In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*,
pp. 749--756, IFAAMAS, Richland, SC, USA, 2009.

An Empirical Analysis of Value Function-Based and Policy Search Reinforcement Learning

Shivaram Kalyanakrishnan
Department of Computer Sciences
The University of Texas at Austin
shivaram@cs.utexas.edu

Peter Stone
Department of Computer Sciences
The University of Texas at Austin
pstone@cs.utexas.edu

ABSTRACT

In several agent-oriented scenarios in the real world, an autonomous agent that is situated in an unknown environment must learn through a process of trial and error to take actions that result in long-term benefit. Reinforcement Learning (or sequential decision making) is a paradigm well-suited to this requirement. Value function-based methods and policy search methods are contrasting approaches to solve reinforcement learning tasks. While both classes of methods benefit from independent theoretical analyses, these often fail to extend to the practical situations in which the methods are deployed. We conduct an empirical study to examine the strengths and weaknesses of these approaches by introducing a suite of test domains that can be varied for problem size, stochasticity, function approximation, and partial observability. Our results indicate clear patterns in the domain characteristics for which each class of methods excels. We investigate whether their strengths can be combined, and develop an approach to achieve that purpose. The effectiveness of this approach is also demonstrated on the challenging benchmark task of robot soccer Keepaway. We highlight several lines of inquiry that emanate from this study.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Algorithms, Experimentation.

Keywords

Reinforcement learning, Temporal difference learning, Policy search, Function approximation.

1. INTRODUCTION

Reinforcement Learning (or sequential decision making from experience) is a very suitable method for autonomous agents to improve their long-term gains as they repeatedly carry out sensing, decision and action while situated in an unknown environment. Reinforcement learning tasks are commonly formulated as Markov Decision Problems (MDPs). The solution of MDPs has benefited immensely from strong a theoretical framework that has been developed over the years. The cornerstone of this framework is the **value function** [2] of the MDP, which encapsulates long-term utilities of decisions. A control policy is induced by the value function; indeed, this approach can guarantee convergence to the optimal policy for finite MDPs [22].

Tasks that occur in the real world invariably involve large, often continuous state spaces that necessitate the use of function approximation. Although certain learning algorithms that employ function approximation provably converge, their resulting performance depends on the accuracy of the approximation, and therefore is not necessarily optimal [9]. Another inevitable handicap to learning in realistic applications is imperfection in the detection of state due to partial observability or sensor noise. Coping with partial observability has merited considerable attention in the literature [4, 10], but is yet to scale to complex tasks with continuous state spaces.

In contrast with value function-based methods, **Policy search** methods for sequential decision making reason directly about parameters that maximize long-term returns, rather than inferring them from a value function. In general, policy search methods may completely bypass the learning of value functions and remain oblivious to the Markovian property of the underlying task. Whereas doing so amounts to ignoring potentially useful state transition information, it can also avoid the pitfalls introduced therein by poor function approximation and partial observability. A variety of policy search algorithms exist in the literature [6, 15], including some that achieve convergence to local optima in the space of policy parameters [1, 3, 7].

Despite the limitations imposed by inaccurate approximation schemes and partial observability, value function-based approaches have recorded numerous successes on tasks as diverse as autonomic resource allocation [21], elevator control [5], and robot soccer [16]. Likewise, policy search methods have been shown to be effective on challenging applications such as quadrupedal locomotion [8] and helicopter control [11]. While much can be understood from the theoretical roots of the algorithms employed in all these cases, complexities inherent in their deployed tasks invariably negate the validity of the theoretical analyses. Thus, given a specific problem instance, practitioners are faced with the *practical* decision of which algorithm to apply. In this paper, we present the results of an empirical study seeking to inform that decision.

As a first step, we define a concrete scope for comparing

Cite as: An Empirical Analysis of Value Function-Based and Policy Search Reinforcement Learning, Shivaram Kalyanakrishnan and Peter Stone, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

sequential decision making algorithms (Section 2). We proceed to construct a class of MDPs with parameters that can be systematically varied to model different degrees of problem size, stochasticity, precision of function approximation, and partial observability (Section 3). We report the results of our discriminative study in Section 4.1. These results lead us to investigate whether value function-based and policy search methods can be combined (Section 4.2), which we further test on a more challenging task (Section 4.3). In Section 5, we present several ways to extend this study. We discuss related work in Section 6 and conclude in Section 7.

2. LEARNING FRAMEWORK

A Markov Decision Problem $M = (S, A, R, T, \gamma)$ comprises a set of states S and a set of actions A available from each state. A reward function $R : S \times A \times S \to \mathbb{R}$ assigns numerical rewards to transitions, which are generated based on a stochastic transition function $T : S \times A \times S \to [0, 1]$. A (deterministic) policy $\pi : S \to A$ specifies an action a to take from a given state s; associated with π is its *action value function* $Q^\pi : S \times A \to \mathbb{R}$, which satisfies, $\forall s \in S, \forall a \in A$:

$$Q^\pi(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma Q^\pi(s', \pi(s'))],$$

where $\gamma \in [0, 1)$ discounts the long-term reward.[1] In reinforcement learning tasks, R and T are not specified, but can be sampled. The objective is to learn an optimal policy π^*, which satisfies $Q^{\pi^*}(s, \pi^*(s)) = \max_\pi Q^\pi(s, \pi(s)), \forall s \in S$. Although π^* and $Q^{\pi^*} = Q^*$ can be learned exactly for finite S and A [22], problems with large, possibly continuous state spaces dictate the adoption of approximate architectures. In such cases, value function-based (VF) methods commonly approximate Q^* through $\hat{Q} : S \times A \to \mathbb{R}$, given by

$$\hat{Q}(s,a) = \rho(\boldsymbol{\phi}(s,a), \mathbf{w_{VF}}), \forall s \in S, \forall a \in A.$$

Here, $\hat{Q}$ is expressed as some function ρ acting on a vector of features $\boldsymbol{\phi}$ representing the agents' state-action space. ρ is parameterized by a vector $\mathbf{w_{VF}}$. In general, VF methods start with some initial $\mathbf{w}_{VF}$, which they progressively refine based on experience. We adopt Sarsa(0) [14] as a representative VF method for our experiments. Sarsa(0) is an on-policy method under which experience gathered along the trajectory $s, a, r, s', a', \ldots$ leads to the following update (if minimizing the squared error):

$$\begin{aligned}\mathbf{w}_{VF,t+1} \quad = \quad & \mathbf{w}_{VF,t} + \alpha_t \cdot \nabla_{\mathbf{w}} \rho(\boldsymbol{\phi}(s,a), \mathbf{w_{VF}}) \cdot \\ & \{r + \gamma \rho(\boldsymbol{\phi}(s',a'), \mathbf{w}_{VF,t}) - \rho(\boldsymbol{\phi}(s,a), \mathbf{w}_{VF,t})\}.\end{aligned}$$

The learning rate $\alpha_t > 0$ is usually annealed over time. Although our implementation of Sarsa(0) does not meet all the conditions necessary for provable convergence [12], we find that in our experiments, Sarsa(0) typically leads to optimality in situations where optimality is indeed achievable (see Section 4).

Whereas VF methods derive the control policy π_{VF} from the learned $\hat{Q}$, policy search (PS) methods directly search for parameters $\mathbf{w}_{PS}$ to maximize the value of the policy π_{PS}. In general, PS methods can update a policy every time an action is taken from some state, and the reward and next state become available. In this paper, however, we only consider the limiting case in which policy updates are made solely based on the long-term reward accrued by a policy. This makes policy updates less sensitive to imperfections in the state representation when compared to a bootstrapping VF method such as Sarsa(0). In principle, VF methods too can make updates on the basis of long-term rewards; for example, Sarsa(λ) with $\lambda > 0$ does so through the use of eligibility traces [10]. In Section 5, we consider incorporating such methods in our study as part of future work.

[1] As a matter of convenience, we use the action value function Q^π in our exposition instead of the more commonly used *value function* $V^\pi : S \to \mathbb{R}$, given by $V^\pi(s) = Q^\pi(s, \pi(s)), \forall s \in S$. The value of the *policy* π is given by $V(\pi) = \sum_{s \in S} D(s) V^\pi(s)$, where $D : S \to [0, 1]$ is the distribution of start states. Also, we note that it is common to set $\gamma = 1$ in episodic MDPs.

The PS method we use in our experiments is the cross-entropy method [6], a general optimization technique under which a population of candidate solutions is sampled from a parameterized distribution; based on the fitness of each sample (in this case an estimate of the value of the policy), the parameters of the distribution are recomputed to generate fitter samples. Like Sarsa, the cross entropy method too enjoys successes on challenging reinforcement learning tasks [18].

In order to conduct a fair comparison between the methods, we enforce that both VF and PS employ the same representation for their policies, given by:

$$\begin{aligned}\pi_{VF}(s) \quad &= \quad \operatorname*{argmax}_{a \in A} \rho(\boldsymbol{\phi}(s,a), \mathbf{w}_{VF}), \text{and} \\ \pi_{PS}(s) \quad &= \quad \operatorname*{argmax}_{a \in A} \rho(\boldsymbol{\phi}(s,a), \mathbf{w}_{PS}).\end{aligned}$$

In effect, this restriction facilitates a *direct comparison* between the converged values of $\mathbf{w}_{VF}$ and $\mathbf{w}_{PS}$, which are reached through very different routes. Under VF, ρ approximates the action value function, and is updated on-line using transition information. Thus, VF is likely to converge more quickly than PS, under which policy updates are only made after enough transitions are available to estimate the value of the policy (In PS, ρ provides action *preferences*, which do not have any particular semantics.). Yet, as Baxter and Bartlett [1] illustrate, the convergence of VF can be sub-optimal even in simple 2-state MDPs. In principle, PS methods may possibly locate choices of $\mathbf{w}$ that may not approximate Q^* well, but still induce π^*. The next section develops experimental apparatus to examine this possibility.

3. A PARAMETERIZED CLASS OF MDPS

Seeking to identify trends in the performance of VF and PS methods as the substrate problem is varied, we construct a *class* of MDPs indexed by parameter settings that can be systematically varied. To facilitate extensive testing, it is desirable that these MDPs be time efficient to solve, while retaining enough richness to discriminate between solutions. Additionally, if optimal solutions to the MDPs can be computed from their specifications, the performance of the learning methods being tested on them can be benchmarked quantitatively.

The class of MDPs we design to meet these criteria consist of simple square grids, each consisting of a discrete number of states. The size of the state space is $s^2 - 1$, where s, the side of the square, serves as a parameter to be varied. Each episode begins with the agent placed in a start state chosen uniformly randomly from among the set of non-terminal states, as depicted in Figure 1(a). The agent can take either of two actions from each state: **North** (**N**) and **East** (**E**). On taking **N** (**E**), the agent moves north (east) with proba-

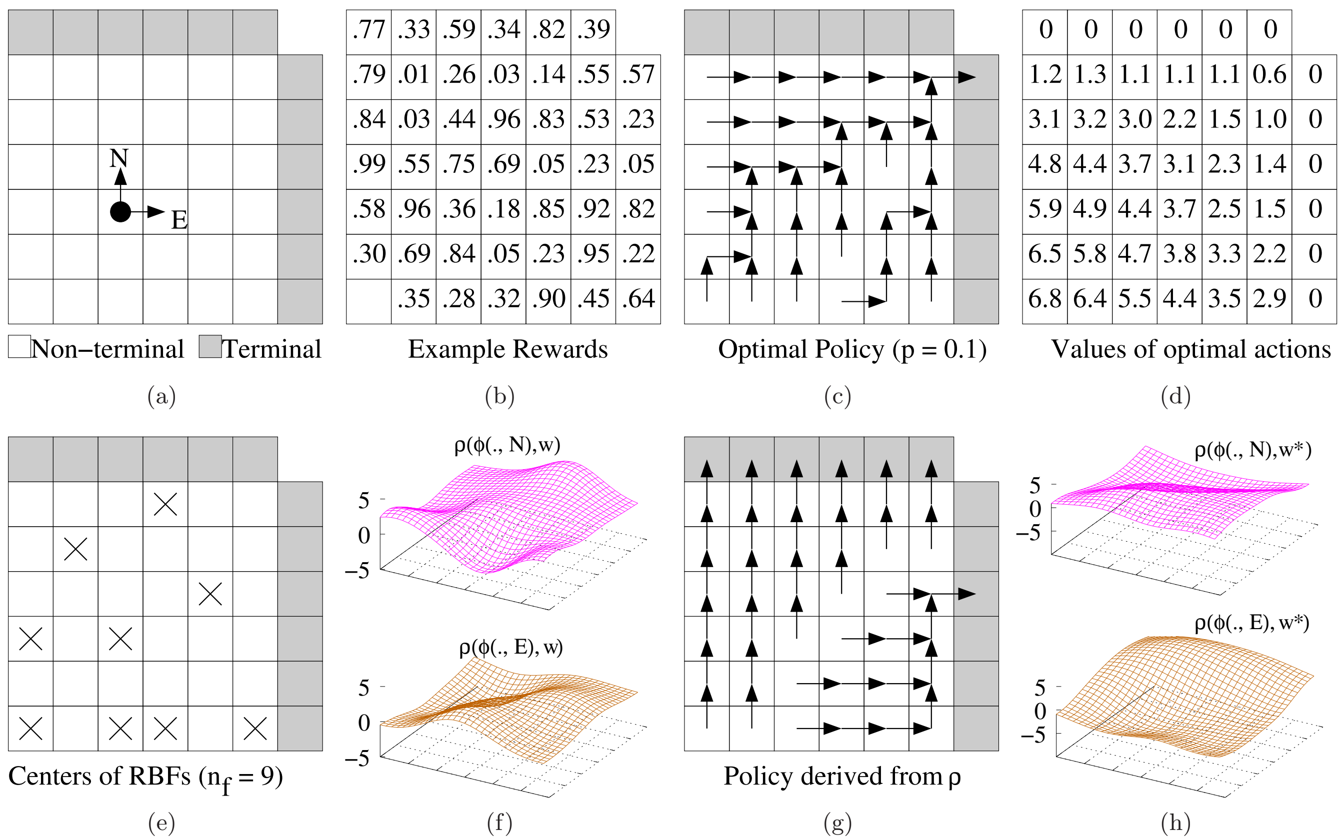


Figure 1: (a) MDP example with $s = 7$. (b) Rewards obtained at "next states" of transitions. (c) Optimal policy when $p = 0.1$. (d) Optimal action values from each state. (e) Subset of cells with RBFs ($\chi = 0.25$) (f) ρ constructed with these features for some w, and (g) Resulting policy. (h) ρ corresponding to optimal policy.

bility p and it moves east (north) with probability $1-p$. The variable p, which controls the stochasticity in the transitions, is also treated as a parameter of the MDP. We notice that irrespective of the value of p, the agent always moves either north or east on each transition before reaching a terminal state; consequently, episodes last at most $2s-3$ steps.

Through the course of each episode, the agent accrues rewards at the states it visits. Each MDP is initialized with a fixed set of rewards drawn uniformly from $[0,1]$, as illustrated in Figure 1(b). Figures 1(c) and 1(d) show the optimal actions to take (along with their values) for this reward structure (assuming $p = 0.1$), obtained using dynamic programming. Needless to mention, it is also quite straightforward in this case to learn the optimal policy based on experience, for example by using a table of action values updated through Q-learning. Yet, the objective of our study is to investigate situations where table-based approaches do not apply and approximate schemes must be employed. Hence, we limit the agents' perceptual information to a vector of n_f features, each corresponding to the activation of a radial basis function (RBF) centered at some fixed non-terminal cell in the grid. The centers of the RBFs are chosen at random; in Figure 1(e), 9 such centers are shown. ρ is implemented as a separate linear combination of the same features for each action a; the policy becomes $\pi(s,a) = \text{argmax}_{a\in A}\, \rho(\boldsymbol{\phi}(s,a), \mathbf{w}) = \text{argmax}_{a\in A}\, \mathbf{w}_a^T \phi(s,a)$, where $\mathbf{w}_a$ are the coefficients for action a.

Although the state space is discrete, the choice of radial basis functions (with standard deviation equal to the width of a grid cell) as features promotes generalization to nearby cells devoid of bases. Figures 1(f) and 1(g) depict instances of ρ and π obtained from an 18-dimensional weight vector allocating one weight to each combination of RBF and action. Figure 1(h) shows an example in which every non-terminal state contains an RBF (thus, $n_f = (s-1)^2$), and the vector of parameters $\mathbf{w}^*$ is optimal, leading to a representation ρ corresponding to the optimal policy. In order to model a gradual decay in the accuracy of the representation, we set $\chi = \frac{n_f}{(s-1)^2}$ as a parameter in our experiments. In Figure 1(e), $n_f = 9$, $s = 7$, and thus $\chi = 0.25$.

In order to study the effect of partial observability in this domain, we add noise uniformly generated from $[-\sigma, \sigma]$ to each component of the feature vector $\boldsymbol{\phi}(s,a)$ visible to the agent. Setting $\sigma > 0$ invalidates the assumption of complete observability of the underlying state, effectively rendering the environment non-Markovian. σ serves as the fourth and final parameter in our study.

The design choices described in this section are the result of a process of trial and error directed towards constructing a suite of instances that allow us to study trends in learning algorithms, rather than constructing instances that are challenging in themselves for learning. Of the parameters we choose, s and p are both attributes of the underlying MDP. χ is usually construed as a property of the learning algorithm, exhibiting the shortcomings of imprecise features and representations. The state noise σ may also owe to inexact representation, and typically includes the physical limitations of the agents' sensors. It exceeds the scope of this paper to develop more precise models of the state noise; yet our simplifying assumption that it is uniformly distributed allows us to *vary* noise levels in a controlled manner. In principle, s, p, χ and σ along with a random seed fix an MDP in the class. By averaging over multiple runs with different random seeds, we estimate the mean performance achieved by learning methods as a function of s, p, χ and σ. The next section reports the results of experiments applying the VF and PS methods identified in Section 2 to the class of MDPs described in this section.

4. EMPIRICAL EVALUATION

For our experiments, we choose algorithm-specific constants for Sarsa(0) and the cross entropy method that work well over a broad range of settings in our test suite. We use an ϵ-greedy policy while learning under Sarsa(0), initializing ϵ to 0.1 at the beginning of training, and decaying it as a harmonic sequence every $50,000$ episodes. Incidentally, the same initial value and annealing schedule also work well for the learning rate α. Under the cross entropy method, we maintain a population of 100 policies derived from parameter vectors drawn from a Gaussian distribution starting with zero mean and a variance of 1.0 along each dimension. Each policy is evaluated for 2000 episodes, after which the sample mean and variance of the best 5 fix the distribution for the subsequent iteration. In our experiments, we run each learning method for a total of 10^6 episodes.

We fix a set of values for the MDP parameters to model a broad range of situations agents encounter in practice. The size of the underlying state space is reflected by the parameter s, which we draw from $\{4, 7, 10, 13, 16\}$. The action noise p is varied likewise from 0 to 0.4, in increments of 0.1. We find it informative to gather more data points varying the function approximation parameter χ; correspondingly, we pick 10 equally spaced points, drawn from $\{0.1, 0.2, \ldots, 1.0\}$. With the state noise parameter σ, we notice a ceiling effect beyond a value of 0.4, so we limit our samples to the set $\{0, 0.1, 0.2, 0.3, 0.4\}$. In aggregate, these choices for the individual parameters lead to a total of $5 \times 5 \times 10 \times 5 = 1250$ settings. Under each such setting, we run 25 independent trials of Sarsa(0) (VF) and cross entropy (PS). Figure 2 shows corresponding learning curves for 12 such settings that facilitate comparisons across changes in each parameter. The policy value is normalized to fit in the interval $[0, 1]$ by scaling it with respect to the optimal and least optimal values: for our class of MDPs, both values are well-defined. Predominantly, standard errors tend to be less than 0.01 for all the curves beyond 20,000 episodes of training; we do not list them explicitly. In Section 4.2 we consider VF+PS, a scheme to integrate VF and PS methods, which is also included in Figure 2.

4.1 Initial Results

The most striking observation from Figure 2 is the disparity in the learning rates of VF and PS. In most cases, VF plateaus within a few thousands of episodes, while PS takes orders of magnitude longer. This phenomenon affirms the intuition that by explicitly accounting for state transitions, VF methods are more sample efficient. We could possibly make PS quicker by reducing the number of episodes over which each policy is evaluated (currently 2000), but we notice that doing so significantly reduces its asymptotic performance. Indeed, a gradual decline in learning speed is apparent for PS as s is varied from 4 to 16 (Figures 2(a), 2(b) and 2(c)). The action noise p increases from 0 to 0.4 in in Figures 2(b), 2(d) and 2(e), while other parameters remain constant. At $p = 0.4$, the learning speed of VF is slower compared to lower values of p, yet the asymptotic performance is still near-optimal. Indeed, the MDPs from Figures 2(a) through 2(e) can all be solved exactly (since $\chi = 1$ and $\sigma = 0$), as hinted by the performance of VF in our experiments.

The sequence of examples in Figures 2(f) through 2(i) contrast with the earlier examples: they alter χ, which controls function approximation, in steps from 0.1 to 1.0. While this affects the asymptotic performance of PS minimally, a dramatic decline is observed in the case of VF with small χ values. Indeed, at $\chi = 0.1$ and $\chi = 0.3$, VF converges to noticeably poorer policies than PS. The ability of PS to achieve good performance even under such impoverished representations makes it a promising candidate in a large number of real-world domains where feature engineering is deficient. We posit that like Baxter and Bartlett's example [1], many of the cases with $\chi < 1.0$ allow for the representation of high-reward policies, but only admit poor approximations of the action value function.

The performance of VF and PS under increasing values of state noise σ is shown in Figures 2(i) through 2(k). As with decreasing χ, VF methods show a sharp fall in performance as σ is increased. PS methods too suffer a fall in performance, albeit more gradually. In the extreme case with $\chi = 0.1$, $\sigma = 0.4$ (Figure 2(l)), both methods fare poorly, which is possibly due to the inability of the representation employed to accommodate perceptual aliasing. Although principled approaches may be followed to cope with partial observability [4, 10], we leave an extension to such a scenario for future work.

4.2 Synthesizing VF and PS Methods

The main inferences we draw from our experiments are that VF methods display superior sample complexity and asymptotic performance under conditions that favor their provable convergence, while PS is more resilient to departures from such an ideal. All these are desirable properties for learning algorithms deployed in practical situations. In an effort to investigate whether the strengths of VF and PS methods can be integrated into an algorithm that enjoys such properties, we devise a simple scheme. Since we observe that VF algorithms tend to converge quickly, we hypothesize that these early converged values can serve as good starting points for PS, which could then proceed to improve upon the initialization.

Since in our experiments, VF and PS are constrained to share a common representation, a straightforward method to initialize PS with a policy learned using VF is to assign $\mathbf{w}_{PS,init} = \mathbf{w}_{VF,converged}$. Although such a scheme does not apply generally across VF and PS algorithms that differ in their representations and assumptions, we conjecture that its resulting performance can still offer insights about synthesizing the merits of VF and PS methods. We conduct experiments on the same suite of MDPs with VF+PS, which combines VF and PS as described above. In these experiments, VF is conducted for 10,000 episodes, followed by a transfer and subsequent switch to PS. Thus, at the switch,

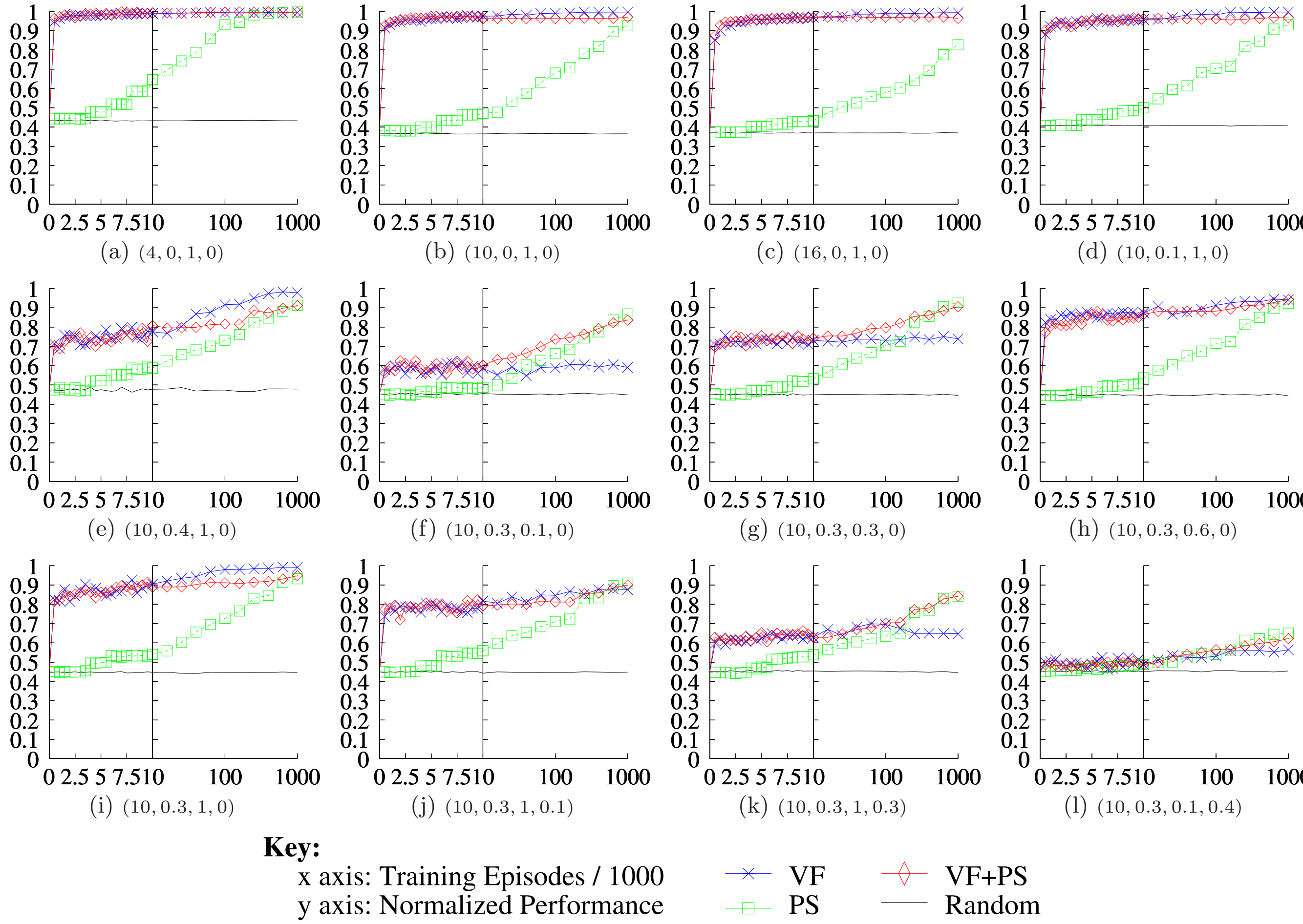
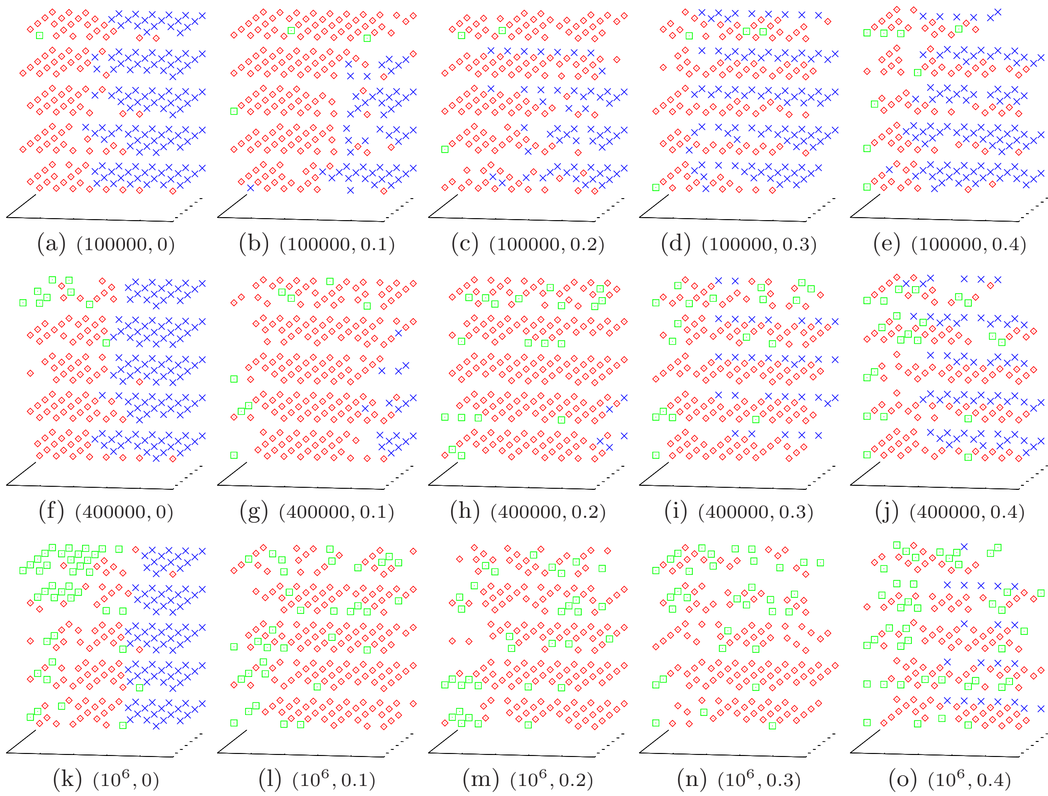


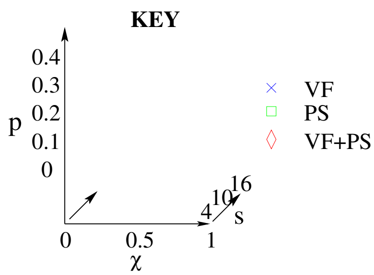
Figure 2: 12 different parameter settings, each abbreviated as (s, p, χ, σ). Each graph shows curves for VF, PS, VF+PS (see Section 4.2) and Random methods. Each curve is an average of 25 independent runs. At each reported point, the learned policy is executed for 10,000 episodes with learning switched off. Note the break in the x axis at 10,000 episodes, beyond which a log scale is adopted.

the mean of the Gaussian distribution initializing the cross entropy method is set to $\mathbf{w}_{VF}$; we set the variance along each dimension to a value of 0.25. VF+PS is compared with VF and PS in Figure 2. In the cases where $\chi = 1$ and $\sigma = 0$ (Figures 2(a) through 2(e)), VF+PS performs on par with VF or falls slightly short in its final performance, likely due to the sampling error of PS introduced by stochasticity. Thus, VF+PS preserves the superior sample efficiency inherited from VS. At the same time, it does not suffer an equal drop in performance when χ is reduced (Figures 2(f) through 2(h)). While all methods show significant losses in asymptotic performance as σ is increased, VF and VF+PS do perform relatively better than VF (Figures 2(j) through 2(l)). This suggests that VF+PS displays the resilience of PS when $\chi < 1$ and $\sigma > 0$.

To facilitate locating patterns in the performance of VF, PS, and VF+PS over our entire suite of test MDPs, we identify the winner for each MDP after given amounts of training time. The results are depicted graphically in Figure 3. Several trends become apparent from this visualization. Under most parameter settings, VF and VF+PS prevail up to 100,000 episodes of training (Figures 3(a) through 3(e)), which we also confirm from Figure 2. Beyond this point, VF+PS continues to remain effective, while the performance of VF begins to fall. Indeed, VF is predominantly outperformed by VF+PS and PS for intermediate values of σ (0.1 to 0.3); at $\sigma = 0.4$, a ceiling effect starts manifesting as all methods show deterioration. For $\sigma = 0$, VF only seems to enjoy an advantage for values of χ greater than 0.5, reinforcing our earlier observation that VF suffers more from deficient function approximation than PS. In Figure 3, no dramatic trends are visible within any given setting as the problem size (s) and stochasticity (p) are varied; we posit that these parameters primarily play an important role very early in training.

From Figure 3, we also notice that VF+PS derives the strengths of VF and PS in the parameter space suited to either method. Indeed, VF+PS outperforms VF and PS individually if we average across all the parameter settings for a given number of training episodes. The fact that VF+PS achieves higher asymptotic performance than PS under several settings suggests that the initial VF phase leads it starting points for PS that are better than random. Another way to interpret this phenomenon is that PS is able to refine the solutions found by VF. Such robustness evinces the potential of VF+PS to apply as a general purpose solution to a





Overall Fraction of Wins after Training Episodes

Method	100,000	400,000	10^6
VF	0.34	0.19	0.09
PS	0.02	0.06	0.15
VF+PS	0.48	0.59	0.52

Figure 3: 15 scatter plots, each corresponding to some number of training episodes e and some value of σ, abbreviated (e, σ). Within each plot, points are indexed by values of χ, s, and p, as shown in the key. For each value of (e, σ, χ, s, p), an algorithm among VF, PS, and VF+PS is shown as the winner if its performance exceeds the others' at level of significance 0.01. Settings for which comparisons are not statistically significant are omitted. For each value of e, the table shows the fraction over the 1250 MDPs that each method wins.

vast number of realistic problems. This inference is derived from experiments on synthetic problems; we proceed to verify its validity on a significantly more complex and popular reinforcement learning benchmark.

4.3 Results from a Complex Benchmark Task

The task we consider for further experiments is Keepaway [16], a challenging benchmark problem for multiagent reinforcement learning. Keepaway is a subtask of soccer in which a team of 3 *keepers* has to keep possession of the ball away from the opposing team of *takers* inside a small rectangular region (Figure 4). The continuous state space is represented through 13 features involving distances and angles among the players, and the keeper with the ball has 3 actions: holding the ball or passing to either teammate. The keepers get rewarded based on the time elapsed between actions, which amounts to maximizing the episode duration. Players have noisy sensors and actuators.

Unlike with our synthetic MDPs, a linear representation is not sufficient for representing competent Keepaway policies, and nor is it clear if optimality is achievable. Our policy is represented through three 13-20-1 neural networks, computing activations for each action. We run Sarsa(0) (VF) with constant values of $\epsilon = 0.01$ and $\alpha = 0.0001$. We choose parameters for the cross entropy methods (PS) such that they maximize the performance after 40,000 episodes of training. We maintain a population of 20 solutions, each evaluated for 125 episodes. The initial Gaussian distribution used for generating the weights of the neural net is $N(0, 1)^{903}$ (each neural net has 301 weights, including biases). After each generation (2500 episodes), the 5 neural networks registering the highest performance are used to determine the mean and variance for the subsequent iteration. Under VF+PS, we transfer the weights of the neural networks after 15,000 episodes of training (and set each variance to 1). Figure 4 shows that on Keepaway too VF+PS dominates both VF

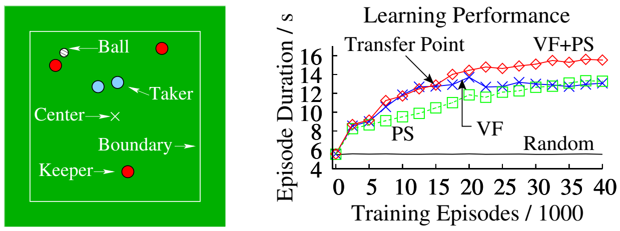


Figure 4: On left, a snapshot from a Keepaway game. On right, learning curves for VF, PS, VF+PS, and Random policies. Each curve is an average of at least 14 independent runs. At each reported point, the learned policy is executed for 500 episodes with learning switched off.

and PS beyond 20,000 episodes of training ($p < 0.01$). This result serves as an existence proof of the successful combination of VF and PS methods on a complex, realistic task, affirming a similar observation inferred from the suite of synthetic test domains.

5. OTHER RELEVANT COMPARISONS

The underlying motivation for this study is that the theoretical proofs of convergence, both of VF and PS methods [6, 12] fail to provide adequate bounds for the performance of the converged solution in the presence of function approximation and partial observability. Yet these two factors affect nearly every sequential decision making task encountered in practice. Thus, it is left to empirical devices to identify patterns in the interplay between these factors and algorithms for sequential decision making.

The experiments conducted in our study take a step in this direction by focusing on test instances that highlight such trends. Nevertheless, the specific algorithms we have considered in this study do not represent the entire spectrum of VF and PS methods (and combinations thereof) that can be applied to decision making; rather, they represent the very ends of that spectrum. Sarsa(0) bootstraps learning updates assuming that given the representation, transitions obey the Markov property. The cross entropy method only uses the aggregate reward accumulated by a policy in evaluating it, and completely ignores individual state transitions. Our results illuminate the divergent properties of these contrasting approaches, while showing that their strengths can be combined. These results provide the incentive for a closer examination of the other methods that make up the spectrum. Here we describe three relevant classes of methods, which we plan to include in future extensions of this study.

- **Actor-Critic Methods** [3, 17] decouple the processes of policy evaluation (performed by the critic) and policy improvement (performed by the actor). Thus, the actor and critic can maintain separate data structures for learning; in particular, the actor can directly update the parameters of a policy (as in PS) while the critic estimates state values (as in VF). This separation can potentially reduce the variance in the policy evaluation step and lead to quicker convergence when compared to our implementation of PS. Actor-critic methods have been shown to be effective in practice [13].
- **Policy Gradient Methods** [1, 7] assume that the control policy is differentiable with respect to the policy parameters (which rules out the use of the "argmax" operator, used by both VF and PS in this study), and perform gradient descent updates to maximize long-term returns. While they do not learn the value function explicitly, they still make updates to the policy based on intermediate state transitions. Careful experimentation is necessary to determine whether this makes them more susceptible to deficient function approximation and partial observability than our PS implementation, which ignores intermediate state transitions.
- **Eligibility Traces** serve as a means to overcome partial observability in VF methods such as Sarsa. Loch and Singh [10] demonstrate their effectiveness in doing so on a number of discrete MDPs. Yet Stone et al. [16] report that eligibility traces do not have a significant impact when Sarsa is applied to Keepaway, which also has partial observability. Thus it becomes interesting to gauge the effect of the eligibility trace parameter λ under different values of the parameter σ used in our experiments to vary partial observability.

6. RELATED WORK

A number of applications from game-playing and robotic control have benefited from both VF and PS solutions; of work which has focused on *comparing* these methods, the most relevant to our approach is that of Taylor *et al.* [19]. Their test domain is also Keepaway, on which they compare Sarsa and NEAT [15], a PS method. As we notice in our experiments, they too observe that Sarsa degrades faster than NEAT when state noise is added. Additionally, they observe the opposite trend when action noise is added. We conjecture that this phenomenon is not significant in our experiments due to the relatively large number of evaluation runs for PS. Their analysis is based on 3 representative settings from a complex domain, while we adopt the alternative approach of conducting extensive experiments (with 1250 settings) on a much simpler class of domains. For comparing VF and PS on an equal footing, we enforce that they share a common representation. However, in the comparison performed by Taylor *et al.* [19], VF uses a tile-coding function approximation scheme, while NEAT uses an evolved neural network representation. Consequently, the absolute performances achieved by our VF and PS implementations are not readily comparable with theirs.

Our evidence that VF and PS methods may be combined successfully also finds support in the literature. Whiteson and Stone [23] consider performing value function-based learning *during* evaluations conducted by policy search; their algorithm, NEAT+Q improves the asymptotic performance of both VF and PS on the Mountain Car and job scheduling tasks. While we initialize PS using a policy learned from VF, Tesauro *et al.* initialize VF itself with a hand-coded policy. This benefits the early phase of training, as demonstrated on a complex resource allocation task. In the impressive application of reinforcement learning for helicopter control [11], a transition model is learned. Although no explicit value function is derived, the transition model is used to simulate experiences for PEGASUS, a PS method. The success of these diverse methods on specific applications align with our results obtained from a broad suite of experiments.

In our implementation of VF+PS, both on the suite of MDPs and on Keepaway, we manually fix the number of episodes after which to switch from VF to PS. This "transfer point" is chosen to coincide roughly with the time that VF begins to plateau. Likely, picking it before VF plateaus will initialize the PS phase with a less optimal policy; at the same time, delaying it past the time VF has "converged" might waste samples. Devising automatic means to determine the optimal "transfer point" becomes an avenue for future work. Indeed, Taylor and Stone [20] face a similar situation while performing behavior transfer between tasks.

7. CONCLUSION

VF and PS methods are contrasting approaches to solve reinforcement learning problems, which model realistic scenarios of autonomous agents seeking long-term gains in their interaction with an unknown environment. Both VF and PS methods rest on solid theoretical foundations, but these fail to provide performance guarantees when the agent has to cope with deficient function approximation and partial observability. We design an extensive experimental setup intended to tease apart the characteristics of VF and PS methods under such *practical* settings. Our experiments illustrate that VF methods enjoy superior sample complexity and asymptotic performance when provided precise function approximators and complete state information. Yet, PS methods display greater resilience to inadequate function approximation and noisy state information. We demonstrate that the desirable qualities of both methods can be combined effectively. The resulting method, VF+PS, is shown to achieve the best performance on a broad range of test settings and also on the challenging Keepaway task. The inferences drawn from our study encourage the further pursuit of research to integrate VF and PS methods. Promising areas of focus as next steps in this line of research have been identified throughout the paper.

8. ACKNOWLEDGMENTS

The authors thank Shimon Whiteson and anonymous reviewers of the current and previous versions of this paper for providing very useful comments. This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104, EIA-0303609 and IIS-0237699), DARPA (FA8750-05-2-0283, FA-8650-08-C-7812 and HR0011-04-1-0035), General Motors, and the Federal Highway Administration (DTFH61-07-H-00030).

9. REFERENCES

[1] J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[2] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, June 1957.

[3] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Incremental natural actor-critic algorithms. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 105–112. MIT Press, Cambridge, MA, 2008.

[4] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 2, pages 1023–1028, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[5] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, pages 1017–1023. MIT Press, 1996.

[6] P. T. De Boer, D. P. Kroese, S. Mannor, and R. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[7] S. Kakade. A natural policy gradient. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1531–1538. MIT Press, 2001.

[8] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, pages 611–616, July 2004.

[9] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[10] J. Loch and S. Singh. Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 323–331. Morgan Kaufmann, 1998.

[11] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[12] T. J. Perkins and D. Precup. A convergent form of approximate policy iteration. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1595–1602. MIT Press, Cambridge, MA, 2003.

[13] J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

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

[15] K. O. Stanley. Efficient evolution of neural networks through complexification. Technical Report AI-TR-04-314, Department of Computer Sciences, University of Texas at Austin, August 2004.

[16] P. Stone, R. S. Sutton, and G. Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

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

[18] I. Szita and A. Lőrincz. Learning Tetris using the noisy cross-entropy method. *Neural Computation*, 18:2936–2941, 2006.

[19] M. Taylor, S. Whiteson, and P. Stone. Comparing evolutionary and temporal difference methods for reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1321–28, July 2006.

[20] M. E. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, New York, NY, July 2005. ACM Press.

[21] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, 10(3):287–299, 2007.

[22] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[23] S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, May 2006.