ARTICLE IN PRESS

ELSEVIER

Robotics and Autonomous Systems 972 (2002) 1–15

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Solving weak transduction with expectation maximization

Yuri A. Ivanov*, Bruce M. Blumberg

MIT Media Laboratory, 20 Ames St., Cambridge, MA 02139, USA

Abstract

In this paper we describe an algorithm designed for learning perceptual organization of an autonomous agent. The learning algorithm performs incremental clustering of a perceptual input under reward. The distribution of the input samples is modeled by a Gaussian mixture density, which serves as a state space for the policy learning algorithm. The agent learns to select actions in response to the presented stimuli simultaneously with estimating the parameters of the input mixture density. The feedback from the environment is given to the agent in the form of a scalar value, or a *reward*, which represents the utility of a particular clustering configuration for the action selection. The setting of the learning task makes it impossible to use supervised or partially supervised techniques to estimate the parameters of the input density. The paper introduces the notion of *weak transduction* and shows a solution to it using an expectation maximization-based framework. © 2002 Published by Elsevier Science B.V.

Keywords: EM algorithm; Weak transduction; Weakly labeled data

1. Introduction

Designing the perceptual system for autonomous agents is often a difficult task. The autonomy of the agent implies that the behavior of the agent is continuously modified as the agent collects new experiences and observes outcomes of the actions that it decides to take. A typical setting of the reinforcement learning task formulates a reward function, or a function according to which the world outside the agent's body rewards or punishes the agent for taking actions.

Since the reward function is rarely known, the agent has to experiment with the world to approximate this function from a just few samples and find a *policy*, which lets the agent select its actions such that the average reward intake is maximized.

To give a concrete example, imagine the situation where an agent is trained by a human trainer to respond to a set of voice commands. After hearing an utterance the agent performs an action. The trainer would like to have the agent respond correctly by providing (possibly noisy) rewards and punishments after observing the actions that it performs in response. In this scenario the agent needs to learn two things: (a) parameterized equivalence classes in the space of utterances; and (b) what action to select upon observing a sample from one of these classes. The first problem is that of *clustering under reward*, while the second is the typical *policy learning* task of reinforcement learning.

There exist a number 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? This paper focuses on the solution to the problem by embedding a simple associative search algorithm into an expectation maximization (EM) paradigm.

The task of *on-line* estimation of the perceptual organization and the policy of action selection is cast

* Corresponding author.
E-mail address: yivanov@media.mit.edu (Y.A. Ivanov).

0921-8890/02/$ – see front matter © 2002 Published by Elsevier Science B.V.
PII: S0921-8890(02)00199-9

here as a problem of a *multi-armed bandit with hidden state* and is solved iteratively within the EM 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.

In addition, in this paper we introduces the notion of *weak transduction* in order to properly place the solution among the existing techniques used for unsupervised and transductive problems.

The paper proceeds as follows: after introducing the multi-state bandit problem, Section 3 describes a modification of the reinforcement pursuit algorithm that allows us to include the estimation of the hidden state. Section 4 justifies modifications to the EM algorithm, which permits the inclusion of the reward information into the parameter estimation and to solve the problem of learning perceptual organization along with policy learning. The two parts of the estimation procedure are brought together in Section 5, which presents the complete reinforcement-driven EM algorithm. Experiments with this algorithm showing the results for different objectives are presented in Section 6. The paper concludes with Section 7, which points out contributions and some of the problems with the algorithm.

1.1. Related work

The work of this paper is based on the EM algorithm [1,6,12] extended to situate it within the context of reinforcement learning and to take advantage of the additional information that is available as a result of interaction with the environment. Neal and Hinton [4] show a view of the EM algorithm that makes the extensions made in this paper possible. This view is concisely presented by Minka in [3].

At a certain parameter setting and binary reward, the algorithm shown here can be viewed as an on-line version of the λEM, presented by Nigam et al. [5], for learning with partially labeled data (albeit for a Gaussian mixture) and transductive inference [10]. However, the problem solved by the algorithm described here is in general more difficult than the problem of transductive clustering, which Nigam's algorithm solves as it does not have access to exact labels for the input data.

To learn the policy of the action selection the learning algorithm developed here uses an N-armed bandit model (see, e.g., [8]). The multi-state policy is estimated with the aid of the reinforcement pursuit algorithm of Thathachar and Sastry [9], which is applied to a set of states simultaneously.

A problem similar to the one presented here was explored by Likas [2], who used a variant of the REINFORCE algorithm [11] 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 [7]. Using this technique, the algorithm presented here constructs a reward-dependent probability distribution to guide the algorithm towards the configuration resulting in higher value of the expected reward.

2. Agent training

Let us for a moment return to the example of the agent training and introduce some notation. The training consists of a series of steps, where superscript indicates the number of the step. At each step the trainer randomly selects a state, s, out of a finite set o states (see Fig. 2a). From that state the trainer produces an observation for the agent, x^n, according to the probability distribution $p^*(x|s)$. The chosen state switches the trainer to one of its "reward modes", where any action, a^n, that the agent might perform in response to x^n will be rewarded, punished or ignored, by dispensing to it a reward, r, from a probability distribution conditioned on both trainer's state and the agent's action, $p^*(r|s, a)$.

To clarify this, let us use an example. "A state" of the trainer corresponds to the trainer's desire to evoke a particular response from the agent. For example, the trainer might choose to have the agent go away. The external manifestation of that state ($s =$ want-it-to-go-away) might be a verbal command "Go away!"(x^n =" Go away!"). At this point the agent might erroneously choose to come up closer (a^n = approach-trainer). The trainer lets the agent to get away with it this time by not punishing it (drawing a value of 0 from the probability distribution $p^*(r|s$ = want-it-to-go-away, a = approached-trainer).

Environment model
(1) Nature can be in one of a discrete set of states $S = \{s_i\}_{i=1}^{K}$;
(2) Nature selects a state, s_i, from a probability distribution $p^*(s)$;
(3) From a conditional distribution $p^*(x|s = s_i)$ nature generates an observation x^n;
(4) Upon observing the agent taking an action a^n, nature produces a reward from the distribution $p^*(r|s = s_i, a = a^n)$.

The goal of the agent is to learn how to respond to the commands given to it by the trainer to maximize the intake of the reward. To solve this problem we propose an agent architecture that consists of two parts—a *perceptual model* and a *policy*. The perceptual model is a set of clusters in the input space, while the policy is a probability distribution, which for a given state gives a probability with which every agent's action would produce a reward.

The agent's task is then to estimate parameters of the model shown in Fig. 2b. In that figure $p(x|s)$ is the probability model of a category s, which is subject to parameter estimation; $p(s|x^n)$ is a *belief state*, calculated for the incoming observation from current model parameters; $p(a|s)$ is a *policy*, that is related to the past correlations between state, action and reward; and $p(a|x^n)$ is the probability distribution from which the action is selected upon observing x^n.

It is important to note that the perceptual organization of the agent cannot be formed by providing it with supervised examples of correct perceptual class assignment. The trainer can only judge the quality of the agent's perceptual organization based in the actions the agent selects in response to the external stimuli.

3. Estimation of the associative policy

The policy in the problem addressed here has a special and simple character—since all observations are assumed to be equally likely and independent—the trainer can produce any observation (e.g., utterance or gesture) at will—there is no need to keep the history as a part of the action context. In other words, the agent's behavior does not change the state of the world, which only changes at random. The task of the policy estimation in such a simplified setting is termed *associative search* and requires no history of prior observations to be stored. However, it is complicated by the necessity of perceptual learning, or state estimation, which places it in the category of *associative search with hidden state*.

If the state space is modeled 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 expected payoff received once the action is performed. With that payoff, update the parameters of the policy of the action selection and of the input distribution. This section will deal with solving the problem of policy estimation for such a setup.

3.1. Multi-state bandit problem

Due to the assumption that the observations are independent of each other, this problem can be formulated as a multi-state N-armed bandit [8]. 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

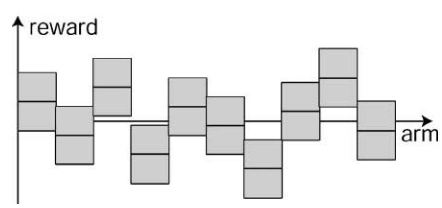

Fig. 1. The 10-armed bandit model. Each of the 10 arms produces a reward by drawing a sample from a corresponding distribution. Each box signifies the reward distribution with some mean (horizontal bar) and variance (height of the box).

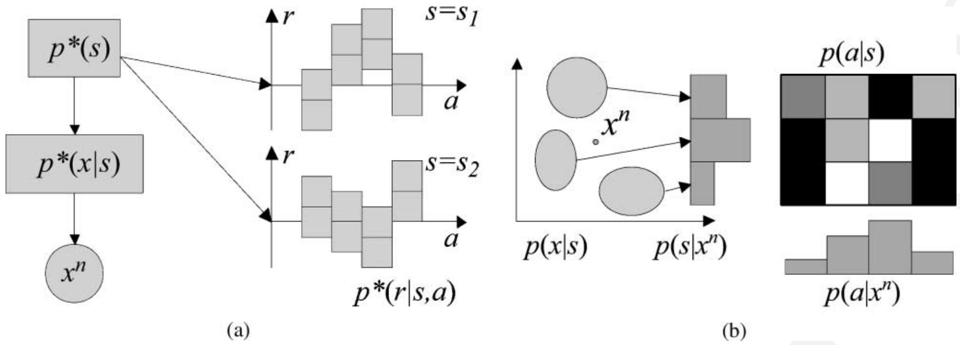

Fig. 2. (a) A generative model of the environment, shown as a 2-state 4-armed bandit. The bandit randomly switches between two states, according to a sample drawn from $p^*(s)$. After selecting the state, s, an observation, x^n is produced from a distribution $p^*(x|s)$. (b) The estimator consists of two parts—a perceptual model and a policy. Upon receiving the observation, x^n, the distribution $p(a|x^n)$ is constructed and an action is selected by drawing a sample from it. Upon delivery of the reward parameters of both the policy and the perceptual model are updated.

the distribution associated with this state (see Fig. 2a). 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.

3.2. Solutions with 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 *reinforcement pursuit*, have been proposed to solve the single-state bandit problem.[1] 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, e.g., an exponentially weighted sample-average method

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

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 to be suboptimal decrease their probability correspondingly. Let $a^*_{t+1} = \arg\max_a(Q_t(a, s))$, then

$$\begin{aligned} \hat{p}_{t+1}(a^*|s) &= \hat{p}_t(a^*|s) + \gamma(1 - \hat{p}_t(a^*|s)), \\ \hat{p}_{t+1}(a|s) &= \hat{p}_t(a|s) + \gamma(0 - \hat{p}_t(a|s)), \quad \forall a \neq a^*. \end{aligned} \qquad (2)$$

The convergence of the pursuit method is dependent upon values of α and γ, which in all experiments of this paper are 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 environments.

3.3. Solutions with 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

[1] A classical method for solving bandit problems, which includes balancing of exploration with exploitation involves computation of the so called Gittins indices. This method provides an optimal solution to a large class of problems, but assumes the knowledge of prior distribution of possible problems.

belief about the state membership of the observation x^n.

Most of the bandit search algorithms allow for formulating a policy, or 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". The action is selected by sampling the conditional probability distribution $p(a|x^n)$, which can be calculated from the belief state and the policy, by marginalizing the joint, $p(a, s|x^n)$

$$p(a|x^n) = \sum_s p(a, s|x^n) = \sum_s p(a|s, x^n)p(s|x^n) = \sum_s p(a|s)p(s|x^n). \quad (3)$$

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 $p(a|x^n)$

$$Q_t(a, s) = Q_{t-1}(a, s) + \alpha(rp(s|x^n) - Q_{t-1}(a, s)). \quad (4)$$

The action preference update equations, Eq. (2) are left unchanged.

4. Clustering under reward

Given the agent's architecture in Fig. 2b it is clear that the agent is not provided with direct supervision for the task of state estimation. This is because the feedback from the trainer rewards its action selection and not the sample classification in its perceptual system. On the other hand, for the purposes of perceptual clustering this situation is better than having no feedback at all, since it does provide some degree of guidance, however small.

State estimation under reward can be performed with the aid of the EM algorithm, which is often used for unsupervised clustering. This section introduces 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.

4.1. Weak transduction

The EM algorithm is a powerful tool for solving unsupervised and transductive problems. It 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, one disregards its utility for the agent.

The utility of a perceptual configuration is measured by the reward that the agent collects while using it. Therefore, an algorithm is sought, which while capable of learning from patterns in the input data alone, can be "directed" with the reward to choose a different configuration providing higher payoff. That is, the solution should be an EM-type algorithm, which would allow the inclusion of reward into its objective for state estimation, while learning the policy of action selection.

The EM algorithm is frequently used for *unsupervised* clustering of data by spatial proximity in the space of features. For a given number of clusters the algorithm proceeds iteratively first to calculate from the current cluster statistics the probability of data to be generated by each cluster, a state posterior, $p(s|x)$; and then to average the data, weighted by this posterior to update cluster statistics.

When the data comes with the known state attribution, $s^n = z$, then the posterior of each data point turns into a deterministic function, having 1 at the slot of the corresponding state and 0 elsewhere:

$$p(s^n = z|x^n) = 1, \qquad p(s^n \neq z|x^n) = 0. \quad (5)$$

Averaging with respect to this posterior, let us call it $p_{01}(s|x)$, results in the parameter estimation to decompose into several independent *supervised* estimation problems.

When only part of the data has an exact label, $s^n = z$, then the solution to the clustering problem results in a mixture of the supervised solution for the labeled data and the unsupervised solution for the unlabeled set. This is an example of *transduction*, where the knowledge from the labeled data is transduced onto the unlabeled. The setting of a transductive classification is illustrated in Fig. 3a. In the figure the supervised solution disregards the additional indirect information provided by the density of the unlabeled data, while

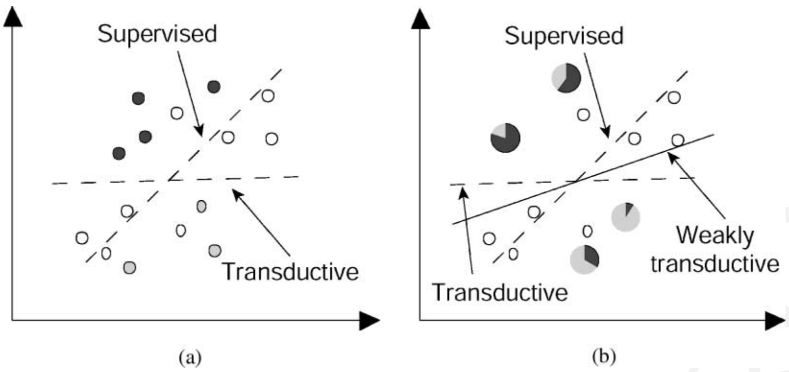

Fig. 3. (a) Illustration of transductive inference for classification. Empty circles indicate the data with unknown class membership while the colored circles indicate the labeled data, which belongs to one of two classes. Classification boundaries are different for estimators with and without unlabeled data. (b) Weak transduction has no direct indication of the class label, but a probability distribution over labels. Clear circles, again, show the unlabeled data, while large colored ones show the data that is weakly labeled. The area of the circle shaded with a particular color is proportional to the probability with which this data point belongs to this class.

transductive inference takes into account the complete data set.

Supervised, unsupervised and transductive learning methods view the label information in a binary fashion—it is either present or absent. In contrast, under the circumstances of the problem where the knowledge about the label is inexact and subjective, the situation is a bit worse than in the transductive setting, but better than unsupervised. With the current setting of the model parameters the posterior $p(s|x^n)$ is computed as the state membership of the query point. If, consequently, some value of reward results from this assignment, it indicates the quality of the posterior given the current parameter settings. This is the situation, which the algorithm being described encounters in the task of estimating a state. That is, in line with the above taxonomy, the data is labeled with a *probability distribution*, as illustrated in Fig. 3b. It is convenient to call data labeled in this fashion *weakly labeled*, and the problem—a *weak transduction*. These terms properly place the problem among traditional machine learning tasks and emphasizes its relation to already existing techniques for learning with labeled, unlabeled and partially labeled data [5].

4.2. 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 the parameter space is difficult. This is the case when we need to learn parameters of a mixture density. In the algorithm of this paper the input space is represented by a mixture density, $p(x;\theta) = \sum_i p(s_i)p(x|s_i;\theta_i)$, parameters of which, θ, need to be estimated. 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 a new cost function is justified, which allows for inclusion of the reward in the traditional EM framework.

4.3. EM as a variational bound optimization

$$f(\theta) = \int p(x,s,\theta)\frac{q(s)}{q(s)}\mathrm{d}s \geq \prod_s \left(\frac{p(x,s,\theta)}{q(s)}\right)^{q(s)} = g(x,\theta). \quad (6)$$

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

$$G(x,\theta) = \int q(s)\log p(x,s,\theta) - q(s)\log q(s)\mathrm{d}s. \quad (7)$$

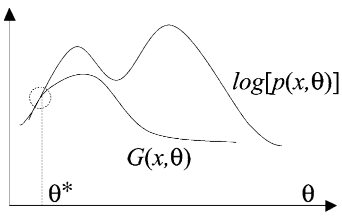

Fig. 4. The step in the direction of the gradient of the lower bound of Eq. (8) is the step in the direction of the gradient of likelihood.

It follows from Eq. (6) that for any density $q(s)$, $G(x, \theta)$ is a lower bound on $\log f(\theta)$. Now the density $q(s)$ needs to be found, which touches $\log f(\theta)$ at θ. The cost function in Eq. (7) can be re-written as follows [4]:

$$G(x, \theta) = -D(q(s) \| p(s|x, \theta)) + \log f(\theta), \tag{8}$$

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 $q(s) = p(s|x, \theta)$, i.e., the bound will be touching the likelihood function at the current θ, as shown in Fig. 4.

4.4. Augmented reward bound

In order to let EM include the expected reward into the optimization, the EM bound shown above needs to be augmented with a reward-dependent term. It is easy to do using the probability matching technique [7].

To learn preferred cluster configurations, one can consider observation-state pairs and construct a reward-dependent probability distribution, $p^*(s|x; r)$. 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 lower 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 $p^*(s|x; r)$, then the output of the algorithm will typically result in higher rewards.

Following this line of reasoning $p^*(s|x; r)$ is made *proportional* to the Boltzmann distribution as shown later in the text. Starting with Eq. (8), an additional term penalizes the estimator for being different from this distribution in the posterior:

$$F(x, \theta) = -D(q(s) \| p(s|x, \theta)) + E_{q(s)}\left[\log \frac{p^*(s|x; r)}{p(s|x, \theta)}\right] + \log f(\theta). \tag{9}$$

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

$$E_{q(s)}\left[\log \frac{p^*(s|x; r)}{p(s|x, \theta)}\right]\Bigg|_{q(s)=p(s|x,\theta)} = -D(p(s|x, \theta) \| p^*(s|x; r)). \tag{10}$$

In fact this term induces a different but very intuitive bound for the likelihood maximization (see Appendix A and Theorem 1 for proof)

$$F(x, \theta) = -D(q(s) \| p^*(s|x; r)) + \log f(\theta). \tag{11}$$

This function has the same form as Eq. (8), which implies that for practical purposes one may simply substitute the EM-induced posterior with the fictitious probability distribution, $p^*(s|x; r)$. 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 $p^*(s|x; r)$ needs to be constructed in a convenient form. The main constraint that should be imposed is that the additional term in Eq. (9) vanishes when after producing a label s for an observation x, the reward r received from the environment is 0. That is,

$$E_{q(s)}\left[\log \frac{p^*_{r=0}(s|x; r)}{p(s|x, \theta)}\right] = 0 \tag{12}$$

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

$$p^*(s|x; r) = \frac{p(s|x, \theta) \exp(\beta r p(s|x, \theta))}{Z_\beta(x, \theta)}. \tag{13}$$

This form of $p^*(s|x; r)$ is used throughout the rest of this paper.

The resulting bound is illustrated in Fig. 5. The augmented bound behaves just like the traditional EM

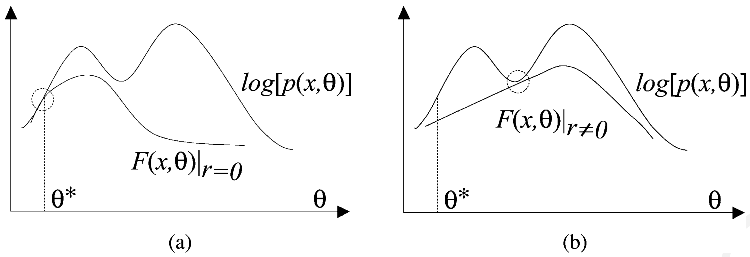

Fig. 5. (a) The augmented bound behaves just like the traditional EM bound when no reward is present. (b) With the reward present the bound is no longer in contact with the likelihood at the current parameter setting, leading uphill in the expected reward.

bound when no reward is present. With the reward present, the bound is no longer in contact with the likelihood at the current parameter setting, leading uphill in the expected reward. The point of contact with the bound is the value of parameter at which the posterior $p(s|x^n)$ equals $p^*(s|x; r)$.

5. Reward-driven EM

Now the two parts of the estimation procedure can be joined to get the complete solution to perceptual learning under reward. The algorithm is shown below and is illustrated in Fig. 6.

The 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 Fig. 2. It proceeds as follows:

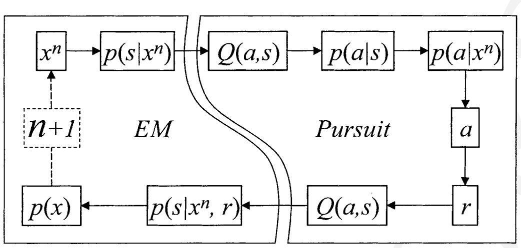

Fig. 6. The reward-driven perceptual learning algorithm breaks out of the expectation step of EM to compute the improved posterior. Then the parameter estimation is performed with respect to $p(s|x^n, r)$.

REM algorithm

---

(1) *Initialize*: set parameters to starting values;
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 $p(a|x^n)$ (Eq. (3));
(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 $p^*(s|x^n, r^n)$ (Eq. (13));
(3) *M-step*: maximize the resulting bound, Eq. (A.1)

In the forward pass of the algorithm the processing breaks out of the EMs expectation step to select an action and update the bandit model as shown in Fig. 6. The yielded payoff serves as a control parameter for the EM.

6. Experiments

The experimental analysis of the algorithm presented in this paper is performed on a series of tasks of increased difficulty. The first experiment does not include policy learning and is designed to simply test estimation of the perceptual model alone for a fixed optimal policy. Next, two experiments are performed, which involve the policy estimation. In the first experiment the reward is delivered by a bandit with only one arm per state producing a unit of reward. In the second experiment the binary restriction on the bandit is removed allowing each arm to produce some value of reward, positive or negative. Finally, an experiment is performed with a variation on the reward structure such that the reward reinforces arbitrary objective, not related to the likelihood of the data.

6.1. EM for state estimation

The first experiment confirms the conclusions of the previous section, showing that it is in fact possible to use the EM framework for partially supervised tasks. It has to be shown that, given the context of the classification task, the algorithm will result in choosing the clustering configuration that provides a higher expected reward.

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

As a source of the data a Gaussian mixture, $q(x) = \sum q(s)q(x|s)$ is used. 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 to classify x^n to be generated by a particular component of $p(x|s_i)$. The experiment proceeds as follows:

---

(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 Eq. (13);
    (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 Fig. 7. The top plot shows the attained likelihood of the data after a number of randomly generated samples. The horizontal axis shows the 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 intuition behind choosing the parameter β is that as it increases, the entropy of the probability distribution from which a label is selected drops. Characteristic behavior of the algorithm can be observed at extreme values of β with $\beta = -\infty$, the distribution over labels is uniform and the label selection is performed purely by chance, with no regard to neither the reward nor mixture parameters. At $\beta = 0$ the distribution over labels exactly equals to the mixture posterior, that is the algorithm disregards the reward completely, performing the unsupervised parameter estimation as mixture parameters dictate. Setting β to $+\infty$ results in a "winner-take-all" label assignment.

The second plot in Fig. 7 complements the likelihood plot by showing the classification accuracy of the algorithm at different values of the parameter β. It is expected that the accuracy of the EM used for classification should 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. It is interesting that none of the settings of the parameter β result in the optimal accuracy of 1. There are two reasons for this. First, any fixed value of β less than ∞ will cause a sub-optimal label to be selected, albeit with small probability. The second reason is related to the fact that even optimal Bayes classifier will not achieve the perfect classification rate as randomly placed source Gaussian components may significantly overlap.

The influence of β is further illustrated in Fig. 8. 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. In this run the source distribution has the component 1 (green) at the position (5, 5) and component 2 (red) at (0, 0), which is correctly identified by the algorithm with large positive value of β.

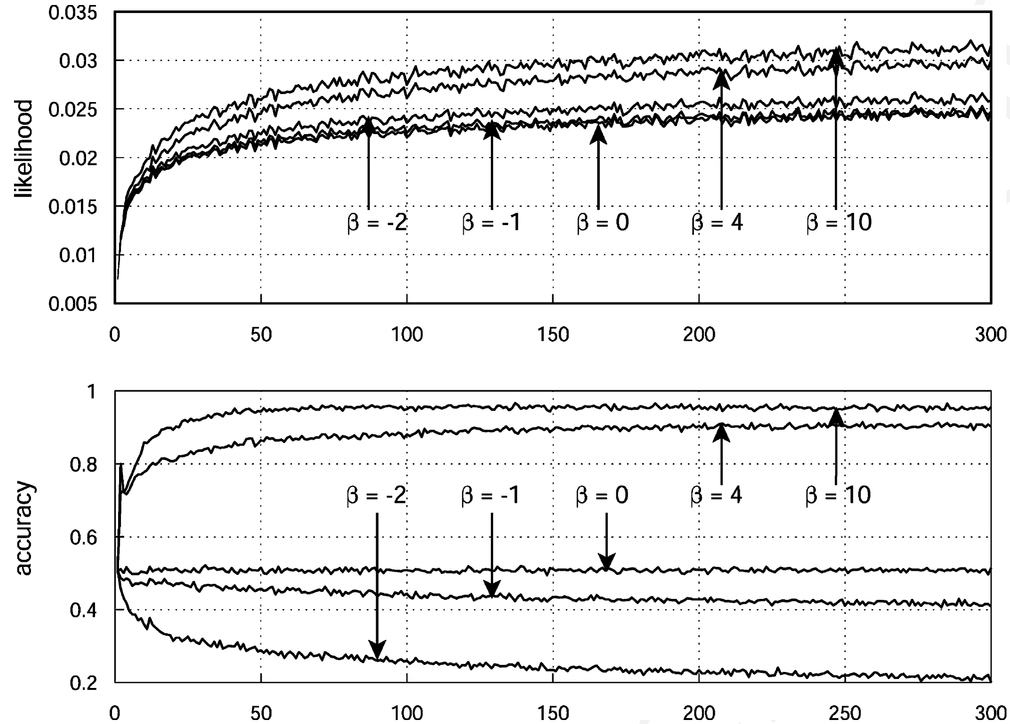

Fig. 7. Performance of the REM averaged over 1000 runs for different values of the parameter β as compared with EM. Curiously, even negative values of β result in higher likelihood than that attained by EM.

6.2. Multi-state bandit with hidden state

In contrast to the previous experiment a policy estimation is now introduced. The estimation of the perceptual state has to be performed on the basis of *indirect* reward attribution, i.e., the state now becomes hidden.

6.2.1. Maximization of the likelihood—binary bandit

This section shows the results on problems in which the reward function is well aligned with the likelihood, i.e., the problems where maximization of the reward results in maximization of the likelihood. Results for this task are shown in Fig. 9. 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.

It is particularly interesting 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

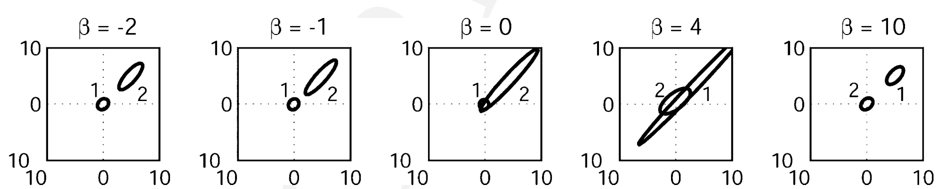

Fig. 8. Results of a run of the algorithm for different values of β starting from the same initial conditions. For coefficients with opposite signs the labeling is reversed, while the EM produces the labeling by chance. In this run the source distribution has the component 1 at the position (5, 5) and component 2 at (0, 0).

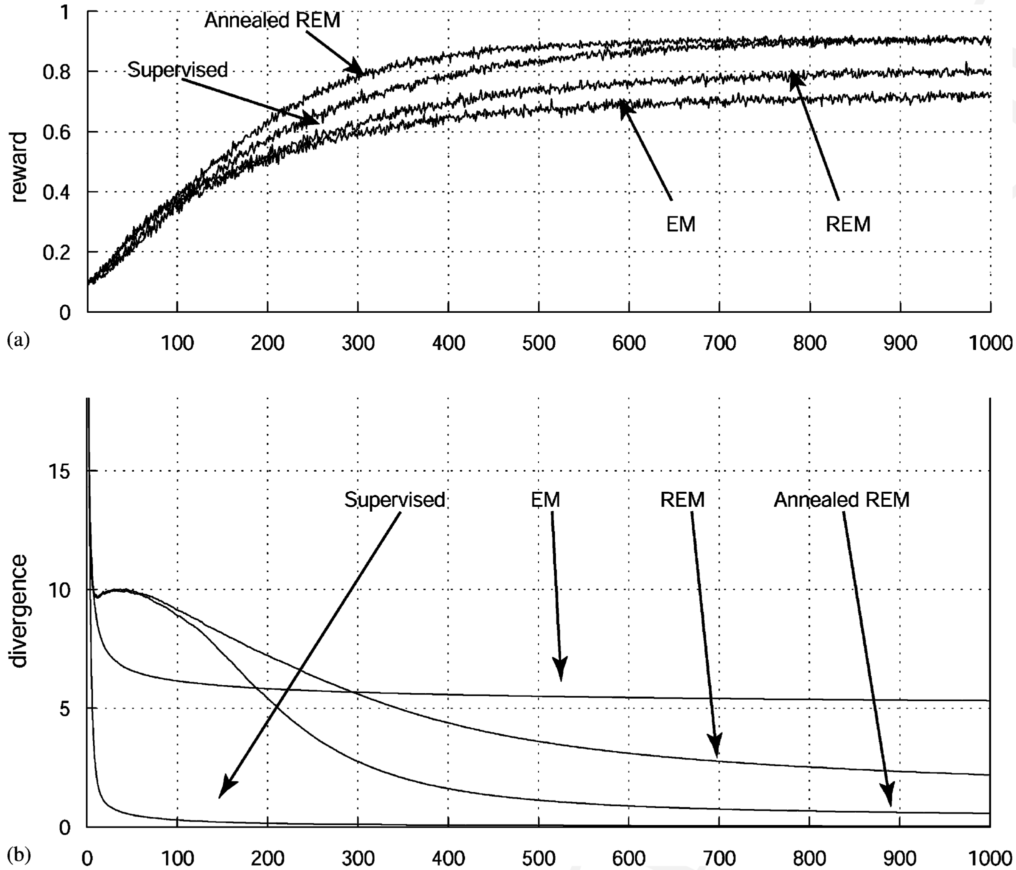

Fig. 9. (a) Performance on the 2-state 10-armed binary bandit. (b) Divergence between estimated and true source distributions.

gets reinforced than the regular EM. In the case of a Gaussian mixture density with a known number of components (known number of states), the fit can be measured with the symmetric KL divergence

$$S(p\|q) = \frac{1}{4}[(\mu_p - \mu_q)^{\mathrm{T}}(\mathbf{\Sigma}_p^{-1} + \mathbf{\Sigma}_q^{-1})(\mu_p - \mu_q) + \mathrm{tr}(\mathbf{\Sigma}_p^{-1}\mathbf{\Sigma}_q + \mathbf{\Sigma}_q^{-1}\mathbf{\Sigma}_p - 2\mathbf{I})]. \quad (14)$$

For a lack of a better analytical method, this quantity is computed for every combination of source and estimator components and the minimum value is selected.

The experiment with a 2-state 10-arm bandit is performed as follows:

(1) *Initialize*: for each state, randomly select a Gaussian observation model: $\mu_i \sim N(\mathbf{0}, 2I)$, $\sigma_i = I$;
(2) *Iterate:*
 (a) randomly choose a generator state, s_k;
 (b) generate an observation, x^n, from $\mathcal{N}(\mu_k, \sigma_k)$;
 (c) using current parameters select an action a^n;
 (d) if a^n is the same as the optimal arm deliver a reward of 1, otherwise -1;
 (e) update parameters of the model;

One variation on the algorithm described in this paper is the REM with the parameter β changing over time. For example, slowly increasing β, starting with the value of 0 will cause the algorithm to not pay any attention to the reward initially, while slowly shifting towards the "winner-take-all" mode after some period of time. Let us call it annealed REM.

Fig. 9a shows the average amount of reward collected by bandits trained with the EM, REM and 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 Eq. (14), given in Fig. 9b. 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.

6.2.2. *Maximization of the likelihood—full bandit*

The algorithm works with the full bandit, where each action taken by the algorithm results in some value of the reward—positive or negative, with no modifications. The results are shown in Fig. 10a. 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 (Fig. 10b), 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.

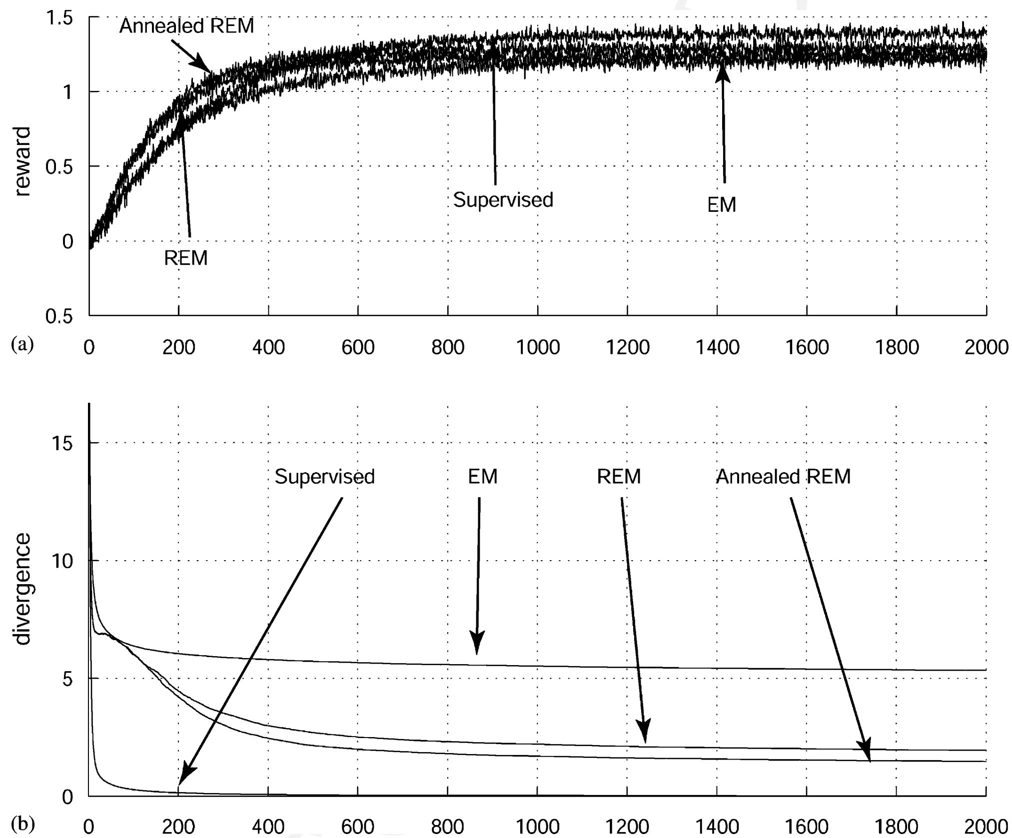

Fig. 10. (a) Performance on the full 2-state 10-armed bandit. (b) Divergence between estimated and true source distributions.

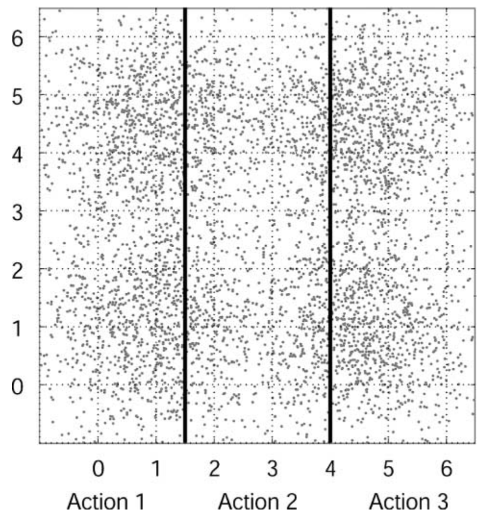

Fig. 11. Source and the reward structure for the reward bound maximization task. The data forms four strong clusters, while the reward is delivered for selecting action 1 if the data comes from the area marked “action 1”, etc.

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.

6.2.3. *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 four 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 $1.5 \leq x_1 < 4$ and a_3 when $x_1 > 4$, as shown in Fig. 11.

The performance of the model on this task is shown in Fig. 12. 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.

Fig. 13 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 kth position of lth row of the matrix indicates the probability of selecting an action k once the point x^n is classified as belonging to the cluster l. Figure (a)–(c) demonstrates 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 three Gaussians (component 4 is never used and therefore has a mixing coefficient 0). The REM uses all four components and aligns them with the reward partitioning. Note that both clusters 2 and 4 select action a_1.

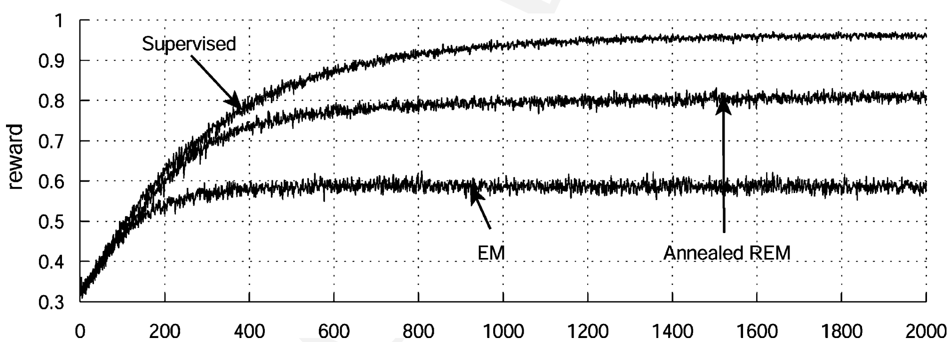

Fig. 12. Performance of EM, REM and a fully supervised estimator on the problem where reward structure does not coincide with the likelihood (averaged over 2000 runs).

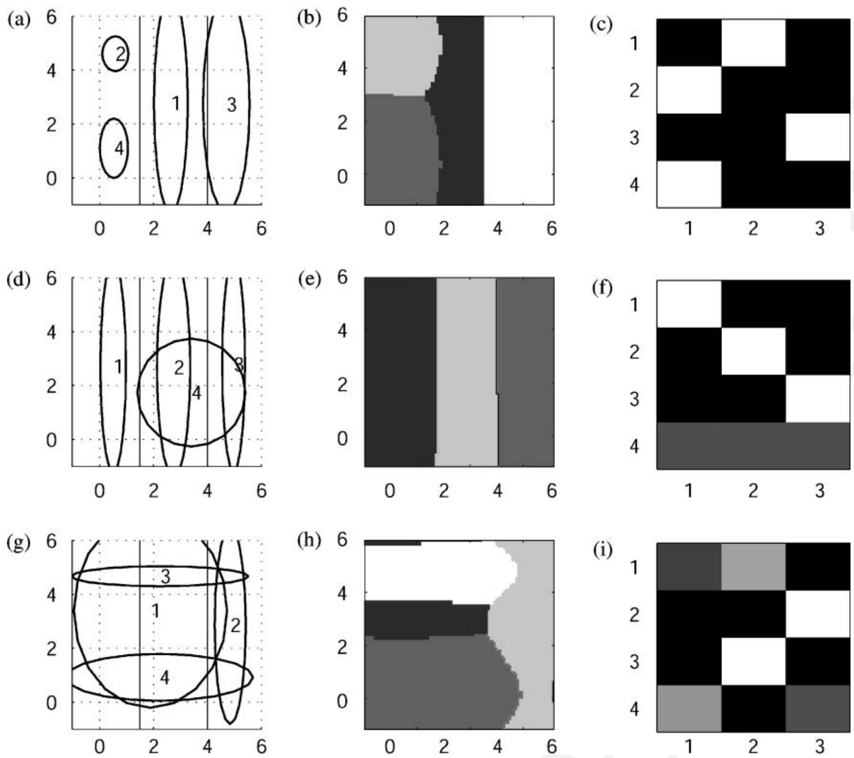

Fig. 13. Final cluster positions (left column), decision regions (middle column) and cluster assignment matrices (right column) for REM (top row), supervised (middle row) and EM (bottom row) estimators after a single run.

7. Conclusions

This paper 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. The justification for entropic variations of the posterior to achieve arbitrary component assignment goals is provided in the text. The algorithm allows for smooth blending between likelihood- and reward-based costs.

The paper shows that inclusion of the reward signal into the process of state estimation is important if we want to design agents without explicit programming of their perceptual states. The feedback is not only important for computing the policy of action selection, but also as a guiding mechanism for developing a robust grounded perception. In contrast to unsupervised techniques, where the final cluster configuration is aligned with the likelihood of the data, in the algorithm shown in this paper the grounding is achieved in a procedure allowing us to develop the configuration with a high utility to the agent.

One of the problems of the algorithm is the appropriate choice of the parameter β. In some cases it is convenient to have an asymmetric schedule for positive and negative rewards, which adds another parameter to the set.

In other cases special 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.

Appendix A

Theorem 1 shows that the a reward objective function $F(x, \theta)$ (Eq. (9)) is a lower bound on a log likelihood, $\log f(\theta)$ and can be used for EM-type estimation.

Theorem 1. *$F(x,\theta)$ is a lower bound on $\log f(\theta)$.*

Proof. Starting from (9), one can write

$$
\begin{aligned}
F(x,\theta) &= -D(q(s)\|p(s|x,\theta)) \\
&\quad + E_{q(s)}\left[\log\frac{p^*(s|x;r)}{p(s|x,\theta)}\right] + \log f(\theta) \\
&= \int q(s)\log\frac{p(s|x,\theta)}{q(s)}\mathrm{d}s \\
&\quad + \int q(s)\log\frac{p^*(s|x;r)}{p(s|x,\theta)}\mathrm{d}s + \log f(\theta) \\
&= \int q(s)\left[\log\frac{p(s|x,\theta)}{q(s)} + \log\frac{p^*(s|x;r)}{p(s|x,\theta)}\right]\mathrm{d}s \\
&\quad + \log f(\theta) = \int q(s)\log\frac{p^*(s|x;r)}{q(s)}\mathrm{d}s \\
&\quad + \log f(\theta) = -D(q(s)\|p^*(s|x;r)) \\
&\quad + \log f(\theta).
\end{aligned} \tag{A.1}
$$

In the last line of Eq. (A.1) the divergence, $D(q(s)\|p^*(s|x;r)) \geq 0$ from which it follows that

$$F(x,\theta) \leq \log f(\theta), \quad \forall q(s),\theta, \text{s.t.} \sum(q(s)) = 1 \tag{A.2}$$

with equality holding iff $q(s) = p^*(s|x;r)$. This concludes the proof. □

References

[1] N.M. Dempster, A.P. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of Royal Statistical Society B 39 (1977) 185–197.
[2] A. Likas, Reinforcement learning approach to online clustering, Neural Computation 11 (1999) 1915–1932.
[3] T.P. Minka, Tutorial: expectation-maximization as lower bound maximization, Tutorial Note, 1999.
[4] R.M. Neal, G.E. Hinton, A view of the EM algorithm that justifies incremental, sparse and other variants, in: M.I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, MA 1998, pp. 355–368.
[5] K. Nigam, A. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learning 39 (2/3) (2000) 103–134.
[6] R. Redner, H. Walker, Mixture densities, maximum likelihood, and the EM algorithm, SIAM Review 26 (2) (1984) 195–239.
[7] P.N. Sabes, M.I. Jordan, Reinforcement Learning by Probability Matching, Vol. 8, NIPS, 1996.
[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[9] M.A.L. Thathachar, P.S. Sastry, A new approach to the design of reinforcement schemes for learning automata, IEEE Transactions on Systems, Man and Cybernetics 15 (1985) 168–175.
[10] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[11] R.J. Williams, A class of gradient-estimating algorithms for reinforcement learning in neural networks, in: Proceedings of the IEEE First International Conference on Neural Networks, San Diego, CA, 1987.
[12] L. Xu, M.I. Jordan, On convergence properties of the EM algorithm for Gaussian mixtures, Neural Computation 8 (1) (1996) 129–151.