Expectation Maximization for Weakly Labeled Data

Yuri Ivanov YIVANOV@MEDIA.MIT.EDU
MIT Media Laboratory, 20 Ames St., E15-390, Cambridge, MA 02139, USA

Bruce Blumberg BRUCE@MEDIA.MIT.EDU
MIT Media Laboratory, 20 Ames St., E15-311, Cambridge, MA 02139, USA

Alex Pentland SANDY@MEDIA.MIT.EDU
MIT Media Laboratory, 20 Ames St., E15-387, Cambridge, MA 02139, USA

Abstract

We call data *weakly labeled* if it has no exact label but rather a numerical indication of correctness of the label "guessed" by the learning algorithm - a situation commonly encountered in problems of reinforcement learning. The term emphasizes similarities of our approach to the known techniques of solving unsupervised and transductive problems. In this paper we present an *on-line* algorithm that casts the problem as a multi-arm bandit with hidden state and solves it iteratively within the Expectation-Maximization framework. The hidden state is represented by a parameterized probability distribution over states tied to the reward. The parameterization is formally justified, allowing for smooth blending between likelihood- and reward-based costs.

1. Introduction

1.1 Partially and Weakly Labeled Data

Suppose that we want to train an agent to respond to a set of voice commands. After hearing an utterance the agent performs an action. We would like to train the agent to respond correctly by providing (possibly noisy) rewards and punishments after seeing actions that it performs in response. In this scenario the agent needs to learn two things: a) equivalence classes in the utterance space; and b) what action to select upon observing a sample from one of these classes.

There exists a variety of algorithms permitting learning of policies of action selection given that the perceptual system provides a good set of features. But how can such features be efficiently estimated while the policy learning is taking place? In this work we attempt to solve the problem by embedding the expectation-maximization based algorithm for state estimation into a reinforcement learning paradigm.

Supervised, unsupervised and transductive learning methods view label information in a binary fashion - it is either present or absent. In contrast, in online reinforcement learning algorithms the label is first "guessed" by the learner. This guess is evaluated by an external source, which indicates its validity by a real-valued signal. We call data labeled in this fashion *weakly labeled*. This term is meant to properly place the problem among traditional machine learning tasks and to emphasize its connections with already existing techniques for learning with labeled, unlabeled and partially labeled data.

The EM algorithm is a powerful tool for solving unsupervised and transductive problems. We would like to extend its functionality to help us with problems where training labels are weak. EM is often used as a clustering algorithm with the objective of maximizing the likelihood of the data. This is a good heuristic to use for learning perceptual organization when no other evidence is available. However, by using an *unsupervised* technique for learning the perceptual organization, we disregard the reward and hence its utility for the agent.

The utility of a perceptual configuration is measured by the reward that the agent collects while using it. Therefore, we are seeking an algorithm which, while capable of learning from patterns in the input data alone, can be "directed" with the reward to choose a different configuration, which would provide higher payoff. That is, we would like an EM-type algorithm,

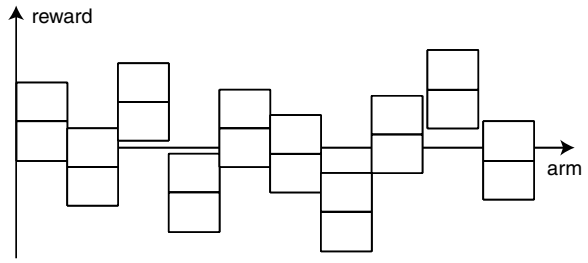

Figure 1: 10-armed bandit model. Each of the 10 arms produces a reward by drawing a sample from a corresponding distribution. Each box signifies the reward distribution with some mean (horizontal bar) and variance (height of the box).

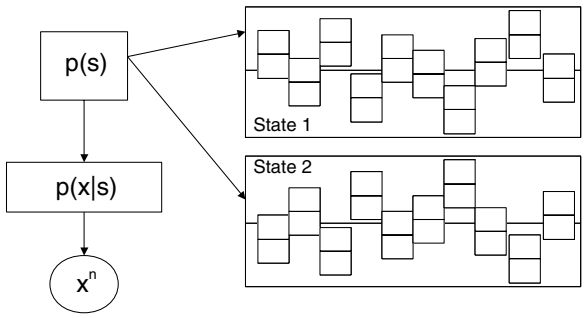

Figure 2: 2-state 10-armed bandit model. The bandit randomly switches between two states, according to a sample drawn from $p(s)$. After selecting the state, s, and observation, x^n is produced from a distribution $p(x|s)$.

which would allow for inclusion of reward into its objective for state estimation, while learning the policy of action selection. This is the main topic of the paper.

The paper proceeds as follows: after introducing the multi-state bandit problem and listing the relevant prior work, section 2 formally justifies modifications to the EM algorithm that allow for inclusion of the reward information into the parameter estimation. Section 3 uses these results to connect the extended EM with a multi-state bandit problem with hidden state. Experiments with these models, showing the results for different objectives, are presented in section 4. We conclude in section 5 with a brief summary of the algorithm, showing some of the problems with the algorithm and indicating future directions of our work.

1.2 Multi-State Bandit Problem

The problem of the agent training, as described earlier, falls into the category of *associative learning with hidden state*. If we model the input space with a mixture distribution, then the problem can be described as follows: given an observation, estimate the state of the world from a finite set of states, $S = \{s_i\}$. Given the belief about the state membership, select an action (label), which will result in the maximum amount of payoff received once the action is performed. With that payoff update parameters of the policy of the action selection, and of the input distribution.

This problem is often thought of as a multi-state N-armed bandit (Sutton & Barto, 1998). The N-armed bandit is a gambling device with a set of N arms (see fig. 1). Each arm has a probability distribution associated with it, according to which the sample reward is drawn every time the arm is pulled. Most frequently the reward generating process is assumed to be stationary or, at most, slowly varying.

Now imagine that the bandit can be in any one of M states, each of which have different distributions of the reward. Before each trial the bandit switches to a new state and produces an observation, x^n, from the distribution associated with this state (see fig. 2). The player's goal is to identify this state and perform action selection and model update for that state. When the state is perfectly known the problem reduces to M independent N-armed bandits. It is more difficult when the state is hidden and must be estimated. It is this problem that we address in this paper.

1.3 Related Work

Our work proposes a variant of the Expectation Maximization algorithm (Dempster & Rubin, 1977; Redner & Walker, 1984; Xu & Jordan, 1996) that allows us to situate it within the context of reinforcement learning. Neal and Hinton (1998) show a view of the EM algorithm that allows for modifications made in this paper. At a certain parameter setting and binary reward our algorithm can be viewed as an on-line version of the λEM, presented by Nigam et al. (2000), for learning with partially labeled data (albeit for a Gaussian Mixture) and transductive inference (Vapnik, 1998). The goal of the algorithm described here, however, is to develop a mechanism of perceptual learning for an agent learning from the environment in a Reinforcement Learning framework. To learn action selection policy we use an N-armed bandit model, e.g. (Sutton & Barto, 1998) estimating its parameters with the reinforcement pursuit algorithm of Thathachar and Sastry (1985), applying it to a set of states simultaneously.

A problem similar to ours was explored by Likas (1999), who used a variant of the REINFORCE algorithm (Williams, 1987) to learn Vector Quantization on the batch data, aided by a reward signal.

The technique of probability matching for reinforcement learning used here is similar to that shown by Sabes and Jordan (1996). Using this technique we can

construct a reward-dependent probability distribution to guide the convergence of the algorithm.

2. State Estimation With Reward

We use an EM algorithm for estimation of the state parameters. In this section we introduce a technique for including the reward function into the EM re-estimation procedure. The new objective function is simply implemented in the EM framework while allowing the algorithm to "fall back" to the unsupervised mode if no reward is provided.

2.1 Reward-driven variational bound

Typically, the EM algorithm for density estimation is used for unsupervised maximization of the likelihood function of a parametric density model when obtaining an analytical solution for the gradient in parameter space is difficult. This is the case when we need to learn parameters of a mixture density. In our algorithm we represent the input space by a mixture density, $p(x;\theta) = \sum_i p(s_i)p(x|s_i;\theta_i)$, parameters of which, θ, we would like to estimate. The goal of the algorithm, however, is to not simply maximize the likelihood of the data, but also take into account the external reward signal if such is present. To do so, in this section we justify a new cost function which allows for inclusion of the reward in the traditional EM framework.

EM as a variational bound optimization. The main idea of EM is based on simple geometric reasoning - if instead of maximizing some difficult function we maximize its convex lower bound that touches the function at a current parameter value, then a step in the direction of the gradient of this bound is also a step in the direction of the local maximum of the function. For a likelihood function, $f(\theta) = p(x,\theta)$, where x is the data set and θ is the vector of parameters, the EM algorithm is based on the following bound (GAM inequality):

$$
\begin{aligned}
f(\theta) &= \int p(x,s,\theta)\frac{q(s)}{q(s)}ds \\
&\geqslant \prod_s \left(\frac{p(x,s,\theta)}{q(s)}\right)^{q(s)} = g(x,\theta)
\end{aligned} \tag{1}
$$

Here, $g(x,\theta)$ is a lower bound of the likelihood, $f(\theta)$, and $q(s)$ is some positive function of s, integrating to 1. Typically, for the purposes of optimization of $f(\theta)$, the logarithm of $g(x,\theta)$ is optimized:

$$
G(x,\theta) = \int q(s)\log p(x,s,\theta) - q(s)\log q(s)ds \tag{2}
$$

It follows from eqn. (1) that for any density $q(s)$, $G(x,\theta)$ is a lower bound on $\log f(\theta)$. Now we need to find the density $q(s)$, which touches $\log f(\theta)$ at θ. The cost function in eqn. (2) can be re-written as follows, (Neal & Hinton, 1998):

$$
G(x,\theta) = -D\left(q(s)||p(s|x,\theta)\right) + \log f(\theta) \tag{3}
$$

where $D(p||q)$ is a Kullback-Leibler Divergence between distributions p and q. From here it is easily shown that $G(x,\theta) = \log f(\theta)$ when $q(s) = p(s|x,\theta)$, that is, the bound will be touching the likelihood function at the current θ.

Augmented reward bound. In order to let EM include the expected reward into the optimization we need to augment the EM bound shown above with a reward-dependent term. We do that by using the probability matching technique (Sabes & Jordan, 1996).

To learn preferred cluster configurations, we consider observation-state pairs and construct a reward-dependent probability distribution, $p^*(s|x;\bar{r})$. The task of the learning algorithm is to select from a set of conditional distributions $p(\mathcal{S}|\mathcal{X},\theta)$, aided by rewards that are provided by the environment for some of the data points. These rewards can be thought of as inverse energies - pairs (s,x) receiving higher rewards correspond to low energy states. Energies can be converted to probabilities via the Boltzmann distribution, which represents the ideal observation-state mapping - (s,x) pairs receiving higher rewards being more likely than pairs receiving low reward. If the parameters of $p(s|x,\theta)$ are adjusted so that it is close to $p^*(s|x;\bar{r})$, then the output of the algorithm will typically result in higher rewards.

We follow this line of reasoning while making $p^*(s|x;\bar{r})$ *proportional* to the Boltzmann distribution as shown later in the text. We need to consider this distribution and penalize the estimator for being different from this distribution in the posterior. We begin with adding an extra term to the equation (3):

$$
\begin{aligned}
F(x,\theta) = &-D\left(q(s)||p(s|x,\theta)\right) + \\
&E_{q(s)}\left[\log\frac{p^*(s|x;\bar{r})}{p(s|x,\theta)}\right] + \log f(\theta)
\end{aligned} \tag{4}
$$

When $q(s)$ is set to the posterior distribution, $p(s|x,\theta)$, the expectation term turns into negative divergence between the posterior and, $p^*(s|x;\bar{r})$:

$$
\begin{aligned}
&E_{q(s)}\left[\log\frac{p^*(s|x;\bar{r})}{p(s|x,\theta)}\right]\Bigg|_{q(s)=p(s|x,\theta)} = \\
&\quad - D(p(s|x,\theta)||p^*(s|x;\bar{r}))
\end{aligned} \tag{5}
$$

In fact this term induces a different but very intuitive bound for the likelihood maximization, which is shown in the proposition 2.1.

Proposition 2.1. *$F(x,\theta)$ is a lower bound on $\log f(\theta)$.*

Proof. Starting from (4), we can write:

(6)
$$
\begin{aligned}
F(x,\theta) =& - D\left(q(s)||p(s|x,\theta)\right) \\
&+ E_{q(s)}\left[\log \frac{p^*(s|x;\bar{r})}{p(s|x,\theta)}\right] + \log f(\theta) = \\
&\int q(s) \log \frac{p(s|x,\theta)}{q(s)} ds \\
&+ \int q(s) \log \frac{p^*(s|x;\bar{r})}{p(s|x,\theta)} ds + \log f(\theta) = \\
&\int q(s) \left[\log \frac{p(s|x,\theta)}{q(s)} + \log \frac{p^*(s|x;\bar{r})}{p(s|x,\theta)}\right] ds + \log f(\theta) = \\
&\int q(s) \log \frac{p^*(s|x;\bar{r})}{q(s)} ds + \log f(\theta) = \\
&-D\left(q(s)||p^*(s|x;\bar{r})\right) + \log f(\theta)
\end{aligned}
$$

In the last line of eqn. (6) the divergence, $D(q(s)||p^*(s|x;\bar{r})) \geqslant 0$, from which follows that

$$F(x,\theta) \leqslant \log f(\theta), \qquad \forall q(s), \theta \tag{7}$$

with equality holding iff $q(s) = p^*(s|x;\bar{r})$. □

This function has the same form as eqn. (3), which implies that for practical purposes we may simply substitute the EM-induced posterior with our fictitious probability distribution, $p^*(s|x;\bar{r})$. It provides the traditional bound for the likelihood function in the absence of the reward. With the reward present, the algorithm performs only a *partial E-step*. However, the step in the direction of the gradient of this bound leads uphill in the future expected reward.

Now we need to construct $p^*(s|x;\bar{r})$ in a convenient form. The main constraint that we want to impose is that the additional term in eqn. (4) vanishes when after producing a label s for an observation x, the reward r received from the environment is 0. That is,

$$E_{q(s)}\left[\log \frac{p^*_{r=0}(s|x;\bar{r})}{p(s|x,\theta)}\right] = 0 \tag{8}$$

which implies that $p^*_{r=0}(s|x;\bar{r}) = p(s|x,\theta)$. We can simply set $p^*_{r=0}(s|x;\bar{r})$ to be proportional to the Boltzman distribution:

$$p^*(s|x;\bar{r}) = \frac{p(s|x,\theta)\exp\left(\beta r(x,y) p(s|x,\theta)\right)}{Z_\beta(x,\theta)} \tag{9}$$

This form of $p^*(s|x;\bar{r})$ is used throughout the paper.

Summary. In this section we derived a new cost function for EM parameter estimation which includes the reward function. This allows to drive learning the perceptual categories to achieve a higher utility with respect to the future reward.

3. Action Selection

Using the results of the previous section to build a state estimator, we connect it with the multi-state bandit, introduced in section 1.2, to learn the policy of action selection as shown below.

3.1 Solutions with the Known State

When the state is exactly known, then the solution for the multi-state bandit is achieved by independently solving a set of single-state bandits. A variety of *action-value* methods, such as *sample average*, *reinforcement comparison* and *pursuit*, have been proposed to solve the single-state bandit problem. The general idea is to stochastically search the action space while updating the estimate of the reward function. A probability distribution over the action space (*action preference*) is built based on this estimate and action selection is done via sampling from this distribution.

The simplest pursuit method, adapted for the multi-state agent, maintains an estimate of the payoff structure of the bandit via action value function, $Q_t(a,s)$. This function is updated at each step based on the reward received from the bandit after pulling the arm a by, for example, an exponentially-weighted sample-average method:

$$Q_t(a,s) = Q_{t-1}(a,s) + \alpha(r - Q_{t-1}(a,s)) \tag{10}$$

Based on the value of $Q_t(a,s)$, the pursuit method updates its action preference model, $\hat{p}_t(a|s)$, such that the action with the highest value of $Q_t(a,s)$ increases the probability of being selected by a small fraction, γ. Actions that are currently found suboptimal decrease their probability correspondingly. Let $a^*_{t+1} = \arg\max_a(Q_t(a,s))$, then:

$$
\begin{aligned}
&\hat{p}_{t+1}(a^*|s) = \hat{p}_t(a^*|s) + \gamma(1 - \hat{p}_t(a^*|s)) \\
&\hat{p}_{t+1}(a|s) = \hat{p}_t(a|s) + \gamma(0 - \hat{p}_t(a|s)), \ \ \forall a \neq a^*
\end{aligned} \tag{11}
$$

The convergence of the pursuit method is dependent upon values of α and γ, which in our experiments we set to be $\alpha = 0.1$ and $\gamma = 0.01$. In addition, it is readily combined with ϵ-greedy techniques to allow for non-stationary environment.

3.2 Solutions with the Hidden state

In the presence of the hidden state the problem of estimating the optimal action becomes more difficult. The uncertainty about the state can be dealt with by distributing the reward proportionally to the current belief about the state membership of the observation x^n.

Most of the bandit search algorithms allow for formulating a probability distribution over actions, given a state, $p(a|s)$. This is an arbitrary distribution which only expresses the current estimate of "action preferences". In our setting we need to map not states, but observations to actions. We can similarly construct a probability distribution $p(a|x)$ via a set of hidden states, s:

$$\begin{aligned} p(a|x^n) &= \sum_s p(a, s|x^n) \\ &= \sum_s p(a|s, x)p(s|x^n) \\ &= \sum_s p(a|s)p(s|x^n) \end{aligned} \tag{12}$$

The action selection now takes into account the uncertainty about the state, encoded in the state posterior. For the purpose of bandit updates, the reward is distributed among M bandits in proportion to their contribution to $p(a|x^n)$: $r(s_i) = rp(s_i|x^n)$.

Thus, our algorithm folds the action selection policy estimation into the Expectation step of the EM algorithm while using the immediate reward signal to control the entropy of the posterior for the Maximization step. The algorithm is iterative and incremental, performing one iteration per data point, keeping only the sufficient statistics about the density function of the input space. The goal of the algorithm is to estimate the structure shown in the figure 2. It proceeds as follows:

1. Initialize
   Set parameters of the M-state Bandit model to starting values; guess initial parameters of the distribution $p(x)$ and iterate the following Expectation and Maximization steps; for each new data point:
2. E-step
   (a) calculate $p(s|x^n)$ using the Bayes rule and the current parameters of the observation model, $p(x)$;
   (b) Forward pass
      (i) compute $p(a|x^n)$ (eqn. 12);
      (ii) select an arm by sampling $p(a|x^n)$
   (c) Backward pass
      (i) collect reward and distribute it among the states in fractions of $p(s|x^n)$;
      (ii) calculate $p^*(s|x^n, r^n)$ (eqn. (9));
3. M-step
   Maximize the resulting cost, eqn. (6), with respect to parameters, θ.

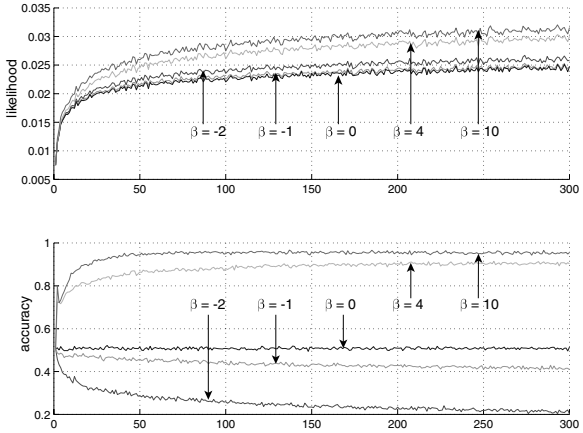

Figure 3: Performance of the REM averaged over 1000 runs for different values of the parameter β as compared with EM.

In the forward pass of the algorithm we break out of the EM's Expectation step to select an action and update the Bandit model. The yielded payoff serves as a control parameter for the EM.

4. Experiments

4.1 EM for state estimation

In the first experiment we confirm the conclusions of the previous section that we can in fact use the EM framework for partially supervised tasks. We need to see that given the context of the classification task the algorithm will result in choosing the clustering configuration, which provides higher expected reward.

In the experiments of this section we compare the performance of the algorithm with the traditional EM. However, it should be understood that this comparison is for a reference only, as the EM is not designed to perform the task, which we are targeting and can only provide the "worst case" performance.

As a source of the data we use a Gaussian mixture, $q(x) = \sum q(s)q(x|s)$. The algorithm estimates the density $p(x) = \sum p(s)p(x|s)$ by adjusting its parameters in an on-line fashion, upon seeing every data point, x^n. The reward is delivered after an attempt of classifying x^n to be generated by a particular component of $p(x|s_i)$. The experiment proceeds as follows:

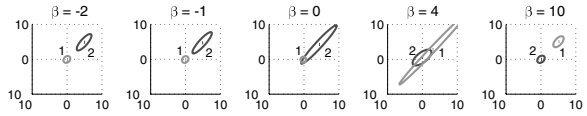

Figure 4: Results of a run of the algorithm for different values of β starting from the same initial conditions. For coefficients with opposite signs the labeling is reversed, while the uncontrolled EM produces the labeling by chance.

1. Initialize the generator mixture, $q(x)$
   for each state, s_i, randomly select a Gaussian observation model - $\mu_i \sim N(\mathbf{0}, 2I)$ and $\sigma_i = I$;
2. Iterate:
   (a) randomly choose a generator state, s_k;
   (b) generate an observation, x^n, distributed with μ_k and σ_k;
   (c) using current parameters of the model, $p(x)$, select a label l^n;
   (d) if $l^n = s_k$, deliver a reward of 1, otherwise, -1;
   (e) update parameters of the model
      (i) compute $p^*(s|x^n; \hat{r})$ via eqn. (9);
      (ii) perform the E-step of the EM algorithm using $p^*(s|x^n; \hat{r})$ in place of $p(s|x^n)$.

The results of the incremental reinforced binary classification experiments are shown in the figure 3. The top plot shows the attained likelihood of the data after a number of randomly generated samples. The horizontal axis shows number of iterations (data points seen so far) with the likelihood plotted along the vertical axis. It is curious to see that the unguided EM (with $\beta = 0$) attains the lowest likelihood. This is partially due to the fact that the EM is more likely to get stuck in the local maxima, while the reward signal delivers some extra energy for the algorithm to get out of it.

The second plot in the figure 3 complements the likelihood plot by showing the classification accuracy of the algorithm at different values of the parameter β. It should be expected that the accuracy of the EM used for classification would not be better than chance, since even when EM converges to the correct set of classes it does not care which source cluster corresponds to which estimated component. Positive values of the parameter β drive the extended EM towards correct labeling, while negative β drives the algorithm away from it, as can be seen in the accuracy plot.

The influence of β is further illustrated in the figure 4. The figure shows the resulting clustering attained with different values of β. It can be seen that the clusters for positive and negative values of β have opposite labeling while zero-valued β is labeled by chance. This is the desired behavior.

Figure 5: a) Performance on the 2-State 10-Armed binary bandit. b) Divergence between estimated and true source distributions.

4.2 Multi-State Bandit with Hidden State

Binary deterministic reward models the situation where the agent always gets a fixed value of reward upon selecting the optimal action. The "partially labeled data", or the problem of transductive inference, is a special case of this setting, as one can regard a binary reward as providing the label.

4.2.1 Maximization of the likelihood - Binary Bandit

This section shows the results on problems, in which the reward function is well aligned with the likelihood. That is, in which maximization of the reward results in the maximization of the likelihood. Results for this task are shown in figure 5. Unlike in the experiments of the previous section, the cluster identity is not important, as long as they correctly partition the input space. The multi-state Bandit essentially implements the mapping from clusters to labels.

We would also like to see if the reward-based estimator of the input density results in a better fit of the resulting observation density to the one that gets reinforced than the regular EM. In the case of a Gaussian mixture density with a known number of components (known number of states), we can measure the fit with the symmetrized KL Divergence:

$$S(p||q) = \frac{1}{4}\left[(\mu_q - \mu_p)^T(\Sigma_q^{-1} + \Sigma_p^{-1})(\mu_q - \mu_p) - tr\left(\Sigma_q^{-1}\Sigma_p + \Sigma_p^{-1}\Sigma_q - 2\mathbf{I}\right)\right] \quad (13)$$

For a lack of a better analytical method, we compute this quantity for every combination of source and estimated components and select the minimum value.

We perform the experiment with a 2-state 10-arm Bandit as follows:

1. Initialize the generator
   for each state, randomly select a Gaussian observation model - $\mu_i \sim N(\mathbf{0}, 2I)$ and $\sigma_i = I$;
2. Iterate:
   (a) randomly choose a generator state, s_k;
   (b) generate an observation, x^n, distributed with μ_k and σ_k;
   (c) using current parameters of the model select an action a^n;
   (d) if a^n is the same as the optimal arm in the generator state, s_k, deliver one unit of reward;
   (e) update parameters of the model;

The figure 5a) shows the average amount of reward collected by Bandits trained with the EM, REM, annealed REM algorithms compared to the case where the input space is estimated via a supervised estimator. As the goal is an accurate reproduction of the source mixture, these plots need to be considered along with the divergence plots (eqn. 13), given in figure 5b). The annealed REM algorithm, which slowly increases the value of the parameter β, performs very well, converging even faster than the supervised case. It is somewhat puzzling, but easily explained by the fact that the annealing amounts to simultaneous exploration of all states of the bandit in the initial stages. This gives a good set of initial conditions for subsequent search in each bandit when β increases.

4.2.2 Maximization of the likelihood - Full Bandit

The algorithm works with the full bandit with no modifications. The results are shown in the figure 6a). As in the case with the binary bandit, the initial convergence of both REM and Annealed REM is faster than the supervised case. The advantage, compared to EM, however, seems less spectacular than in the binary case. The divergence plots (figure 6b)), as before, show better fit of REM and Annealed REM to the source distribution.

This experiment shows the worst case scenario for the algorithm. The reward structure here has many local maxima and is "distracting" for the on-line search. The search becomes more difficult and the limitations of the search algorithm become the deciding factor in the achieved performance. However, despite the inconsistencies in the reward, the perceptual system captures the input distribution better when aided by the reward than when no feedback is given.

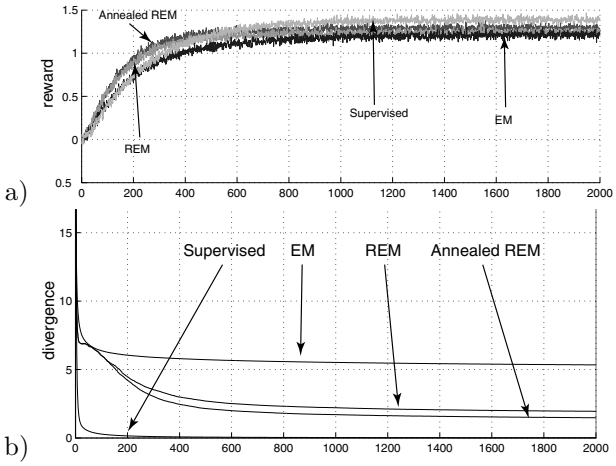

Figure 6: a) Performance on the full 2-State 10-Armed bandit. b) Divergence between estimated and true source distributions.

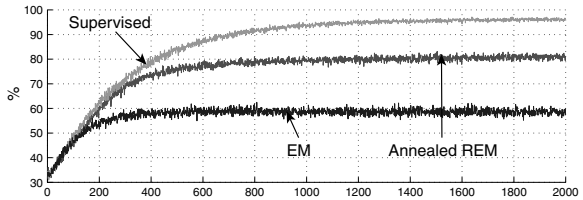

Figure 7: Performance of EM, REM and a fully supervised estimator on the problem where reward structure does not coincide with the likelihood (averaged over 2000 runs).

4.2.3 Maximization of the reward

It is interesting to see how this model performs on a problem in which the reward function is not aligned with the likelihood. The problem in this section is as follows - the input data is generated from 4 2-dimensional Gaussians. However the reward is delivered in such a way that action a_1 is rewarded when $x_1^n < 1.5$, a_2 - when $1.5 \leqslant x_1 < 4$ and a_3 when $x_1 > 4$.

The performance of the model on this task is shown in the figure 7. After 2000 iterations the EM estimator yields an average reward of 0.58, Annealed REM - 0.82 and supervised estimator - 0.96 with the maximum possible reward of 1.

Figure 8 shows results of a single run of the algorithm. The left column of the figure shows the resulting positions and outlines of the mixture components. The middle column shows the classification decision regions corresponding to the clustering shown on the left. The right column shows the "cluster assignment" - matrices that map states to actions, $p(a|s)$. A value in k-th position of l-th row of the matrix indicates the probability of selecting an action k once the point x^n is classified as

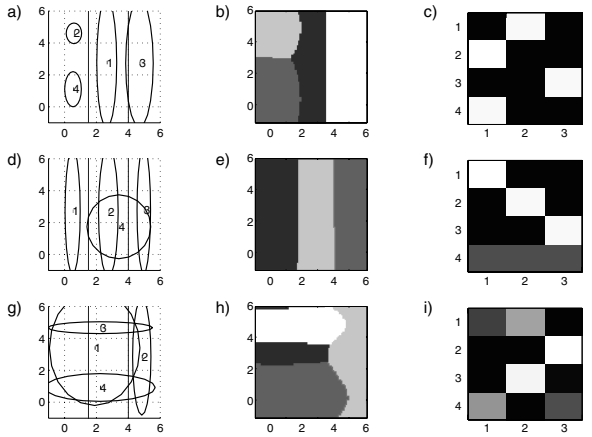

Figure 8: Final cluster positions (left column), decision regions (middle column) and cluster assignment matrices (right column) for REM (top row), supervised (middle row) and EM (bottom row) estimators after a single run.

belonging to the cluster l. Figures (a–c) demonstrate the performance of the annealed REM algorithm, (d–f) - that of the supervised model, and the bottom row (g–i) - the performance of the unguided EM. The supervised case gives the best possible partitioning of the input while using 3 Gaussians (component 4 is never used and therefore has a mixing coefficient 0). The REM uses all 4 components and aligns them with the reward partitioning. Note that both clusters 2 and 4 select action a_1.

5. Conclusions

We have presented an extension to the EM algorithm that allows for solving a range of learning tasks - from fully unsupervised, to fully supervised, including the partially and weakly labeled data. We provided the justification for entropic variations of the posterior to achieve arbitrary component assignment goals. The algorithm allows for smooth blending between likelihood- and reward-based costs.

Among the problem areas for the algorithm we can name the relatively large amount of data necessary to estimate the input density and learn the policy of action selection. Although the learning is performed online, it still takes a large number of iterations, particularly for a larger numbers of states. Another problem is the appropriate choice of the parameter β. In some cases it is convenient to have asymmetric schedule for positive and negative rewards, which adds another parameter to the set.

In some cases especial care must be taken about the fact that both reward signal for the clustering algorithm and the state assignment for the action selection are non-stationary. These problems are easily solved, but not discussed in the paper in the interest of space. The interested reader is encouraged to contact the authors if this information is necessary.

6. Acknowledgments

The authors would like to thank the reviewers of this paper for helping to make the presentation of this paper more clear. We would also like to extend our gratitude to Leslie Pack Kaelbling of MIT AI Lab and Aaron Bobick of Georgia Institute of Technology for their invaluable help in formulating ideas presented here. The idea of the reward maximization experiment is due to Leslie Pack Kaelbling.

References

Dempster, A.P. Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, *39*, 185–197.

Likas, A. (1999). Reinforcement learning approach to online clustering. *Neural Computation*, *11*, 1915–1932.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan (Ed.), *Learning in graphical models*, 355–368. Cambridge, MA: MIT Press.

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

Redner, R., & Walker, H. (1984). Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, *26(2)*, 195–239.

Sabes, P. N., & Jordan, M. I. (1996). Reinforcement learning by probability matching. *NIPS 8*.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Thathachar, M. A. L., & Sastry, P. S. (1985). A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, *15*, 168–175.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Williams, R. J. (1987). A class of gradient-estimating algorithms for reinforcement learning in neural networks. *ICNN 87*. San Diego, CA.

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