What Good are Actions? Accelerating Learning using Learned Action Priors

Benjamin Rosman
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh, UK.
Mobile Intelligent Autonomous Systems (MIAS)
CSIR, South Africa
B.S.Rosman@sms.ed.ac.uk

Subramanian Ramamoorthy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh, UK
s.ramamoorthy@ed.ac.uk

***Abstract*—The computational complexity of learning in sequential decision problems grows exponentially with the number of actions available to the agent at each state. We present a method for accelerating this process by learning action priors that express the usefulness of each action in each state. These are learned from a set of different optimal policies from many tasks in the same state space, and are used to bias exploration away from less useful actions. This is shown to improve performance for tasks in the same domain but with different goals. We extend our method to base action priors on perceptual cues rather than absolute states, allowing the transfer of these priors between tasks with differing state spaces and transition functions, and demonstrate experimentally the advantages of learning with action priors in a reinforcement learning context.**

I. Introduction

A major goal of the artificial intelligence community is the development of autonomous robots capable of performing a wide variety of tasks and operating autonomously for extended periods of time. Such flexibility requires a large repertoire of skills — potentially including both preprogrammed skills, increasingly transferred between robots by standard operating systems such as ROS, and skills acquired by the robot autonomously over the course of its operational lifetime. One way to view the developmental process is that it is primarily concerned with the accumulation of skills.

However, access to a rich palette of possible skills comes with a cost: it can significantly increase the computational complexity of learning and/or planning. The robot is faced with a decision-making problem that has complexity exponential in the number of possible actions in each situation. This additional complexity has the potential to fatally undermine the advantages of either starting with a wide range of skills, or acquiring a large number of skills over time.

Humans face a similar problem. A very large number of behaviours could be invoked by a person at any point in time, and although all are in theory useful, in most cases only a handful of context-relevant actions make sense at that point in time. For example, when confronted with a closed door, one may think of pushing it, pulling it, unlocking it, turning the handle, etc. Although these are all physically possible, the idea of licking the door or punching it are unlikely to occur to the person. It seems likely that people control the computational explosion arising from reasoning through chains of actions by restricting themselves to prioritising small sets of "useful" actions. Which actions are useful in a situation can be determined by introspecting over what has been previously learned in the same or similar settings, and using the past experience of optimal behaviours from many different tasks to guide exploration. This introspection could be a gradual process over a lifetime, allowing the person to become an expert in that particular scenario. A similar phenomenon has been noted in chess [1], where experts are able to quickly limit the set of moves to be considered from any given position; an ability that seems lacking in less experienced players.

This problem is an excellent example of one where a developmental and life-long approach yields benefits. By considering the statistics of action choices over a lifetime, we obtain restrictions to search that may not have been obvious if each task was solved in isolation. Also, as the human example demonstrates, there is reason to believe that this is an important ingredient in the development of expertise. Furthermore, we suggest conditioning this information on local perceptual information, the *context* of the agent, as this is more flexible than relying on global state identification and is known to help facilitate knowledge transfer [2].

We present two methods for learning prior action selection distributions over the actions available to an agent, depending on either the state of the agent or the agent's perception (the context). These priors arise from previous experience in similar settings, and can be used to bias action selection. We pose our problem in the setting of reinforcement learning in discrete Markov Decision Processes. The topic we wish to address is *the extent to which priors over an action set can be learned from other policies in the same environment, and used to accelerate subsequent learning*. This is similar to the idea of ordering actions according to relevance in the domain [3].

Our proposed approach involves combining a set of policies learned in a single domain, by constructing action priors as a Dirichlet distribution. This is a form of lifelong learning, as the priors improve with the number of tasks the agent carries out in that domain. It can also be regarded as a form of transfer

learning [4] because these priors are learned from previous policies, although we are interested in transferring preferences, rather than the policies themselves.

The first method we present conditions the action priors on the state, and the second has them conditioned on the observations of the agent. This allows us to pool the priors from different, yet perceptually similar states to enable faster learning, and allows for priors to be transferred to different domains with different state spaces. Grounding the action priors in perception provides the learning agent with the ability to perform fast look-ups from any known or unknown location, to determine base preferences for action selection. Indeed, this approach is compatible with more sophisticated spatial knowledge representations [5], [6], in that richer observations provide more structure in which to embed these priors.

We demonstrate the advantages of using action priors in a navigation maze experiment, where we show that exploration using action priors in the reinforcement learning paradigm provides significant improvements in convergence speed compared to the standard approach of using uniform priors.

II. Preliminaries

In keeping with the standard formalism of reinforcement learning, we assume an environment specified by a Markov Decision Process (MDP). An MDP is defined as a tuple (S, A, T, R, γ), where S is a finite set of states, A is a finite set of actions which can be taken by the agent, $T : S \times A \times S \to [0, 1]$ is the state transition function where $T(s, a, s')$ gives the probability of transitioning from state s to state s' after taking action a, $R : S \times A \times S \to \mathcal{R}$ is the reward function, where $R(s, a, s')$ is the reward received by the agent when transitioning from state s to s' with action a, and $\gamma \in [0, 1]$ is a discount factor. As T is a probability function, $\sum_{s' \in S} T(s, a, s') = 1, \forall a \in A, \forall s \in S$.

A Markovian policy $\pi : S \times A \to [0, 1]$ for an MDP is a mapping from states to actions. The return, or utility, generated from an episode of running the policy π is the accumulated discounted reward $U^\pi = \sum_k \gamma^k r_k$, for r_k being the reward received at step k. The goal of reinforcement learning is to learn an optimal policy $\pi^* = \arg\max_\pi U^\pi$ which maximises the total expected return of an MDP, where typically T and R are unknown.

Many approaches to learning an optimal policy involve learning the value function $Q^\pi(s, a)$, giving the expected return from selecting action a in state s and thereafter following the policy π, and then using this to improve the policy.

III. Learning Priors over Action Selection

We define a domain by the tuple $D = (S, A, T, \gamma)$, and a task as the MDP $M = (D, R)$ such that the state set, action set and transition functions are fixed for the whole domain, and each task varies only in the reward function. Given an arbitrary set of tasks $\mathcal{M} = \{M\}$ and their corresponding optimal policies $\Pi = \{\pi^*_M\}$, we wish to learn for each state $s \in S$ a distribution $\theta_s(A)$ over the action set, representing the probability of each action in A being used in an optimal policy in the state s, aggregated over tasks. This is then used to prioritise the actions in each state.

To provide an intuition into how this distribution represents action "usefulness", we say that an action a_1 is more useful than an action a_2 in a state s, if a_1 is used in s by more optimal policies than a_2. The setting in which we study the phenomenon of accelerated learning in a lifelong sense is that the tasks seen so far are sampled from a distribution of all possible tasks, and are representative of that task space. By studying the optimal policies that arise from multiple tasks in the same domain, we hope to learn about the structure of the underlying domain.

A. Combining Policies

Consider the setting in which the agent has prolonged experience in the domain D. This means the agent has had to solve a set of tasks in D, and we use the set of optimal Q-functions to extract the action priors as a form of structural information about the domain.

For each state $s \in S$ in the domain, we model the action priors $\theta_s(a)$, $\forall a \in A$ using a Dirichlet distribution [7]. A Dirichlet distribution is a conjugate prior of the multinomial distribution, and so the posterior distribution has a closed-form solution. For each state s, we maintain a count $\alpha_s(a)$ for each action $a \in A$. The initial values of $\alpha_s(a) = \alpha^0_s(a)$ are known as the pseudocounts, and can be initialised to any value by the system designer to reflect prior knowledge. If these counts are the same for each action in a state, i.e. $\alpha_s(a) = k$, $\forall a \in A$ this returns a uniform prior, which results in each action being equally favourable.

The α counts are updated whenever a new Q-function Q^{new} is available:

$$\alpha^{new}_s(a) \quad \longleftarrow \quad \begin{cases} \alpha_s(a) + 1 & \text{if } a = \arg\max_a Q^{new}(s, a) \\ \alpha_s(a) & \text{otherwise.} \end{cases}$$

In this way, $\alpha_s(a)$ reflects the number of times a was considered the best choice of action (leading to the highest return) in state s in *any* Q-function, added to the count priors $\alpha^0_s(a)$.

To obtain the action priors $\theta_s(a)$, sample from the Dirichlet distribution: $\theta_s(a) \sim \text{Dir}(\alpha_s)$. Note that $\theta_s(a)$ is sampled as a probability distribution over A, and so $\sum_a \theta_s(a) = 1, \forall s \in S$.

This process could be viewed as a form of averaging the Q-functions. However, naive averaging is known to be problematic, and often gives detrimental results. Reward functions may, in principle, differ by orders of magnitude, and an averaged policy may not even be feasible. For example, given a state s in front of an obstacle, Q-function Q_1 may suggest moving around the obstacle to the left, while Q-function Q_2 indicates movement to the right. Averaging suggests moving forward, straight into the obstacle. We instead infer that both moving left and moving right are feasible choices to be later explored, whereas moving forward is never the correct choice. The action priors should consequently place more weight on 'left' and 'right' than on 'forward', reflecting the preferences elicited from Q_1 and Q_2. This is learned from experience.

B. Trajectory-based Priors

The assumption that the agent has access to the full policy set Π for the task set $\mathcal{M}$ is a strong one. We can relax this by providing the agent with access to trajectories, rather than full policies, sampled from some policies in the domain. These are preferably near optimal, and may be demonstrated by some domain expert or by the agent itself. This could easily be the case during the process of long term development.

Define a trajectory as a sequence of states-action pairs (with the final action being null): $\tau = (s_0, a_0), (s_1, a_1), \ldots, (s_T, \emptyset)$. Given a trajectory τ^{new}, and using the same Dirichlet model, the counts are updated using

$$\alpha_s^{new}(a) \quad \longleftarrow \quad \begin{cases} \alpha_s(a) + 1 & \text{if } (s,a) \in \tau^{new} \\ \alpha_s(a) & \text{otherwise.} \end{cases}$$

The counts obtained using full Q-functions and those from sampled trajectories are compatible, which results in the agent being able to combine knowledge from both these sources. This is useful in the case of a long-lived agent who may, for example, first be provided with some demonstrations, and later learn policies for various tasks.

IV. Using the Action Priors

An action prior provides the agent with knowledge about which actions are sensible in situations in which the agent has several choices to explore. As a result, they are useful for seeding search in a policy learning process. We demonstrate this with an adaptation of traditional Q-learning [8], called ϵ-greedy Q-learning with State-based Action Priors (ϵ-QSAP), and shown in Algorithm 1. Note, in this algorithm, $\alpha^Q \in [0, 1]$ denotes the learning rate, and should not be confused with the Dirichlet distribution counts $\alpha_s(a)$. The parameter $\epsilon \in [0, 1]$ controls the trade-off between exploration and exploitation. Both α^Q and ϵ are typically annealed after each episode.

Algorithm 1 ϵ-greedy Q-learning with State-based Action Priors (ϵ-QSAP)

Require: action prior $\theta_s(a)$
1: Initialise $Q(s, a)$ arbitrarily
2: **for** every episode $k = 1 \ldots K$ **do**
3: Choose initial state s
4: **repeat**
5: $a \longleftarrow \begin{cases} \arg\max_a Q(s,a), & \text{w.p. } 1-\epsilon \\ a \in A, & \text{w.p. } \epsilon\theta_s(a) \end{cases}$
6: Take action a, observe r, s'
7: $Q(s,a) \longleftarrow Q(s,a) + \alpha^Q[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
8: $s \longleftarrow s'$
9: **until** s is terminal
10: **end for**
11: **return** $Q(s, a)$

The difference between this and standard ϵ-greedy Q-learning can be seen on line 5. This is the action selection step, consisting of two cases. The first case deals with exploiting the current policy stored in $Q(s, a)$ with probability $1 - \epsilon$, and the second case with exploring other actions $a \in A$ with probability ϵ. The exploration case is typically handled by choosing the action uniformly from A, but instead we choose with probability based on the prior $\theta_s(a)$ to shape the action selection based on what were sensible choices in the past. This is aided by the fact that $\theta_s(a)$ is a probability distribution.

The effect is that the agent exploits the current estimate of the optimal policy with high probability, but also explores, and does so with each action proportional to the number of times that action was favoured in previous tasks.

V. Conditioning Priors on Perception

The primary limitation of the state-based action priors described in Section III is that the reuse of these priors is restricted to specific, previously visited states in the same domain D. In order to extend the portability of these priors, we generalise their use by associating them with perceptual features, rather than more specific absolute state information. This is possible by noting that in many cases there is a dependence between priors of different states, possibly in the form of physical constraints or general commonsense behaviour. This enables cross domain transfer, as the priors may be used in different state spaces, provided the observation space remains the same. The assumption is that even though the action priors could be grounded in the state, they actually relate to observations, in that the current observables dictate useful actions, rather than some arbitrary state identification. This change also allows action priors to be used in partially-observable scenarios.

Therefore, instead of learning $\theta_s(a), \forall s \in S$, we learn $\theta_o(a), \forall o \in \mathcal{O}$, where $\mathcal{O}$ is the observation space. In this case, the observations are a function of the state $o(s)$, and depend on the sensing capabilities of the agent. We assume there is redundancy in the observation space, and so the inverse mapping from observation to state is non-unique. The Dirichlet model works just as before, with the equivalent learning process:

$$\alpha_{o(s)}^{new}(a) \quad \longleftarrow \quad \begin{cases} \alpha_{o(s)}(a) + 1 & \text{if } a = \arg\max_a Q^{new}(s,a) \\ \alpha_{o(s)}(a) & \text{otherwise.} \end{cases}$$

Note that the priors from multiple states will map to the same perception-based action priors, increasing the speed of learning. This is because if the same observation is obtained at different states, the action prior information is updated better due to this, as more data is acquired for that observation during a training episode. The assumption we have made is that the world, which is potentially infinite in states, is composed of a finite number of observational patterns.

The perceptual action priors can then similarly be used in a variant of Q-learning. This modification, ϵ-greedy Q-learning with Perception-based Action Priors (ϵ-QPAP), is shown in Algorithm 2.

This variant differs from ϵ-QSAP in that the perceptual action priors $\theta_o(a)$ are used instead of the state action priors $\theta_s(a)$. The effect is that the state space S and state transition

Algorithm 2 ϵ-greedy Q-learning with Perception-based Action Priors (ϵ-QPAP)

Require: perceptual action prior $\theta_o(a)$

1: Initialise $Q(s, a)$ arbitrarily
2: **for** every episode $k = 1 \dots K$ **do**
3: Choose initial state s
4: **repeat**
5: $o \longleftarrow$observations(s)
6: $a \longleftarrow \begin{cases} \arg\max_a Q(s,a), & \text{w.p. } 1-\epsilon \\ a \in A, & \text{w.p. } \epsilon\theta_o(a) \end{cases}$
7: Take action a, observe r, s'
8: $Q(s,a) \longleftarrow Q(s,a) + \alpha^Q[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
9: $s \longleftarrow s'$
10: **until** s is terminal
11: **end for**
12: **return** $Q(s, a)$

function T can be different to those used in the training process, provided the observation space $\mathcal{O}$ remains consistent.

Line 5 refers to obtaining perceptual information (context) from the current state. The observations repeat throughout the domain, whereas the state information is global and unique. The observations used in our experiments simulate simple range scans.

Note that learning still occurs over the state space, but the actions are defined over the observation space. The action priors do not in fact require that the system is fully observable, and so our approach should be easily applicable to partially observable domains. This is because only the perceptual information available to the agent is used, whether or not this is an absolute and complete description of the state of the environment and agent.

VI. Experiments

Spatial navigation is one setting in which we believe an agent stands to make significant gains by using action priors. Our state-based experiment therefore takes place in a 23×23 cell maze where every second row and column is passable space, and the remaining cells are obstacles, creating a lattice. Each task involves a random goal location which has to be reached by the agent, and each episode of a task initialises the agent in a random location in free space. The set of actions available to the agent is to move one cell North, South, East or West. The state available to the agent is the unique number of the cell the agent is occupying. Reaching the goal provides a reward of 100, walking into a wall a reward of -10, and each action taken results in -1 reward.

The nature of this maze world is such that in most cells, only two or three of the available actions are beneficial (do not result in colliding with a wall). Typically, every action is tried with equal probability during learning. However, any optimal policy has already learned to avoid collisions, and it is this knowledge we transfer to a new learning instance.

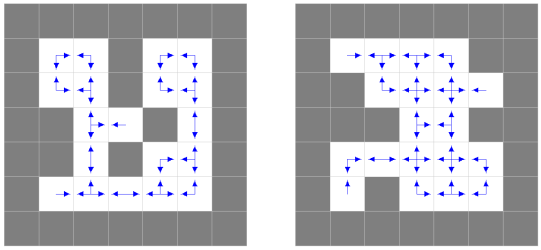

Fig. 1. Example perception-based action priors learned for 7×7 maze worlds, from 50 random optimal Q-functions. The indicated directions in each cell are those with a non-negligible probability mass, but in every cell the agent has the choice of executing any of four directional movements. Grey cells are obstacles, and white cells are free space.

To illustrate perception-based action priors learned from optimal Q-functions, Figure 1 demonstrates the result of using our methods on 7×7 maze worlds (smaller than our actual 23×23 experimental worlds for ease of visualisation), extracted from Q-functions which were the optimal solutions to 50 random tasks. An arrow in a cell is drawn in a direction only if any mass was allocated to that direction by any Q-function. Note that this results in the "useful" actions of the domain, being the actions that do not cause collisions with obstacles. The use of action priors effectively reduces the set of actions from four in each cell to the subset which were useful in the training tasks (55.26% and 63.16% of the full action sets respectively in the examples shown in Figure 1).

The experimental procedure is as follows. We generate a set of tasks in the domain, and allow the agent to learn the corresponding optimal Q-functions. We then extract the action priors $\theta_s(a)$ from these, using the method described in Section III, and finally provide the agent with a new set of tasks which use the priors for exploration as discussed in Section IV.

Figure 2 shows the improvement in convergence speed obtained through the addition of action priors. These results were averaged over learning of 10 different tasks, and this speed-up was achieved using the policies obtained from 10 training task optimal policies. One important observation is that the ϵ-QSAP algorithm immediately receives positive return, unlike Q-learning, as it has learned to avoid harmful actions. The results do not include the training times for ϵ-QSAP, which can be considered a once-off overhead which need not be repeated for any future task. Alternatively, if the priors are updated with each new task instance then ϵ-QSAP starts with no advantage, but after 10 tasks achieves the performance benefits shown.

Note that the Q-learning baseline is equivalent to ϵ-QSAP with a uniform action prior. Additionally, the fact that we are modelling the action prior as a Dirichlet distribution means that none of the action probabilities ever decrease to zero (assuming they started with a uniform prior and $\alpha_s^0(a) > 0, \forall s, a$). As a result, an action is never wholly excluded from the action set, and so all convergence guarantees are retained.

The second experiment demonstrates the advantages of ϵ-QPAP in tasks with different state spaces and transition functions. The domain here was a similar 23×23 gridworld, but instead of the lattice configuration, a random percentage

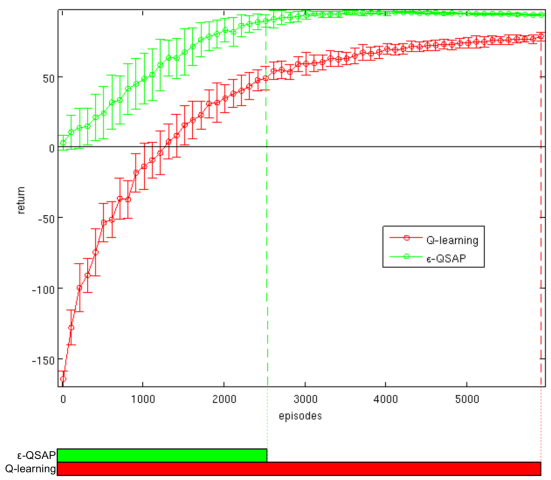

Fig. 2. Comparing the learning rates of Q-Learning with State-based Action Priors (ϵ-QSAP) to Q-learning, in the 23×23 maze domain. The results are averaged over 10 different random tasks. The shaded bars below indicate episodes taken for each method to reach 90% of optimal return. The error bars indicate one standard deviation.

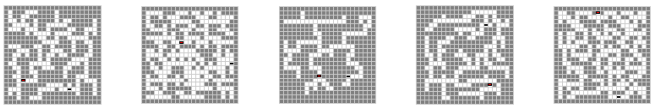

Fig. 3. Example world instances for the second experiment

of the cells are obstacles, while preserving the connectivity of the free space. Example mazes are shown in Figure 3.

In this domain, ϵ-QSAP is at a disadvantage, as the structure of the world differs greatly between trials, and in particular the connectivity of each free cell may be very different (if that cell indeed remains free between different trials). However, the perceptual contexts of the agent remain the same across trials, and so any priors learned for these percepts can be transferred between trials. We consider here simple perception of the occupancy of the 8 cells surrounding the current location of the agent. This simulates a scaled down sonar, and results in a smaller observation space than state space. The results are shown in Figure 4, and clearly show there is no marked difference in performance between Q-learning and ϵ-QSAP, but ϵ-QPAP shows greatly accelerated convergence of learning.

VII. Related Work

Work by Sherstov and Stone [9] has similar aspirations to those of our own methods. In problems with large action sets ($|A| \sim 100$, often from parametrised actions), they also try to either cut down the action set, or bias exploration in learning. The difference is that the reduced action set, or the relevance of an action, is determined from the training data of optimal policies for the *entire* domain, rather than for each state or observation. This has the effect of pruning away actions that are always harmful in the domain, but the pruning is not context-specific.

Other work on learning policy priors [10] also has similar aspirations to our own. They propose a policy search algo-

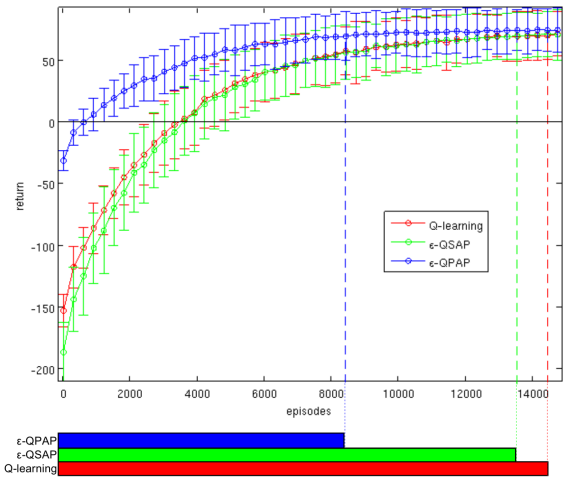

Fig. 4. Comparing the learning rates of Q-Learning with Perception-based Action Priors (ϵ-QPAP), Q-Learning with State-based Action Priors (ϵ-QSAP) and Q-learning, in the 23×23 noise domain, with 50%-80% noise. The results are averaged over 10 different random tasks. The shaded bars below indicate episodes taken for each method to reach 90% of optimal return. The error bars indicate one standard deviation.

rithm, based on MCMC, which learns priors over the problem domain. The method requires the specification of a hyperprior (abstract prior knowledge) over the prior, with the effect that the method learns priors which it is able to share among states. For example, the method can discover the dominant direction in a navigation domain, or that there are sequences of motor primitives which are effective and should always be prioritised during search.

Learning action priors is related to the idea of dividing a problem (or set thereof) into an agent space and a problem space [11]. The agent space refers to commonalities between the problems, and the problem space is specific to each task. This formulation involves learning a reward predictor which, in a sense, can be used to guide action selection.

Where our approach reduces computation by biasing and restricting the search over action space, similar benefits have been found by only searching over limited aspects of the state space, particularly in relational planning problems. Notable examples include reasoning only in terms of the subset of objects that are relevant for current planning purposes (relevance grounding) [12], or using variables to stand in for the objects relevant to the current actions (deictic references) [13].

Options [14] are a popular formalism of hierarchical reinforcement learning, and are defined as temporally extended actions with initiation sets where they can be invoked, and termination conditions. There are many approaches to learning these [15]. Although there are similarities between learning the initiation sets of options and action priors, they are distinct, in that an initiation set defines where the option *can physically be instantiated*, whereas an action prior describes regions where the option is *useful*. For example, while punching a door may always be physically possible, it would seldom make sense for a robot to do this, but that choice would not be ruled out by

options. Consequently, action priors not only augment options, but are beneficial when using large sets of options to mitigate the negative impact of exploration with a large option set.

One approach to reusing experience is to decompose an environment or task into a set of subcomponents, learn optimal policies for these common elements, and then piece them together [16], possibly applying transforms to make the subcomponents more general [17]. This is the philosophy largely taken by the options framework. Our method differs by discovering a subset of reasonable behaviours in each perceptual state, rather than one optimal policy. Our priors can thus be used for a variety of different tasks in the same domain, although the policy must still be learned. As a result, our method is complementary to this decomposition approach.

VIII. Conclusion

We have shown that by learning priors over actions, an agent can improve performance in learning tasks in a single domain. These priors are learned by extracting structure from the policies (or trajectories) used to accomplish other tasks in the domain. By maintaining these distributions over the action space, exploration is guided towards behaviours that have been successful previously.

Associating action priors with observations has the added advantage of enabling transfer between not just different tasks in the same domain, but different state spaces as well. As a result, the agent can perform a fast query of a situation that has not previously been encountered, and provided it resembles an explored state perceptually, the action priors can be transferred.

The notion of perceptions as used here could easily be extended to encompass topological descriptors of the environment [18]. These richer percepts could then be structured into contexts, by means of topic models [19] or related methods, where each observation is generated from some distribution over topics. By first extracting structural contexts from the environment, these would then act as the seeds for learning action priors.

This approach to learning the usefulness of actions in different situations is model free, as the definition of usefulness is based on previous experience of the agent in the environment, and which behaviours were used to achieve tasks optimally. A more powerful learning algorithm may involve a model based approach wherein action usefulness is defined by some objective measure. This is a topic of our current and future work, where we are working to model this with the regret incurred by not taking an action in a certain state.

These results would be expected to improve with a larger action space. Our results show that this approach can yield a substantial improvement in learning, even in a case with only four actions. In richer environments, with more capable agents, one would expect the benefits of this method to increase.

The same principle demonstrated here in reinforcement learning could additionally be applied to planning, similar to learning initial conditions [20], and the action priors could be used to either prune certain actions or provide a preference list for depth-first search. In either case, this should guide the search toward solutions that have been previously encountered, allowing a deeper focused search with minimal additional computational overhead.

Acknowledgements

This work has taken place in the Robust Autonomy and Decisions group within the School of Informatics, University of Edinburgh. Research of the RAD Group is supported by the UK Engineering and Physical Sciences Research Council (grant number EP/H012338/1) and the European Commission (TOMSY Grant Agreement 270436, under FP7-ICT-2009.2.1 Call 6).

The authors gratefully acknowledge the anonymous reviewers for their helpful and insightful comments.

References

[1] H. A. Simon and W. G. Chase, "Skill in Chess: Experiments with chess-playing tasks and computer simulation of skilled performance throw light on some human perceptual and memory processes," *American Scientist*, vol. 61, no. 4, pp. 394–403, July-August 1973.

[2] M. Stolle and C. G. Atkeson, "Finding and transferring policies using stored behaviors," *Autonomous Robots*, vol. 29, pp. 169–200, 2010.

[3] D. Dey, T. Y. Liu, B. Sofman, and J. A. Bagnell, "Efficient Optimization of Control Libraries," *AAAI*, pp. 1983–1989, 2012.

[4] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[5] H. M. Dee, D. C. Hogg, and A. G. Cohn, "Scene Modelling and Classification Using Learned Spatial Relations," *COSIT-09, Lecture Notes in Computer Science*, no. 5756, pp. 295–311, 2009.

[6] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 712–722, June 2010.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[9] A. A. Sherstov and P. Stone, "Improving Action Selection in MDP's via Knowledge Transfer," *AAAI*, pp. 1024–1029, 2005.

[10] D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum, "Bayesian Policy Search with Policy Priors," *International Joint Conference on Artificial Intelligence*, 2011.

[11] G. D. Konidaris and A. G. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," *Proceedings of the 23rd International Conference on Machine Learning*, pp. 489–496, 2006.

[12] T. Lang and M. Toussaint, "Relevance Grounding for Planning in Relational Domains," *European Conference on Machine Learning*, 2009.

[13] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, no. 1, pp. 309–352, May 2007.

[14] D. Precup, R. S. Sutton, and S. Singh, "Theoretical results on reinforcement learning with temporally abstract options," *European Conference on Machine Learning*, 1998.

[15] M. Pickett and A. G. Barto, "PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning," *International Conference on Machine Learning*, pp. 506–513, 2002.

[16] D. Foster and P. Dayan, "Structure in the Space of Value Functions," *Machine Learning*, vol. 49, pp. 325–346, 2002.

[17] B. Ravindran and A. G. Barto, "Relativized Options: Choosing the Right Transformation," *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[18] B. S. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, September 2011.

[19] D. M. Blei, "Probabilistic Topic Models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, April 2012.

[20] M. D. Schmill, T. Oates, and P. R. Cohen, "Learning Planning Operators in Real-World, Partially Observable Environments," *International Conference on Artificial Planning and Scheduling*, pp. 246–253, 2000.