Context-based policy search: transfer of experience across problems

Leonid Peshkin *pesha@ai.mit.edu*
Harvard Center for Artificial Intelligence
Dworkin Bld. 134, Cambridge, MA 02138

Edwin D. de Jong *edwin@cs.brandeis.edu*
Computer Science Department
Brandeis University, Waltham, MA 02454-9110

Abstract

An important question in reinforcement learning is how generalization may be performed. This problem is especially important if the learning agent receives only partial information about the state of the environment. Typically, the bias required for generalization is chosen by the experimenter. Here, we investigate a way for the *learning method* to extract bias from learning one problem and apply it in subsequent problems. We use a gradient-based policy search method, and look for controllers that consist of a context component and an action component. Empirical results on a two-agent coordination problem are reported. It was found that learning a bias made it possible to address problems that were not solved otherwise.

1. Introduction

Reinforcement learning problems with large state spaces typically require the use of generalization. Any form of generalization implies that choices must be made regarding the similarity of different situations, and any such choices constitute a bias. The success of a particular method for generalization depends on whether this bias is appropriate for the problem at hand [12].

For standard *Markov Decision Processes* (MDPs) the observation of the learner captures all relevant information about the state of the environment. In *Partially Observable* MDPs (POMDPs), the choice of the appropriate action may in principle depend on all previous observations. Thus, the learner has to extract relevant information from the history of its interactions with the environment. This renders the need for appropriate generalization even more pressing.

While in general it may not always be possible to recover all required information about the state of the environment from observations, a certain class of POMDPs can be reduced to MDPs by distinguishing among different *contexts* in which the agent may find itself, and providing the current context as an extra input. A context here refers to a subset of the possible histories of interaction between the agent and its environment, which consist of observations, actions, and rewards. If contexts with the above property can be extracted from the interaction history, then the remaining problem can be addressed using the standard *reinforcement learning* methods. For information about reinforcement learning in general the reader may consult [15, 16, 10].

For problems having the above structure, the optimal policy can be described as a set of behaviors, each of which corresponds to a particular context. In this paper, we suggest that this structure has a potential for meta-learning. If related environments require similar behaviors, while differing in their correspondence between contexts and input observations, then behaviors can be transferred from previously solved problems to more difficult variants of those problems. Thus, we have identified a particular mechanism for extracting useful bias from related problems (see e.g. [5, 4, 2, 3, 6]), and a corresponding class of problems which could benefit from this mechanism. We investigate this method for a simple case, and report empirical results.

[0]The current version and related work available from `http://www.eecs.harvard.edu/~pesha/papers.html`

2. Architecture and Learning Mechanism

Partial observability of the environmental state requires the policy representation to include some form of memory in order to specify the optimal policy. In our case, this is achieved by using a finite state controller (see e.g. [11]). Furthermore, in order to *find* optimal policies for partially observable problems, the learning method must be able to search for policies that use memory. To this end, we employ a gradient-based method for policy search, see [13, 1] for the general method and [14] for the discussion of cooperation without explicit coordination in agents controlled by FSCs.

The main idea of this paper is to search for policies that can be described as a set of contexts and corresponding behaviors, so that the behaviors can be transferred between problems. To make this possible, we construct the agent from two components, see figure 1. The first component, called the *context component*, determines the current context from the interaction history. In general this history may include past observations, actions and rewards. In this particular problem, the context component uses the current observation and the context at the previous time step to determine the current context. The *action component* receives the current context and, optionally, the observations as input, and specifies a behavior or partial policy for each context.

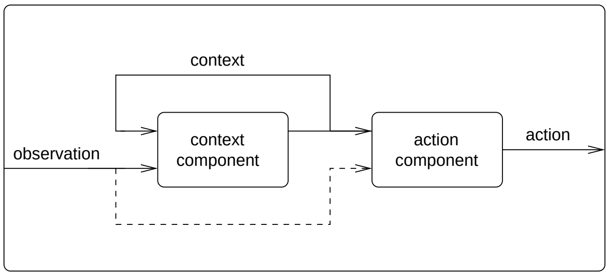

Figure 1. Proposed architecture. The *context component* determines the current context from the previous observation and context. The *action component* uses this information, possibly in combination with the current observation, to produce a corresponding behavior.

Separating the context and action components in the representation of a policy allows these components to be learned separately. Other examples of taking advantage of such controller structure are presented in [9] and [7, 8]. The former uses a controller consisting of two disjoint neural networks, while the latter demonstrates how to use the natural continuity of the Euclidean coordinate system to generalize over the observation space.

In order to transfer experience, we find an optimal controller on a small instance of the problem at hand, and retain the action component of the controller when scaling up to a larger instances of the problem. This provides a natural way to incorporate bias into the learning process.

The finite state controller will now be described in more detail. The controller consists of an internal state transition function that determines the context from observations, and an action function that associates each context with a corresponding behavior.

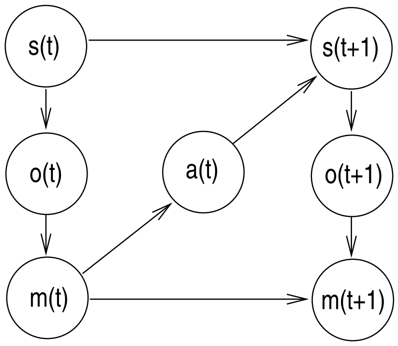

Figure 2. An influence diagram for agent with FSCs in POMDP.

The complete controller for an agent with action space A and observation space O is a tuple $\langle M, \mu_a, \mu_m \rangle$, where M is a finite set of internal controller states (contexts); $\mu_m : M \times O \to \mathcal{P}(M)$ is the internal state transition function that maps an internal state and observation into a probability distribution over internal states; and $\mu_a : M \to \mathcal{P}(A)$ is the action component that maps an internal state into a probability distribution over actions. We assume that both the internal state transition function and the action function are stochastic, that their derivatives exist, and that these are bounded away from zero. Figure 2 depicts an influence diagram for an agent using this controller.

We consider series of problems that require similar behaviors. Thus, by retaining the part of the policy μ_a that specifies the behaviors learned on a small problem, a useful starting point for addressing larger but similar problems is obtained. The context component μ_m on the other hand must be learned anew, since the particular observations that identify each context may vary across different environments. In the following sections, we first describe the task in detail, than experiments are reported that illustrate the advantage of aforementioned learning with bias.

3. Task Description: Block Moving

The metaphor for the task used in the experiments is *"block moving"* which is a multi-agent problem derived from the "load-unload" problem [11, 13]. It requires the cooperation of two agents and constitutes a *coordination problem* in the sense that the agents need to adjust their actions to one another. This produces a non-trivial form of partial observability. In the real-world version of this problem, the activity of the two people lifting a heavy block should be synchronized for satisfactory results, and thus requires coordinating the moment at which to start moving. The information about the state of each agent is exchanged through (possibly non-verbal) communication.

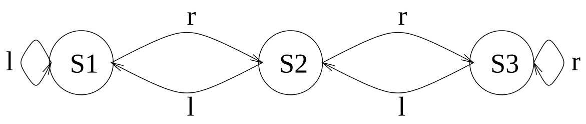

Figure 3. State transition function.

In the simple, abstract task we study here, the process is modeled as follows. Each agent can move independently between several locations (see figure 3 for three locations example). When two agents are in state S3, they automatically load a block. From then on, they must move in synchrony in order not to drop the load. Since the agents perceive only their own location and not that of the other agent, communication is necessary.

The reward function for the task is depicted in figure 4. The states in this diagram are collective states, specifying the state of both the first and second agent (A1 and A2). Connections without arrows mark transitions that can have either direction. The projection of the diagram onto either horizontal axis yields the state transition diagram for a single agent shown above in figure 3.

The reward function not only depends on the current state of both agents, but also on a history of both of their states. The only aspect of this history that is relevant to the reward function is whether the load has been lifted and not dropped since it was lifted. The two collective states with this property are in the upper level of the diagram, marked *loaded.* All remaining collective states are at the lower, unloaded level.

A potential for communication is provided as follows. In addition to the action an agent can select to move between states, each agent has a single bit that it can set or reset. Each agent reads the bits of all other agents, in addition to the sensor information specifying its own location. If both agents would always

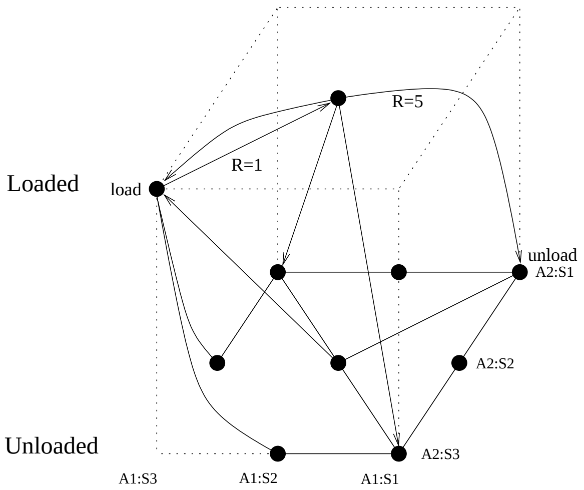

Figure 4. The reward structure for "block moving" domain.

start in state $S1$, they could learn to move to state $S3$ as quickly as possible, and return to $S1$ from there. This ballistic policy would cause them to move in synchrony, without any need for communication. In order to disallow this strategy, the initial position of each agent is selected randomly from all locations except "load".

For the three-location instance depicted in figure 3 the optimal strategy has the following form. Each agent moves to state $S3$, informs the other agent of its presence there and, if necessary, waits for the other agent to arrive. Then each agent moves to $S2$, and from there back to $S1$. If both agents execute this policy, they move together after picking up their load. Thus, by the use of communication, the agents can synchronize their behavior and collect the maximum amount of reward.

Part of the difficulty of this task is that it requires establishing a convention. While it is clear that the only possibility for communication is for each agent to set or reset its bit, the sending and receiving agent must converge on the same choice of which setting to use for which case. Since both agents are equivalent and may function both as sender and as receiver, this problem needs to be solved twice (not necessarily in the same way). A greater difficulty than the selection of such a convention however is that of settling on sending and receiving behavior simultaneously; while the first agent to arrive can only determine whether the second agent has arrived through communication, the second agent can only discover that signaling this information is useful if the first agent acts on it, using the same convention. The necessity to simultaneously arrive at compatible communication and action policies makes

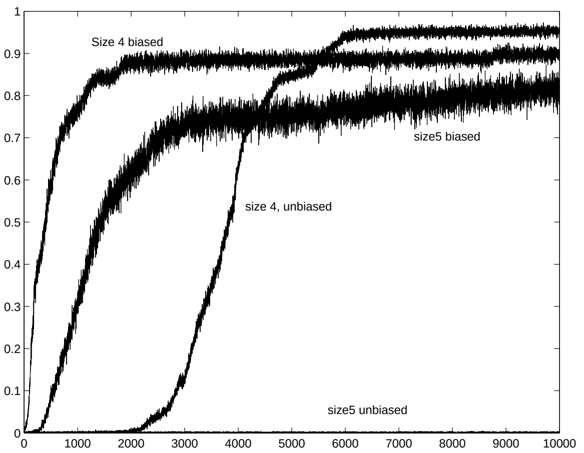

Figure 5. Empirical comparison of biased and unbiased learning by policy search in FSCs for the "block lifting" environment of various size.

the *block moving* problem a challenging one.

The difficulty of the problem strongly depends on the size of the problem instance. A version of the problem with three states has been described, involving a "load" state (S3), an intermediate state (S2), and an "unload" state (S1). To obtain larger versions of the problem, we introduce one or more additional intermediate states. Since the part of the policy space that must be considered grows exponentially with the length of the smallest optimal policy, even adding one or two states makes the problem considerably more difficult.

4. Empirical Results

In our experiments we begin by learning the optimal policy with no initial bias in the instance of the problem with three locations all together, counting **load** and **unload** locations. For this trivial instance of the domain even an exhaustive evaluation of all policies would be feasible, so it comes as no surprise that agents rapidly converge to a good policy. The resulting action function is used as a bias - a starting point for action functions in learning policies for larger instances of the domain.

Figure 5 illustrates the advantage of learning with bias in our domain. All plots are averaged over 10 runs. The learning rate is kept constant. A policy is learned in the space of three-state finite space controllers. For the domain with four locations, both biased and unbiased learning converge, but unbiased does so after a significant delay. For the domain with five locations unbiased learning never picks up for the trial length used. Using the bias learned on a smaller version of the problem makes it possible to learn even on this difficult instance, and turns out to be even more efficient than unbiased learning for a smaller domain with four locations.

In the experiments, the action function from a smaller problem instance was used as a starting point in the learning process, and could in principle be modified. In practice, we found that the action component did not change, and only the context component was learned anew.

5. Discussion

The signal from the other agent in our environment formed a part of the observation. A potential change in the representation of the controller would be to separate the location from the signal and feed the signal part of the observation directly into the action function. This would be justified by the fact that as we increase the size of the domain, the number and the semantics of the messages does not change. The location numbering on the other hand does change, and needs to be learned for every instance. Another alternative approach to the approach that has been investigated would be to fix the action function and only learn internal state transition function. We plan to experiment with some of these variants of the setup in future experiments.

In a sense, an agent that develops a categorization into different situations and learns what behavior to use in each situation, establishes a convention with itself in the form of a mapping between contexts and behaviors. Given a mapping from possible world states to internal states, there is a corresponding mapping from internal states to partial policies that in combination leads to the optimal complete policy.

The issue of arising at a convention becomes more pronounced in problems such as that used here, where multiple agents have the ability to exchange signals. This variant of establishing a mapping between internal states and signals to produce can be seen as a rudimentary form of communication development; while any consistent signaling convention is sufficient to allow agents to produce optimal behavior, and the specific signaling convention that will be used is therefore arbitrary, arriving at such a convention is a difficult coordination problem.

6. Conclusion

We investigated an approach to partially observable reinforcement learning problems where the representation of the policy is separated into a context component and an action component.

Useful bias was extracted from a simple version of the problem by retaining the action component. When put to use on more difficult problems, this method solved problems which an unbiased approach was unable to address. In future work, we hope to explore other mechanisms for extracting bias from learning and utilizing it to aid subsequent learning. We believe this may in principle make it possible to address problems that can not practically be addressed by conventional learning methods.

Acknowledgement

EdJ gratefully acknowledges an NWO Talent-fellowship.

References

[1] Leemon C. Baird. *Reinforcement Learning Through Gradient Descent.* PhD thesis, CMU, Pittsburgh, PA, 1999.

[2] Eric Baum. *Neural Networks and Machine Learning*, chapter Manifesto for an Evolutionary Economics of Intelligence, pages 285–344. Springer-Verlag, 1998.

[3] Eric Baum. Toward a model of intelligence as an economy of agents. *Machine Learning*, 35:155–185, 1999.

[4] Eric B. Baum. Evolution of cooperative problem solving in an artificial economy. *Neural Computation*, 12:2743–2775, 2000.

[5] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[6] Edwin D. De Jong and Jordan B. Pollack. Utilizing bias to evolve recurrent neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, pages 2667–2672, 2001.

[7] G. Z. Grudic and L. H. Ungar. Localizing policy gradient estimates to action transitions. In *Proceedings of the Seventeenth International Conf. on Machine Learning*. Morgan Kaufmann, 2000.

[8] G. Z. Grudic and L. H. Ungar. Localizing search in reinforcement learning. In *Proceedings of the Seventeenth National Conf. on Artificial Intelligence*, 2000.

[9] Dean F. Hougen, Maria Gini, and James Slagle. An integrated connectionist approach to reinforcement learning for robotic control. In *Proceedings of the Seventeenth International Conf. on Machine Learning*. Morgan Kaufmann, 2000.

[10] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–277, 1996.

[11] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie P. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 427–436. Morgan Kaufmann, 1999.

[12] Tom M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University, New Brunswick, New Jersey, May 1980.

[13] Leonid Peshkin. *Reinforcement Learning by Policy Search.* PhD thesis, Brown University, Providence, RI, 2001.

[14] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conf. on Uncertainty in Artificial Intelligence*, pages 307–314, San Francisco, CA, 2000. Morgan Kaufmann.

[15] Christian R. Shelton. *Importance Sampling for Reinforcement Learning with Multiple Objectives.* PhD thesis, MIT, Cambridge, MA, 2001.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, MA, 1998.