Conjugate Markov Decision Processes

Philip S. Thomas PTHOMAS@CS.UMASS.EDU
Andrew G. Barto BARTO@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01002 USA

Abstract

Many open problems involve the search for a mapping that is used by an algorithm solving an MDP. Useful mappings are often from the state set to some other set. Examples include representation discovery (a mapping to a feature space) and skill discovery (a mapping to skill termination probabilities). Different mappings result in algorithms achieving varying expected returns. In this paper we present a novel approach to the search for *any* mapping used by *any* algorithm attempting to solve an MDP, for that which results in maximum expected return.

1. Introduction

Although there have been successful applications of mechanisms for solving Markov decision processes (MDPs) to real-world problems, there remain difficulties. These difficulties include *representation discovery*: finding a feature space conducive to learning; *motor primitive discovery*: finding a low-dimensional action space; and *skill discovery*: the creation of temporally extended actions. Although we adopt reinforcement learning (RL) terminology (Sutton & Barto, 1998) and refer to any mechanism for solving an MDP as an *agent*, we do not assume that the agents use RL.

Maximizing the expected return is the goal of an agent, and therefore also the goal when solving any of the aforementioned problems. However, methods are typically heuristic, in that they attempt to optimize some heuristic that *may* be correlated with the expected return. For example, one might search for feature spaces that best preserve the distance metric of state transitions (Mahadevan & Maggioni, 2007), motor primitives that best recreate observed policies (Todorov & Ghahramani, 2003), or skills that reach bottleneck states (McGovern & Barto, 2001).

Rather than relying on a heuristic optimization to improve expected return, we propose direct optimization of the expected return for these and other problems by phrasing the problem in a general way: the search for a mapping, f, from the state set to some other set or space $\mathcal{U}$. For representation discovery, $\mathcal{U}$ would be the feature space; for motor primitive discovery, it would be the action set of the MDP[1]; for skill discovery, it would be skill termination probabilities (Sutton et al., 1999). Our goal is to find the mapping, f^*, that maximizes an agent's expected return.

We show how an algorithm can take advantage of the structure of the underlying problem to perform an informed search for f^*. By doing so, the search problem itself becomes an MDP, which we call a Conjugate MDP (CoMDP). We then propose that one agent solve the original MDP while a second agent (coagent) solves the CoMDP.

2. Background

Sequential decision problems are often formulated as MDPs, each a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of possible states and actions respectively, $\mathcal{P}$ gives state transition probabilities: $\mathcal{P}(s,a,s') = \Pr(s_{t+1}{=}s'|s_t{=}s, a_t{=}a)$, where t is the current time step, and $\mathcal{R}(s,a,s',r) = \Pr(r_t{=}r|s_t{=}s, a_t{=}a, s_{t+1}{=}s')$ is the reward distribution. $\mathcal{R}$ represents the reward distribution rather than the expected reward to facilitate proofs in the appendix. If $\mathcal{S}, \mathcal{A}$, or $\mathcal{U}$ are uncountable, replace the corresponding probability distributions with probability density functions, summations with integrals, and mixima with suprema. An agent, A, with time-varying parameters $\theta_t \in \Theta$ (typically function approximator weights, learning rates,

[1] We define a motor primitive to be a mapping from a set with one element to a high-dimensional action space. This is equivalent to a *constant map* from the state space to the action space. We expound upon this in Section 8.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011. Copyright 2011 by the author(s)/owner(s).

etc.) observes the current state, s_t, selects an action, a_t, based on s_t and θ_t, which is used to update the state according to $\mathcal{P}$. It then observes the resulting state s_{t+1}, receives uniformly bounded reward r_t according to $\mathcal{R}$, and updates its parameters to θ_{t+1}.

We define $\mathbb{P}$ to be the space of all *policies*: mappings from states to probabilities of selecting each possible action. The agent's parameterized policy is $\pi(\theta)$, $\pi(\theta) : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$, where $\pi(\theta)(s,a) = \Pr(a_t{=}a|s_t{=}s)$, and $\pi : \Theta \rightarrow \mathbb{P}$ is a *parameterized policy generator* (PPG). The agent attempts to find a $\theta \in \Theta$ that approximates[2] a policy, μ^*, called an *optimal policy*, which maximizes the expected return:

$$\mu^* = \arg\max_{\mu \in \mathbb{P}} E\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \mu, d_0\right], \tag{1}$$

where $\gamma \in [0,1)$ is a discount parameter and $d_0(\cdot)$ is the initial state distribution. We only consider problems for which μ^* exist, which precludes some MDPs that have continuous actions. The *value function* for policy μ and MDP M maps states to the expected return from that state if actions are selected according to μ:

$$V_M^\mu(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \mu, s_0 = s\right]. \tag{2}$$

Hereafter, we abuse notation by interchanging $\pi(\theta)$, and θ as superscripts, as they both describe a policy.

We augment this formulation by allowing A access to a mapping, f, from $\mathcal{S}$ to some set $\mathcal{U}$. For generality, we allow the mapping to be probabilistic by making f a conditional probability distribution over $\mathcal{U}$ given the current state s_t, so $f : \mathcal{S} \times \mathcal{U} \rightarrow [0,1]$, where $f(s,u) = \Pr(u_t{=}u|s_t{=}s)$. For convenience, we call f a *probabilistic mapping* from $\mathcal{S}$ to $\mathcal{U}$. We define $\mathbb{F}$ to be the space of all such fs. Thus, A has access to both s_t and $u_t \in \mathcal{U}$ at each time step. A's policy becomes $\pi(\theta) : \mathcal{S} \times \mathcal{U} \times \mathcal{A} \rightarrow [0,1]$ for all $\theta \in \Theta$. We define A's function for updating its parameters given recent observations to be $\Xi : \mathcal{S}\times\mathcal{U}\times\mathcal{A}\times\mathcal{R}\times\mathcal{S}\times\mathcal{U}\times\mathcal{A}\times\Theta \rightarrow \Theta$, so $\theta_{t+1} = \Xi(s_t, u_t, a_t, r_t, s_{t+1}, u_{t+1}, a_{t+1}, \theta_t)$. We place no restrictions on how A uses u_t within π and Ξ. This definition encompasses batch methods because A may store state histories and time counters in θ_t, as well as function approximator weights. Because an agent can be entirely described by its current parameters, PPG, and its update function, we define an agent to be $A_t = (\theta_t, \pi, \Xi)$. The canonical formulation without f is a degenerate case in which $\mathcal{U}$ is a singleton.

[2]We do not concern ourselves with *how* A approximates an optimal policy, as we place no restrictions on A.

If an agent attempts to solve M, it faces a new MDP, which we call the *augmented MDP*, $M^A = \big(\mathcal{S} \times \mathcal{U}, \mathcal{A}, \mathcal{P}^A, \mathcal{R}^A\big)$, where $\mathcal{S} \times \mathcal{U}$ is the new state set, $\mathcal{P}^A$ gives transition probabilities, and $\mathcal{R}^A$ is the reward distribution. We define $\mathcal{P}^A$ to update the $\mathcal{S}$ portion of the state according to $\mathcal{P}$, and to then sample the $\mathcal{U}$ component of the state from $f(s_t, \cdot)$. Similarly we define $\mathcal{R}^A$ to be equivalent to $\mathcal{R}$, where the $\mathcal{U}$ component of the state is ignored. If we vary f during our search, M^A is nonstationary because the transition function depends on f. The value function of M^A is independent of the $\mathcal{U}$ component of its state given the $\mathcal{S}$ component, and thus is the same as that of M.

Although $\mathcal{U}$ is part of the state, we place no restrictions on A, so we do not require it to use u_t as an additional feature. For example, A may use u_t to encode the probability of an exploratory action or the termination probability of a temporally extended action. Similarly, A may ignore the $\mathcal{S}$ component of its state. This lack of constraint on A leaves the responsibility of selecting a reasonable A to the researcher. Here we are interested in finding a good f regardless of the A selected.

3. Problem Statement

Given an MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, called the *base MDP*, and agent $A_0 = (\theta_0, \pi, \Xi)$, our goal is to find a (θ, f) that maximizes the expected return on M:

$$\arg\max_{(\theta,f)\in\Theta\times\mathbb{F}} E_M\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \theta, f, d_0\right], \tag{3}$$

where E_M denotes the expected value assuming the dynamics of M. We constrain the problem by requiring that Expression 3 exist and θ_t only be updated according to Ξ during the search for an optimal (θ, f).

This is a unified description of the problems listed in the introduction: for an agent that uses some probabilistic mapping f, we want to find the optimal f, with optimality defined in terms of expected return. Because applications typically involve continuous state or action spaces, we forgo attempts to search for global optima and instead search for a locally optimal (θ, f).

4. Conjugate MDPs

A common method for solving multivariate optimization problems is grouped coordinate ascent (Bezdek et al., 1987), in which the variables are partitioned into two disjoint subsets, one of which is fixed while the objective function is maximized over the other (which we refer to as a *partial optimization* hereafter), and the process repeated with the other subset fixed. Repeated application of this process guarantees a local maximum if each partial optimization reaches a global maximum (Bezdek et al., 1987).

To apply grouped coordinate ascent to Expression 3, we must provide methods for performing each partial optimization,

$$\arg\max_{\theta\in\Theta} E_M\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| \theta, f, d_0\right], \tag{4}$$

for fixed f, and

$$\arg\max_{f\in\mathbb{F}} E_M\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| \theta, f, d_0\right], \tag{5}$$

for fixed θ. Expression 4 is the standard problem of solving an MDP: the solutions are the θs that maximize the return on M^A. In practice, the agent A aims to maximize this return by using the rule Ξ to update θ_t. However, because we place no restrictions on A, these updates may not always produce an increase in objective value, and we must therefore forfeit convergence guarantees. In principle, appropriate choices of A will provide such a guarantee for problems with finite state and action sets, and will tend to increase the objective function for infinite MDPs.

To find solutions for Expression 5, we construct a new MDP, the *Conjugate MDP* (CoMDP), such that each f is a policy for the CoMDP, and an optimal policy is an optimal f. The CoMDP is defined as $M^C = \left(\mathcal{S}, \mathcal{U}, \mathcal{P}^C, \mathcal{R}^C\right)$, where $\mathcal{S}$ remains the state set, $\mathcal{U}$ is now the action set, $\mathcal{P}^C$ gives state transition probabilities,

$$\mathcal{P}^C(s,u,s') = \sum_a \pi(\theta)(s,u,a)\mathcal{P}(s,a,s'), \tag{6}$$

and where $\mathcal{R}^C : \mathcal{S}\times\mathcal{U}\times\mathcal{S}\times\mathbb{R}\to[0,1]$ is the reward distribution when transitioning from s to s' via action u:

$$\mathcal{R}^C(s,u,s',r) = \frac{\sum_a \pi(\theta)(s,u,a)\mathcal{P}(s,a,s')\mathcal{R}(s,a,s',r)}{\sum_a \pi(\theta)(s,u,a)\mathcal{P}(s,a,s')}. \tag{7}$$

Notice that $\mathcal{R}^C(s,u,s',\cdot)$ is the reward distribution given the observation of s, u and s', but not a, for M with fixed θ.

Because the CoMDP M^C has state space $\mathcal{S}$ and action space $\mathcal{U}$, a policy for M^C is a probabilistic mapping $\mu^C(s,u) = \Pr(u_t{=}u|s_t{=}s)$, just like f. In the appendix, we prove in Theorem 1 that the state-value function for M^C is the same as that of M and then use this result to show in Theorem 2 that the optimal policies for M^C are the solutions to Expression 5.

5. Approximate Coordinate Ascent

In the previous section, we proposed the use of grouped coordinate ascent to perform the multivariate optimization problem of Expression 3. This involves fixing f and using A to solve M^A, then fixing θ and solving M^C using any algorithm for solving MDPs, and repeating. We call the agent solving the CoMDP the *coagent*, C.

However, coordinate ascent is impractical for at least three reasons: it requires convergence tests to determine when to switch variables being optimized, it could result in poor performance at the beginning of each partial optimization, and the first partial optimization of θ may result in a locally optimal (θ, f) before the space of fs has been searched, because the agent has tuned its parameters specifically for the initial f.

To overcome these drawbacks, we propose an approximation to grouped coordinate ascent for *on-line* methods in which the partial optimizations each run for k steps of M. For large k, this approaches grouped coordinate ascent. When $k = 1$ the agents take turns training every other time step. We define $k = 0$ to mean that both agents train during every time step of M. Notice that k scales the nonstationarity of the MDPs. We call this process ***AP****proximate* ***C****oordinate* ***A****scent* (k), or APCA(k), the pseudocode for which is provided in Algorithm 1, where θ_A, θ_C, Ξ_A, Ξ_C, π_A, and π_C are the parameters, update functions, and PPGs of A and C respectively, and the outer loop is over episodes. Notice that, while both A and C use the formulation of agents from Section 2, C's state is $\mathcal{S}$ while A's augmented state is $\mathcal{S}\times\mathcal{U}$.

If M^A and M^C have finite state and action spaces, and A and C are guaranteed to converge to an optimal policy for M^A and M^C respectively (e.g., tabular Q-learning), then APCA(∞), though impractical, is guaranteed to converge to a local optimum. Theorem 3, in the appendix, shows that the policy gradients for M^A and M^C are the components of the policy gradient for M, suggesting that policy gradient methods may provide convergence guarantees for APCA(0).

6. Coagent Networks

We have presented a framework in which a coagent, C, and an agent, A, work together to solve M. Given a state of M, they coordinate to produce an action and use the resulting state and reward from M to perform an update. Thus, we can view A and C together as one larger agent, A'. Notice that A' fits the definition of an agent given in Section 2.

Algorithm 1 Approximate Coordinate Ascent (k)

```
1: Initialize $\theta_A, \theta_C$.
2: $count \leftarrow 0$
3: loop
4:   $s \sim d_0$
5:   $u \sim \pi_C(\theta_C)(s, \cdot)$
6:   $a \sim \pi_A(\theta_A)(s, u, \cdot)$
7:   repeat
8:     $s' \sim \mathcal{P}(s, a, \cdot)$
9:     $r \sim \mathcal{R}(s, a, s', \cdot)$
10:    $u' \sim \pi_C(\theta_C)(s', \cdot)$
11:    $a' \sim \pi_A(\theta_A)(s', u', \cdot)$
12:    if $k = 0$ or $\lfloor count/k \rfloor \mod 2 = 0$ then
13:      $\theta_A \leftarrow \Xi_A(s, u, a, r, s', u', a', \theta_A)$
14:    end if
15:    if $k = 0$ or $\lfloor count/k \rfloor \mod 2 = 1$ then
16:      $\theta_C \leftarrow \Xi_C(s, u, r, s', u', \theta_C)$
17:    end if
18:    $s \leftarrow s', u \leftarrow u', a \leftarrow a'$
19:    $count \leftarrow count + 1$
20:  until $s$ is terminal
21: end loop
```

We can then ask whether an additional input, for example an additional feature, could be useful to A'. If so, we can create a new coagent, C', that searches for this mapping. C' will then be solving a new CoMDP. This process can be repeated an arbitrary number of times to create a structure, called a *coagent network* (CN), consisting of multiple coagents and one agent, all working together to solve M. Because the state-value function is the same for all CoMDPs (Theorem 1), it need only be computed by one coagent. This allows for architectures in which a subset of the coagents compute TD errors and broadcast them to the others, much like the dopamine system in animals (Schultz, 1998). To see how adding additional coagents can actually make a problem easier, see Section 8.

7. Alternate Views

In this section we present alternate views of coagents and CoMDPs. First, although it may seem that the CoMDP framework is introducing an unnecessary temporal component to an inherently non-temporal problem—searching for an optimal mapping f—this is not the case. Rather, by phrasing the problem as an MDP, we are taking advantage of structure in the underlying problem.

Consider the application of a general optimization technique, such as simulated annealing, to Expression 5. One must compute a heuristic, $h(f)$: an estimate of the expected return for various f. If we use the observed return for a finite time τ: $h(f) = \sum_{t=0}^{\tau} \gamma^t r_t$, small τ will result in high bias, while large τ will result in high variance. Furthermore, because the agent must wait τ steps to estimate $h(f)$, policy improvement slows as τ increases. All these problems can be mitigated by storing an estimate $V^f : \mathcal{S} \to \mathbb{R}$ of the expected future return, observing the return for small τ, and using V^f to estimate the remainder of the return. V^f is the value function for M^C, suggesting that the search for f takes the form of the underlying problem: an MDP.

The second interpretation is to view agents as black boxes that learn to output whatever is necessary to achieve maximal expected return in their environments, and which can cope with mild amounts of non-stationarity. One may then ask what would happen if two boxes were connected such that one's (coagent) output was given as input to the second (agent), and both observed the same states and rewards. One would expect the coagent to learn the output patterns necessary to maximize its expected return, which is the same as the agent's expected return. The environment, as seen by the coagent, is the CoMDP.

Third, the CoMDP framework is a fully-cooperative multi-agent reinforcement learning (MARL) approach (Busoniu et al., 2008), much like (Barto & Jordan, 1987). It can also be viewed as a meta-learning system (Vilalta & Drissi, 2002).

8. Case Study

To empirically validate our method, we avoid canonical domains such as cart-pole, mountain-car, and pendulum swing-up because they are too simple for modern methods, leaving little room for improvement. We therefore create a more difficult domain conducive to representation and motor primitive discovery: a simulated navigation task for a high-dimensional robot in a 10×10 continuous room. Rather than providing it with its x, y coordinates, we provide it with a high-dimensional self-centric state representation: simulated LIDAR data. We shoot 20 rays from the agent at equally spaced angular intervals and compute the distance before the rays strike a wall. These 20 real-valued numbers in the range $[0, \sqrt{200}]$ make up the state representation $\mathcal{S}$.

Real-world tasks can also require the coordination of many actions. For example, movement of a human arm requires the coordinated activation of over 100 muscle elements. To make our problem more realistic, we therefore introduce a high-dimensional action space. Rather than moving directly in the four cardinal directions, the agent has 50 actuators pointing at equally spaced angles. Each actuator can be either on or off, and, when on, moves the agent in the actuator's direction at a velocity of $1/4$. We use a time step of $\Delta t = 0.1$, and randomly scale movement velocities by up to 10%. The effects of the actuators are additive, so, when randomly activated, they tend to cancel out and result in only small movement. In order to produce rapid movement, the agent must coordinate its actions to turn on only the actuators on one side.

The agent's goal is to reach a terminal state: any within one unit of $(5, 5)$, starting from a random initial state. The reward function is 1 for reaching a terminal state, plus a *misleading* shaping reward with potential function proportional to the negative squared distance from $(3, 6)$. One approach to problems with such high-dimensional state and action spaces is to perform simultaneous representation and motor primitive discovery.

The 20-dimensional real-valued state makes fixed bases without domain specific knowledge impractical, while methods for computing feature spaces based on state trajectories will also face difficulties because random initial policies result in slow trajectories. To solve the representation problem, we therefore create 1000 Q(λ) coagents, F_i, $i \in \{1...1000\}$, each learning one binary feature used by the agent. That is, they search for a probabilistic mapping $f : \mathcal{S} \times \{0, 1\} \to [0, 1]$, the result of which, when applied to the current state, is used by A as a feature. Rather than searching for learning parameters for these coagents, we select them randomly from a distribution of reasonable parameters. This works because the agent only requires a few good features to represent the value function, so only a few coagents must learn well. We call the resulting feature space $\mathcal{F} = \{0, 1\}^{1000}$.

The motor primitive discovery problem arises because there are 2^{50} possible actions. Rather than searching this space, our agent searched over 10 actions, each of which results in a pattern of activation over the 50 actuators. We created 10 coagents, P_i, $i \in \{1, ..., 10\}$, one to learn each of these motor primitives, which are probabilistic mappings from $\mathcal{F}$ to $\mathcal{A}$. Each of these 10 coagents is composed of 50 coagents, $B_{i,j}$, $j \in \{1, ..., 50\}$, each of which learns a probabilistic mapping from the feature space to the action for one actuator for one motor primitive, $f : \mathcal{F} \times \{0, 1\} \to [0, 1]$. These coagents are actor-critics (Sutton & Barto, 1998), with state-independent policies, which result in state invariant motor primitives. We then maintain one critic, *Critic*, for all 500 of these actor-critic coagents. A and *Critic* both use the feature space $\mathcal{F}$ as a substitute for the state space $\mathcal{S}$. When $|\mathcal{F}|$ is large, this substitution is admissible because the representation will likely remain Markov.

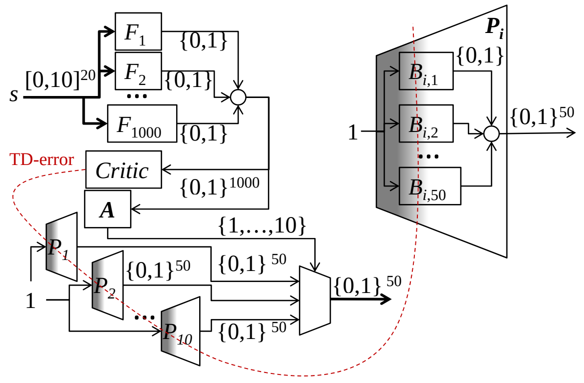

Figure 1. Diagram of the coagent network used. Bold lines denote inputs and outputs to M, the upper right trapezoid depicts the contents of each P_i, circles denote concatenation, and the non-gradiated trapezoid is a multiplexer.

Our final system has 1500 coagents, one agent, and one critic, all learning simultaneously via APCA(0). The CoMDPs all have state space $\mathcal{S}$ and action space $\{0, 1\}$. Thus, the CoMDP framework has allowed us to decompose a high-dimensional task into many low-dimensional, though nonstationary, tasks, such that successful solutions for a few will result in good overall performance. The resulting coagent network is depicted in Figure 1. Notice that the only information we have provided the coagent network *a priori* is that it should create 1000 features and 10 motor primitives.

We also created a modified task in which an actuator only produces velocity if both of its neighbors are not activated, though the velocity produced by each actuator is increased to $1/2$. Maximum velocity is then achieved by turning on every other actuator on one side. Although the coagents must already coordinate to achieve large velocities, this modified problem emphasizes the necessity for coagent coordination.

As the number of steps to reach the goal increases, the discounted lifetime return decreases, so we plot the time to reach the goal during training for APCA(0) on the original and modified problem in Figure 2. Figure 3 depicts typical motor primitives (outputs of each P_i) after training the coagent network for 2000 episodes. Random actuator settings result in an average velocity of 0.78 and 1.02 on the original and modified tasks respectively, and the maximum possible velocity is 3.98 for both. The motor primitives learned result in the agent achieving average movement speeds of 2.91 and

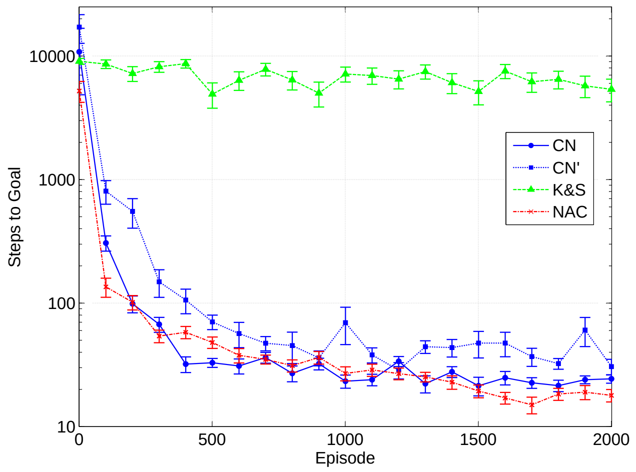

Figure 2. Average steps to reach the goal during training episodes when using APCA(0) on a CN for the original (CN) and modified (CN') problems, including standard error bars (30 trials). Results are also provided for K&S on the original problem (20 trials), though the horizontal axis is scaled by a factor of 5, so it ranges from 0 to 10000 episodes. A least-squares linear fit to the K&S data estimates a slope of -0.35, suggesting that its policy is improving, though performance appears flat when plotted with a logarithmic vertical axis. Lastly, results are provided for NAC on the original problem (20 trials), however this curve is deceiving. In 1/3 of NAC trials, the agent failed to complete an episode within a million time steps and the trial was terminated. The plot shown excludes these failures.

3.03, respectively. During control trials with 1000 random fixed features, the agent failed to learn.

We were unable to compare our results to Q(λ), Sarsa(λ), least squares policy iteration (Lagoudakis & Parr, 2002) or any other method that requires evaluation of $\arg\max_{a\in\mathcal{A}}$ because the large action space makes them impractical. Wire-fitting (Baird & Klopf, 1993) and Product of Experts (Sallans & Hinton, 2004) are not applicable because the actions are discrete and state continuous, respectively.

We therefore compare our results to a simple yet robust policy gradient method (Kohl & Stone, 2004), which we call K&S, as well as a state of the art natural policy gradient method, the natural actor-critic (NAC) with LSTD-Q(λ) (Peters & Schaal, 2008). For the former we searched the space of parameters $t \in \{1, 2, 5, 10, 50, 100, 200, 500, 1000, 2000, 5000\}, \eta \in \{0.001, 0.01, 0.1, 0.2, 0.5, 1, 2, 5\}, n \in \{0, 1, 2, 3\}$, and $\epsilon \in \{0.001, 0.01, 0.1, 0.2, 0.5, 1, 2, 5\}$ using the uncoupled Fourier basis (Konidaris et al., 2011) of order n, uncoupled polynomial basis of order n, and the identity basis.

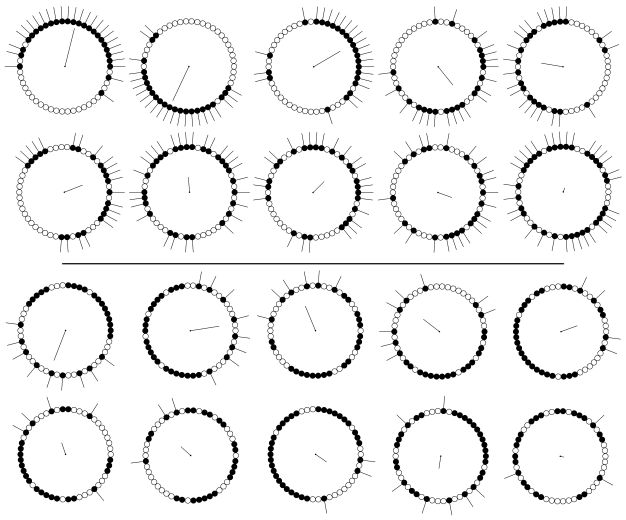

Figure 3. Individual motor primitives from a typical trial on the original task (top) and modified task (bottom). Filled circles denote activated actuators, while unfilled circles denote deactivated actuators. Rays from each actuator denote the velocity vector produced by that actuator. Notice that the first three in both sets result in a desirable movement basis: rapid movement at equally spaced angles.

Due to the computational complexity of NAC with LSTD-Q(λ), we were limited to the identity basis and a smaller parameter search: $\beta \in \{0, 0.5, 0.9, 0.99\}, \alpha \in \{0.001, 0.01, 0.1, 0.5, 1.0\}, \lambda \in \{0, 0.5, 0.9\}$, and $\gamma \in \{0.99, 0.9\}$. Additionally, because the angle between consecutive gradient estimates failed to converge in a reasonable time, we updated the policy after every $\epsilon \in \{3000, 6000, 10000, 20000, 50000\}$ time steps. This optimization took over a year of CPU time on a cluster with 60 eight-core Xeon 5355 2.66 GHz CPUs.

Results using the best parameters found for NAC and K&S are both provided in Figure 2. K&S performs poorly because it was intended for problems with fewer parameters and for which a reasonable initial policy is known. Though NAC performs as well as the coagent network 2/3 of the time, it failed during 1/3 of the trials, as discussed in the caption of Figure 2. NAC is also computationally expensive, requiring ten times as many multiplications and additions per time step as the entire coagent network, plus the additional cost of inverting a 1071×1071 matrix for policy improvement steps, making parameter optimizations impractical without access to a computational cluster.

Notice that, although we performed optimizations for K&S and NAC, we did not optimize the parameters nor the structure of the coagent network. Thus, there may be room for improvement over the coagent network results reported. Complete implementation details, including source code and all parameters used, can be found at http://www-anw.cs.umass.edu/pubs.shtml.

9. Conclusion

We have presented a novel method for searching for a mapping f that is used by an agent solving an MDP for that which maximizes the agent's expected return. We presented a case study showing that this method can outperform other state of the art methods, suggesting that it deserves further study. Future work could perform comparisons to feature and motor primitive discovery methods independently, or could use APCA(k) to find other mappings, such as skill termination sets or exploration policies for nonstationary tasks.

Acknowledgments

We would like to thank the members of the Autonomous Learning Lab for their valuable feedback and useful discussions, especially Scott Niekum, Bruno Castro da Silva, and George Konidaris. We would also like to thank the reviewers for their feedback and corrections. Andrew Barto was supported by the Air Force Office of Scientific Research under grant FA9550-08-1-0418.

References

Baird, L. C. and Klopf, A. H. Reinforcement learning with high-dimensional, continuous actions. Technical Report WL-TR-93-1147, Wright-Patterson Air Force Base, 1993.

Barto, A. G. and Jordan, M. I. Gradient following without back-propagation in layered networks. In Caudill, M. and Butler, C. (eds.), *Proceedings of the First IEEE Annual Conference on Neural Networks*, pp. II–629–II–636, San Diego, CA, 1987.

Bezdek, J. C., Hathaway, R. J., Howard, R. E., Wilson, C. A., and Windham, M. P. Local convergence analysis of grouped variable version of coordinate descent. *Journal of Optimization Theory and Applications*, 54(3): 471–477, 1987.

Busoniu, L., Babuska, R., and De Schutter, B. A comprehensive survey of multiagent reinforcement learning. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, volume 38, pp. 156–172, 2008.

Kohl, N. and Stone, P. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.

Konidaris, G. D., Osentoski, S., and Thomas, P. S. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, August 2011.

Lagoudakis, M. G. and Parr, R. E. Model-free least-squares policy iteration. In *Advances in Neural Information Processing Systems*, volume 2, pp. 1547–1554, 2002.

Mahadevan, S. and Maggioni, M. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007.

McGovern, A. and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *International Conference on Machine Learning*, pp. 361–368, 2001.

Peters, J. and Schaal, S. Natural actor-critic. *Neurocomputing*, 71:1180–1190, March 2008.

Sallans, B. A. and Hinton, G. E. Reinforcement learning with factored states and actions. *Journal of Machine Learning*, 5:1063–1088, 2004.

Schultz, W. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80:1–27, 1998.

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

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

Todorov, E. and Ghahramani, Z. Unsupervised learning of sensory-motor primitives. In *IEEE Engineering in Medicine and Biology Society*, 2003.

Vilalta, R. and Drissi, Y. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18:77–95, 2002.

Appendix

For the subsequent proofs, we require additional notation. Let $E_5^f[\cdot]$ denote an expected value assuming the dynamics of M with fixed θ and the specified f, i.e., the dynamics of Expression 5. Similarly, let $E_C^f[\cdot]$ denote an expected value assuming the dynamics of M^C with the provided f as the policy. We also define $d_5^t(s, d_0)$ to be the probability that $s_t = s$ given initial state distribution d_0 and assuming the dynamics of Equation 5, with a suppressed dependence on f. Similarly, we define $d_C^t(s, d_0)$ to be the probability assuming the dynamics of M^C. Lastly, we define d_s to be the state distribution where $\Pr(s_t{=}s) = 1$.

Because we have not yet established that the composition of M and the agent with fixed θ produces an MDP with actions u, we must redefine the value function,

$$V_5^f(s) = E_5^f\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s,\right]. \tag{8}$$

Notice that $E^f_5, E^f_C, V^f_5, V^f_{M^C}, d^t_5$, and d^t_C all assume that θ is fixed and provided.

For Theorem 3, let f be parameterized by θ_C, $\rho = [\theta, \theta_C]$, and let $J_M(\rho) = E[V^\rho_M(s_0)|d_0]$ be the expected discounted return for M from initial state distribution d_0, where ρ induces a policy on M. Similarly, $J_{M^A}(\theta) = E[V^\theta_{M^A}(s)|d_0]$, and $J_{M^C}(\theta_C) = E[V^{\theta_C}_{M^C}|d_0]$. M^A and M^C have suppressed dependencies on θ_C and θ respectively, so J_{M^A} and J_{M^C} are both functions of θ and θ_C.

Lemma 1: $E^f_5[r_t|d_t] = E^f_C[r_t|d_t]$, for arbitrary t and state distribution d_t, where $d_t(s) = \Pr(s_t = s)$.
Proof:

$$\begin{aligned}E^f_5[r_t|d_t] =& \sum_r r \Pr(r|d_t)\\ =& \sum_r r \sum_s d_t(s) \sum_u f(s,u) \sum_a \pi(\theta)(s,u,a)\times\\ & \sum_{s'} \mathcal{P}(s,a,s')\mathcal{R}(s,a,s',r). \qquad (9)\end{aligned}$$

$$\begin{aligned}E^f_C[r_t|d_t] =& \sum_{r,s,u,s'} r\, d_t(s) f(s,u)\mathcal{P}^C(s,u,s')\mathcal{R}^C(s,u,s',r)\\ =& E^f_5[r_t|d_t], \qquad (10)\end{aligned}$$

by substituting from Equations 6 and 7 into Equation 10. □

Lemma 2: $d^t_5(s, d_0) = d^t_C(s, d_0)$ for all states s, times t, and initial state distributions d_0.
Proof: The base case is $d^0_5(s, d_0) = d_0 = d^0_C(s, d_0)$. The inductive step is to show that $d^{t+1}_5(s, d_0) = d^{t+1}_C(s, d_0)$ if $d^t_5(s, d_0) = d^t_C(s, d_0)$:

$$\begin{aligned}d^{t+1}_5(s,d_0) =& \sum_{\bar{s}} d^t_5(\bar{s},d_0) \sum_u f(\bar{s},u)\times\\ & \sum_a \pi(\theta)(\bar{s},u,a)\mathcal{P}(\bar{s},a,s) \qquad (11)\\ d^{t+1}_C(s,d_0) =& \sum_{\bar{s}} d^t_C(\bar{s},d_0) \sum_u f(\bar{s},u)\mathcal{P}^C(\bar{s},u,s)\\ =& d^{t+1}_5(s,d_0), \qquad (12)\end{aligned}$$

by substituting Equation 6 for $\mathcal{P}^C$. □

Theorem 1: $V^f_5(s) = V^f_{M^C}(s)$ for all f and s.
Proof:

$$V^f_5(s) = \sum_{t=0}^{\infty} \gamma^t E^f_5\left[r_t|d^t_5(\cdot, d_s)\right], \qquad (13)$$

$$V^f_{M^C}(s) = \sum_{t=0}^{\infty} \gamma^t E^f_C\left[r_t|d^t_C(\cdot, d_s)\right], \qquad (14)$$

which by Lemmas 1 and 2 allows us to conclude that $V^f_5(s) = V^f_{M^C}(s)$. □

Notice that $V^f_5(s)$ is the expected return on M for the current f and θ: the value function for M.

Theorem 2: The optimal policies for M^C are the solutions to Expression 5.
Proof: Optimal fs for Expression 5 satisfy

$$\begin{aligned}&\arg\max_{f\in\mathbb{F}} E_M\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| \theta, f, d_0\right]\\ =& \arg\max_{f\in\mathbb{F}} \sum_s d_0(s)V^f_5(s). \qquad (15)\end{aligned}$$

Similarly, an optimal policy for M^C satisfies

$$\arg\max_{f\in\mathbb{F}} \sum_s d_0(s)V^f_{M^C}(s). \qquad (16)$$

By Theorem 1, we conclude that Expressions 15 and 16 are equal. □

Theorem 3: If $\partial J_M(\rho)/\partial\rho$ exists, and the parameterized policies, $\mu^A(\theta)$ for M^A and $\mu^C(\theta_C)$ for M^C, are differentiable with respect to their parameters, then the policy gradients for M^A and M^C are the components of the policy gradient for M.
Proof:

$$\begin{aligned}\frac{\partial J_M(\rho)}{\partial\theta} =& \frac{\partial}{\partial\theta}\sum_s d_0(s)V^\rho_M(s) = \frac{\partial}{\partial\theta}\sum_s d(s)V^\theta_{M^A}(s)\\ =& \frac{\partial J_{M^A}(\theta)}{\partial\theta}, \qquad (17)\end{aligned}$$

and

$$\begin{aligned}\frac{\partial J_M(\rho)}{\partial\theta_C} =& \frac{\partial}{\partial\theta_C}\sum_s d_0(s)V^\rho_M(s) = \frac{\partial}{\partial\theta_C}\sum_s d(s)V^{\theta_C}_{M^C}(s)\\ =& \frac{\partial J_{M^C}(\theta_C)}{\partial\theta_C}, \qquad (18)\end{aligned}$$

because $V^\rho_M = V^\theta_{M^A}$ (see Section 2), $V^{\theta_C}_{M^C} = V^f_5$ by Theorem 1, and $V^f_5 = V^\rho_M$, trivially. If A and C compute their respective policy gradients, $\frac{\partial J_{M^A}(\theta)}{\partial\theta}$ and $\frac{\partial J_{M^C}(\theta_c)}{\partial\theta_C}$, they would therefore be computing

$$\frac{\partial J_M(\rho)}{\partial\rho} = \left[\frac{\partial J_{M^A}(\theta)}{\partial\theta}, \frac{\partial J_{M^C}(\theta_C)}{\partial\theta_C}\right], \qquad (19)$$

which is the gradient of Expression 3 and the policy gradient of M. □

If ρ is updated by $\rho \leftarrow \rho + \alpha\frac{\partial J_M(\rho)}{\partial\rho}$, then ρ will converge to a local optimum under the usual conditions for decreasing the step-size parameter α.