Machine Learning, 27(1):97-119, 1997.

A Comparison of New and Old Algorithms for A Mixture Estimation Problem

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

ROBERT E. SCHAPIRE schapire@research.att.com
AT&T Labs, 600 Mountain Avenue, Murray Hill, NJ 07974

YORAM SINGER singer@research.att.com
AT&T Labs, 600 Mountain Avenue, Murray Hill, NJ 07974

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

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 polynomial bound on the 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 [5]. 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 [4], [16].

We derive the standard iterative algorithms for the unsupervised mixture proportions estimation problem by placing them in a common hill-climbing framework. This framework is analogous to the one developed by Kivinen and Warmuth [8] for supervised on-line learning. Our goal is to maximize the log likelihood of the observations as a function of the mixture vector $\mathbf{w}$, denoted by $\mathrm{LogLike}(\mathbf{w})$. This is computationally hard and requires iterative methods. In the tth iteration we approximate the log-likelihood $\mathrm{LogLike}(\mathbf{w}_{t+1})$ at the new mixture vector $\mathbf{w}_{t+1}$ by $\mathrm{LogLike}(\mathbf{w}_t)+\nabla\mathrm{LogLike}(\mathbf{w}_t)\cdot(\mathbf{w}_{t+1}-\mathbf{w}_t)$, which is the Taylor expansion of the log-likelihood around the old mixture vector $\mathbf{w}_t$. It is now easy to maximize this approximated log-likelihood. However the approximation degrades the further we move from the old mixture vector $\mathbf{w}_{t+1}$. Thus we subtract a penalty term $d(\mathbf{w}_{t+1}, \mathbf{w}_t)$ which is a non-negative function measuring the distance between the new and old mixture vector. This penalty term keeps $\mathbf{w}_{t+1}$ close to $\mathbf{w}_t$ as measured by the distance function d. In summary we are maximizing the function

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

The relative importance between the penalty term and increasing the log-likelihood is governed by the positive parameter η, called the *learning rate.*

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 for the simple mixture estimation problem considered in this paper. Our experimental evidence suggests that setting $\eta > 1$ results in a more effective update. These results agree with the infinitesimal analysis in the limit of $n \rightarrow \infty$ based on a stochastic approximation approach [14], [15], [16].

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 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 rate of convergence results usually apply only when the algorithm is started with a vector near the ML solution, whereas our results show convergence for *any* initial probability vector with strictly positive components.

The derivations of the learning rules using the above framework are simple and can readily be applied to other settings. They are similar to previous derivations found in the literature [16], [13].

2. Definitions and Problem Statement

Let $\mathbb{R}$ represent the real numbers. 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 $\mathbf{v}$:

$$\begin{aligned} \mathrm{d}_{EUC}(\mathbf{u}||\mathbf{v}) &\stackrel{\text{def}}{=} \tfrac{1}{2}\sum_{i=1}^{N}(u_i - v_i)^2 = \tfrac{1}{2}||\mathbf{u}-\mathbf{v}||^2 \\ \mathrm{d}_{RE}(\mathbf{u}||\mathbf{v}) &\stackrel{\text{def}}{=} \sum_{i=1}^{N} u_i \ln \frac{u_i}{v_i} \qquad \text{and} \\ \mathrm{d}_{\chi^2}(\mathbf{u}||\mathbf{v}) &\stackrel{\text{def}}{=} \tfrac{1}{2}\sum_{i=1}^{N} \frac{(u_i - v_i)^2}{v_i} \ . \end{aligned}$$

All three distance functions are non-negative and zero iff $\mathbf{u} = \mathbf{v}$. The first one is half of the square of the Euclidean length of the vector $\mathbf{u} - \mathbf{v}$. The second one is the standard *relative entropy* and the last one is a second order Taylor approximation (at $\mathbf{u} = \mathbf{v}$) of the relative entropy called the χ^2-*distance*. These distance functions are used in Section 3 to derive the updates used in this paper (See discussion at the end of Section 3 and Figure 1).

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. It is easy to see that the Hessian of the log-likelihood is negative semi-definite. Thus there are no spurious local maxima and the ML solutions form a convex region. We use $\mathbf{u}$ to denote an arbitrary ML solution, and call $\mathbf{u}$ “the ML solution” for brevity. 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 $\mathbf{w}_t$,

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

3. The Updates

Kivinen and Warmuth [8] studied a general framework for on-line learning in which they derived algorithms for a broad class of loss functions. Here, we apply their method specifically to negative log-likelihood.

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 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:

$$\hat{F}(\mathbf{w}_{t+1}) = \eta \text{LogLike}(\mathbf{w}_{t+1}) - d(\mathbf{w}_{t+1}, \mathbf{w}_t), \quad \eta > 0 \ . \tag{2}$$

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 computationally hard 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 Equation (1).

To maximize this function F, we add a Lagrange multiplier for the constraint that the components of $\mathbf{w}_{t+1}$ sum to one, leading us to maximize

$$\begin{aligned}\tilde{F}(\mathbf{w}_{t+1}, \gamma) \ = \ & \eta\left(\text{LogLike}(\mathbf{w}_t) \ + \ \nabla\mathcal{L}(\mathbf{w}_t) \cdot (\mathbf{w}_{t+1} - \mathbf{w}_t)\right) \\ & -d(\mathbf{w}_{t+1}, \mathbf{w}_t) \ + \ \gamma\left(\sum_{i=1}^{N} w_{t+1,i} - 1\right) .\end{aligned}$$

This is done by setting the N partial derivatives to zero and enforcing the normalization constraint. So our framework consists of solving the following $N+1$ equations for the N coefficients of $\mathbf{w}_{t+1}$:

$$\frac{\partial \tilde{F}(\mathbf{w}_{t+1}, \gamma)}{\partial w_{t+1,i}} = \eta \nabla \mathcal{L}(\mathbf{w}_t)_i - \frac{\partial d(\mathbf{w}_{t+1}, \mathbf{w}_t)}{\partial w_{t+1,i}} + \gamma = 0 \tag{3}$$

and

$$\sum_{i=1}^{N} w_{t+1,i} = 1 \ . \tag{4}$$

We now derive all updates used in this paper by plugging different distance functions into the above framework. For the standard *gradient projection update* (which we abbreviate GP_η) we use the distance function $\mathrm{d}_{EUC}(\mathbf{w}_{t+1}||\mathbf{w}_t) = \frac{1}{2}||\mathbf{w}_{t+1} - \mathbf{w}_t||^2$. In this case the equations (3) become

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

By summing the above N equalities and using the identities $\sum_{i=1}^N w_{t,i} = \sum_{i=1}^N w_{t+1,i} = 1$ we see that $\gamma = \frac{\eta}{N}\sum_{i=1}^N \nabla \mathcal{L}(\mathbf{w}_t)_i$ and obtain the update

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

If we use the relative entropy, $\mathrm{d}_{RE}(\mathbf{w}_{t+1}||\mathbf{w}_t) = \sum_{i=1}^n w_{t+1,i} \ln(w_{t+1,i}/w_{t,i})$, as a distance function then the equations (3) become

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

By solving for the $w_{t+1,i}$ we have

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

Enforcing the normalization constraint (4) gives a new update which we call the *exponentiated gradient*[2] (EG_η) update:

$$w_{t+1,i} = \frac{w_{t,i} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_i}}{\sum_{j=1}^N w_{t,j} e^{\eta \nabla \mathcal{L}(\mathbf{w}_t)_j}} \ . \tag{6}$$

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 $\mathrm{d}_{\chi^2}(\mathbf{w}_{t+1}||\mathbf{w}_t) = \frac{1}{2}\sum_{i=1}^N (w_{t+1,i} - w_{t,i})^2 / w_{t,i}$. Now the equations (3) become

$$\eta \nabla \mathcal{L}(\mathbf{w}_t)_i - \left(\frac{w_{t+1,i}}{w_{t,i}} - 1 \right) + \gamma = 0 \ .$$

By solving for the $w_{t+1,i}$ we get

$$w_{t+1,i} = \eta w_{t,i} \nabla \mathcal{L}(\mathbf{w}_t)_i + w_{t,i}(\gamma + 1) \ .$$

We can now sum the above N equalities and use the constraints that $\sum_{i=1}^N w_{t,i} = 1$ and $\sum_{i=1}^N w_{t+1,i} = 1$. Our particular mixture estimation problem has the invariant[3] $\sum_{i=1}^N w_{t,i} \nabla \mathcal{L}(\mathbf{w}_t)_i = 1$. Thus $\gamma = -\eta$ and we obtain the update

$$w_{t+1,i} = w_{t,i} \left(\eta \left(\nabla \mathcal{L}(\mathbf{w}_t)_i - 1 \right) + 1 \right) \ . \tag{7}$$

We call Equation (7) the EM$_\eta$-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. The EM$_1$ update can be motivated by the likelihood equations, and the generalization to arbitrary η was studied by Peters and Walker [14], [15].

Since the χ^2 distance approximates the relative entropy it may not be surprising that the EM$_\eta$-update (7) also approximates the EG$_\eta$-update (6). 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$_\eta$-update can be viewed as a first order approximation of the EG$_\eta$-update. The approximation is accurate when the exponents $\eta(\nabla \mathcal{L}(\mathbf{w}_t)_j - 1)$ are small. The advantage of the EM$_\eta$-update is that it is computationally cheaper as it avoids the exponentiation. However the EG$_\eta$-update is easier to analyze. Our experiments indicate that these two update rules tend to approximate each other well.

Each of the different distance functions leads to a different bias that is encoded in the update. In Figure 1 we plot the three distance functions $\mathrm{d}_{EUC}(\mathbf{w}_{t+1} || \mathbf{w}_t)$, $\mathrm{d}_{RE}(\mathbf{w}_{t+1} || \mathbf{w}_t)$ and $\mathrm{d}_{\chi^2}(\mathbf{w}_{t+1} || \mathbf{w}_t)$ as a function of $\mathbf{w}_{t+1}$ for the three dimensional problem (with a triangle as the feasible region for $\mathbf{w}_{t+1}$). The contour lines for the distance function d_{EUC} are circles and the contour lines for d_{χ^2} are ellipses that become more degenerate as the old weight vector $\mathbf{w}_t$ approaches the boundary of the feasible region. The contour lines for d_{RE} are deformed ellipses that bend towards the vertices of the triangular feasible region.

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 [2]: $w_i = e^{r_i} / \sum_{j=1}^N e^{r_j}$ and maximize the function

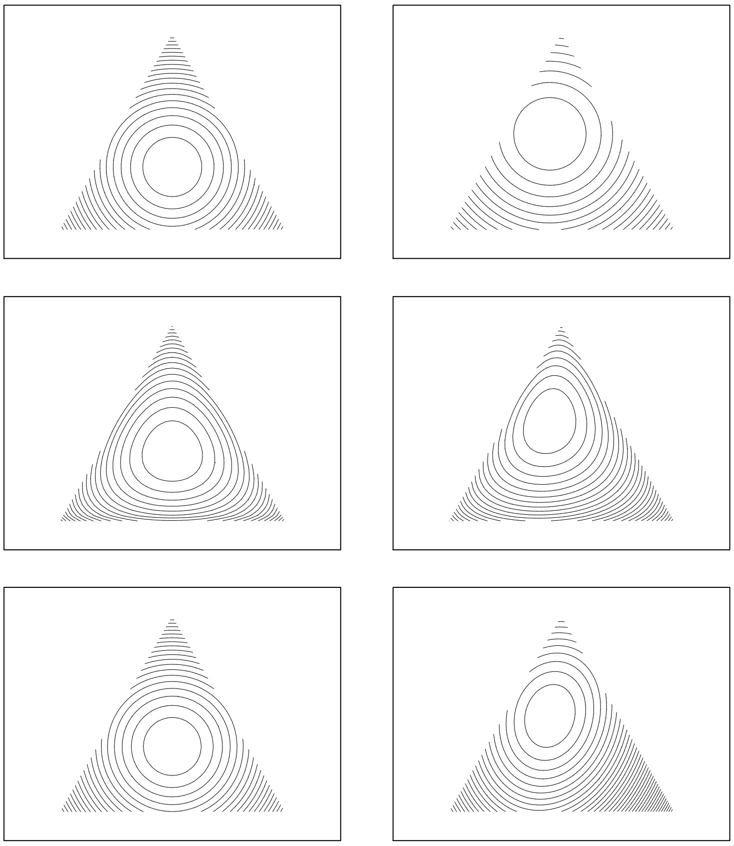

Figure 1. The figure contains plots of the three distance functions $\mathrm{d}_{EUC}(\mathbf{w}_{t+1}||\mathbf{w}_t)$ (first row), $\mathrm{d}_{RE}(\mathbf{w}_{t+1}||\mathbf{w}_t)$ (second row) and $\mathrm{d}_{\chi^2}(\mathbf{w}_{t+1}||\mathbf{w}_t)$ (third row) as a function of $\mathbf{w}_{t+1}$. The dimension is three and the non-negativity constraint on the three components of $\mathbf{w}_{t+1}$ plus the fact that the component must sum to one result in a triangle as the feasible region for $\mathbf{w}_{t+1}$. The corners of the triangle correspond to the vector $\mathbf{w}_{t+1} = (0,0,1)$ at the top vertex and vectors (1,0,0) and (0,1,0) at the left and right bottom vertices. The plots are contour plots of the distance function while looking at the triangle from above. The left column gives the distance from the uniform vector $\mathbf{w}_t = (1/3, 1/3, 1/3)$ which is at the center of the triangle and the right column the distance from the point (0.3, 0.2, 0.5). Note that contour lines may represent different distances in different diagrams.

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

(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 gradient descent update 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 can also be derived in our framework by approximately maximizing a function corresponding to $\hat{F}$ (Equation (2)):

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

For this maximization, we use $d(\mathbf{r}_{t+1}, \mathbf{r}_t) = \frac{1}{2}||\mathbf{r}_{t+1} - \mathbf{r}_t||^2$ 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, where 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) [4], [12]. Neal and Hinton [13] considered another extension of EM 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.

Note that in the above derivations of the updates we ignored the non-negativity constraints on the new weights $w_{t+1,i}$. For the EG_η update and for the gradient descent update with exponential parameterization the non-negativity constraints follow from the non-negativity of the previous weights $w_{t,i}$. However for EM_η and GP_η the learning rate η has to be sufficiently small to assure the non-negativity of the $w_{t+1,i}$. In particular, the standard EM algorithm (using $\eta = 1$) has the property that the non-negativity constraints are always preserved.

4. Convergence and Progress

In this section we discuss the convergence properties of the algorithms. Using standard methods (with the usual assumptions for convergence proofs) as in Luenberger [11], it can be shown that all updates described in the previous section converge locally to an optimal ML solution, provided that the current mixture vector $\mathbf{w}_t$ is close to the ML solution and given the usual assumptions. Moreover, using techniques similar to those in [15], [16], it can be shown that it is better to use a learning rate $\eta > 1$ rather than the rate $\eta = 1$. This implies that the EM algorithm is not optimal for this family of update rules. This analysis is supported by the experimental results presented in the next section, where choosing $\eta > 1$

leads to faster convergence, even when the current mixture vector is far from the ML solution.

These methods suffer from a number of limitations. For instance, the proof of convergence is only valid in a small neighborhood of the solution. In this section, we present a different technique for proving the convergence of the EG_η update and (under non-negativity assumptions) the GP_η updates.

If an update is derived with a distance function d then it is natural 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 $x_{p,i} \geq r > 0$. In the following section 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.

In contrast, we prove a similar bound for the GP_η update[4] in Section 4.2 showing that the average additional loss during T trials of the algorithm GP_η above the loss of the ML solution is at most

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

However, the analysis assumes that the GP_η algorithm does not produce mixture vectors with negative components. This assumption may not always hold 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 $1/r$, and even though we don't know of any matching lower bounds, they suggest 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-Bayesian 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 (below) 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*

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

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

$$-\sum_{t=1}^{T} \frac{1}{P} \sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p) \leq -\frac{T}{P} \sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + \frac{\sqrt{2T\ln N}}{2r} . \tag{9}$$

Proof: We have that

$$\begin{aligned}
\mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_{t+1}) - \mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_t) &= -\sum_i u_i \ln(w_{t+1,i}/w_{t,i}) \\
&= -\sum_i u_i \left(-\ln Z_t + \frac{\eta}{P} \sum_{p=1}^{P} \frac{x_{p,i}}{\mathbf{w}_t \cdot \mathbf{x}_p} \right) \\
&= -\frac{\eta}{P} \sum_{p=1}^{P} \frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p} + \ln Z_t .
\end{aligned} \tag{10}$$

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:

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

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

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

This fact can be proved by repeated application of Hölder's inequality.[5]

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

$$\begin{aligned} &\prod_{p=1}^{P}\left(\sum_{i=1}^{N}\left(w_{t,i} - \left(1 - \exp\left(\frac{\eta}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)\right) w_{t,i} x_{p,i}\right)\right)^{1/P} \qquad (11) \\ &= \prod_{p=1}^{P}\left(1 - \mathbf{w}_t \cdot \mathbf{x}_p \left(1 - \exp\left(\frac{\eta}{\mathbf{w}_t \cdot \mathbf{x}_p}\right)\right)\right)^{1/P} . \end{aligned}$$

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

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

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

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 $f(x) \geq 0$. 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 Equation (11), the upper bound on Z_t, and then applying Lemma 1 gives us

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

since r is a lower bound on $\mathbf{w}_t \cdot \mathbf{x}_p$. Plugging into Equation (10) we obtain

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

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

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

which implies the first bound (8) stated in the theorem. The second bound (9) follows by straightforward algebra, noting that $\mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_1) \leq \ln N$ when $\mathbf{w}_1$ is the uniform probability vector. ∎

Note that if any other upper bound K_{RE} on $\mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_1)$ is known a priori (possibly for some other choice of $\mathbf{w}_1$), then by tuning η as a function of K_{RE} the $\ln N$ term in the bound (9) of the theorem can be replaced by K_{RE}. This gives a bound of

$$\frac{\sqrt{2TK_{RE}}}{2r} \tag{12}$$

of the additional loss of the algorithm over the ML solution.

It follows from Theorem 1 that, if we run for T iterations, then the *average* loss (or average 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 $T = (\ln N)/(2\epsilon^2 r^2)$ guarantees that at least one of the $\mathbf{w}_t$'s will have a log-likelihood within ϵ of $\mathbf{u}$. Furthermore, it is easy to find the best candidate $\mathbf{w}_t$ that maximizes the likelihood among $\mathbf{w}_1, \ldots, \mathbf{w}_T$ by simply computing the likelihood of each.

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. Although each $\mathbf{w}_{t+1}$ is produced from the previous $\mathbf{w}_t$ as above, the algorithm outputs the modified 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 $\widetilde{\mathrm{EG}}_{\alpha,\eta}$.

THEOREM 2 *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*

$$-\sum_{t=1}^{T} \frac{1}{P}\sum_{p=1}^{P} \ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) \leq -\frac{T}{P}\sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + 2\alpha T + \frac{d_{RE}(\mathbf{u}||\mathbf{w}_1)}{\eta} + \frac{\eta T N^2}{8\alpha^2}. \quad (13)$$

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

$$-\sum_{t=1}^{T} \frac{1}{P}\sum_{p=1}^{P} \ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) \leq -\frac{T}{P}\sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) \ + \ 2(2N^2 \ln N)^{1/4}(T)^{3/4} \ . \quad (14)$$

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,

$$\begin{aligned} -\ln(\tilde{\mathbf{w}}_t \cdot \mathbf{x}_p) &\leq -\ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) - \ln(1-\alpha+\alpha/N) \\ &\leq -\ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) + 2\alpha \end{aligned} \tag{15}$$

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

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

$$-\sum_{t=1}^{T} \frac{1}{P} \sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \tilde{\mathbf{x}}_p) \leq -\frac{T}{P} \sum_{p=1}^{P} \ln(\mathbf{u} \cdot \tilde{\mathbf{x}}_p) + \frac{\mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_1)}{\eta} + \frac{\eta T N^2}{8\alpha^2} \tag{16}$$

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 inequalities (15) and (16), and summing over all t, this gives the first bound (13) of the theorem. The second bound follows from the fact that $\mathrm{d}_{RE}(\mathbf{u}||\mathbf{w}_1) \leq \ln N$ when $\mathbf{w}_1$ is the uniform probability vector. ■

From Theorem 2, it follows that the average additional loss of the $\mathbf{w}_t$'s for this algorithm over that of the ML solution is at most

$$O\left(\left(\frac{N^2 \ln N}{T}\right)^{1/4}\right) .$$

This is unfortunately a rather weak bound.

4.2. Convergence proofs for gradient-projection algorithms

In this section, we prove a convergence result for the gradient-projection algorithm. The setup is exactly as in Section 4.1.

The update rule used by GP_η is

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

where $\eta > 0$ is the learning rate, and $\mathbf{1}$ is the all 1's vector. We assume that $w_{t,i}$ remains non-negative.

THEOREM 3 *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* $\eta > 0$, *assume that* GP_η *produces a sequence of probability vectors* $\mathbf{w}_1, \ldots, \mathbf{w}_T$ *so that all components of each are nonnegative. Then*

$$-\sum_{t=1}^{T} \frac{1}{P} \sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p) \leq -\frac{T}{P} \sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + \frac{\eta N T}{2r^2} + \frac{d_{EUC}(\mathbf{u}||\mathbf{w}_1)}{\eta}. \tag{17}$$

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

$$\eta = \frac{r}{\sqrt{NT}}$$

then $d_{EUC}(\mathbf{u}||\mathbf{w}_1) \leq \frac{1}{2}$ *and*

$$-\sum_{t=1}^{T} \frac{1}{P} \sum_{p=1}^{P} \ln(\mathbf{w}_t \cdot \mathbf{x}_p) \leq -\frac{T}{P} \sum_{p=1}^{P} \ln(\mathbf{u} \cdot \mathbf{x}_p) + \frac{1}{r}\sqrt{NT}. \tag{18}$$

Proof: We use $\mathrm{d}_{EUC}(\mathbf{u}||\mathbf{w}_t) = \frac{1}{2}||\mathbf{u} - \mathbf{w}_t||^2$ as the potential function since it is the distance function used to derive the GP_η update. We can bound the change in potential at time t using straightforward algebra as follows.

$$\begin{aligned}
&\frac{1}{2}||\mathbf{u} - \mathbf{w}_{t+1}||^2 - \frac{1}{2}||\mathbf{u} - \mathbf{w}_t||^2 \\
&= \frac{\eta}{P} \sum_{p=1}^{P} \left(1 - \frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) + \frac{\eta^2}{2} \left\| \frac{1}{P} \sum_{p=1}^{P} \frac{1}{\mathbf{w}_t \cdot \mathbf{x}_p} \left(\mathbf{x}_p - \frac{\mathbf{1}}{N} \sum_{i=1}^{N} x_{p,i}\right) \right\|^2 \\
&\leq -\frac{\eta}{P} \sum_{p=1}^{P} \ln\left(\frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) + \frac{\eta^2}{2P} \sum_{p=1}^{P} \left\| \frac{1}{\mathbf{w}_t \cdot \mathbf{x}_p} \left(\mathbf{x}_p - \frac{\mathbf{1}}{N} \sum_{i=1}^{N} x_{p,i}\right) \right\|^2
\end{aligned}$$

In the second step we used the convexity of the function $||\cdot||^2$, and the fact that $1 - e^x \leq -x$ for all real x. Since $x_{p,i} \in [r, 1]$, and assuming that $w_{t,i} \geq 0$, it follows that this is bounded by

$$-\frac{\eta}{P} \sum_{p=1}^{P} \ln\left(\frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) + \frac{\eta^2 N}{2r^2}.$$

Thus, summing over all $t \leq T$, we get

$$\frac{1}{2}||\mathbf{u} - \mathbf{w}_{T+1}||^2 - \frac{1}{2}||\mathbf{u} - \mathbf{w}_1||^2 \leq -\frac{\eta}{P} \sum_{t=1}^{T} \sum_{p=1}^{P} \ln\left(\frac{\mathbf{u} \cdot \mathbf{x}_p}{\mathbf{w}_t \cdot \mathbf{x}_p}\right) + \frac{\eta^2 N T}{2r^2}.$$

So

$$\sum_{t=1}^{T}\sum_{p=1}^{P}\ln\left(\frac{\mathbf{u}\cdot\mathbf{x}_p}{\mathbf{w}_t\cdot\mathbf{x}_p}\right) \le \frac{P}{2}\left(\frac{\eta NT}{r^2}+\frac{\|\mathbf{u}-\mathbf{w}_1\|^2}{\eta}\right) = P\left(\frac{\eta NT}{2r^2}+\frac{\mathrm{d}_{EUC}(\mathbf{u}\|\mathbf{w}_1)}{\eta}\right)$$

which implies the bound in Equation (17). The derivation of the second bound in Equation (18) follows by straightforward algebra. ■

When tuning η to obtain bound (18), we used the fact that $\mathrm{d}_{EUC}(\mathbf{u}\|\mathbf{w}_1)$ is at most $\frac{1}{2}$. If a better upper bound, K_{EUC}, on this distance is available a priori, then we can tune η in (17) accordingly to obtain the bound of

$$\frac{\sqrt{2NTK_{EUC}}}{r} \tag{19}$$

on the additional loss of GP_η above that of the ML solution.

One way to compare the bound for EG_η (12) and the bound for GP_η (17) is to assume that both algorithms know the true distance to the ML solution, so that $K_{RE} = \mathrm{d}_{RE}(\mathbf{u}\|\mathbf{w}_1)$ and $K_{EUC} = \mathrm{d}_{EUC}(\mathbf{u}\|\mathbf{w}_1)$. In this case each algorithm can use the value of η minimizing its bound. If the algorithms are tuned in this way and the starting vector $\mathbf{w}_1$ is $(1/N,\ldots,1/N)$, then one can show that the bound for $\mathrm{EG}_\eta$is never higher that the bound for GP_η, i.e.:

$$\frac{\sqrt{2T\mathrm{d}_{RE}(\mathbf{u}\|\mathbf{w}_1)}}{2r} \le \frac{\sqrt{2NT\mathrm{d}_{EUC}(\mathbf{u}\|\mathbf{w}_1)}}{r}.$$

The above may be seen as theoretical support for our observation that that EG_η always converges faster than GP_η when the start vector is uniform and both algorithms use the (empirically found) best fixed learning rate .

Theorem 3 assumes a lower bound on the $x_{p,i}$. When no such lower bound r is available, then we can use similar techniques to those described in Section 4.1.

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 ($P \ge 100$) from at least 5 distributions ($N \ge 5$) 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

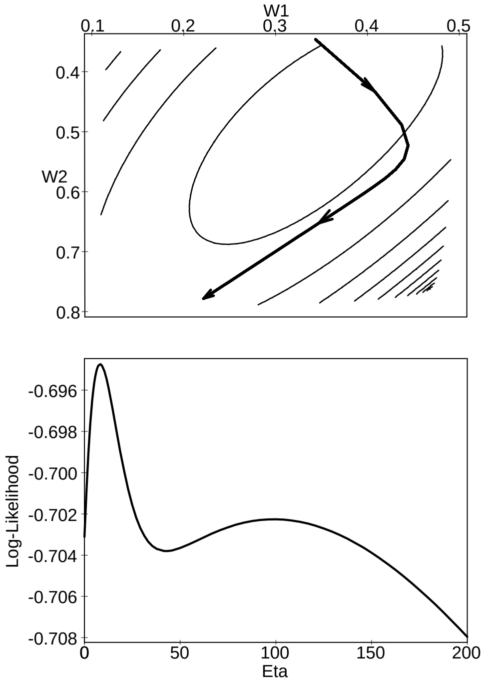

Figure 2. When the EG_η update is used, the log-likelihood as a function of η may have local maxima. At the bottom part of the figure, the log-likelihood is plotted as a function of η for a given $\mathbf{w}_t$. At the top, the corresponding path is plotted over the log-likelihood as a function of the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

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 2, so the searches must be careful to pick the global maximum.

The optimal learning rate determined by the line-searches tended to oscillate, as shown at the bottom part of Figure 3. When a momentum term was added, the oscillations were damped and the convergence was accelerated.[6]

Using fixed scheduling turned out to be a competitive alternative to the expensive line-searches. Furthermore, we found that the learning rates used for deriving the bounds in the previous section are too conservative. For the fixed scheduling experiments reported in this section we used a higher learning rate in the range [1..5]. All these phenomena are depicted at the top part of Figure 3.

The gradient ascent update with exponential parameterization appears inferior to all other methods. 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

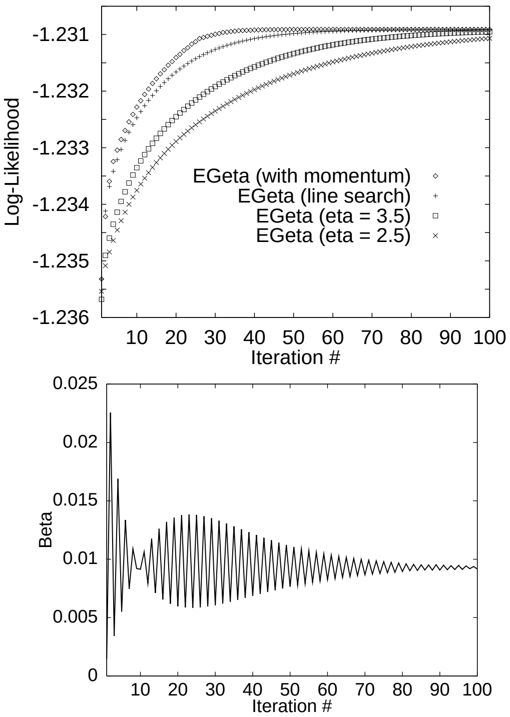

Figure 3. Top: The log-likelihood using the exponentiated gradient algorithm, with line-searches, line-searches plus a momentum term, and fixed scheduling with $\eta = 3.5$ and $\eta = 2.5$. The fixed schedulings are only slightly worse than setting the rate by expensive line-searches, while adding a 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.

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 4).

When the uniform start vector and (empirically found) best fixed learning rates for each algorithm are used, then EG_η (as well as EM_η) always converge faster than GP_η in the experiments we have done (see Figure 5). The bounds at the end Section 4 may be seen as theoretical support for this behavior. However when the start vector is not uniform and the ML solution $\mathbf{u}$ is close to the uniform vector then we have observed cases where GP_η converges faster than EG_η (and EM_η).

One of the main observation in the experiments is the following: EG_η and EM_η *clearly outperform* GP_η when the solution is sparse (see Figure 5). This is consistent with other settings [8], [9], [10], where updates derived using the relative entropy distance outperform gradient-descent-type updates when the solution is "sparse".

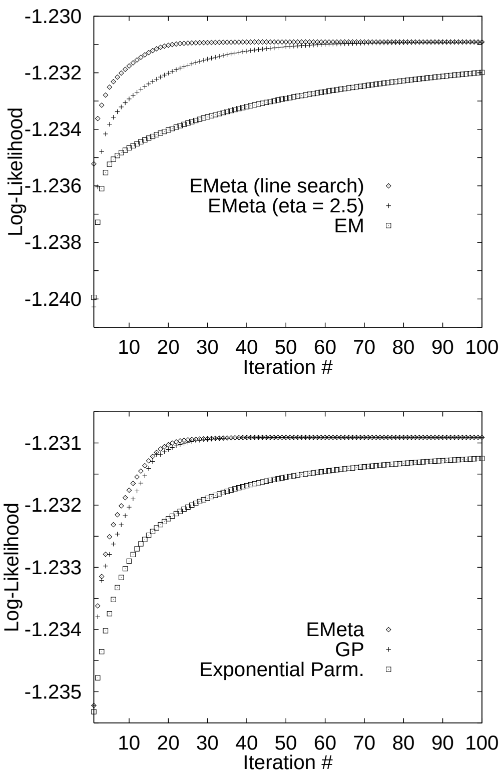

Figure 4. Top: Comparison of the performance of the EM_η-update algorithm and the standard EM algorithm. The EM_η-update clearly outperforms the standard EM algorithm, even when a fixed conservative scheduling is used. Bottom: comparison of the EM_η-update with gradient ascent algorithms. The gradient-projection is comparable to the EM_η-update and the gradient ascent update with exponential parameterization is inferior.

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

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

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

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

The right-hand side is minimized at,

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

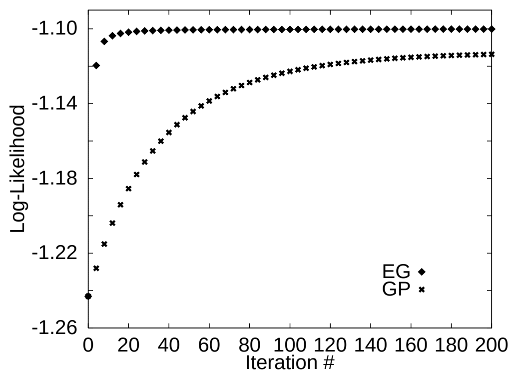

Figure 5. Comparisons of the performance of the EG_η and GP_η algorithms. Consider 10 Gaussian distributions with their centers equally spaced on the unit circle. The observations were generated from a perturbation of the uniform mixture on only 5 of the 10 Gaussians and the algorithms are started with the uniform start vector. This (and similar experiments) indicate that the EG_η algorithm performs better when the optimum mixture vector $\mathbf{u}$ has few large components.

This is the basic Newton method, which requires calculations of second order derivatives and inversion 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. We found that when N is sufficiently large ($N \geq 10$) the EM_η and EG_η algorithms converged more rapidly than the basic Newton's method when running time (rather than number of iterations) is considered. In Figure 6 we plotted N iterations of EM_η against one iteration of Newton's method. In this qualitative comparison again EM_η outperform Newton.

6. Applications and future research

We investigated various algorithms for learning the proportion vector which maximizes the likelihood of a mixture of given densities. This is a very simple mixture estimation problem since the parameters of the densities don't have to be learned as well. We presented some new algorithms called EG_η and EM_η. The EG_η algorithm uses the gradient of the log-likelihood in the exponent and the EM_η is a first-order approximation of the latter algorithm that replaces the exponentiation by a multiplication. When the learning rate η of the EM_η algorithm is set to one then we get the standard EM algorithm for our simple mixture estimation problem.

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 wonder what are the distance functions most appropriate for a particular situation. One important area for future research is identifying good distance functions when the

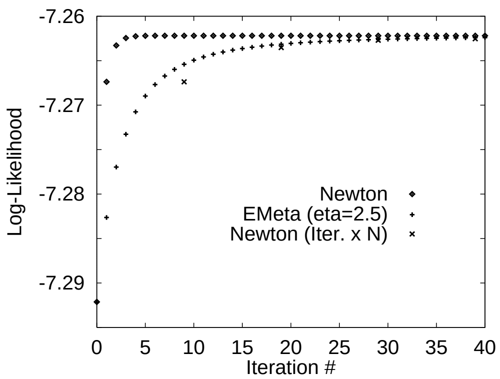

Figure 6. Comparison of EM_η-update with second order algorithms (Newton methods). Second order methods require fewer iterations than either the EM_η-update algorithm with fixed scheduling or EM_η with line searches. However, the basic second order method requires that the Hessian be calculated as well as its inverse. Since these calculations require $O(PN^2 + N^3)$ time and the EM_η algorithm with fixed scheduling requires only $O(PN)$ time per iteration we can compare a single Newton iteration with N iterations of EM_η. This qualitative comparison shows that the EM_η-update performs better even when fixed scheduling is used for the learning rate. The EG_η algorithm behaves essentially the same as EM_ηin the experiments, however it requires slightly more time because of the exponentiation.

parameters do not form a probability vector. We have already applied our methodology for deriving updates to more complicated mixture estimation problems such as training hidden Markov models [17] and we are currently applying 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 which is the EM_ηalgorithm with $\eta = 1$ We have also applied the on-line versions of our algorithms to a portfolio selection problem [7] investigated by Cover [3]. Although Cover's analytical bounds appear better than ours, 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.

Acknowledgments

We thank Jyrki Kivinen for helpful discussions and the anonymous referees for their useful comments. Yoram Singer acknowledges the Clore Foundation for its support. Part of this research was done while he was at the Hebrew University of Jerusalem, and visiting the Computer and Information Sciences department at the University of California, Santa Cruz. Manfred K. Warmuth received funding from NSF grant IRI-9123692.

Notes

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.
2. A similar update for the case of linear regression was first given by Kivinen and Warmuth [8].
3. $\sum_{i=1}^{N} w_{t,i} \nabla \mathcal{L}(\mathbf{w}_t)_i = \sum_{i=1}^{N} \frac{1}{P} \sum_{p=1}^{P} \frac{w_{t,i} x_{p,i}}{\mathbf{x}_p \cdot \mathbf{w}_t} = \frac{1}{P} \sum_{p=1}^{P} \frac{\mathbf{w}_t \cdot \mathbf{x}_p}{\mathbf{x}_p \cdot \mathbf{w}_t} = 1.$
4. This algorithm's performance was analyzed in the PAC model in [1].
5. In one form, Hölder's inequality states that, for non-negative a_i, b_i,

$$\sum_i a_i b_i \leq \left(\sum_i a_i^p \right)^{1/p} \left(\sum_i b_i^q \right)^{1/q}$$

for any positive p, q satisfying $1/p + 1/q = 1$.
6. The *conjugate gradient* search is a method for iteratively searching a quadratic cost function [11], [6]. 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 $K \rightarrow \infty$).

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. J. Bridle. Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. In F. Fogelman-Souli and J. Hérault, editors, *Neuro-Computing: Algorithms, Architectures, and Applications.* New York: Springer Verlag, 1989.
3. T. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991.
4. 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.
5. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.
6. G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns-Hopkins University Press, 1989.
7. D. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. On-line portfolio selection using multiplicative updates. In *Proc. 13th International Conference on Machine Learning*, pages 243–251. Morgan Kaufmann, San Francisco, 1996.
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. X.L. Meng and D.B. Rubin. Recent extensions of the EM algorithm (with discussion). In J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith, editors, *Bayesian Statistics, 4*. Oxfod: Clarendon Press, 1992.
13. R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Unpublished manuscript, 1993.
14. B. C. Peters and H. F. Walker. An iterative procedure for obtaining maximum-likelihood estimates of the parameters for a mixture of normal distributions. *SIAM Journal of Applied Mathematics*, 35:362–378, 1978.
15. 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.
16. R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *Siam Review*, 26:195–239, 1984.
17. Y. Singer and M. K. Warmuth. Training algorithms for hidden markov models using entropy based distance functions. To appear in Advances in Neural Information Processing Systems, 8, 1996.