EM for Perceptual Coding and Reinforcement Learning Tasks

Yuri Ivanov Bruce Blumberg Alex Pentland

{yivanov bruce sandy}@media.mit.edu
MIT Media Laboratory
20 Ames Street, E15-368A
Cambridge, MA 02139, US

Abstract. The paper presents an algorithm for an EM-based reinforcement-driven clustering. As shown here it is applicable to the reinforcement learning setting with continuous state/discrete action space. E-step of the algorithm computes the posterior given the data and the reinforcement. Although designed to discover intrinsic states, the algorithm performs action selection without explicit state identification.

Learning algorithms are an important area of research in intelligent robotic systems. Intelligence requires that the system be able to adapt to its environment. One particularly difficult aspect of adaptation is the problem of selective attention and the ability of the system to generalize. Efficient generalization in the space of observations leads to emergence of observation states and results in more efficient action selection algorithms.

In this paper we present our work on an EM-based reinforcement learning algorithm which allows for action selection in the presence of a hidden state. The algorithm provides a probabilistic framework that unifies the state identification task with action selection.

The algorithm helps an agent learn a simple association task - given sensory data and a discrete set of actions the agent needs to find a mapping from observations to actions so that the reward is maximized[1]. We approach this problem via a set of latent variables. For the purposes of this paper we restrict our attention to the case where the number of states is known. In the future we plan to estimate that from data as well.

1 Related work

A similar problem of reinforcement-driven clustering was addressed by Likas in [4]. The reinforcement learning algorithm was combined with a neural network to improve vector quantization in the input space. In contrast, this paper extends the Expectation-Maximization (EM) algorithm first introduced in [2].

The problem of action selection in the presence of a hidden state is developed in [5]. In our current experiments we are not concerned with the reinforcement learning itself, but rather we present a useful connection between an un-supervised EM algorithm and any sort of traditional state-based reinforcement learning algorithm. However, we are planning to fully explore the reinforcement learning algorithms in our future work. For these purposes [3] and [7] provide an overview of the field of reinforcement learning. An application of multiple-model Q-learning and a POMDP model in computer vision for selective attention was introduced by Darrell and Pentland in [1] which uses this model for gesture recognition.

2 EM for perceptual coding

State identification is necessary in many traditional reinforcement learning algorithms. In continuous perceptual spaces there often exist task-dependent natural perceptual categories. This paper is concerned with discovery of such perceptual categories while allowing the agent to perform the action selection in the presence of unknown perceptual states. Without the task context such categories can be efficiently found in a framework of density estimation by an Expectation Maximization algorithm with a mixture distribution. We assume that observations come from a mixture, each tied to a state s, $p(x|s)$, weighted by corresponding prior state probabilities, $p(s)$:

[1] We want to use this algorithm as a part of a learning system of a synthetic creature.

$$p(x) = \sum_s p(s)p(x|s) \tag{1}$$

The EM algorithm lets us estimate the parameters of the mixture iteratively guaranteeing the improvement in the likelihood at each step until it converges to a local maximum. In order to do so, it requires computing the posterior distribution $p(s|x)$ over the hidden variables s (E-step), and then maximizing the expected error with respect to this distribution (M-step).

Getting every new observation x^n we can now compute our best belief about the actual state which generated this observation, $p(s|x^n)$, a "belief state". Once the state is determined, we can proceed with any traditional reinforcement learning algorithm to learn action selection.

However, the main drawback of this scheme is that the received reward in no way influences the class membership function $p(s|x)$. The clustering might as well be performed independently from the action selection estimation procedure. The following section is devoted to an algorithm that fixes this problem and takes into account the information that comes in the form of the reinforcement signal from the environment. A simple intuition here is that we want to form the state space that is found significant with respect to the reinforcement.

3 Q-table and action sampling

The algorithm presented here is largely based on a common interpretation of the Q-table (a table containing an expected reward for each state-action pair) that gives rise to the conditional density function. This allows for presenting exploration and exploitation of the action selection in the framework as sampling from this pdf. In this interpretation, the conditional pdf is computed from the Q-function directly, interpreting each value of the state-action pair as a multinomial count of the reward, from which, for fixed state, s, we can get a Maximum Likelihood estimate of the probability distribution over the set of discrete actions $p(a|s)$:

$$p(a_i|s) = \frac{Q(s,a_i)}{\sum_j Q(s,a_j)} \tag{2}$$

Alternatively, a softmax function can be applied to the Q-table, which can be varied in accordance with the average reward:

$$p(a_i|s) = \frac{e^{f(E[r])Q(s,a_i)}}{\sum_j e^{f(E[r])Q(s,a_j)}} \tag{3}$$

This interpretation has the advantage of representing the action selection policy as sampling from this distribution. For instance, ϵ-greedy policy, [7], is implemented as sampling from the distribution:

$$p(a_i|s) = \begin{cases} 1-\epsilon & \text{if } a_i = \arg\max_j(Q(s,a_j)), \\ \epsilon/(|a|-1) & \text{otherwise.} \end{cases} \tag{4}$$

Figure 1a) shows the probability distributions of interest. In the center of it is the table, representing the conditional pdf, $p(a|s)$(white color represents a value close to 1, black - close to 0). In the figure, rows of $p(a|s)$ correspond to states and columns - to actions. Summation of this table along its columns pre-multiplied with $p(s|x^n)$ corresponds to marginalization of $p(a,s|x^n)$ over s (see eqn. 5). It is easy to see independencies in this structure, showing it as a graphical model (figure 1b)), most important being that action, a, is conditionally independent of observation, x, given the state, s, which will be used in the subsequent sections.

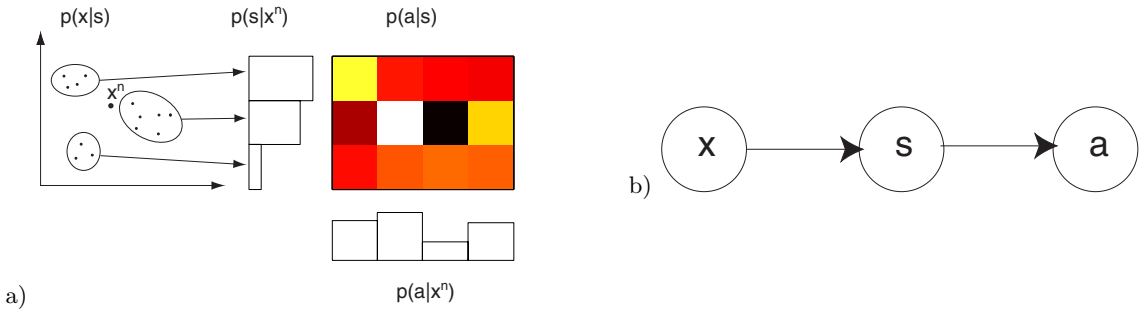

Fig. 1.: a) probability distributions of interest in the reinforcement-driven EM. The white color represents a value close to 1, black - close to 0; b) its representation as a graphical model.

4 Reward propagation

Having computed $p(a|s)$ from the Q-table, for each new observation, x^n, we can find the probability distribution over actions, $p(a|x^n)$, by a brute force approach - first computing the joint pdf, $p(x^n, s, a)$, and then marginalizing out the state, s. The resulting joint pdf, $p(x^n, a)$, is easily converted into the conditional pdf.

A better alternative is to utilize factorizations in the joint pdf, $p(x, s, a)$. The corresponding simple graphical model is shown in figure 1b). Each new observation, x^n, lets us compute $p(a|x^n)$ via $p(a|s)$. After receiving the reward and updating $p(a|s)$, the effect of the reward can be propagated back to the node x to compute and update $p(x)$.

Below we detail the steps of the algorithm.

4.1 Action selection - computing $p(a|x^n)$

The goal of the action selection is given an observation x^n to choose an optimal action a according to a probability distribution $p(a|x^n)$. When the mapping $x^n \mapsto s_i$ is known, the action selection is done by sampling $p(a|s_i)$. However in the case of an on-line EM action selection is typically more difficult because the state is hidden and is being constantly re-estimated. The distribution $p(a|x^n)$ that is sampled to select an action in this case is not the same as $p(a|s_i)$, but is computed by marginalizing $p(a, s|x^n)$ as follows:

$$\begin{aligned} p(a|x^n) &= \sum_s p(a, s|x^n) \\ &= \sum_s p(a|s, x^n) p(s|x^n) \\ &= \sum_s p(a|s) p(s|x^n) \end{aligned} \tag{5}$$

The first factor of the resulting equation is the conditional pdf computed from the Q-function, the second - the posterior probability distribution computed in the E-step of the EM for the new piece of data, x^n. To select an action a we sample the resulting density, $p(a|x^n)$, according to a chosen sampling strategy as shown in section 3.

4.2 Parameter update - computing $p(s|x^n, a^n)$

After the action is taken, the reward is received and the Q-table is updated, we can compute an improved estimate of the posterior to replace the posterior $p(s|x^n)$ to take into account the evaluation of the action selection. The intuition is that if no reward is received, it is not considered a new evidence and the posterior should not change, while with an updated $p(a|s)$, the posterior should take into account that attributing x^n to the state which produced the reward was beneficial. In other words, in the M-step of EM we replace $p(s|x^n)$ with $p(s|x^n, a^n)$, which is propagated back through the Q-table as follows (reversing the arrows in 1b)):

$$
\begin{aligned}
p(s|x^n, a^n) &= \frac{p(s, a^n|x^n)}{p(a^n|x^n)} \\
&= \frac{p(a^n|s, x^n)p(s|x^n)}{p(a^n|x^n)} \\
&= \frac{p(a^n|s)p(s|x^n)}{p(a^n|x^n)}
\end{aligned} \tag{6}
$$

The improved posterior, $p(s|x^n, a^n)$, is readily computed from the data which is already available - the first factor in the numerator is a conditional pdf computed from the updated Q-table, while the second one is the old posterior.

4.3 Summary of the algorithm

1. Initialize
   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) compute $p(s|x^n)$ using the Bayes rule and the current parameters of the model;
   (b) convert Q-table to $p(a|s)$;
   (c) compute $p(a|x^n)$, as given in eqn. (5);
   (d) select an action according to $p(a|x^n)$, collect reward and update the Q-table;
   (e) convert Q-table to $p(a|s)$;
   (f) compute $p(s|x^n, a^n)$ as given by eqn. (6);
3. M-step
   Update the model parameters to maximize the expected log likelihood $p(x^n, s)$ with respect to the distribution $p(s|x^n, a^n)$.

5 Experimental results

In this section we show results of the algorithm in two settings. In the first, the input space is modeled by a mixture of Gaussians. The two sets of results are shown on synthetic data and the well-known IRIS data set. In the second setting observations are modeled by a mixture of Hidden Markov Models.

We compare the proposed algorithm with an unguided clustering performed by EM algorithm with no reinforcement feedback. It is not completely fair to compare performances of the two algorithms, however, it is clear that performance of the unguided EM should provide a lower bound for the performance of the proposed extension. In all comparisons against EM algorithm parameter estimation started from identical initial conditions for both models.

In our current experiments we only attempt to learn direct associative relationship between observations and actions. This implies that there is no need to estimate state dynamics, since the probability of being in a particular state is independent of the previous state. With this in mind we simplify the estimation of the Q-table as follows:

$$
Q(s, a) \leftarrow \gamma Q(s, a) + \delta(a, a^n)p(s|x^n)r \tag{7}
$$

with $0.8 < \gamma < 0.98$. This effectively distributes the reward for taking the action a^n among states in proportion with their posterior, $p(s|x^n)$.

5.1 Gaussian mixture model

The first set of experiments is run on synthetic data generated from a Gaussian mixture probability distribution. The experiment is set up as follows:

1. randomly initialize K normal distributions for the data source;
2. sample this mixture randomly switching between these K Gaussians to generate the complete data set;
3. reinforce the action selection such that if the x^n came from distribution k, selecting the action a_k is rewarded. This results in a classification scheme performance of which we can evaluate.

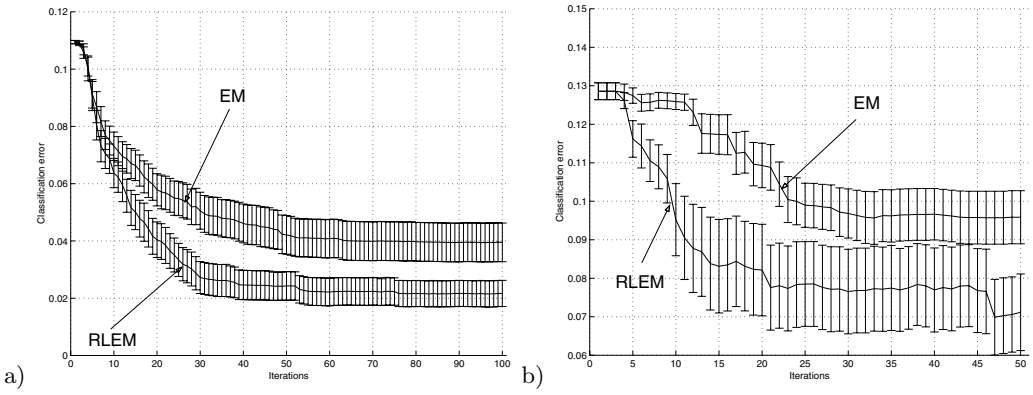

Fig. 2.: Performance of the algorithm compared to the case where the EM algorithm was used for perceptual coding with no feedback. a) 5-states/5-action setup with $\mu_i \sim \mathcal{N}(0, 2)$, $\sigma_i^2 \in [0.5, 1]$. b) 3-states/3-action setup with $\mu_i \sim \mathcal{N}(0, 1)$, $\sigma_i^2 \in [0.5, 1]$.

The figure 2 shows comparison of the algorithm presented here with a vanilla EM. Both algorithms perform fairly well when the means of the generating sources are well separated (figure 2a). In this case the alignment of EM with clusters in the data is more likely. In the second case, where the data clusters significantly overlap, such an alignment is less likely, and this is the more important case that our algorithm handles well (figure 2b). Our algorithm typically converges to the true source distributions (figure 3a).

An unguided EM does not typically result in a good estimate of the Q-table, while our algorithm results in the Q-table, which is almost deterministic (figure 3b).

	3 states	4 states	5 states
RLEM	91.67%	98%	97.47%
EM	67.33%	78.27%	77.07%

Table 1.: Classification accuracy results for the EM and RLEM algorithm on the IRIS data.

Further comparisons can be made on one of the standard data sets. We chose the well known IRIS data set. It consists of 150 4-dimensional samples. The measurements come from three classes of plants. As in the previous experiment with synthetic data the class information is only used to indicate that the algorithm made the right guess assigning a label to a particular observation as described in the earlier sections. Plots in figure 4 show the performance of the algorithm averaged over 10 runs while using 3, 4 and 5 states respectively. The table 1 shows the classification accuracy of RLEM and the EM with no feedback.

5.2 Mixture of HMMs

In the final set of experiments we address the problem of reinforcement-driven classification of the speech utterances. The goal is to learn to respond to simple voice commands used for dog training. We need to learn how to differentiate between these commands having no initial labeling such that the agent's actions

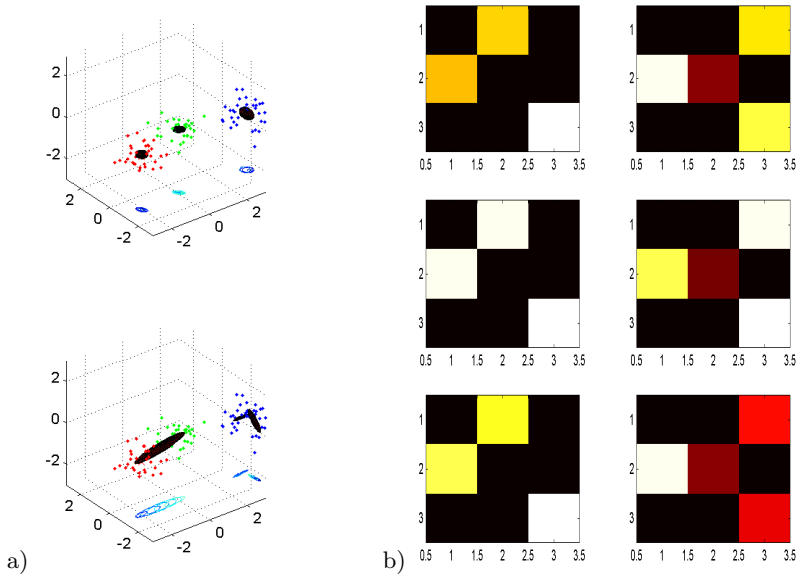

Fig. 3.: a) Typical result of the training of the proposed algorithm (top) and the usual EM (bottom). It can be seen that EM did not arrive to the correct cluster assignment, while integration of the reinforcement signal into parameter estimation resulted in the correct assignment. b) First column - Q-table, $p(a|s)$ and the joint pdf $p(a, s)$ for the RLEM. Second column - the same for EM. The confusion of EM about true clusters is reflected in the Q-table.

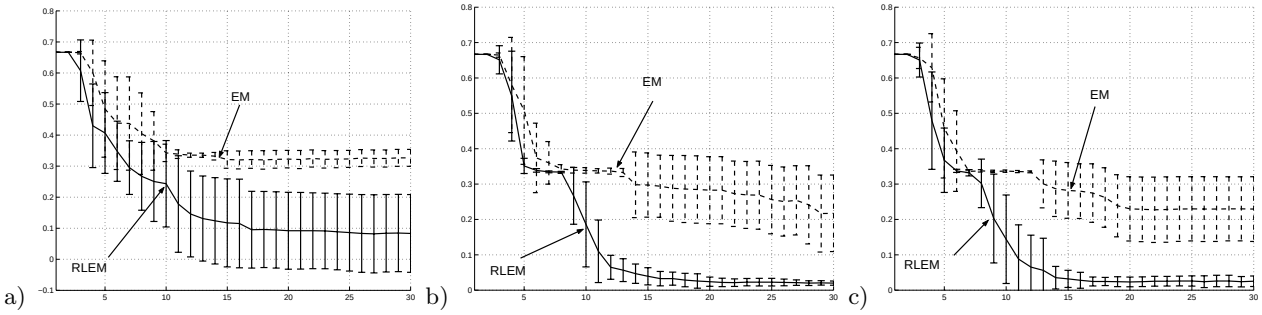

Fig. 4.: Performance of the algorithm in the classification task on the IRIS data. a) Performance with 3 Gaussian components. b) 4 Gaussian components. c) 5 Gaussian components.

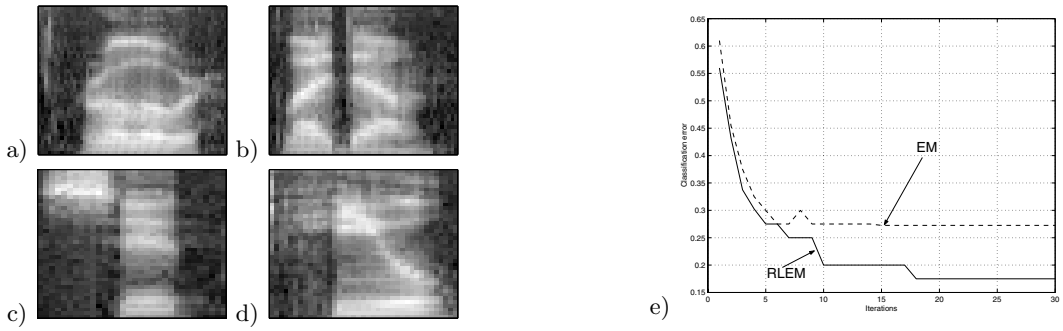

Fig. 5.: a)-d) Mel-scale filter bank responses of utterances (a) 'roll over', (b) 'lie down', (c) 'sit' and (d) 'heel'. e) Classification accuracy of the reinforcement-driven clustering and a mixture estimated by EM with no feedback.

yield maximum reward. For the purposes of this paper we limit the task to simply identifying the action we are going to take and use the action selection error as a measure of classification accuracy.

The input to the model is a sequence of spectral coefficients of a mel-scale filter bank. Figures 5a)-d) show examples of the spectrum of utterances used in the experiments. The data set consisting of 10 examples of each utterance is projected onto the sub-space of three largest eigenvectors of a large sample of unconstrained speech. Each component of a mixture distribution is modeled by an HMM ([6]).

In the training of HMM mixture parameters we weigh each data point with the posterior $p(s|x^n, a^n)$, similarly to the case of the Gaussian mixture. For instance, new values of the transition probability matrix, A, are essentially averaged with respect to the posteriors:

$$A_{ij} = \frac{\sum\limits_n p(s|x^n, a^n)\tilde{A}^n_{ij}}{\sum\limits_n p(s|x^n, a^n)} \tag{8}$$

where $\tilde{A}^n_{ij}$ is a contribution to A_{ij} due to the sequence x^n and is estimated in the usual manner:

$$\tilde{A}^n_{ij} = \frac{\sum_{t=1}^{T^n} p(q_t = i, q_{t-1} = j|x^n, \lambda)}{\sum_{t=1}^{T^n} p(q_t = i|x^n, \lambda)} \tag{9}$$

The rest of the parameters of each HMM is estimated in a similar fashion. Figure 5e) shows the classification accuracy of the algorithm compared to having no reinforcement signal affect parameter estimation. It shows a 10% increase in accuracy due to the feedback from action selection.

6 Discussion

The algorithm presented in this paper builds a useful connection between reinforcement learning algorithms utilizing a Q-function and an EM-based state estimator. The advantage afforded by the algorithm comes from being able to compute an estimate of the partitioning function $p(s|x^n)$ with additional evidence of an assignment of each x^n to the correct class given by the reinforcement signal.

One thing to note is that if we can iterate through the data set infintely many times and at each iteration have access to the reinforcement, then it can be shown that the exact partitioning can be determined in only $K - 1$ iterations. Of course, such an estimation is impossible on-line since only partial information about correct labeling will be available (we can only guess a label for each data point once), while the algorithm presented here will have no problems.

Another concern is that in the context of the current set of experiments, another approach is possible - since the reward is immediate and serves as an indication of the correct class membership of each data point, x^n, we can use this information deterministically. That is, we can mark the input data which we assigned correctly and never change their membership afterwards. This technique is reasonable when immediate reward is available, however, with delayed reward, the problem of temporal credit assignment arises preventing us from using it in such a straightforward manner. In order for the algorithm to remain general we need to compute the component density estimates probabilistically, via $p(a|s)$.

7 Future work

In our future work we plan to automatically select the number of components based on the information content of the Q-table.

Another direction is to explore state transitions to perform the full reinforcement learning task, comparing it to the full POMDP approach.

8 Acknowledgments

The authors would like to thank Leslie Pack Kaelbling and Andrew Barto for fruitful discussions about this and future work. We would also like to extend our gratitude to the rewievers of the paper for their comments which helped to make the presentation more clear.

References

1. T. Darrell and A. Pentland. Multiple-model q-learning for active recognition tasks. In *13th IEEE Intl. Conference on Pattern Recognition*, Vienna, Austria, 1996.
2. N.M. Dempster, A.P. Laird and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39:185–197, 1977.
3. L.P. Kaelbling, L.M. Littman, and A.W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
4. A. Likas. Reinforcement learning approach to online clustering. *Neural Computation*, 11:1915–1932, 1999.
5. A.K. McCallum. *Reinforcement learning with selective perception and hidden state*. Ph.d., University of Rochester, 1995.
6. L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice Hall, Englewood Cliffs, 1993.
7. R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

This article was processed using the TEX macro package with SIRS2000 style