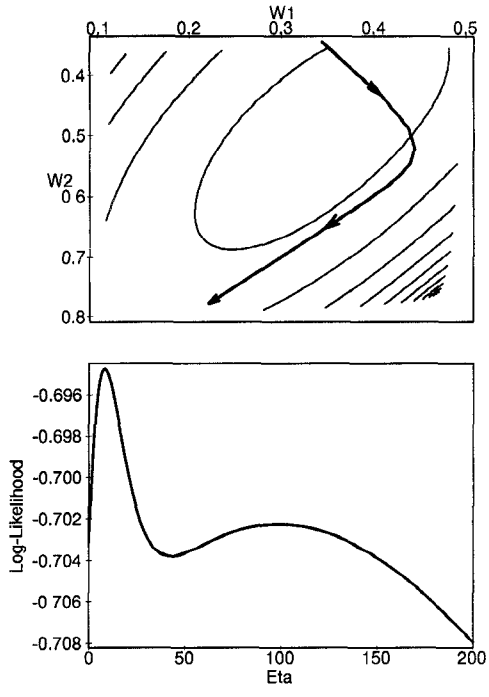

Figure 1: When the EG_η update is used, the log-likelihood as a function of η may have local maxima. At the bottom part of the figure, the log-likelihood is plotted as a function of η for a given $\mathbf{w}_t$. At the top, the corresponding path is plotted over the log-likelihood as a function of the first two weights $w_{t+1,1}$ and $w_{t+1,2}$ (denoted in the figure by W1 and W2).

5 Experimental Results

In this section we briefly present and discuss a few of the empirical tests we performed. In order to compare the various algorithms, data was synthetically created from N normal distributions evenly spaced on the unit circle in $\mathbb{R}^2$. The ith distribution was generated from a normal distribution with a mean vector $\vec{\mu} = \left(\sin(\frac{2\pi i}{N}), \cos(\frac{2\pi i}{N})\right)$. Each observation was created by uniformly picking one of the distributions, and sampling that distribution to obtain a point $\vec{\xi} = (\xi_1, \xi_2) \in \mathbb{R}^2$. The corresponding row of X contains the probability density at $\vec{\xi}$ for each of the N distributions. The examples presented in this section were obtained by generating hundreds of observations ($P \geq 100$) from at least 5 distributions ($N \geq 5$) each with variance 1. The same qualitative results are obtained when using matrices of different sizes and other stochastic sources (such as the uniform distribution). We tested all the described algorithms. The algorithms were tested using both fixed scheduling and line-searches to find the best choice of η on each iteration. The line-searches allow us to compare the updates when they are optimally tuned. Note that when the EG_η-update is used, the likelihood may have two local maxima as a function of η as shown in Figure 1, so the searches must be careful to pick the global maximum.

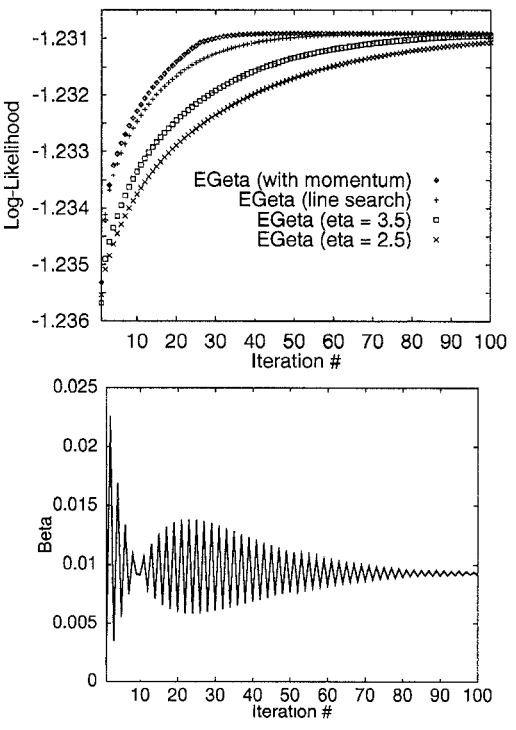

Figure 2: Top: The log-likelihood using the exponentiated gradient algorithm, with line-searches, line-searches plus a momentum term, and fixed scheduling with $\eta = 3.5$ and $\eta = 2.5$. The fixed schedulings are only slightly worse than setting the rate by expensive line-searches, while adding a momentum term accelerates the increase in the likelihood. Bottom: The values of $\beta = e^{-\eta}$ when using line-searches for the exponentiated gradient update. The β-value oscillates, eventually converging to a typical value. This anomaly is common with gradient ascent algorithms.

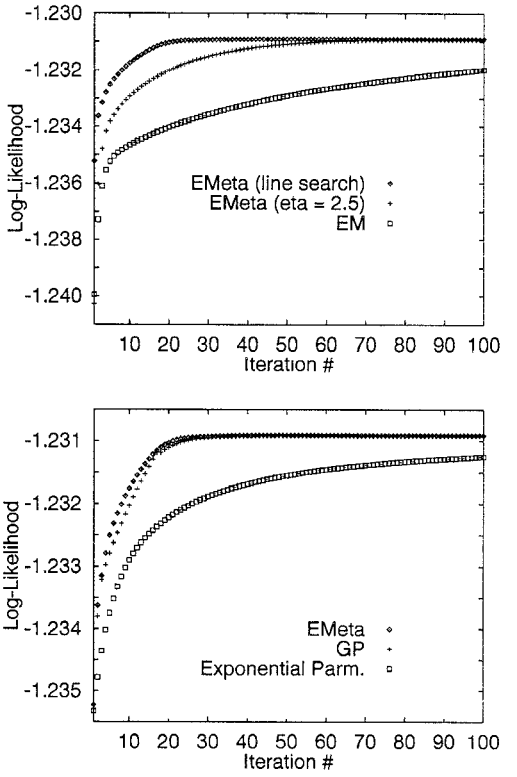

Figure 3: Top: Comparison of the performance of the EM_η-update algorithm and the standard EM algorithm. The EM_η-update clearly outperforms the standard EM algorithm, even when a fixed conservative scheduling is used. Bottom: comparison of the EM_η-update with gradient ascent algorithms. The gradient-projection is comparable to the EM_η-update and the gradient ascent update with exponential parameterization is inferior.

The optimal learning rate determined by the line-searches tended to oscillate, as shown at the bottom part of Figure 2. When a momentum term was added, the oscillations were damped and the convergence was accelerated.[5]

Using fixed scheduling turned out to be a competitive alternative to the expensive line-searches. All these phenomena are depicted at the top part of Figure 2.

The gradient ascent update with exponential parameterization appears inferior to all other methods. This may be due to the ad-hoc exponential parameterization. A good fixed scheduling for that method is difficult to obtain as the optimal learning rate has large oscillations. The EM_η and EG_η updates have about the same performance, which is expected as the EM_η update approximates the EG_η update. Both methods outperform the EM algorithm and the EM_η and EG_η updates are superior to the EM algorithm even when η is set to a fixed value greater than one (see Figure 3).

The EM_η and EG_η updates are competitive with the gradient projection update, and in fact there is no clear winner. As discussed in Section 4, we have some theoretical evidence that the gradient projection update beats both EM_η and EG_η when the algorithms are started at the uniform vector and most of the components of the ML solution are large. However, when most of the components of the ML solution are very small, the EM_η and EG_η updates tend to be better. Thus, as in other settings [8, 9, 10], updates based on the relative entropy tend to be the best when the solution is sparse (see Figure 4).

We also compared the performance of the various updates with second order methods. Second order methods (also known as Newton methods) are based on a quadratic approximation of the objective function. Near

[5]The *conjugate gradient* search is a method for iteratively searching a quadratic cost function [11, 7]. When the cost function is non-quadratic, as is the likelihood function in our case, a variant of the conjugate gradient method can be devised. This variant, termed partial conjugate gradient (PCG), is restarted after every K conjugate gradient steps, so that the search direction every K iteration becomes the gradient. Adding a momentum term can be seen as an approximation of the partial conjugate gradient algorithm, with no restarts (i.e., the PCG method with $K \to \infty$).

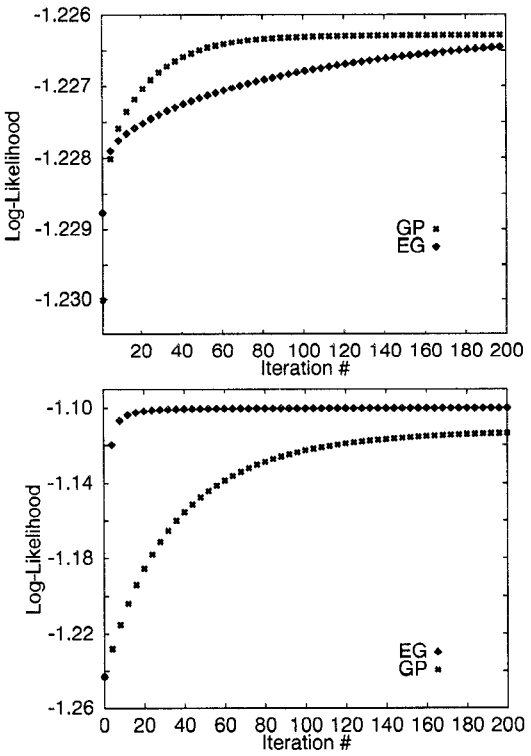

Figure 4: Comparisons the performance of the EG_η and GP_η algorithms. In both plots there were 10 Gaussian distributions with centers on the unit circle. On the left, the observations were generated from a (slightly perturbed) uniform mixture of the Gaussians. On the right, the observations were generated from a perturbation of the uniform mixture on only 5 of the Gaussians. This (and similar experiments) indicate that the EG_η algorithm performs better when the optimum mixture vector $\mathbf{u}$ has few large components.

the solution we can approximate the log-likelihood by the truncated Taylor series,

$$\begin{aligned}\text{LogLike}(\mathbf{w}) \approx\ & \text{LogLike}(\mathbf{w}_t) + \nabla\mathcal{L}(\mathbf{w}_t)^T(\mathbf{w}-\mathbf{w}_t) \\ & +\frac{1}{2}(\mathbf{w}-\mathbf{w}_t)^T H(\mathbf{w}_t)(\mathbf{w}-\mathbf{w}_t)\ ,\end{aligned}$$

where $H(\mathbf{w}_t)$ is the Hessian calculated at $\mathbf{w}_t$,

$$H_{ij}(\mathbf{w}_t) = \frac{\partial^2}{\partial w_{t,i}\partial w_{t,j}}\text{LogLike}(\mathbf{w}_t)\ .$$

The right-hand side is minimized at,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - H(\mathbf{w}_t)^{-1}\nabla\mathcal{L}(\mathbf{w}_t)\ .$$

This is the basic Newton method, which requires calculations of second order derivatives and inversions of the Hessian. Newton methods converge to a vector close to the solution in fewer updates than the EM_η and EG_η updates. However, the EM_η and EG_η updates can often do significantly more iterations than Newton methods with the same computational effort. When N is sufficiently large (we found experimentally that it is enough

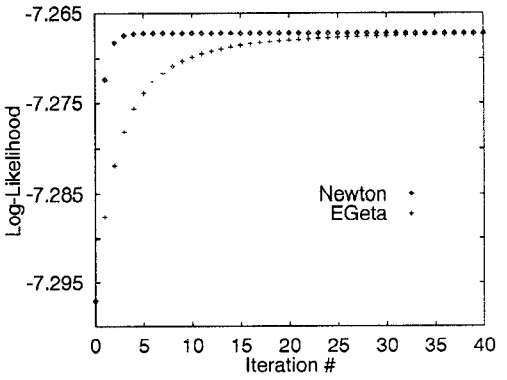

Figure 5: Comparison of EM_η-update with second order algorithms (Newton methods). Second order methods are better than the EM_η-update algorithms with line searches however they require second order derivative calculations and expensive $N \times N$ matrix inversions.

that $N \geq 10$), the EM_η and EG_η outperform Newton methods when computational cost (rather than number of iterations) is considered.

6 Applications and future research

A thorough understanding of the mixture proportions problem has many important applications. For example, an s-state hidden Markov model (HMM) may be viewed as $2s+1$ essentially independent mixtures: one for the initial state distribution, one for the transitions leaving each state, and one for the output probabilities at each state. Our distance functions are easily generalized to the HMM case by summing the distances for each of the $2s+1$ probability vectors defining the s state HMM. Thus, the EM_η-update is derived using the sum of the χ^2 distances over all $2s+1$ mixtures. Similarly, the EG_η-update for HMMs is obtained by summing the relative entropies as the distance function. Again, the EM_1-update is EM (which is usually called Baum-Welch in this context) and the EM_η-update is a first order approximation to the EG_η-update.

Identifying the distance function associated with an update helps explain what the update is doing and facilitates comparisons between iterative methods. After explaining the standard algorithms using distance functions we might ask what are the distance functions most appropriate for a particular situation. One important area for future research is identifying good distance functions when the parameters do not form a probability vector. In particular, we are attempting to apply this methodology to mixtures of Gaussians with arbitrary mean and variance. In this more complicated setting we need distance functions that depend on the means and variances given to the Gaussians as well as the mixture probabilities assigned to them.

Our framework naturally leads to on-line versions of our algorithms where only a single observation (instead of

the whole matrix) is used each iteration. In particular, we have derived an on-line version of EM_η. Experimentally, this version outperforms the known on-line versions of EM (called Generalized-EM). We have also applied the on-line version of our algorithms to a portfolio selection problem investigated by Cover [4]. Although Cover's analytical bounds appear better than ours, preliminary experimental results indicate that EM_η and EG_η outperform Cover's algorithm on historical stock market data. Furthermore, our algorithms are computationally efficient while Cover's algorithm is exponential in the number of possible investments. A complete analysis of this on-line version of the mixtures problem will be presented in a companion paper.

Acknowledgments

We thank Jyrki Kivinen for helpful discussions. Yoram Singer acknowledges the Clore Foundation for its support. Part of this research was done while he was visiting AT&T Bell Labs, and the Computer and Information Sciences department at the University of California, Santa Cruz. Manfred K. Warmuth received funding from NSF grant IRI-9123692.

References

[1] N. Abe, J. Takeuchi, and M. K. Warmuth. Polynomial learnability of probablistic concepts with respect to the Kullback-Leibler divergence. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 277–289. Morgan Kaufmann, 1991.

[2] S. Amari. *Differential Geometrical Methods in Statisitcs.* Springer-Verlag, 1985.

[3] S. Amari. The EM algorithm and information geometry in neural-network learning. *Neural Computation*, 7(1):13–18, 1995.

[4] T. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B39:1–38, 1977.

[6] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns-Hopkins University Press, 1989.

[8] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, 1995.

[9] J. Kivinen and M. K. Warmuth. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proceedings of the Eighth Annual Workshop on Computational Learning Theory*, July 1995.

[10] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[11] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, 1984.

[12] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Unpublished manuscript, 1993.

[13] S. J. Nowlan. *Soft Competative Adaption: Neural Network Learning Algorithms Based on Fitting Statistical Mixtures.* PhD thesis, Carnegie Mellon University, 1991.

[14] B. C. Peters and H. F. Walker. The numerical evaluation of the maximum-likelihood estimates of a subset of mixture proportions. *SIAM Journal of Applied Mathematics*, 35:447–452, 1978.

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