Machine Learning, 49, 325–346, 2002

Structure in the Space of Value Functions

DAVID FOSTER* djfoster@mit.edu
Centre for Neuroscience, University of Edinburgh, Edinburgh, UK; Gatsby Computational Neuroscience Unit, London, UK

PETER DAYAN dayan@gatsby.ucl.ac.uk
Gatsby Computational Neuroscience Unit, London

Editor: Satinder Singh

Abstract. Solving in an efficient manner many different optimal control tasks within the same underlying environment requires decomposing the environment into its computationally elemental fragments. We suggest how to find fragmentations using unsupervised, mixture model, learning methods on data derived from optimal value functions for multiple tasks, and show that these fragmentations are in accord with observable structure in the environments. Further, we present evidence that such fragments can be of use in a practical reinforcement learning context, by facilitating online, actor-critic learning of multiple goals MDPs.

Keywords: dynamic programming, value functions, reinforcement learning, unsupervised learning, density estimation, mixture models

1. Introduction

Hierarchical structures have recently been the focus of substantial work in the field of reinforcement learning (Watkins, 1989; Singh, 1992; Dayan & Hinton, 1993; Kaelbling, 1993; Sutton, 1995; Thrun & Schwartz, 1995; Dietterich, 1998; Hauskrecht, 1998; Hauskrecht, Meuleau, Boutilier, Kaelbling, & Dean, 1998; Parr, 1998; Parr & Russell, 1998; Precup & Sutton, 1998; Precup, Sutton, & Singh, 1998; Sutton, Precup, & Singh, 1998; Moore, Baird, & Kaelbling, 1999). Decompositions, in some of a whole variety of forms, are seen as being critical for solving large and complex control problems, an insight borrowed and extended from the literatures on traditional planning methods in artificial intelligence (e.g. Fikes, Hart, & Nilsson, 1972; Tate, 1977; Korf, 1985; Currie & Tate, 1991) and on stochastic optimisation in large control problems (e.g. Forestier & Varaiya, 1978).

Although the case can be made that decompositions are always important for solving large-scale Markov decision problems (MDPs), they are compelling even on the small-scale when many similar problems must be solved. For instance, consider a maze task in which the only difference between different problem instances is the location of the goal (this is often called a 'multiple-goals MDP'). Information relevant for getting to one goal is likely to be useful for getting to other goals. In particular, a decomposition that reflects

*Author is currently at the Center for Learning and Memory, Dept. of Brain and Cognitive Sciences, Massachusetts Institute of Technology, E 18-366, Ames Street, Cambridge, MA 02139.

the underlying structure of the maze will facilitate learning, even to a novel goal location. In this paper, we consider methods for decomposing multiple-goals MDPs.

Much of the recent work on hierarchies and decomposition in reinforcement learning has focused on *temporal abstraction*, in such forms as macro-actions, options, and hierarchical abstract machines (HAMs). Underlying them all is the idea is that it is inefficient always to plan at the finest level of granularity of actions. In many circumstances, more abstract abilities (such as knowing how to traverse some particular room in a maze optimally, or at least having some information about how *not* to do so) can be used in order to reduce significantly the complexity of planning, and, in particular, the amount of exploration required to learn the task well. Temporal abstraction has its most direct effect on *policies*, for instance by specifying as elements, complex, multi-step actions.

The main alternative to temporal abstraction is *structural abstraction* (also known as state abstraction or aggregation). In conventional forms of this, groups of states are lumped together in some manner so that the effective size, and thereby the complexity, of the optimisation problem is reduced. Structural abstraction has been particularly well studied in the case that the Markov decision problem admits a formal decomposition, for instance if the transition matrices can be represented by a collection of sparsely connected belief networks. Boutilier, Dean, and Hanks (2000) provide an excellent review of the way that structural abstraction provides a link between notions of planning in artificial intelligence and optimal control in MDPs.

Structural abstraction can be useful for things other than just throwing away all the differences between functionally similar states (this might be called *lossy* state aggregation). In this paper, we ultimately consider using structural decompositions as a way of specifying multi-resolution representations of state, for use in temporal difference learning. Provided that the more abstract levels in the hierarchy are used to *augment* rather than *replace* a basic representation of state (i.e. lossless state aggregation), they may offer the benefits, in terms of generalisation, of more rigidly hierarchical learning methods (such as feudal Q-learning; Dayan & Hinton, 1993), without incurring the cost of partial observability that is often associated with throwing away state information.

Structural abstraction as presently practised mostly works in a top-down manner, starting from a formal decomposition of the MDP. In this paper we consider it from a bottom-up perspective, unleashing *unsupervised learning* on collections of optimal value functions. As has been well recognised before (e.g. Drummond, 1998), value functions, particularly for a few different goals, contain substantial information about the functional texture of the underlying MDP, amounting, in many cases, to an appropriate structural decomposition of the problem. Probabilistic methods in unsupervised learning are concerned exactly with finding such structure inherent in a collection of input examples, and might thus be appropriate as a bottom-up method of decomposition.

Section 2 describes the families of multiple-goals navigation problems that we use to illustrate our analysis; Sections 3 and 4 present the probabilistic models we use to capture structures in value functions together with results demonstrating their utility. Section 5 considers an extended use of this same structure in a task involving a succession of novel goals. Finally Section 6 considers the significance of these results.

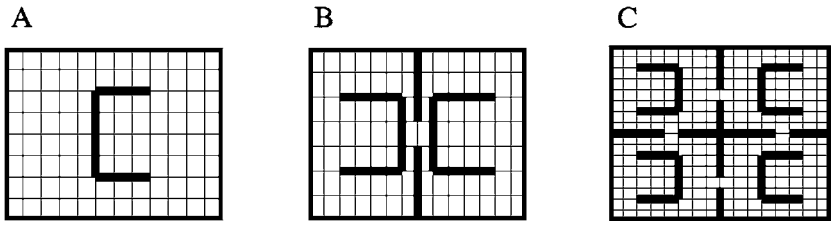

Figure 1. Simple mazes. The thin grid lines outline the states the agent can occupy; the thick lines show impassable walls. Manhattan moves (North, South, East and West) are permitted, but transitions are only stochastically successful.

2. Multiple-goals MDPs

Figure 1 shows three examples which we use to illustrate the underlying ideas. Each involves a simple discrete grid-world (with 96, 98 and 256 states), although their structures are rather different. Note that mazes are by no means the only MDPs susceptible to our methods—they are, however, the ones for which it is easiest to appreciate the resulting structural decompositions. In the mazes, Manhattan moves are possible everywhere except at the boundaries and across the barriers (thick lines). *Stochasticity* affects transitions in that legal action choices (i.e. those that do not attempt to cross barriers or go outside the maze) succeed 80% of the time, and lead to a randomly chosen legal action 20% of the time. Illegal action choices also lead to a randomly chosen legal action.

The key point about multiple-goals MDPs is that any of the states can be a goal—i.e. we are interested in solving many similar MDPs. The difference between different MDPs lies in the reward structure and in the dynamics of what happens at the goal, which is absorbing. We used costs $r(x, a)$ of 1 for each move from state x using action a to a non-goal state, and 0 to a goal state. Figure 2 shows optimal value functions for the three mazes for two different locations of the goal. The optimal value function for state x when the goal is g is

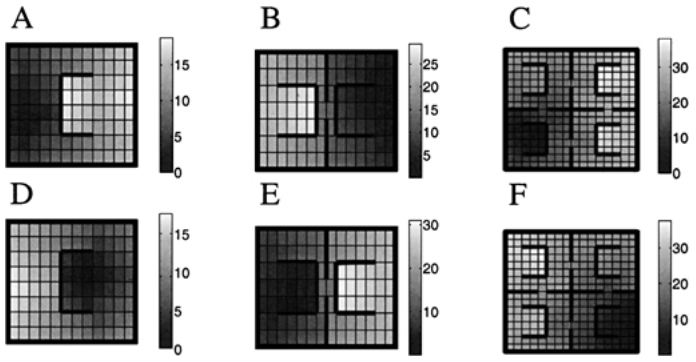

Figure 2. Grayscale value functions for the mazes in figure 1 for two different goals in each case. Note the different scales for each of the mazes. The goals are evident as the darkest points in the mazes.

defined as

$$V^g(x) = \min_a \left\{ r(x,a) + \sum_y P_{xy}(a) V^g(y) \right\}$$

where $P_{xy}(a)$ is the transition matrix reporting the probability of state y following state x, after applying action a (and, strictly, should be written $P^g_{xy}(a)$, given that the process absorbs at the goal g). The optimal value function, in conjunction with $r(x,a)$ and $P_{xy}(a)$, can be used to determine optimal actions at each state.

Figure 2 shows clearly how the optimal value functions lay bare the underlying structure of the environments. For each pair of value functions, even though the values within each segment of the maze are quite different, the underlying discontinuities across the barriers are the same. This is why unsupervised learning of the value functions should extract appropriate structural decompositions, even without a priori knowledge such as that the mazes are posed in two dimensional environments.

In common with many other applications of unsupervised learning, it can be hard to evaluate the fragmentations that emerge. If the state space can be seen in some convenient two dimensional representation, like a maze, then the fragments can certainly be inspected. For instance, figure 4 (which is explained below) shows one simple fragmentation of the maze in figure 1(B) into four pieces $\pi_i(x)$ for $i = 1 \ldots 4$. The pieces can be seen to conform closely to the apparent structure of the environment.

However, because we are interested in using the fragmentations to enhance the representation of state in reinforcement learning, we have a way to test their utility directly. We do this by solving the multiple-goals MDPs using temporal difference (TD) learning and an actor-critic architecture (Barto, Sutton, & Anderson, 1983; Barto, Sutton, & Watkins, 1990). We train a separate actor-critic on each of the MDPs defined by a set of novel goals, using either a conventional, table look-up state representation, or an *augmented* representation resulting from having both conventional states and fragment membership as inputs to the system.

3. The flat model

We consider the task of learning abstractions as one of unsupervised learning of the structure inherent in the set of *optimal* value functions $\mathcal{D} = \{V^g(\cdot)\}$ for a collection of goals g. A large fraction of modern methods of unsupervised learning is based on maximum likelihood estimation in parameterised probabilistic models, i.e. finding the parameters θ^* that maximise

$$\log \mathcal{P}[\mathcal{D}; \theta] \tag{1}$$

under a model $\mathcal{P}[\mathcal{D}; \theta]$ for the probability density or probability distribution over $\mathcal{D}$. In our case, the parameters θ include one group (called **a**) devoted to goal-*independent* fragmentation, and another (called $\mathbf{e}^g$) to goal-*dependent* values.

There is no unique best probabilistic model, since value functions do not really come from the sort of probabilistic generative processes underlying our models. Different models effectively express different prior expectations about how structure might appear. In this and the next section, we show how some very simple probabilistic models can be used to extract the relatively simple structure inherent in the navigation tasks described in Section 2. More sophisticated MDPs will clearly require more sophisticated models. As a particular approximation, the models treat the value functions for different goals as being completely independent, so permitting $\log \mathcal{P}[\mathcal{D}; \theta]$ to be written as

$$\log \mathcal{P}[\mathcal{D}; \theta] = \sum_g \log \mathcal{P}[V^g(\cdot) \mid \theta]. \tag{2}$$

The models also generally treat the values of states for a particular goal as independent, hence:

$$\log \mathcal{P}[\mathcal{D}; \theta] = \sum_g \sum_x \log \mathcal{P}[V^g(x); \theta] \,. \tag{3}$$

In the first model, each element of the value function $V^g(x)$ is treated as an independent sample from a mixture of Gaussians distribution (e.g. McLachlan & Basford, 1988), such that

$$\mathcal{P}[V^g(x); \theta] = \sum_i \pi_i(x) \frac{1}{\sqrt{2\pi\omega_i^g}} e^{-(V^g(x) - e_i^g)^2 / 2\omega_i^g} \tag{4}$$

Here, the different components of the mixture are like spatial *fragments.* The two groups of parameters are the goal-independent **a**, which governs the inclusion of state x within fragment i according to:

$$\pi_i(x) = \frac{e^{a_i(x)}}{\sum_j e^{a_j(x)}} \tag{5}$$

and the goal-dependent values e_i^g and variances ω_i^g which are attached to mixture component i. Although they could also be optimised, in this paper, we fix the variances $\omega_i^g = \omega = 10$ throughout learning.

Under the probabilistic model of Eq. (4), the mean of the value function is

$$\bar{V}^g(x) = \sum_i \pi(x) e_i^g \tag{6}$$

and so this model might be described as a mixture of constants. The variance ω can be seen as capturing the residuals between the true $V^g(x)$ and these mean values.

Fitting the model in this case means maximising $\log \mathcal{P}[\mathcal{D}; \theta]$ with respect to both goal-independent and goal-dependent parameters. However, the fragmentation that results is then given only by the optimal values of the fragmentation parameters **a**. We employ an online,

gradient ascent method for optimising these parameters, each iteration of which uses the optimal values for a goal drawn randomly from a training set of goals (which, in many cases, is far smaller than the set of all possible goals). On each iteration, however, we first choose the optimal set of parameters $\mathbf{e}^{g*}$ which maximises $\sum_x \log \mathcal{P}[V^g(x) \mid \mathbf{a}, \mathbf{e}]$ using a form of the expectation-maximisation algorithm (EM; Dempster, Laird, & Rubin, 1977).

In the E phase of EM, posterior probability or responsibility

$$q_i^g(x) = \frac{\pi_i(x) e^{-(V^g(x)-e_i^g)^2/2\omega}/\sqrt{\omega}}{\sum_j \pi_j(x) e^{-(V^g(x)-e_j^g)^2/2\omega}/\sqrt{\omega}} \tag{7}$$

is calculated for each state x and fragment i. This represents the extent to which fragment i is responsible for $V^g(x)$, given both its prior weight $\pi_i(x)$ and the proximity of e_i^g to $V^g(x)$. In the M phase, the set of parameters $\mathbf{e}^g$ is changed to minimise

$$E(\mathbf{e}^g) = \sum_x \sum_i q_i^g(x) \left(\frac{\left(V^g(x) - e_i^g\right)^2}{\omega} \right) \tag{8}$$

Calculating the global minimum with respect to all the parameters in the M phase is computationally straightforward. E and M phases are alternated for a few (about 10) iterations, which is usually ample to get adequately near convergence. Although the EM steps are guaranteed not to decrease the likelihood (for fixed **a**) there is no equivalent guarantee that EM will find a global minimum—we have not yet tried to use sophisticated initialisation methods or multiple restarts or the like to avoid bad local minima.

Given the optimising values $\mathbf{e}^{g*}$ for the goal-dependent parameters, the fragmentation parameters **a** are changed according to

$$\Delta a_i(x) \propto q_i^{g*}(x) - \pi_i(x) \tag{9}$$

where $q_i^{g*}(x)$ is defined according to Eq. (7) using the values $\mathbf{e}^{g*}$ produced at the end of the EM procedure. One intuition underlying this is that the prior responsibility $\pi_i(x)$ should be the mean value of the posterior responsibilities $q_i^{g*}(x)$, averaging across all the goals—and Eq. (9), is the appropriate method for achieving this. The whole process is repeated for uniformly randomly chosen goal positions from the set. If EM was guaranteed to find essentially the same local minimum with respect to $\mathbf{e}^g$ each time the same goal is selected and a suitably small learning rate was used in Eq. (9), then the likelihood would increase, on average. Of course, EM can, in principle, find different local minima each time, a concern that proved not to present a problem in practice. We initialise $e_i^g = 10$ at the start of each complete optimisation for goal g.

We choose the number of components in the mixture at the outset. However, one major focus of current work in unsupervised learning is model selection, in this case, the automatic determination of the number of components (e.g. Ghahramani & Beal, 2000; Attias, 2000), and these methods would apply directly.

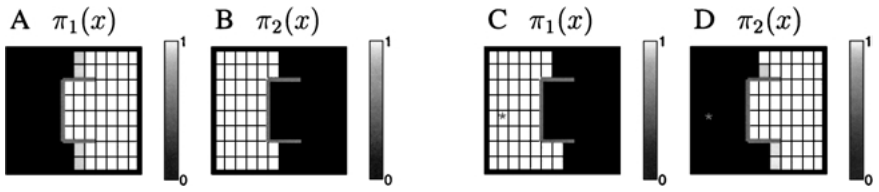

Figure 3. Structural decompositions into two flat regions of the maze in figure 1(A). (A;B) $\pi_1(x)$ and $\pi_2(x)$ when trained on all the goals. (C;D) $\pi_1(x)$ and $\pi_2(x)$ when trained on just one goal (shown by the star). The internal barriers are shown in gray.

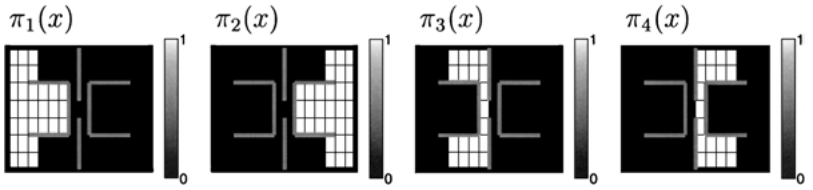

Figure 4. Structural decompositions into four flat regions ($\pi_1(x), \pi_2(x), \pi_3(x)$ and $\pi_4(x)$) of the maze in figure 1(B), trained on all the goals.

3.1. Structure results

Figure 3(A) and (B) show the results of applying the flat model to the maze of figure 1(A) in the case that there are two fragments. The figures show $\pi_1(x)$ (A) and $\pi_2(x)$ (B). The unsupervised learning procedure has taken the states and found a structural decomposition that seems appropriately sensitive to the structure of the environment, taking account of spatial proximity and, where appropriate, the barriers. Although there is no explicit term in the likelihood forcing $\pi_i(x)$ to adopt extreme values, one can see that this is, in fact, what happens. Even though random factors such as the initial values of the parameters can in general affect the solutions, they all closely resemble figure 3.

The structure of this rather simple environment can actually be gleaned from just a single goal. Figure 3(C) and (D) show $\pi_1(x)$ and $\pi_2(x)$ after training on the optimal values for only one goal, indicated by the star. The resulting decomposition closely resembles that found using all the goals.

Figure 4 shows the decomposition that results from applying the flat model with four components to the two-room maze in figure 1(B). Once again, the decomposition is appropriately sensitive to the underlying structure of the task, and can be captured using only 2 goals (figure 5; goal positions shown by the stars). Note that we have provided no prior information about the spatial structuring of states, such as their position within a coordinate frame—all necessary such information is contained within the training values. A diverse enough sample of goals is likely to be desirable, as suggested by the poorer fragmentation found using two goals that were rather close together, in effect occupying the same fragment (figure 6).

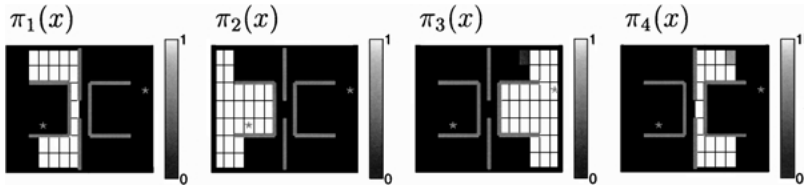

Figure 5. Structural decompositions into four flat regions ($\pi_1(x), \pi_2(x), \pi_3(x)$ and $\pi_4(x)$) of the maze in figure 1(B), using the value functions for only two goals whose positions are indicated by the stars.

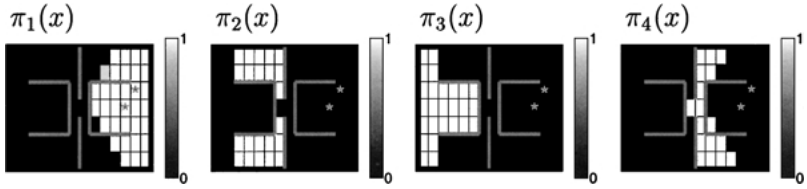

Figure 6. Structural decompositions into four flat regions ($\pi_1(x), \pi_2(x), \pi_3(x)$ and $\pi_4(x)$) of the maze in figure 1(B), using the value functions for two nearby goals (positions indicated by stars).

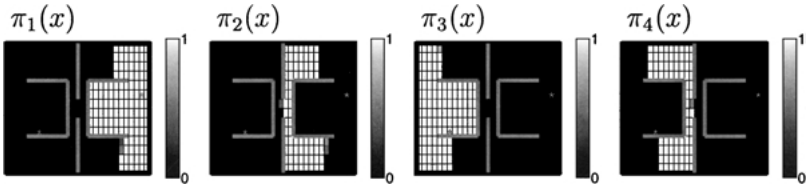

Figure 7. Structural decompositions into four flat regions ($\pi_1(x), \pi_2(x), \pi_3(x)$ and $\pi_4(x)$) of a maze resembling that of figure 1(B) but containing 392 states, again using the value functions for only two goals whose positions are indicated by the stars.

Some insight into the nature of these decompositions can be gained by fitting the same number of fragments to a similarly structured environment that contains four times as many states (392 states). The resulting decomposition is essentially equivalent to that for the smaller state space (compare figure 5 with figure 7), demonstrating that the decompositions scale almost perfectly with the essential complexity of the environmental structure rather than with the size of the state space. The unsupervised learning algorithm we use actually learns more quickly for the larger state space (i.e. in fewer iterations) than for the smaller, indicating that this kind of scaling is not necessarily a problem, although the larger state space clearly uses a greater number of optimal values, the determination of which takes longer than for the small state space.

3.2. Reinforcement learning results

As we described above, we judge the fragmentations partly by using them to augment the state representation for an actor-critic system, comparing performance with that of a

conventional actor-critic, and also with an actor-critic augmented with a randomly assigned fragmentation formed by assigning a random real number between 0 and 0.5 to each $a_i(x)$, which, in turn, lead to random, but normalised, $\pi_i(x)$, through Eq. (5).

For the critic, for each goal, we consider the linear combination of a table-lookup representation (labelled S) and the fragmentation (labelled F):

$$U(x) = u^{\mathrm{S}}(x) + \sum_i \pi_i(x) u_i^{\mathrm{F}} \tag{10}$$

and update the parameters according to the TD(0) rule (Sutton, 1988)

$$\Delta u^{\mathrm{S}}(x(t)) = \epsilon \delta(t) x(t) \tag{11}$$

$$\Delta u_i^{\mathrm{F}} = \epsilon' \delta(t) \pi_i(x(t)) \tag{12}$$

$$\delta(t) = r(x(t), a(t)) + U(x(t+1)) - U(x(t)) \tag{13}$$

where $\delta(t)$ is the temporal difference error, ϵ and ϵ' are learning rate parameters for the table-lookup and the fragmentation respectively. Updates are made in an on-line manner, after every step in the maze. The unadorned and randomly adorned representations are used in just the same way (dropping the fragmentation component from Eq. (10) for the unadorned case).

For the actor, each action choice $a(t) \in A$ is made probabilistically using the Boltzmann-like distribution $P(a(t) = a; x(t)) = \frac{e^{10z(a,x(t))}}{\sum_{a' \in A} e^{10z(a',x(t))}}$, where

$$z(a, x(t)) = z^{\mathrm{S}}(a, x(t)) + \sum_i \pi_i(x(t)) z_i^{\mathrm{F}}(a) \tag{14}$$

the components of which are updated like the critic parameters after every step taken, according to

$$\Delta z^{\mathrm{S}}(a(t), x(t)) = \eta \delta(t) x(t) \tag{15}$$

$$\Delta z_i^{\mathrm{F}}(a(t)) = \eta' \delta(t) \pi(x(t)) \tag{16}$$

where $\delta(t)$ is given by Eq. (13). The learning rates, ϵ, ϵ', η and η', were jointly optimised.

In each simulation run, we average the performance over all goals *except the two goals used to find the fragmentation*. For each goal, all weights are set to zero before 200 learning trials, each starting in a randomly chosen state, and terminating with either the goal state, or a time-out of 5000 steps. Every 10 trials, the mean number of steps is found over 98 non-learning trials, one from each possible state.

Figure 8(A) shows the mean and standard error for 50 simulation runs, revealing that learning is faster when the state representation is augmented with the learned fragmentation (solid line), than without it (dashed line) or with a random fragmentation (dotted line). The significance of this improvement is clearer from figure 8(B) which displays the same data, but in terms of the average *excess* steps to the goal, expressed as a percentage of the number of steps taken by the fully augmented actor-critic. It is notable that such coarse fragments

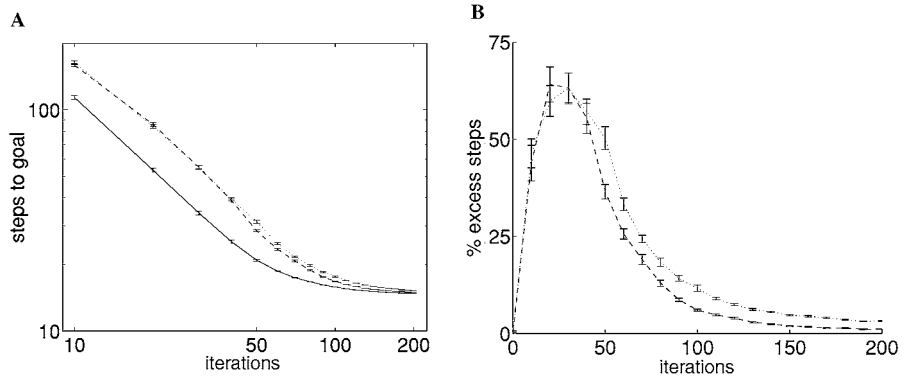

Figure 8. Faster actor-critic learning with the augmented representation. The flat model decomposition of figure 5 (4 fragments trained on 2 goals) was used in the 2-room environment of figure 1(B). (A) The curves show the mean number of steps, measured over non-learning trials for every state and for 96 goals, for the augmented actor-critic (solid), conventional actor-critic (dashed) and an actor-critic augmented with a random fragmentation (dotted). The number of steps taken by a naive actor (i.e. before any learning) is 643 ± 8. (B) The same results shown in terms of the *percent excess steps*, i.e. the number of extra steps taken by the conventional (solid line) and randomly augmented (dotted line) actor-critics, expressed as a percentage of the performance of the correctly augmented fragmentation. The optimised learning rates in each case were as follows, in the format $(\epsilon, \eta; \epsilon', \eta')$ (see text for key): Augmented = (0.742, 2.97; 0.085, 0.0065). Conventional = (0.875, 3.5;-,-). Random = (.928, 3.71; .68, .052).

(as shown in figure 5) learned from only two goals, are nevertheless of considerable value for reinforcement learning.

Further insight into the mechanism by which state augmentation speeds learning can be gained by considering the behaviour of the value function. Figure 9 shows the *mean* values (over 1000 independent simulation runs to the same, novel goal position) at different stages of learning, for an actor-critic adorned with the fragmentation shown in figure 5 (left column) and an unadorned actor-critic (right column). State augmentation serves to transmit value information throughout the state space more quickly—by the end of trial 2, a clear improvement over the unadorned values is already evident. However, because information about individual states remains available, the full value function is flexibly captured, as shown by the virtually indistinguishable value functions at trial 40. By contrast, methods based on lossy state aggregation cannot capture the true value function in this way.

4. The hierarchical model

In many cases, environments are too complicated for their structure to be captured by the sort of flat representation that we have hitherto considered. These mixture models generate *exclusive* decompositions, in which the $\pi_i(x)$ are near 0 or 1 for most states x, and are not substantially distributed. A natural generalisation is to a hierarchical fragmentation in which states are free to belong simultaneously to many fragments at different levels, reflecting, for example, their position in rooms, in particular parts of rooms, and the like.

For simplicity, we consider a 2-level hierarchy only. We use a hierarchical fitting scheme which is similar in some respects to the hierarchical mixture of experts model of Jordan

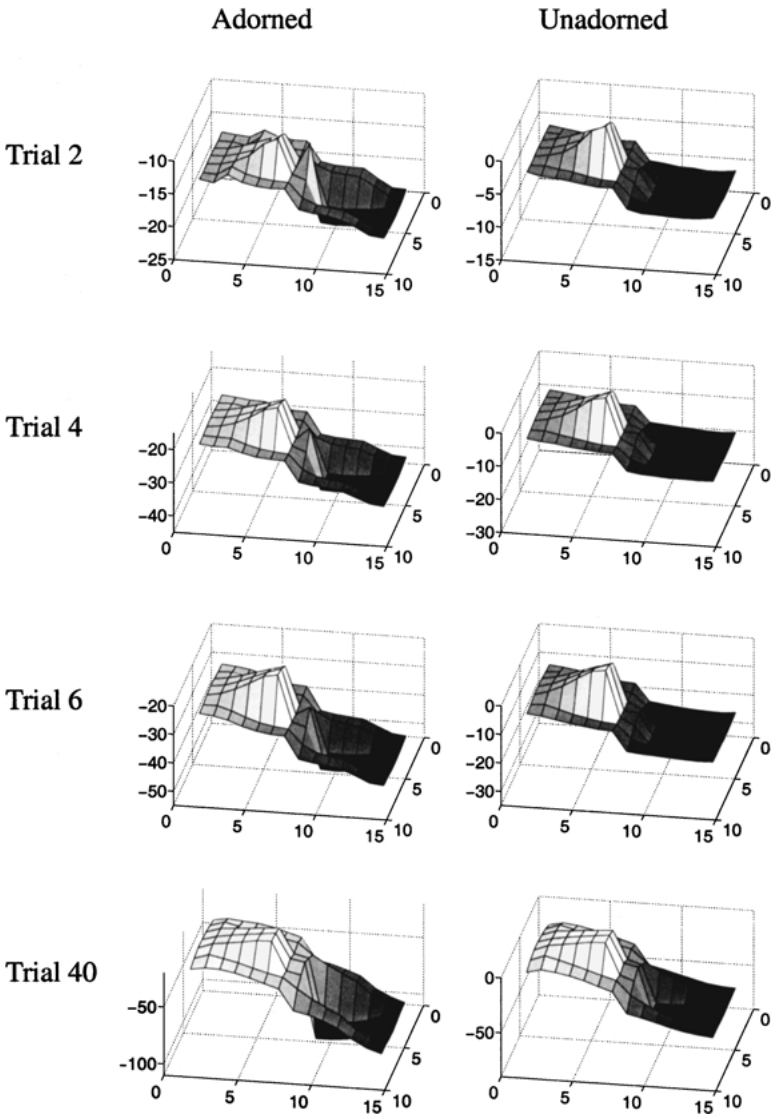

Figure 9. The contribution of the fragmentation to value learning. Shown are the values of states, averaged over 1000 simulated runs to the same goal position, after trials 2, 4, 6 and 40. In the adorned case (left column), value information is transmitted rapidly throughout the environment. In the conventional actor-critic (right column), it spreads out more slowly, only eventually reaching the farthest states. Since we are using a discount factor of 1, the absolute values of the surfaces are irrelevant.

and Jacobs (1993), in that low-level fragment membership probabilities are treated as conditional on higher level ones, but similar in other respects to the pyramidal representation of Burt and Adelson (1983), in that actual value contributions (i.e. experts) are modeled at all levels of the hierarchy. The solution to the problem of fitting a hierarchy of contributing

experts is under-constrained and so, inspired by Burt and Adelson, we first train at the higher level and then train lower levels on the *residual error* from the higher level. The hierarchical model incorporates prior information that the flat model did not: that there should be a certain number of low-level fragments tied to each high-level fragment.

We take the residuals after fitting the best decomposition $\pi_i^*(x)$ and the best parameters e_i^{g*} for each goal g, and find a new fragmentation

$$\pi_{j|i}(x) = \frac{e^{a'_{ij}(x)}}{\sum_k e^{a'_{ik}(x)}} \tag{17}$$

with fragmentation parameters $\mathbf{a}'$ designed to fit the new data set of residual value functions, $U^g = (V^g - V^{g*})$, where

$$V^{g*}(x; \boldsymbol{\pi}^*, \mathbf{e}^{g*}) = \sum_i \pi^*(x) e_i^{g*}$$

The lower level fragmentation is trained using a gradient ascent procedure with an EM inner loop as before. However, in the E phase, the joint posterior probabilities are given by

$$q_{ij}^g(x) = \frac{\pi_i(x)\pi_{j|i}(x)e^{-(U^g(x)-e'^g_{ij})^2/2\omega'}/\sqrt{\omega'}}{\sum_k \pi_k(x) \sum_l \pi_{l|k}(x)e^{-(U^g(x)-e'^g_{kl})^2/2\omega'}/\sqrt{\omega'}} \tag{18}$$

where the low-level fragments are modeled as mixtures of gaussians, with parameters e'^g_{ij} and ω' (here, $\omega' = 7$). Similarly, the marginal probabilities $q_i^g(x)$ and conditional probabilities $q_{j|i}^g(x)$ may be calculated. In the M phase, the set of low-level parameters $\mathbf{e}'^g$ is changed to minimise

$$E(\mathbf{e}'^g) = \sum_x \sum_i \sum_j q_{ij}^g(x) \left(\frac{\left(U^g(x) - e'^g_{ij}\right)^2}{\omega'} \right) \tag{19}$$

In the outer loop, the fragmentation parameters $\mathbf{a}'$ are changed according to

$$\Delta a'_{ij}(x) \propto q_i^{*g}(x)\left(q_{j|i}^{*g}(x) - \pi_{j|i}(x)\right) \tag{20}$$

Again, the whole process is repeated for uniformly randomly chosen goal positions from a training set of goals, which is often smaller than the set of all goals.

4.1. Structure results

Figure 10 shows the consequence of fitting a hierarchy of four flat fragments in the upper level, and eight flat fragments at the lower level (which are trained on the residual values from the upper level), using four training sets, each with a larger number of goals. From the all-goals decomposition, one can see that the natural intuition about an appropriate higher level decomposition—that it should correspond to the four rooms—turns out to be correct. Moreover, although the one goal decomposition is rather different, as the number of training goals is increased, the decomposition clearly comes to resemble the all goals

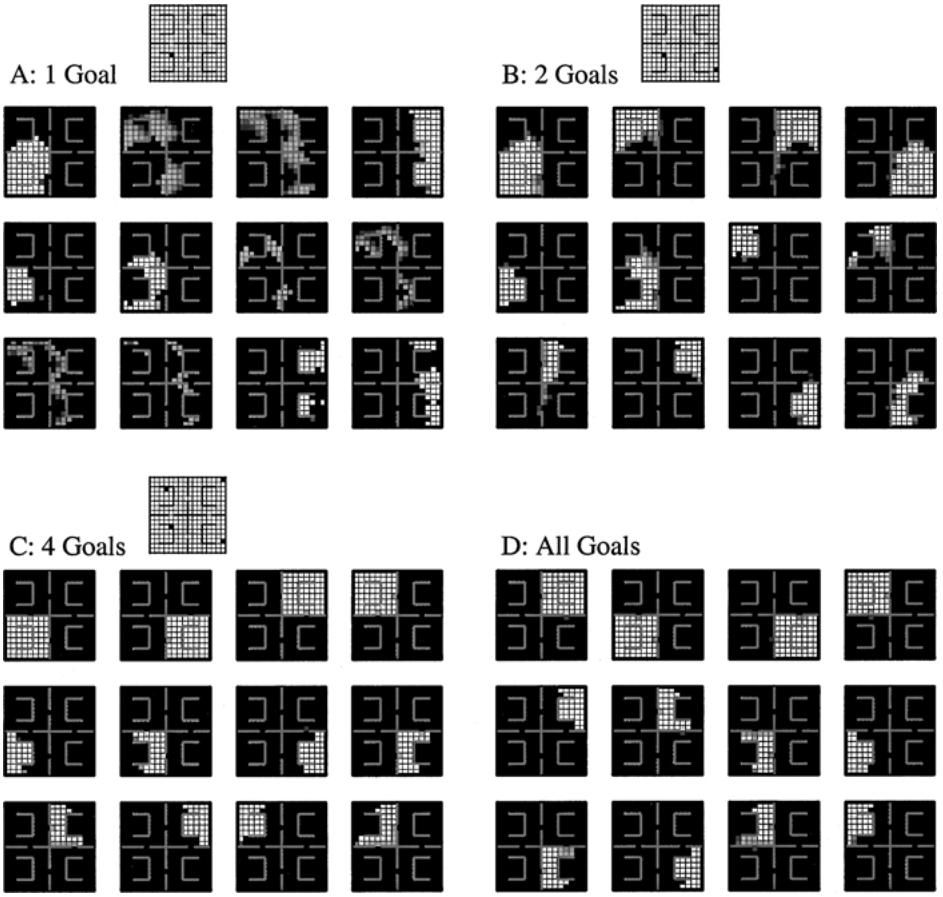

Figure 10. Hierarchical structural decompositions for the four room environment, found using optimal values from one goal (A), two goals (B), four goals (C) and all possible goals (D). In each case, the top row shows the decomposition at the higher level (*rooms*), while the remaining two rows show the decomposition at the lower level (*fragments*). The decompositions from four goals and from all goals closely resemble each other, indicating that such structure as can be found with this method only requires optimal values from a subset of possible goals (4 out of 256). The iconic mazes in A; B; C show the locations of the goals.

decomposition, and moreover does so with a much smaller number of goals (4) than all (256). As before, this number would be expected to increase if more than one goal occupied the same room.

4.2. Reinforcement learning results

Once again, the utility of the decompositions found was tested using an actor-critic scheme, which learns to navigate to all goals within the large environment. For the hierarchical case, the linear combination of table-lookup representation, low-level and high-level fragmentation was considered, effectively adding a third component to the quantities in Eqs. (10)

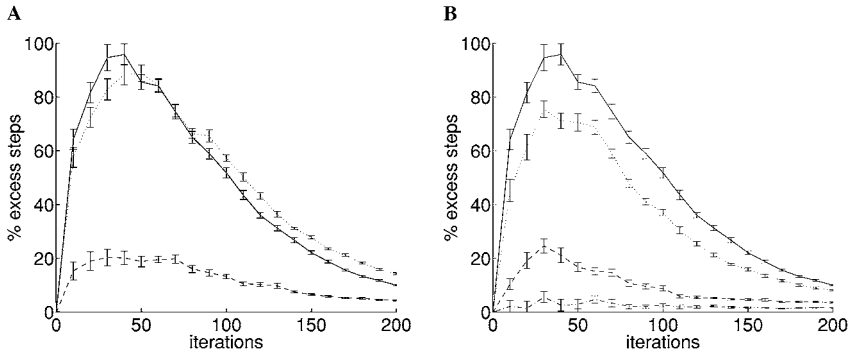

Figure 11. Hierarchical decompositions and actor-critic learning: (A) The percent excess steps, relative to an actor-critic adorned with the fragmentation found using all goals (figure 10(D)), taken by a conventional, unadorned actor-critic (solid line), a randomly, hierarchically adorned actor-critic (dotted line) and an actor-critic adorned with structure found by the flat model fitting 8 fragments to the larger environment (dashed line; flat-model trained on all goals, fragmentation not shown). (B) The percent extra steps, relative to the same baseline as for figure A, taken by a conventional, unadorned actor-critic (solid line), an actor-critic adorned with a hierarchical fragmentation found from just 1 goal (figure 10(A); dotted line), an actor-critic adorned with the fragmentation found from 2 goals (figure 10(B); dashed line), and an actor-critic adorned with the fragmentation found from 4 goals (figure 10(C); dash-dotted line). The optimised learning rates in each case were as follows, in the format (ϵ, η; ϵ', η'; ϵ'', η''): Hierarchical augmentation (same for all numbers of training goals) = (0.74, 2.97; 0.085, 0.0065; 0.037, 0.0012). Flat augmentation = (0.74, 2.97; 0.085, 0.0065;-,-). Conventional = (0.875, 3.5;-,-;-,-). Random hierarchical augmentation = (0.93, 3.71; 0.68, 0.052; 0.074, 0.0024).

and (14). Three sets of learning rates were optimised (ϵ and η, ϵ' and η' for the low-level fragmentation, and ϵ'' and η'' for the high-level fragmentation). Again, all the goals considered were novel (except for the model trained on all goals), since the four goals shown in figure 10(C) were dropped from the training set.

Figure 11(A), showing the result of 25 simulations in each case, reveals that learning is significantly speeded up for the actor-critic augmented with a hierarchical fragmentation, when compared with a conventional actor-critic (solid line), and an actor-critic augmented with a random hierarchical fragmentation (dotted line). Moreover, the importance of the hierarchy can be gauged relative to the performance of an actor-critic augmented with a flat fragmentation of 8 fragments (dashed line).

For the learned hierarchical fragmentation, the degree of speed-up increases with the number of goals upon which the fragmentation was trained (figure 11(B)). This can be related to the structural result evident in figure 10(A) to (D), and provides some evidence for a link between the apparent correctness of the representations and their usefulness for reinforcement learning.

5. Goal generalisation

The efficacy of the structure-finding methods we have presented can be further illustrated by considering generalisation not just between states, but also between goals, in a task

composed of successive navigation problems to novel goals. We consider a combinatorial, look-up table representation for every combination of current location and goal (labelled SS). However, an adorned actor-critic can augment this look-up table with three additional combinatorial fragmentation representations, so that, for example, the value of occupying location x with respect to goal g is estimated as:

$$U(x, g) = u^{\mathrm{SS}}(x, g) + \sum_i \pi_i(x) u_i^{\mathrm{FS}} + \sum_j \pi_j(g) u_j^{\mathrm{SF}} + \sum_i \sum_j \pi_i(x) \pi_j(g) u_{ij}^{\mathrm{FF}} \quad (21)$$

The critic, actor and update rules for each are exactly analogous to those described in Eqs. (11) to (16), with updates to u_i^{FS}, $z_i^{\mathrm{FS}}(a)$, u_j^{SF} and $z_j^{\mathrm{SF}}(a)$ proportional to the fragment membership probabilities in each case; for the FF representation, updates are proportional to the product $\pi_i(x)\pi_j(g)$ for each u_{ij}^{FF} and $z_{ij}^{\mathrm{FF}}(a)$.

We make use of the same decompositions as previously (medium-sized environment, figure 1(B); four flat fragments learned from 2 goals, figure 5). As before, each simulation run involves 96 goal-finding problems, each to a novel goal (with the two goals used to find the fragmentation excluded). However, weights are only set to zero at the beginning of the simulation run, and not between goals, and the choice of goals follows a randomised, non-repeating sequence. Moreover, in order to gauge the extent to which the new combinatorial actor-critic can apply prior experience to novel problems, performance is only measured over the last 76 goal-finding problems (i.e. results from the first 20 goal problems are excluded). The average numbers of steps for both conventional and adorned actor-critic are shown in figure 12(A), though note that learning trials are measured after every trial, the better to observe rapid learning. The rapid learning effect is clearly evident. For comparison

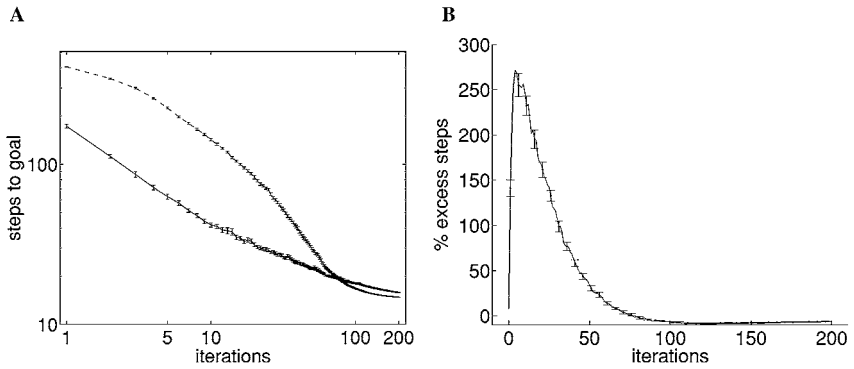

Figure 12. Generalisation to novel goals problems using a combinatorial representation of state and goal, with both components augmented by a learned fragmentation (see text): (A) The number of steps taken by an augmented actor-critic (solid line) and an actor-critic using only a conventional, look-up table representation (dashed line), averaged over 76 novel goals. (B) The same data expressed in terms of the number of excess steps taken by the conventional actor-critic, as a percentage of the number of steps taken by the augmented actor-critic. The optimised learning rates in each case were as follows, in the format (ϵ^{SS}, η^{SS}; ϵ^{FS}, η^{FS}; ϵ^{SF}, η^{SF}; ϵ^{FF}, η^{FF}): Augmented actor-critic = (0.75, 6; 0.005, 0.00125; 0.1, 1; 0.01, 0.1). Conventional actor-critic = (0.75, 3;-,-;-,-;-,-).

with previous results, the same results are shown in terms of the percent excess steps taken by the conventional actor-critic over the adorned actor-critic (figure 12(B)).

6. Discussion

We have presented a new method for extracting structure in MDPs in which unsupervised learning techniques are applied to a collection of value functions, decomposing them into their elemental pieces. The technique is rather general—we illustrated it using some simple multiple-goals MDPs and flat and hierarchical statistical structural models, which we fit using EM-based maximum likelihood techniques. Fragmentations inferred on the basis of only a handful of goals have been shown to offer ways of augmenting conventional representations of state in a form of lossless state aggregation. Such fragmentations significantly speed up simple temporal difference learning methods for finding optimal policies.

The method poses a number of questions. First, there is a formal equivalence between the sort of maximum likelihood modeling we have performed and various methods of minimum description length modeling (MDL; Rissanen, 1989; Hinton & Zemel, 1994). Indeed, Moore, Baird, and Kaelbling (1999) justify their method of temporal abstraction partly by considering the savings in the representation of the optimal policies for all the goals, although they do not count representational cost using MDL methods. In the simplest model of the link between our method and MDL, one might imagine communicating all the value functions by communicating the fragmentation $\mathbf{a}$, then the optimised values $\mathbf{e}^{g*}$ and the residual errors in the model of the value functions $V^g(x)$ for each goal. More sophisticated and cheaper codes involving whole distributions over $\mathbf{e}^g$ would also be conceivable (Hinton & Zemel, 1994; Frey & Hinton, 1997). Good fragmentations involve only a few components, and so only require a few bits to communicate $\mathbf{e}^{g*}$, and offer accurate renditions of the value function, and so only require a few bits to communicate the residual errors. Minimising, or approximately minimising, the complete description length also offers a principled way of optimising the *number* of fragments (Ghahramani & Beal, 2000; Attias, 2000), although we have not addressed this issue directly.

However, describing our method in terms of coding involves a certain prevarication, since MDL is explicitly unconcerned with computational complexity. The true MDL message for the value functions for multiple goals would probably consist merely of a description of the transition matrix and reward structure for the MDP followed by a list of the relevant goals, just enough information so that the receiver itself can perform dynamic programming to calculate the relevant value functions. In fact, using the value functions in order to choose appropriate actions at states requires knowing the transition matrix and reward structure in any case, making the MDL argument even less appealing. Communicating the value functions directly is expensive in terms of bits, but valuable in terms of finding structural decompositions. Finding computational justifications for good representations is a constant theme of unsupervised learning—density estimation is really a surrogate for inference about underlying structure (e.g. Hinton & Ghahramani, 1997), and it can sometimes be hard to do substantially more than what we have done, i.e. indicating the evident appropriateness of structural decompositions.

A more telling point is that the underlying Gaussian statistical models we have used are wrong in two significant respects. Our models explain the correlations in the values $V^g(x)$ of related states x across many goals g by positing the existence of underlying structure. However, they do not capture correlations in the value functions associated with *goals* that are related. Obviously, goals that are nearby give rise to similar value functions—similarities of which we are not taking direct advantage. Thinking in terms of coding, one way to take such correlations into account is to notice that there will be generally be similarities in the optimal values of $\mathbf{e}^{g_*}$ and $\mathbf{e}^{g'_*}$ if goals g and g' are effectively close (i.e. taking barriers into account), and therefore cheaper codes can be generated by taking advantage of this. The structural similarities in these values will be additional sources of structural knowledge about the underlying MDP. However, at least in multiple-goals navigation tasks, this hides the symmetry between states x and goals g, which we take advantage of by using the original fragmentation, i.e. that generated for states, to represent the goal also (Section 5). A further issue when using a combinatorial representation is that the number of state parameters is greatly increased, and includes many which might be irrelevant to learning optimal performance, e.g. a representation of the combination of a currently occupied state and a very distantly located goal state. An extension to the scheme might learn which combinations were most relevant. However, the various ways of treating goal representation remain a topic for future work.

Of course, it would actually be more appropriate to derive the statistical model for representing value functions less according to computational convenience (which favors the sort of simple mixture models we have adopted) and more on the basis of prior expectations as to the underlying MDPs. Doing this in general requires substantially richer prior expectations as to the structure of the tasks. However, in some cases, there may be only limited advantage in incorporating prior expectations of this sort. For example, preliminary studies extending our fragment fitting to the case of *linear* fragments (where the contribution of the fragment is no longer a single value e_i^g, but some function $e_i^g + \mathbf{d}_i^g \cdot \mathbf{x}$ over an assumed underlying coordinate system) indicate that fragments do not greatly differ from those found using the purely flat scheme. Moreover, we have found that applying linear fragments within an actor-critic learning scheme fails to improve performance at all beyond that achieved with flat fragments, because of the difficulty of using linear pieces to learn value functions, a result noted elsewhere (e.g. Boyan & Moore, 1995).

A second problem with the Gaussian statistical models is that they incorrectly suggest that the cost of coding residual errors varies with the square of the residual error. In fact, the exact quantities of the value function $V^g(x)$ are irrelevant—all that matters is that the policies derived from the value function will be correct. In many cases, even just the *ordinal* relationships amongst the values neighboring a state will suffice. One way to address this problem is to use a more sophisticated error model (or quantisation scheme in the context of coding). A slightly more radical departure would be to think in terms of rate distortion theory (see, for example, Cover & Thomas, 1991). This is the extension of standard information-theoretic communication theory to the case that not all the input information (e.g. not all the details of the value function) need be coded, but rather in which there is a quality constraint on some quantity extracted from the code. In this case, the natural quality constraint would

be the expected reward achieved by the policy extracted from the coarsely quantised value function given a distribution over initial states in the MDP.

Thrun and Schwartz (1995) suggest a model for finding temporal abstractions that has something of the flavor of rate distortion theory. They consider communicating *policies* (rather than value functions) for multiple goals, in circumstances under which there are macro-actions (called *skills*), which express whole sets of actions at whole sets of states, and that are intended to be useful for many of the goals. Thrun and Schwartz (1995) construct a single cost function for judging the net policy for each goal, where the policy includes a (possibly stochastic) choice of macro-actions and the truly optimal fine-grain actions. The cost function consists of two terms, one totalling the difference between the optimal values for states and the value for the given policy, the other counting a form of description length for the policy. The more the re-use of the macro-actions for different goals, the less the total description length for the policies, since the actions at many states need not be separately communicated. The relative weighting of the two terms can be seen as a Lagrange multiplier in a rate-distortion theory minimisation problem.

Thrun and Schwartz (1995)'s model has some distinct similarities with the present model. It differs most critically by considering policies rather than value functions, and by treating methods that are designed to give approximations to the optimal policies rather than the exact value function. Further, by not charging any cost for representing the choice between the different macro-actions at states, it has rather little pressure to find the appropriate structural decompositions. It would certainly be possible to use a rather similar method to the one described above to learn structural decomposition by performing unsupervised learning on sets of optimal policies rather than values functions (and indeed, reduced descriptions of value functions such as those consisting only of ordinal relationships amongst states, can be seen as a step in this direction). We considered value functions because they are richer (since they take on real values) and because of the simplicity of not having to consider the possibility of there being multiple optimal actions at each state (a consideration also elided by Thrun & Schwartz, 1995).

Another method which is closely related to ours is that of Drummond (1998). This dissects value functions in two-dimensional mazes using a visual processing algorithm (called the 'snake', Cohen & Cohen, 1993) to search for features such as the discontinuities that mark boundaries in the environment. It then uses the resulting 'rooms' to build a more abstract, graph-based, representation of the underlying MDP. The method constructs an initial approximation to the value function for a new goal by explicitly manipulating (i.e. stretching and rotating) the segments of the value functions for the original goal within each room. Multiple decompositions and value functions are stored, and a matching process is used to find the most appropriate one for a new goal.

Drummond's decomposition method, by relying on a visual processing algorithm that originated to segment two dimensional images, is rather more restricted than our unsupervised learning method, which should be able to find structure in arbitrary problems, and also structure more subtle than just discontinuities. It can also more easily integrate information across multiple value functions to find a single good decomposition, rather than being left with a set of possible decomposition candidates. On the other hand, by directly calculating an approximate value function for a new goal, Drummond's method is rather more ambitious

than ours, since we merely use the fragmentation to augment a standard representation of state for a reinforcement learner. It would be interesting to generalise Drummond's manipulation operators to work in more arbitrary domains which our unsupervised learning method could also handle.

We have described our method on a set of stochastic shortest-path problems, which are the obvious candidates for multiple goals MDPs. However, we believe that useful decompositions also exist for other MDPs, and can be found using unsupervised learning of value functions. Consider, for instance, a task like elevator scheduling (Crites & Barto, 1996) in the cases of different characteristic load patterns (e.g. morning *versus* evening), or busy *versus* quiet periods. Structure in the space of value functions should come from aspects of the task, such as the limit on the number of people each elevator can carry, or the time it takes for an elevator at the top of a building to reach the bottom. Finding this structure should again be useful in generalising the solution for one load pattern to others. Less certain is whether it is possible to take a single large MDP and perform structural decomposition as one finds the single solution. In theory, decompositions can help make exploration more efficient (see, for example, the arguments in Dayan & Hinton, 1993), and one could imagine using unsupervised learning on just a single value function as it changes over learning (rather than a set of optimal value functions for different goals). Demonstrating this is a task for the future.

Although it is valuable to have abstract methods for structural decomposition as a tool for understanding something essential about the underlying tasks, it is interesting to see a way that the resulting abstractions can be at all useful in a practical reinforcement learning context. We studied this issue by using the abstractions to construct augmented representations of the state for a standard on-line reinforcement learner (the actor-critic). To the extent that the structural decompositions carve the MDPs at their true computational joints, augmenting the basic representation of a state with information about how the state fits into the higher level structure of the MDP makes for faster learning. Of course, the fragmentations have been fit to optimal value functions, rather than the suboptimal value functions of approximate policy iteration or asynchronous value iteration that determine the intermediate stages of reinforcement learning solutions. Nevertheless, our results suggest that even these intermediate value functions are reflections of the underlying structure of the environment, since the decompositions are useful for the intermediate as well as the final stages of reinforcement learning. Although we use the actor-critic, there is nothing about the fragmentation that makes it specially well adapted to this form of reinforcement learning. It is a quite general phenomenon in machine learning that good representations make for good (i.e. fast, less overfitting, better generalising) learning, and so we can expect the fragmentations to help other reinforcement learning systems such as Q-learning (Watkins, 1989) and SARSA (Rummery & Niranjan, 1994; Sutton & Barto, 1998).

Augmenting the representation of the state with extra information is rather different from the standard notion of lossy state aggregation, in which acquisition of the optimal policy for an MDP is speeded by *ignoring* distinctions between certain states, even though in the flat model at least, the nature of the extra information is just a single value associated with all the states in a given fragment. Further, although the theory of the effects of state aggregation has been quite well developed (e.g. Singh, Jaakkola, & Jordan, 1995; Gordon, 1996), there is

little theory as to how augmentation can help (or indeed hinder) learning. In our actor-critic investigations, we have not considered some of the recent methods (e.g. see Kivinen & Warmuth, 1997) that have been developed for working out which parts of a representation are relevant towards making predictions, something that should help the course of learning.

The advantage of augmenting a complete representation rather than replacing it is that one avoids falling into the trap of partial observability—although the disadvantage is that the number of states involved in planning, and therefore the direct complexity, has not been reduced. The literature contains few suggestions as to which states should be aggregated—the methods in this paper may be seen as contributions in this direction. From a different perspective, methods such as backpropagation acquire hidden representations of states in the service of learning value functions. The probabilistic unsupervised learning methods in this paper can be seen as systematic approaches along these lines.

The most important theoretical direction for the future is to build models of value functions that are more comprehensively hierarchical. We have found that a hierarchy of the form of Section 4 works well, both as a model for value functions, and as augmented representations. Preliminary results even suggest that appropriately augmented representations offer all the advantages in terms of speed-up of feudal Q-learning, at least in simple maze tasks. Our hierarchy is strict in the probabilistic sense that given the state at the upper level, just one of the lower level units can be selected; it is also slightly unconventional in that the lower level is used to fit residual errors based on the estimate from the upper level. The utility of rich, strict, hierarchies of methods of temporal abstraction such as HAM (Parr & Russell, 1998), MAXQ (Dietterich, 1998) and the airport hierarchy (Moore, Baird, & Kaelbling, 1999) suggests that it is important to widen our scope.

Acknowledgments

We are most grateful to Satinder Singh for critical formative discussions about the idea of unsupervised learning in the space of value functions and helpful suggestions on earlier versions of the paper. We also thank three anonymous reviewers for their most helpful comments and for pointing us to the related work of Drummond (1998). Funding was from the Gatsby Charitable Foundation. The present address of DJF is Department of Brain and Cognitive Sciences, E18-366, Massachusetts Institute of Technology, Cambridge, MA 02139.

References

Attias, H. (2000). Learning structure of latent variable models by variational Bayes. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 13.* Cambridge, MA: MIT Press.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning problems. *IEEE Transactions on Systems, Man, and Cybernetics, 13*, 834–846.

Barto, A. G., Sutton, R. S., & Watkins, C. J. C. H. (1990). Learning and sequential decision making. In M. Gabriel & J. Moore (Eds.), *Learning and computational neuroscience: Foundations of adaptive networks.* Cambridge, MA: MIT Press, Bradford Books.

Boutilier, C., Dean, T., & Hanks, S. (2000). Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research, 11*, 1–94.

Burt, P. J., & Adelson, E. H. (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications, 31*, 532–540.

Cohen, L. D., & Cohen, I. (1993). Finite element methods for active contour models and balloons for 2d and 3d images. *IEEE Transactions in Pattern Analysis and Machine Intelligence, 15*, 1131–1147.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory.* New York, NY: Wiley.

Crites, R. H., & Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing, 9.* Cambridge, MA: MIT Press.

Currie, K. W., & Tate, A. (1991). O-Plan: The open planning architecture. *Artificial Intelligence, 52*, 49–86.

Dayan, P., & Hinton, G. E. (1993). Feudal reinforcement learning. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, 5*. (pp. 271–278). San Mateo, CA: Morgan Kaufmann.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B, 39*, 1–38.

Dietterich, T. G. (1997). Hierarchical reinforcement learning with the MAXQ value function decomposition. In *Proceedings of the 15th International Conference on Machine Learning.* San Mateo, CA: Morgan Kaufmann Publishers.

Drummond, C. (1998). Composing functions to speed up reinforcement learning in a changing world. In C. Nedellec & C. Rouveirol (Eds.), *Lecture Notes in Artificial Intelligence, 1398*, 370–381.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence, 3*, 251–288.

Forestier, J.-P., & Varaiya, P. (1978). Multilayer control of large Markov chains. *IEEE Transactions on Automatic Control, AC-23*, 298–304.

Frey, B. J., & Hinton, G. E. (1997). Efficient stochastic source coding and an application to a Bayesian network source model. *The Computer Journal, 40*, 157–165.

Ghahramani, Z., & Beal, M. J. (2000). Variational inference for Bayesian mixtures of factor analyzers. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 13.* Cambridge, MA: MIT Press.

Gordon, G. J. (1996). Stable fitted reinforcement learning. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, 8.* (pp. 1052–1058). Cambridge, MA: MIT Press.

Hauskrecht, M. (1998). Planning with temporally abstract actions. Report CS-98-01, Department of Computer Science, Brown University, Providence, RI.

Hauskrecht, M., Meuleau, N., Boutilier, C., Kaelbling, L. P., & Dean, T. (1998). Hierarchical solution of Markov-decision processes using macro-actions. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence.*

Hinton, G. E., & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society, Series B, 352*, 1177–1190.

Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems, 6.* San Mateo, CA: Morgan Kaufmann.

Kaelbling, L. P. (1993). Hierarchical reinforcement learning: Preliminary results. In *Proceedings of the 10th International Conference on Machine Learning.* (p. 163). San Francisco, CA, USA: Morgan Kaufmann Publishers.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation, 132*, 1–63.

Korf, R. E. (1985). Macro-operators: A weak method for learning. *Artificial Intelligence, 26*, 35–77.

McLachlan, G. J., & Basford, K. E. (1988). *Mixture models: Inference and applications to clustering.* New York, NY: Marcel Dekker.

Moore, A. W., Baird, L., & Kaelbling, L. P. (1999). Multi-value functions: Efficient automatic action hierarchies for multiple goal MDPs. *International Joint Conference on Artificial Intelligence*.

Parr, R. (1998). Hierarchical control and learning for Markov decision processes. Ph.D. Thesis, Computer Science Division, UC Berkeley.

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 11.* Cambridge, MA: MIT Press.

Precup, D., & Sutton, R. S. (1998). Multi-time models for temporally abstract planning. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 11.* (pp. 1050–1056). Cambridge, MA: MIT Press.

Precup, D., Sutton, R. S., & Singh, S. P. (1998). Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of the 10th European Conference on Machine Learning.* (pp. 382–393). Berlin, Germany: Springer-Verlag.

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. Singapore:World Scientific.

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Singh, S. P. (1992). Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the 10th National Conference on Artificial Intelligence.* (pp. 202–207). Menlo Park, CA: AAAI Press/MIT Press.

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1995). Reinforcement learning with soft state aggregation. In G. Tesauro, D. S. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7.* (pp. 361–368). Cambridge, MA: MIT Press.

Sutton, R. S. (1995). TD models: Modeling the world at a mixture of time scales. In *Proceedings of the Twelfth International Conference on Machine Learning.* (pp. 531–539). San Francisco, CA, USA: Morgan Kaufmann Publishers.

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

Sutton, R. S., Precup, D., & Singh, S. P. (1998). Between MDPs and semi-MDPs: Learning, planning and representing knowledge at multiple temporal scales. Report 98-74, Department of Computer Science, University of Massachusetts, Amherst, MA.

Tate, A. (1977). Generating project networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence.* (pp. 888–893). Cambridge, MA: IJCAI.

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. In G. Tesauro, D. S. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7.* Cambridge, MA: MIT Press.

Watkins, C. J. C. H. (1989). Learning from delayed rewards. Ph.D. thesis, University of Cambridge, Cambridge, UK.

Received March 15, 1999
Revised May 10, 2000
Accepted February 21, 2001
Final manuscript September 24, 2001