# Incremental Structure Learning in Factored MDPs with Continuous States and Actions

**Christopher M. Vigorito**
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
vigorito@cs.umass.edu

**Andrew G. Barto**
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
barto@cs.umass.edu

Technical Report UM-CS-2009-029

## Abstract

Learning factored transition models of structured environments has been shown to provide significant leverage when computing optimal policies for tasks within those environments. Previous work has focused on learning the structure of factored Markov Decision Processes (MDPs) with finite sets of states and actions. In this work we present an algorithm for online incremental learning of transition models of factored MDPs that have continuous, multi-dimensional state and action spaces. We use incremental density estimation techniques and information-theoretic principles to learn a factored model of the transition dynamics of an FMDP online from a single, continuing trajectory of experience.

## 1 Introduction

The factored Markov Decision Process (FMDP) is a formalism for sequential decision problems that allows for the explicit representation of environmental structure not possible in the traditional Markov Decision Process (MDP). Much work in the reinforcement learning literature has focused on algorithms that exploit this structure to more efficiently learn or compute optimal solutions to such problems [1, 2]. In some cases this has allowed for the application of reinforcement learning techniques to environments much larger than could otherwise be handled feasibly by methods that do not exploit structure. These approaches generally make use of structure by ignoring regions or entire dimensions of the state space that are irrelevant to solving a particular task. Many of these approaches make use of factored transition and reward models to represent these independencies.

The majority of work with FMDPs has focused on discrete environments; i.e., those with a finite number of states and actions. Some of these approaches have presented methods for learning a factored transition model online from experience, and for using that model to compute optimal policies [3, 4, 5]. A smaller body of work has considered computation of optimal policies in FMDPs with continuous state and action spaces, as well as hybrid environments with both continuous and discrete components [6]. These methods are offline approaches, however, and assume that access to an accurate transition model is given a priori. The work presented here provides an algorithm for learning factored transition and reward models of continuous state and action FMDPs in an incremental, online fashion, and thus paves the way for applying online methods for learning optimal policies that make use of these models and their inherent structure.

Our approach uses incremental density estimation techniques and information theoretic principles to learn these models online from a single trajectory of experience. In the following section we present relevant background material and outline the formalism for continuous FMDPs that we adopt. Sections 3 and 4 discuss the details of our approach, and Section 5 presents an empirical evaluation in a stochastic, structured environment. Finally we summarize our approach and discuss limitations of the algorithm and future work in Section 6.

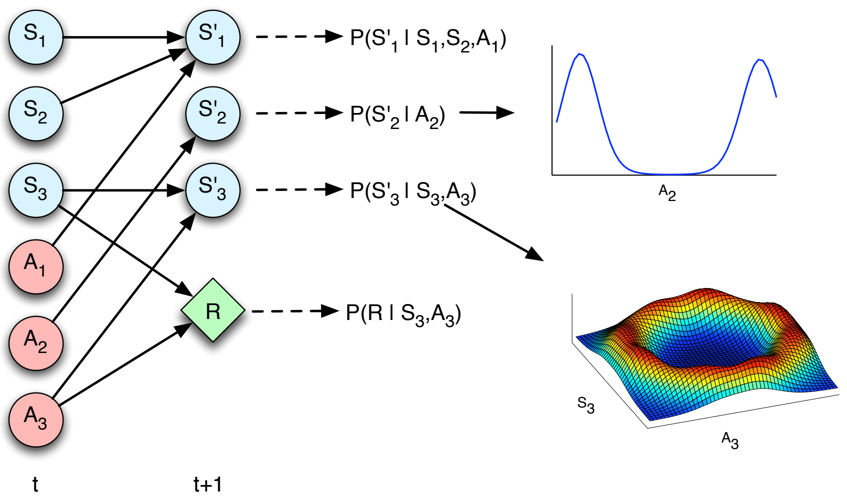

Figure 1: An example DBN for an FMDP with a 3-dimensional observation space and 3-dimensional action space. Example conditional probability distributions are shown for two variables.

## 2 Continuous Factored MDPs

A Markov Decision Process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, where $S \subseteq \Re^n$ is a set of states, $A \subseteq \Re^m$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a one-step transition model that provides a distribution over successor states given a current state and action, $R : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ is a one-step expected reward model that specifies the reward an agent receives for taking a given action in a given state, and $\gamma$ is a discount factor whose significance is irrelevant to our approach. While in the MDP formalism the transition function $P$ is necessarily high-dimensional for high-dimensional state and action spaces, if there is significant structure in the dynamics of the MDP so that the values of certain dimensions are irrelevant to computing the expected value of others at the following time step, $P$ can often be represented in factored form. This leads to the factored Markov Decision Process (FMDP) formalism.

In an FMDP, each dimension of the state space is represented as a random variable $S_i \in \mathbf{S}$, and states are thus vectors in $\Re^n$ corresponding to assignments to each of these variables. The transition and reward models of an FMDP are often represented graphically using a Dynamic Bayesian Network (DBN) [7]. A DBN is a two-layer directed acyclic graph with nodes in layers one and two representing the variables of the FMDP at times $t$ and $t + 1$, respectively (see Figure 1). We will henceforth denote the set of state variables at time $t$ as $\mathbf{S} = \{S_1 \dots S_n\}$ and those at time $t + 1$ as $\mathbf{S}' = \{S'_1 \dots S'_n\}$. Edges in a DBN represent dependencies between variables. We make the common assumption that variables within the same layer do not influence each other. While models of finite FMDPs are often represented with one DBN per action, this is not possible in continuous FMDPs. Rather we represent these models using a single DBN with a set of action variables $\mathbf{A}$ that influence the state variables at the next time step, as shown in Figure 1.

If we let $f_{\mathbf{X}}(\mathbf{s}, \mathbf{a})$ denote the projection of state-action pair $(\mathbf{s} \in \mathcal{S}, \mathbf{a} \in \mathcal{A})$ onto a set of variables $\mathbf{X}$ (i.e., the values the variables in $\mathbf{X}$ take on in $\mathbf{s}$ and $\mathbf{a}$), $f_{\mathbf{X}}(\mathbf{s})$ similarly denote the projection of state $\mathbf{s} \in \mathcal{S}$ onto $\mathbf{X}$, and $Par(Y)$ denote the set of parents of variable $Y$ in the DBN, then the transition function $P$ can be expressed in factored form as

$$P(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = \prod_i^n P(S'_i = f_{S'_i}(\mathbf{s}')|Par(S'_i) = f_{Par(S'_i)}(\mathbf{s}, \mathbf{a})). \tag{1}$$

This form represents the transition function as the product of several conditional distributions, each of which may have much lower dimension than the full joint distribution being modeled. It is this fact that leads to the computational savings associated with these models. Note that the reward model $R$ can be represented in a similar fashion (the diamond-shaped node in Figure 1). It remains to be shown how to represent the component conditional distributions that comprise the full model (right-hand side of Figure 1). In finite FMDPs, these are often represented as decision trees, one for each variable, with internal nodes corresponding to the parents of the given variable and leaves containing probability mass functions corresponding to the conditional distributions. Again, this is not possible in continuous FMDPs for obvious reasons. The following section presents the representation we adopt for the conditional density functions and an incremental method for estimating these densities online, as well as a method for computing mutual information between sets of random variables modeled by these estimators.

## 3 Online Incremental Density and Information Estimation

### 3.1 Density Estimation

There are many choices for the form of density function we may adopt to represent the factors of an FMDP's transition model. We choose the mixture of Gaussians (MOG) model for several reasons, the first being that there are existing methods for both online, incremental estimation of these models and for efficient computation of their entropies, which we make use of in the following section. Secondly, these models provide a natural, efficient way of obtaining conditional probabilities from the joint distributions they represent, which is useful when employing them in reinforcement learning algorithms. Additionally, it has been shown that given a sufficient number of components, the MOG model can represent arbitrary densities [8].

A MOG model with $k$ components gives the probability of a vector $\mathbf{x} \in \Re^n$ as

$$p(\mathbf{x}|\Theta) = \sum_{i=1}^{k} \pi_i p_i(\mathbf{x}|\theta_i), \tag{2}$$

where $\Theta = \{\theta_1, \ldots, \theta_k\}$ is a set of parameter vectors, one for each component, $\pi_i$ is the mixing coefficient (or prior) of component $i$, and $p_i(\mathbf{x}|\theta_i)$ is the component-conditional density of component $i$, which is given as

$$p_i(\mathbf{x}|\theta_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right], \tag{3}$$

where $\theta_i = \{\mu_i, \Sigma_i\}$ represents the mean vector and covariance matrix of component $i$.

Learning the parameters of MOG models is a difficult problem with much work devoted to it. While many approaches use some form of the Expectation-Maximization (EM) algorithm for this task [9], this solution does not lend itself to an efficient online setting, and has often found to be oversensitive to parameter initialization. One alternative approach uses a modification to the Self-Organizing Map (SOM) neural network architecture, called the Self-Organizing Mixture Network (SOMN) [10], to incrementally update the parameters of the model via a stochastic gradient-descent method. Although the SOMN is more general than a MOG model in that it can make use of non-Gaussian mixture components, we present here only the details of its operation for the case of Gaussian components.

After each new training example is observed, the parameters of a SOMN are updated so as to minimize the Kullback-Leibler divergence between the true ($p$) and estimated ($\hat{p}$) densities via stochastic approximation methods. Let $\hat{p}$ and $\hat{p}_i$ be the SOMN's estimates of (2) and (3), $\hat{\pi}_i^t$, $\hat{\mu}_i^t$, and $\hat{\Sigma}_i^t$ be the estimates of the prior, mean vector, and covariance matrix of component $i$, respectively, after $t$ training examples have been observed, and $\hat{P}(i|\mathbf{x}) = \frac{\pi_i \hat{p}_i(\mathbf{x}|\theta_i)}{\hat{p}(\mathbf{x}|\Theta)}$ be the posterior probability of component $i$ given training example $\mathbf{x}$. When a new training example $\mathbf{x}$ is observed, the parameters of each component are updated according to

$$\hat{\pi}_i^{t+1} = \hat{\pi}_i^t + \alpha[\hat{P}(i|\mathbf{x}) - \hat{\pi}_i^t] \tag{4}$$

$$\hat{\mu}_i^{t+1} = \hat{\mu}_i^t + \beta[\mathbf{x} - \hat{\mu}_i^t]\hat{P}(i|\mathbf{x}) \tag{5}$$

$$\hat{\Sigma}_i^{t+1} = \hat{\Sigma}_i^t + \gamma[(\mathbf{x} - \hat{\mu}_i^t)(\mathbf{x} - \hat{\mu}_i^t)^T - \hat{\Sigma}_i]\hat{P}(i|\mathbf{x}) \tag{6}$$

where $0 < \alpha, \beta, \gamma < 1$ are step size parameters. Alternatively, for computational efficiency a winning component may be selected based on the posterior probabilities, and only those components within a local neighborhood of the winner need be updated.

Experiments with the SOMN have shown that it is very robust to parameter initialization, and so it is common to initialize the priors evenly (i.e., $\pi_i = 1/k$, $\forall i$), the mean vectors randomly, and the covariance matrices to $\sigma\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and $\sigma$ is a scalar. The SOMN does require the number of mixture components $k$ to be pre-specified, however, which is a significant limitation. We discuss possible remedies to this problem in our discussion section.

### 3.2 Mutual Information Estimation

Given the form of density function described above, we now present a method for incrementally estimating the mutual information between sets of random variables whose joint distribution is modeled by a SOMN. One way to express the mutual information $I(X, Y)$ between two (sets of) random variables $X$ and $Y$ is as a sum of entropies:

$$I(X, Y) = H(X) + H(Y) - H(X, Y), \tag{7}$$

where for a real-valued random variable $X$, $H(X) = -\int p(X) \log p(X) dX$ is the Shannon differential entropy of $X$. The joint differential entropy of random variables $X$ and $Y$ is given similarly as $H(X, Y) = -\int p(X, Y) \log p(X, Y) dX\, dY$.

Unfortunately, for our choice of density function, computing mutual information based on Shannon entropies is infeasible. However, for the case of a MOG density model as we have assumed, there is an approximation to Shannon entropy that has a particularly nice closed form. This is the quadratic Renyi entropy, a specific instance of a class of generalized entropies described in [11], and for random variable $X$ is given as $H_{R_2}(X) = -\int P(X)^2 dX$.

If we let $G(\mathbf{x} - \mu_i, \Sigma_i)$ represent the value of Gaussian mixture component $i$ evaluated at $\mathbf{x}$, then note that $\int_X G(X - \mu_i, \Sigma_i) G(X - \mu_j, \Sigma_j) dX = G(\mu_i - \mu_j, \Sigma_i + \Sigma_j)$. Thus, for a mixture of $k$ Gaussian densities, the quadratic Renyi entropy of the mixture density can be computed as

$$\begin{aligned} H_{R_2}(X) &= -\log \int P(X)^2 dX \\ &= -\log \int \left( \sum_i^k \pi_i G(X - \mu_i, \Sigma_i) \right)^2 dX \\ &= -\log \sum_{i=1}^k \pi_i \sum_{j=1}^k \pi_j G(\mu_i - \mu_j, \Sigma_i + \Sigma_j), \end{aligned} \tag{8}$$

so that the computation reduces to pairwise interactions between mixture components. Additionally, only half of these need to be computed in practice because of symmetry.

Although there is a similar closed form for the joint quadratic Renyi entropy of two random variables we could use to compute mutual information, we will take a slightly different tactic to obtain the joint entropy. Suppose we have a vector-valued random variable $X \in \Re^{r+s}$ so that $X = [X_1\ X_2]^T$, $X_1 \in \Re^r$, and $X_2 \in \Re^s$. If $X \sim \mathcal{N}(\mu, \Sigma)$ is multivariate Gaussian with $\mu = [\mu_1\ \mu_2]^T$ and $\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$ so that $\mu_1 \in \Re^r$, $\mu_2 \in \Re^s$, $\Sigma_{11} \in \Re^{r \times r}$, $\Sigma_{12} \in \Re^{r \times s}$, $\Sigma_{21} \in \Re^{s \times r}$, and $\Sigma_{22} \in \Re^{s \times s}$, then the marginal distribution of $X_1$ is itself a multivariate Gaussian distribution with mean $\mu_1$ and covariance $\Sigma_{11}$. The marginal of $X_2$ is similarly modeled with mean $\mu_2$ and covariance $\Sigma_{22}$.

This means that we may obtain the joint entropy between two (sets of) variables $X$ and $Y$ by modeling their joint distribution explicitly, and their marginal entropies by applying (8) to the appropriate marginal distributions obtained from the joint, as just described. We will use this fact when evaluating potential dependencies in a DBN, as described in the following section.

## 4 Online Incremental Structure Learning

We now turn to our application of the techniques outlined so far to the problem of incremental structure learning in FMDPs with continuous states and actions. Previous work on learning factored transition models of finite FMDPs has taken the approach of adding dependencies to a DBN model of an FMDP one at a time when the mutual information between two variables is significantly high [3, 12]. We take a similar approach, although our techniques will be different because we are dealing with continuous states and actions.

Recall that mutual information can be expressed as a difference between the sum of two marginal entropies and their joint entropy, as in (7). Let $S_i'^{+} = S_i' \cup Par(S_i')$ be the union of a state variable $S_i' \in \mathbf{S}'$ in the DBN at time $t+1$ and its set of parents (in $\mathbf{S} \cup \mathbf{A}$) at time $t$. Our strategy will be to maintain, for each $S_i'$, an estimate of the information between $S_i'^{+}$ and each other state and action variable in $\mathbf{S} \cup \mathbf{A}$ not already in $Par(S_i')$. We term each of these extra variables a *candidate* variable, and whenever the information between a candidate variable and $S_i'^{+}$ exceeds a pre-specified value, we add that variable to $Par(S_i')$ and remove it from the list of candidate variables for $S_i'$.

In order to do this we will maintain an estimate of the joint distribution of each possible combination of $S_i'^{+}$ and candidate variable $X \in (\mathbf{S} \cup \mathbf{A}) - Par(S_i')$. This initially requires the instantiation of $n^2 m$ SOMN models ($nm$ models per state variable), where $n$ and $m$ are the dimensionalities of the state and action spaces of the FMDP, respectively. At time step $t$, each SOMN modeling a distribution containing $S_i'$ is given a training example that is the concatenation of the values in the previous state and action vector corresponding to the current parents of $S_i'$ and that distribution's associated candidate variable, and the value of $S_i'$ in the current state vector. Initially this will result in the estimation of the joint distributions corresponding to each element of $\mathbf{S}' \times (\mathbf{S} \cup \mathbf{A})$. The techniques described in Section 3 now provide us with the means to compute the mutual information between each $S_i'^{+}$ and each of its candidate variables from these joint distributions.

**Algorithm 1** LearnStructure

```
Initialization:
$\mathcal{M} \leftarrow \{\}$
**for** each $S_i' \in \mathbf{S}'$ **do**
  **for** each $X \in \mathbf{S} \cup \mathbf{A}$ **do**
    initialize a 2-dimensional SOMN $m_{S_i',X}$ to model $p(S_i'^{+}, X)$
    $\mathcal{M} \leftarrow \mathcal{M} \cup m_{S_i',X}$
  **end for**
**end for**
s $\leftarrow$ initial state
Maintenance:
**for** t=1 to $\infty$ **do**
  $\mathbf{a} \leftarrow$ choose action
  $\mathbf{s}' \leftarrow$ next state
  **for** each $m_{S_i',X} \in \mathcal{M}$ **do**
    concatenate $f_{S_i'}(\mathbf{s}')$, $f_X(\mathbf{s},\mathbf{a})$, and $f_{Par(S_i')}(\mathbf{s},\mathbf{a})$ into training example $\mathbf{x}$
    update $m_{S_i',X}$ with $\mathbf{x}$ using (4), (5), and (6)
    compute $I(S_i'^{+}, X)$ using (7) and (8) (see text)
    **if** $I(S_i'^{+}, X) > \eta$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} - m_{S_i',X}$
      $S_i'^{+} \leftarrow S_i'^{+} \cup X$ (add $X$ as parent of $S_i'$)
      **for** each $Y \notin S_i'^{+}$ **do**
        extend $m_{S_i',Y}$ to model new dimension $X$ (see text)
      **end for**
    **end if**
  **end for**
  $\mathbf{s} \leftarrow \mathbf{s}'$
**end for**
```

At every time step, after updating the density models, we evaluate each candidate variable $X$ for each $S_i'$ by computing the three entropies $H(S_i'^{+})$, $H(X)$, and $H(S_i'^{+}, X)$ using (8) and the appropriate marginals obtained from the SOMN model for each $X \notin S_i'^{+}$, and then calculating the resulting information. For each $S_i'$, the candidate variable $Y$ with the highest information gain above a pre-specified threshold $\eta$ (if there is one) is added to the parents of $S_i'$ (and thus to $S_i'^{+}$), and the SOMN modeling the joint distribution of $S_i'^{+}$ and $Y$ is removed from the set of SOMN models. Then, each of the other candidate distributions for $S_i'$ is extended to incorporate $Y$ by extending the mean vectors and covariance matrices of each component in each model by one dimension. The values of the parameters for the new dimension can be initialized in various ways. We describe the method we used in our experiments in Section 5. Algorithm 1 shows the pseudo code for our approach.

The reader may wonder why one would not just maintain $n$ SOMN models, each modeling the joint distribution between a given $S_i'$ and all of its possible parent variables, and then simply evaluate the information between its current set of parents and each other variable by using the appropriate marginal distributions. The reason we do not do this is that as the dimensionality of the SOMN models increases, the accuracy of the density estimate becomes more difficult to maintain with relatively few mixture components. This is simply a consequence of the curse of dimensionality. The idea behind maintaining a larger number of low-dimensional models is to keep the number of mixture components necessary for an accurate estimate at a reasonable number. This is justified to some degree by the assumption that the environment we are trying to model does in fact contain structure in its dynamics, and that the number of parents of any given $S_i'$ will in general be much smaller than the total number of state and action variables.

We would like to note, as we mentioned above, that our choice of the MOG model to represent the factors of the transition function $P$ results in a very simple method for obtaining the conditional probability of a state $\mathbf{s}'$ given the previous state $\mathbf{s}$ and action $\mathbf{a}$. Note that for a multivariate Gaussian random variable $X = [X_1\ X_2]^T$ as defined in the example in Section 3, the conditional probability of $X_1$ given $X_2$ is also a multivariate Gaussian with mean $\mu_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2)$ and covariance $\Sigma_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$, where $\mu$ and $\Sigma$ and their components are also defined as they were in the example in Section 3. Thus, since each $S_i'$ is modeled jointly with its parents in the DBN by a MOG, one can obtain each component of $\mathbf{s}'$ given $\mathbf{s}$ and $\mathbf{a}$ by conditioning on its parents in this manner.

## 5 Experiments

We evaluated our approach on a structured environment which was actually multiple, independent instantiations of a single MDP whose state and action spaces were conglomerated into a single FMDP. The replicated MDP was a continuous "grid-world" with two state dimensions (horizontal and vertical position) and two action dimensions (horizontal and vertical movement). The state dimensions ranged from 0 to 1 and the action dimensions from $-0.1$ to $0.1$, Each action dimension changed the position of the agent in the appropriate dimension by its amount plus some mean-zero Gaussian noise with $0.01$ standard deviation. All action dimensions were executed concurrently and so each action was vector-valued.

The state (action) vector for the full FMDP was constructed by concatenating the state (action) vectors of each MDP into a single vector. We then also added three dimensions to the resulting state vector that were independent of the dynamics of any of the component MDPs, and three action dimensions that had no effect on any of the state dimensions of the full FMDP. The three added state dimensions output at each time step, respectively, a random value in $[0, 1]$, a constant value $(0.5)$, and a value (initialized to 0) that added Gaussian noise to it's previous value (with a mean of $0.05$ and a variance of $0.001$) and that wrapped around to 0 when the value reached 1. These extra state dimensions were just a few arbitrary ways of adding independent dimensions to the FMDP. The values of all state and action dimensions for each of the MDPs were normalized to be in $[0, 1]$ when provided as training samples to the individual SOMN models.

In both experiments we initialized the SOMN models with 9 mixture components arranged in a regular grid over $[0, 1]^2$ and set the initial covariance matrices to $0.3\mathbf{I}$. When distributions were extended by a dimension, we set the values of the mean-vectors for the new dimension to be evenly spaced over $[0, 1]$ and set the last row and column of the covariance matrix to be all zeros, but with 0.3 in the last position. We set the threshold $\eta$ to 3.0 in both experiments. Figure 2 shows the number of correct dependencies as a function of time step for the first environment, averaged over 30 runs. The curve shows that our algorithm was able to discover the correct structure of this environment in a reasonably short period of time. No incorrect dependencies were added by our algorithm at any point.

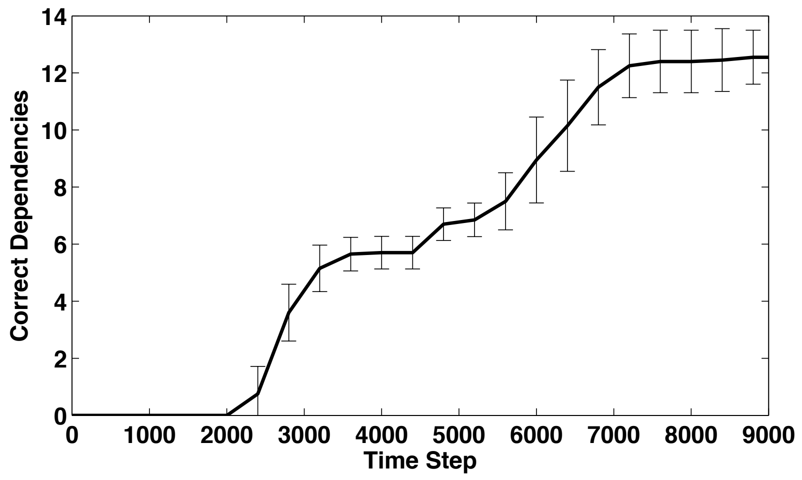

Figure 2: Results for the 3-MDP domain.

## 6 Discussion

We have presented a method for online, incremental learning of transition models of FMDPs with continuous state and action spaces. We use incremental density estimation techniques to model the factored transition function and information theoretic principles to add dependencies to a DBN model of the FMDP. Our experimental results show that our approach is able to discover the correct structure of a non-trivial, continuous, structured environment efficiently.

One limitation of our method is the use of the SOMN model, which requires a pre-specified number of mixture components to model a given density. A possible remedy for this is the incorporation of methods used in other SOM approaches which allow the number of units in the map to vary as new data are received [13]. If successful, each mixture model need only use as many components as necessary to obtain an accurate density estimate, potentially reducing computation time.

The complexity of our approach is quadratic in the number of state and action dimensions, which will obviously pose a problem in very high-dimensional environments. It is possible, however, to incorporate prior knowledge about an environment either in the form of pre-specified dependencies or, less restrictively, constraints on the set of candidate variables considered. This could potentially reduce the computational load significantly. Additionally, the calculation of mutual information need not be done at every time step—only when one desires to evaluate candidate variables.

Our approach may be used in a feature selection setting for value function approximation in reinforcement learning, particularly in the case of learning abstract, temporally extended actions, or options [14]. Although traditionally much of the work on options has assumed the same state representation for each option in a given MDP, recent work has focused on the scenario in which each option has its own state abstraction [2, 15]. The possibility of state abstraction not only reduces the difficulty of learning a given option by reducing the number of variables over which the value function must be supported, but also increases the efficiency with which an agent learns to use an option by generalizing its value across states that differ along irrelevant dimensions.

The structural dependencies learned by our algorithm provide the subset of observation and action dimensions relevant to manipulating a particular set of dimensions in the environment. If the objective of an option is to set such a set of dimensions to a value in a specific range, then our approach provides an appropriate (reduced) subspace of the state-action space over which a value function may be approximated. That subspace will likely be significantly smaller than the full state-action space, greatly reducing the difficulty of learning that option.

**Acknowledgments**

The authors would like to thank Scott Kuindersma, George Konidaris, and Pippin Wolfe for their useful comments and discussions. The work presented here was supported by the National Science Foundation under Grant No. IIS-0733581. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] C. Boutilier, R. Dearden, and M. Goldszmidt, "Stochastic dynamic programming with factored representations," *Artificial Intelligence*, vol. 121, no. 1, pp. 49–107, 2000.

[2] A. Jonsson and A. G. Barto, "Causal graph based decomposition of factored MDPs," *Journal of Machine Learning Research*, vol. 7, pp. 2259–2351, 2006.

[3] ——, "Active learning of dynamic bayesian networks in markov decision processes," in *Lecture Notes in Artificial Intelligence: Abstraction, Reformulation, and Approximation - SARA 2007*, vol. 4612, 2007, pp. 273–284.

[4] T. Degris, O. Sigaud, and P.-H. Wuillemin, "Learning the structure of factored markov decision processes in reinforcement learning problems," in *The 23rd Annual International Conference on Machine Learning*, Pittsburgh, Pennsylvania, 2006.

[5] A. L. Strehl, C. Diuk, and M. L. Littman, "Efficient structure learning in factored-state MDPs," in *Proceedings of the Twenty-second American Association of Artificial Intelligence*, Vancouver, British Columbia, 2007.

[6] B. Kveton, M. Hauskrecht, and C. Guestrin, "Solving factored MDPs with hybrid state and action variables," *Journal of Artificial Intelligence Research*, vol. 27, p. 153201, Oct. 2006.

[7] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, pp. 142–150, 1989.

[8] D. M. Titterington, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. New York: Wiley, 1985.

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[10] H. Yin and N. M. Allinson, "Self-organizing mixture networks for probability density estimation," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 405–411, Mar. 2001.

[11] A. Renyi, "On measures of entropy and information," in *Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1961, pp. 547–561.

[12] A. P. Wolfe and A. G. Barto, "Decision tree methods for finding reusable MDP homomorphisms," in *National Conference on Artificial Intelligence (AAAI)*, Jul. 2006.

[13] H. Xiong, M. N. S. Swamy, M. O. Ahmad, and I. King, "Branching competitive learning network: A novel self-creating model," *IEEE Transactions on Neural Networks*, vol. 15, no. 2, Mar. 2004.

[14] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[15] G. D. Konidaris and A. G. Barto, "Efficient skill learning using abstraction selection," in *International Joint Conference on Artificial Intelligence (IJCAI)*, Jul. 2009.