In *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 17--24, IEEE, 2011.

On Learning with Imperfect Representations

Shivaram Kalyanakrishnan and Peter Stone
Department of Computer Science, The University of Texas at Austin
1616 Guadalupe St Suite 2.408 Austin Texas 78701 USA
{shivaram, pstone}@cs.utexas.edu

***Abstract*—In this paper we present a perspective on the relationship between learning and representation in sequential decision making tasks. We undertake a brief survey of existing real-world applications, which demonstrates that the classical "tabular" representation seldom applies in practice. Specifically, several practical tasks suffer from state aliasing, and most demand some form of generalization and function approximation. Coping with these representational aspects thus becomes an important direction for furthering the advent of reinforcement learning in practice. The central thesis we present in this position paper is that in practice, learning methods specifically developed to work with imperfect representations are likely to perform better than those developed for perfect representations and then applied in imperfect-representation settings. We specify an evaluation criterion for learning methods in practice, and propose a framework for their synthesis. In particular, we highlight the degrees of "representational bias" prevalent in different learning methods. We reference a variety of relevant literature as a background for this introspective essay.**

I. Introduction

Sequential decision making from experience, or reinforcement learning (RL) [1], is a well-suited paradigm for agents seeking to optimize long-term gains as they carry out sensing, decision and action in an unknown environment. RL tasks are commonly formulated as Markov Decision Problems (MDPs). The solution of MDPs has benefited immensely from a strong theoretical framework that has been developed over the years. The cornerstone of this framework is the *value function* of the MDP [2], which encapsulates the long-term utilities of decisions. Control policies can be suitably derived from value functions; indeed several algorithms provably converge to optimal policies in finite MDPs [3], [4]. Further, near-optimal behavior can be achieved after collecting a number of samples that is polynomial in the size of the state space ($|S|$) and the number of actions ($|A|$) [5], [6], using a memory bounded in size by $O(|S||A|)$ [7].

Unfortunately, a large section of the RL tasks we face in the real world cannot be modeled and solved exactly as finite MDPs. Not only are the traditional objectives of convergence and optimality inapplicable to a predominant number of tasks occurring in practice, in many of these tasks we cannot even ascertain the best performance that can be achieved, or how much training is necessary to achieve given levels of performance. What is the chief difference between the mainstream theory of RL — involving the learning of value functions — and its practice? The answer becomes apparent from a cursory survey of instances of RL in practice. In Table I we list a collection of applications of RL from the last two decades, gleaned by browsing several relevant journals, conference proceedings and citations contained therein. In all cases we ensure that the application itself, rather than the RL algorithm employed, is the primary focus of the relevant publication. Even if not comprehensive, Table I represents applications from a wide spectrum of domains.[1]

An inevitable handicap to learning in realistic applications is state aliasing (or *partial observability*), which affects a majority of the applications listed in Table I. In complex systems such as stock markets [22], physical environments [16], and cellular tissue [26], available measurements seldom suffice to capture all the information that can affect decision making. Nearly every agent embedded in the real world [18], [19], [21] receives noisy sensory information. The inadequacy of the sensory signal in identifying the underlying system state hinders the assumption of a Markovian interaction between the agent and the environment, on which the theoretical guarantees associated with most learning methods rely. Whereas coping with partial observability in a systematic manner is a well-studied problem, it is yet to scale to complex tasks with high-dimensional, continuous state spaces [30], [31].

Among the 20 applications presented in Table I, 11 involve continuous state spaces, which necessitate the use of function approximation in order to generalize. Indeed among the nine applications that have discrete state spaces, too, seven use some form of function approximation to represent the learned policy: their state spaces are too large for enumeration, and possibly even infinite. The use of function approximation negates the theoretical guarantees of achieving optimal behavior. Often the function approximation scheme used might not be capable of representing an optimal policy for a task; even when it is, seldom is it guaranteed that a learning algorithm will discover such a policy. Although there exist convergence guarantees for certain algorithms that use linear function approximation schemes [32], [33], [34], these do not provide practically effective lower bounds for the values of the learned policies. Also, convergence results rarely extend to situations in which non-linear representations such as neural networks are used; yet non-linear representations are used commonly in practice, as apparent from Table I.

[1]Other independently-compiled surveys of sequential decision making applications corroborate the observations we draw based on Table I. Langley and Pendrith [8] describe several RL applications presented at a symposium organized around the topic; Szepesvári lists numerous applications from the control and approximate dynamic programming literature at this URL: http://www.ualberta.ca/~szepesva/RESEARCH/RLApplications.html.

TABLE I

CHARACTERIZATION OF SOME POPULAR APPLICATIONS OF REINFORCEMENT LEARNING. "POLICY REPRESENTATION" DESCRIBES THE UNDERLYING REPRESENTATION FROM WHICH THE POLICY IS DERIVED. A "NEURAL NETWORK" REPRESENTATION IS NON-LINEAR, INCORPORATING AT LEAST ONE HIDDEN LAYER OF UNITS. UNDER TILE CODING, "#FEATURES" INDICATES THE NUMBER OF STATE VARIABLES, RATHER THAN THE NUMBER OF INDIVIDUAL TILES.

Task	**State Observability**	**State Space**	**Policy Representation (#Features)**
Backgammon [9]	Complete	Discrete	Neural network (198)
Job-shop scheduling [10]	Complete	Discrete	Neural network (20)
Tetris [11], [12]	Complete	Discrete	Linear (20-50)
Elevator dispatching [13]	Partial	Continuous	Neural network (46)
Acrobot control [14]	Complete	Continuous	Tile coding (4)
Dynamic channel allocation [15]	Complete	Discrete	Linear (100's)
Active guidance of finless rocket [16]	Partial	Continuous	Neural network (14)
Fast quadrupedal locomotion [17]	Partial	Continuous	Parameterized policy (12)
Robot sensing strategy [18]	Partial	Continuous	Linear (36)
Helicopter control [19]	Partial	Continuous	Neural network (10)
Dynamic bipedal locomotion [20]	Partial	Continuous	Feedback control policy (2)
Robot obstacle negotiation [21]	Partial	Continuous	Linear (10)
Optimized trade execution [22]	Partial	Discrete	Tabular (2-5)
9 $\times$ 9 Go [23]	Complete	Discrete	Linear ($\approx$1.5 million)
Ms. Pac-Man [24]	Complete	Discrete	Rule List (10)
General game playing [25]	Complete	Discrete	Tabular (over part of state space)
Adaptive epilepsy treatment [26]	Partial	Continuous	Extremely randomized trees (114)
Computer memory scheduling [27]	Complete	Discrete	Tile coding (6)
Motor skills [28]	Partial	Continuous	Motor primitive coefficients (100's)
Combustion Control [29]	Partial	Continuous	Parameterized policy (2-3)

The theoretical justifications of algorithms derived to work with the classical "tabular" representation are nullified in practice by several factors (e.g., demands on exploration, constraints on computation and memory, and the dimensionality of the action space). As the "first order" factors among these, we focus on state aliasing and generalization, which determine the quality of the representation used for learning. By the *representational bias* of a learning method we mean the extent to which the method's success depends on having a good representation. Thus, a method that will only succeed with a perfect tabular representation (say it will diverge if there is any state aliasing at all) has the strongest representational bias; methods that can still achieve reasonable performance in the presence of state aliasing and poor generalization would have a weak representational bias.

As evidenced by the references included in Table I, a common strategy adopted in practice is to apply algorithms derived under strong representational biases, and to empirically verify that they remain effective when assumptions on the representation are relaxed. To facilitate this approach, much manual effort is expended in designing schemes to mitigate the adverse effects of partial observability and function approximation. In addition, active lines of research focus on developing adaptive methods for reducing state aliasing [31] and improving function approximation [35].

It remains that even in situations where representation can be adaptively improved, the undesirable effects of state aliasing and function approximation are invariably only reduced, and not eliminated. Recognizing that this is an inevitable shortcoming in practice, we propose that representation discovery needs to be complemented by developing learning methods that can operate with varying degrees of representational strength. As depicted in Figure 1, we view the relationship between representation and learning as a conjunctive one, aimed at maximizing performance in the task at hand. The learning mechanism is in charge of adapting a set of "weights" (typically real-valued), while other modules undertake feature and hierarchy construction, state estimation, and so on. In this paper we specifically consider the natural implication of this relationship on the design and analysis of learning algorithms:

> How well do different reinforcement learning methods perform in the *presence* of state aliasing and function approximation; can we develop methods that are both sample efficient and capable of achieving high asymptotic performance in their presence?

The main contributions of this paper are (a) the insight and detailed argument that we need to develop learning algorithms specifically for imperfect representations, (b) an extensive survey of relevant background literature, and (c) the outline of broad research agenda for developing performance-oriented learning methods. This paper is organized as follows. We survey existing lines of research that have specifically addressed issues in state aliasing (Section II), and generalization and function approximation (Section III). Concluding that a perfect representation is an impractical ideal, in Section IV we specify the desirable properties for a learning method to possess in practice. Section V then lays down an instructive framework for developing effective learning methods for practical sequential decision making tasks. We conclude with a summary in Section VI.

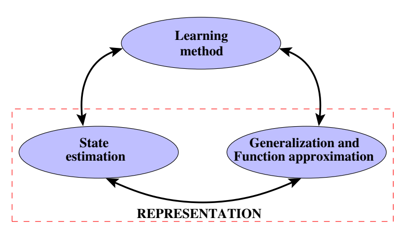

Fig. 1. A schematic depiction of the conjunctive relationship between learning and representation discovery (itself comprising state estimation and generalization). The modules need to complement each other in order to achieve high long-term reward on a specified task.

II. State Aliasing

In an MDP the current system state and action completely determine the dynamics of the ensuing transition. However, in a number of RL applications, perceptual aliasing [36] and noisy sensors [21] deny an agent direct access to the underlying system state. Thus, it becomes necessary to distinguish the agent's *observation* at any instant of time from the environment's *state* at the same instant. For example, in a soccer game, the state of the system might be fixed by the positions and velocities of the players and the ball. However, for a player who can only see within a restricted field of view, his visual percepts do not convey information about objects that lie outside it. Even with an unrestricted field of view, a single visual snapshot would not identify the velocities of the players and the ball. Further, a camera with finite resolution and noisy pixels would lead to incorrect estimates of the objects' positions, too. In all these cases and in a vast number of practical applications, the agent's observation at any instant does not uniquely identify the underlying system state.

In practice, significant manual effort is expended in designing an observed state signal that encapsulates as much information as possible about the underlying state. Past observations and actions, and other useful sources of information are combined to construct the observed state, making extensive use of domain knowledge. Thus, it is common for robot soccer agents to use particle filtering to estimate positions and velocities of objects based on their past configurations, and by applying the laws of physics.

In principle, an agent can use a record of its entire history of past observations and actions in order to reconstruct the system state. The seminal work of Åström [37] demonstrates that by keeping a "belief state" that is updated based on incoming observations, an agent can eventually disambiguate states perfectly. However, the complexity of solving the resulting Partially Observable MDP (POMDP) [38] is forbidding even in the context of planning (with known transition dynamics) [30], and is yet to scale to large problems [31].

In early work in the context of *learning* under state aliasing, McCallum [39] designs a Utile Distinction Memory (UDM) in which states are distinguished only if doing so increases the utility of decision making. The appeal of this approach is that the agent has a dynamic memory size that is adapted based on the needs of decision making, rather than on the underlying complexity of the state space. Prediction suffix trees (PSTs) can be used as the data structure indexing memory: they can speed up learning significantly [40] while achieving perfect disambiguation of states in finite MDPs with deterministic transition and observation functions [41].

It remains that the principled approaches discussed above seldom scale to problems with large, possibly continuous state spaces. With the small, finite number of samples typically available to a learning agent in practice, it is rarely possible to disambiguate aliased states perfectly. The predominant approach adopted in these cases — for example, in several applications listed in Table I — is to treat an "observed state" as though it were a state obeying the Markov property, even if it does not. Of course, this approach devalues the theoretical justifications for the learning algorithm being used, and the performance achieved essentially depends on the (unknown) degree of the devaluation [22]. If observed states do not contain all the "sufficient statistics" of the true underlying state, is it better to learn policies using optimization methods that do not treat observed states as states in an MDP?

Glickman and Sycara [42] show that even in severely occluded maze tasks, a small neural network with recurrent connections, interpreted as a stochastic policy, can be evolved to achieve excellent performance. Loch and Singh demonstrate that the use of eligibility traces — through Sarsa(λ) with high values of λ, which favor true returns more than bootstrapped estimates — perform reasonably well on a suite of benchmark POMDPs. In Section V we essentially argue the need to develop similar learning methods, which explicitly account for the possibility of state aliasing. The other important aspect affecting the performance and properties of learning methods is function approximation, which we next visit.

III. Generalization and Function Approximation

State aliasing handicaps learning by invalidating the Markov property in observed state transitions. However, even in fully observable worlds — when an agent's observation can unambiguously identify the system state — the sheer size of the state space can pose a formidable challenge to learning. Consider: a policy maps observed states to actions, and hence a general representation for learning must maintain a data structure that can assign an *arbitrary* action to *every* observed state. The number of parameters that would have to be stored and updated during learning in order to do so would be too large for most tasks occurring in practice. This necessitates the use of generalization, whereby a smaller set of parameters approximates the learned policy. Even if generalization is strictly not necessary for learning over finite state and action spaces (as it is for infinite state and action spaces), it can still promote quicker learning in finite MDPs.

As with other areas of artificial intelligence and machine learning, feature design has a significant effect on the performance of RL algorithms [9]. In nearly every deployed application, much manual effort is invested in designing effective features and a representation. In addition, several ideas have been proposed towards adaptive generalization and function approximation. These include architectures for state abstraction [43], subgoal discovery [44], and the hierarchical organization of control [45]. Temporal abstraction is achieved through the framework of *options* [46]. Mahadevan's "proto-value" functions [47] identify low-dimensional manifolds in the state space. Other feature selection and discovery approaches apply in the specific contexts of kernels [48], variable resolution discretization [49], genetic programming [50] and Gaussian processes [51].

It is convenient to conceptually separate the overall representation of a policy into (a) a vector of features, ϕ, describing the observed state, (b) a functional form, ρ, such as a linear representation, a decision tree or a multi-layer neural network, and (c) a vector of weights $\mathbf{w}$, which are the parameters updated during learning. As seen in examples cited previously, in some cases these components are conflated, and adapted simultaneously. Yet, separating the learning agent's architecture into $\boldsymbol{\phi}$, ρ, and $\mathbf{w}$ as above enables us to ponder how $\mathbf{w}$ should be treated if $\boldsymbol{\phi}$ and ρ are necessarily *insufficient* for representing the optimal value function or policy. Within the mainstream RL literature, the predominant focus has been on updating $\mathbf{w}$ such that the representation *approximates the value function* well. In the remainder of this section, we summarize the long line of research in value function approximation.

A bulk of the research in learning with function approximation has focused on linear architectures, and mostly so in the context of prediction, i.e., estimating the value function of a fixed policy (without policy improvement). An early result due to Sutton [52] establishes that TD(0) with linear function approximation converges when the features used are linearly independent; Dayan and Sejnowski [53] extend this result to TD(λ), $\forall \lambda \in [0, 1]$, while Tsitsiklis and Van Roy [54] show convergence for the more realistic case of infinite state spaces and linearly dependent features. Although most results for the convergence of linear TD (temporal difference) learning are for estimating values of the policy that is used to gather experiences, the more general case of off-policy learning has also been addressed [55], [56].

The main challenges in learning approximate value functions on-line arise due to the nonstationarity and bias *in the targets* provided to the function approximator [57]. As a consequence, the best theoretical guarantees for learning control policies with approximate schemes come with several restrictions. Most results are again limited to linear function approximation schemes; in addition, some make demands such as Lipschitz continuity of the policy being learned [33], and favorable initial conditions [58]. In recent work, Maei *et al.* [34] introduce the Greedy-GQ algorithm, which provably converges while making off-policy learning updates to a linear function approximator. However, Greedy-GQ requires that the policy followed while learning stay fixed, preventing the agent from actively exploring based on the experiences it gathers. Such a requirement curtails the algorithm's practical utility.

In addition to on-line TD methods, batch and model-based methods have also been studied for use in conjunction with linear function approximation. For instance, recent work has provided convergence guarantees under certain restrictions for a linear implementation of Dyna [59], a classical model-based RL algorithm. In the context of batch RL, the most popular approaches with linear architectures are least squares methods. Least Squares Temporal Difference (LSTD) learning [60] is an efficient procedure to compute a value function from a given batch of transition data; given infinite data the algorithm converges to the same set of weights as TD(0). Least Squares Policy Iteration (LSPI) [61] extends LSTD to control problems, computing a sequence of policies from a *fixed* set of data, such that successive iterations are likely to yield better policies. An advantage of batch least squares approaches when compared to on-line methods is that they involve less parameter tuning (they do not need a learning rate parameter). In turn, batch methods incur greater overheads in memory and computation.

In general, approximate architectures are incapable of representing the optimal action value function Q^*. Even with architectures capable of representing Q^*, it is not guaranteed that an optimal set of parameters $\mathbf{w}$ will be found, as *intermediate* value functions encountered during learning might yet prove problematic for approximation [62]. Further, even if a value function is approximated well, as defined by its Bellman error weighted by some distribution of states, greedy action selection might yet pick suboptimal actions in regions of inaccurate approximation, resulting in low long-term returns [63]. Thus, guarantees on the convergence of algorithms do not necessarily translate into effective guarantees about the long-term reward that will be accrued at convergence [33], [34].

Typical bounds on the sub-optimality of a policy that is greedy with respect to the best approximate value function [64], [65] are (a) directly proportional to ϵ, an error term expressing the inherent limitations of the feature combination, and (b) inversely proportional to $(1-\gamma)$, where γ is the discount factor in the MDP. These bounds are extremely loose in most practical settings: typically ϵ is not even known, and γ is set close to 1 (often 0.9 or higher [26], [27]). Interestingly enough, it is *not necessary* to approximate the optimal action value function well in order to induce the optimal policy [66]. This observation motivates our specification of what a learning method need indeed strive to achieve, which we next present.

IV. Demands of a Practical Learning Method

Let us assume that a learning agent employs an underlying function representation ρ, and at each time step t, receives as input a vector of features $\boldsymbol{\phi}_t$, depending on the state s_t. The applications mentioned in Table I and the discussion in the previous sections affirm that it is realistic, and perhaps most appropriate, to expect that ρ and $\boldsymbol{\phi}_t$ facilitate *some* amount useful generalization, but they are not perfect. Given this limitation, what is demanded of the learning method in adapting the representation parameters $\mathbf{w}$?

The output of the learning agent at time t is an action a_t, based on which the environment will generate a next state s_{t+1} (partially visible to the agent through features $\boldsymbol{\phi}_{t+1}$) and a reward r_t. In some situations it is necessary to measure the rewards accrued while learning (on-line) [27], while in others we can assume that the learner outputs an entire policy at every step, which can then be evaluated off-line, with no further learning [13], [19]. In either case, the learner is evaluated based on the long-term reward it accrues. In fact, the greatest appeal of RL [1] lies in the manner desirable behavior can be specified succinctly through a scalar-valued reward function. Correspondingly, we find it natural that the objective of learning must be to maximize expected long-term reward. Surely, the developers of the 20 applications mentioned in Table I will agree!

Of late, several problems in machine learning have profited through "declarative" specifications of an objective function to optimize by adapting a vector of parameters. For example, classification is posed as the problem of finding real-valued coefficients to maximize a separating margin, or to minimize squared error. Our specification for learning in sequential decision making problems is similar:

> Given ϕ and ρ, adapt $\mathbf{w}$ based on experience such that the expected long-term reward of the resulting policy is maximized, while the number of learning samples is minimized.

It is amply clear that if an agent possesses a perfect representation, maximizing long-term reward is exactly achieved by learning the optimal action value function [2], which it is possible to achieve in a sample-efficient manner [6]. However, the key implication of our definition is on learning with *imperfect* representations. If state aliasing and generalization handicap the representation, how can $\mathbf{w}$ be adapted such that a policy with high long-term reward is achieved with relatively few training samples? *Approximating* the model or value function is no longer the obvious, provably correct answer to our question. In some cases, the predictive power of such approaches might be desired as an end in itself, but here we solely demand performance, a pragmatic measure.

Our formulation of the problem as above does not automatically imply a solution strategy.[2] In fact, our formalisms of ϕ, ρ and $\mathbf{w}$ are not even defined rigorously enough that performance and sample efficiency can be characterized precisely in terms of them. Nevertheless, our rough, qualitative statement is a fair reflection of the true needs of a majority of practical learning agents. Equally, we believe a quest for answers must be in the spirit exhorted by the famed American statistician, John Tukey [67, see pp.13–14]:

> *"... Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise."*

In the following section, we outline a framework for better understanding and utilizing relationships between learning methods and representations. We consider such a strategy a practical approach for developing general, automatic RL solutions for realistic tasks.

V. Towards Learning Methods for Practice

In an early paper on the subject of representation in RL, Cobb [68] separates the inductive biases in a reinforcement learner into "language" and "procedural" biases. In rough terms, the former bias corresponds to the representation used by the learner (what we call "representational" bias), while the latter bias corresponds to the learning method itself. The argument that falls out, and which this paper reiterates, is the need to appropriately match the two biases in order to improve performance. A wealth of evidence suggests that the greatest benefit can be achieved by adapting the representation while learning [35], [69].

With the ultimate aim of maximizing an agent's long-term gains, indeed we believe representation and learning must go hand in hand. Methods to develop better representations have long been researched; yet, as detailed in Sections II and III, we believe that there are practical limits to what representation discovery alone can achieve. In this paper we consider the complementary question: how learning methods must be adapted to work with imperfect representations. We propose a two-stage program for organizing future research on this question. First, we argue the need for systematically evaluating the representational bias of existing classes of RL methods: in Section V-A we put forth a candidate hypothesis, derived from qualitative arguments, to serve as a basis for an evaluative study. Next, in Section V-B we motivate the need for developing general strategies of selecting and integrating learning methods for sequential decision making.

A. Evaluative Study

Solutions to sequential decision making problems are policies, which are mappings from states (implicitly assumed *observed*) to actions. In order to learn policies, methods generalize based on a set of experiences that register the effects of actions an agent has taken from the states it has visited. We posit that the fewer the assumptions a method makes about associations between states (and between states and values), the less it will suffer due to state aliasing and function approximation, both of which constrain the scope of functions that can be learned over the state space. On the other hand, exploiting associations between states is precisely the route to achieving sample efficiency. Thus, there exists a tension between sample efficiency on the one hand, and resilience to imperfect representations on the other (henceforward, referred to as a method's *robustness*). How do different classes of methods trade off these conflicting objectives? Figure 2 summarizes our candidate hypothesis for ordering different classes of methods, which we now describe.

Model-based methods [1, see ch. 9] rely on simulated experiences that are generated by an environmental model; this model is itself computed from transition samples. Hence, the assumption on which model-based methods bank is that an accurate model of the environmental transition and reward functions can be represented and learned based on observed experiences. This is a stronger assumption than that made by on-line model-free value function-based (VF) methods [3], [70], which only assume that observed samples can be used in computing state-action *values*. We consider batch RL methods [61], [71] similar to model-based methods in that they perform multiple passes, or aggregate updates, based on a set of experiences. Note that VF methods make only a single update based on each transition. In line with the view that the data used by batch methods themselves constitute an implicit model [7], [72], we treat model-based and batch methods as

[2] Throughout this paper, by "reinforcement learning" we refer to the *problem* of sequential decision making from experience, and not to any specific class of learning methods. Such an interpretation is consistent with Sutton and Barto [1, see ch. 3].

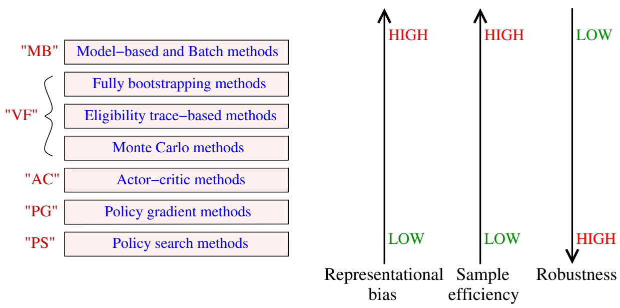

Fig. 2. A candidate hypothesis ordering classes of learning methods based on the strength of their reliance on observed samples for learning. Eligibility trace methods span a segment of the spectrum between fully bootstrapping methods ($\lambda = 0$) and Monte Carlo methods ($\lambda = 1$).

one category (abbreviated "MB"): the one with the highest representational bias.

VF methods make progress towards the goal of approximating the value function by constantly shifting their current estimates towards "better" estimates derived from transition samples. Among this class itself, the extent of bootstrapping could determine the sample efficiency and robustness of different methods. Fully bootstrapping methods such as Sarsa(0) [70] estimate values of states using the estimated values of next states, while Monte Carlo methods such as Sarsa(1) estimate state values based on samples of long-term reward. This makes Monte Carlo methods less dependent on the state signal and value function representation than fully bootstrapping methods. In general, controlling the eligibility trace parameter λ yields intermediate methods, such as Sarsa(λ), which implement varying extents of bootstrapping.

While VF methods still try to estimate state-action values, policy gradient (PG) methods [73], [74] only estimate the *gradient of the value of a policy* with respect to its parameters, which is achieved by summing actual returns, as under Monte Carlo methods. Policies contain less information than their action value functions, and consequently, can be expressed with a smaller number of parameters. The aggregation of information gathered from multiple transitions to perform an update on a small number of parameters has the effect of making PG methods less dependent on each individual transition. Actor-critic (AC) methods [32] are much like PG methods, but they rely on a "critic" learning values (to reduce the variance in gradient estimates), as under VF methods. The values learned by the critic do not directly yield a control policy; rather, they guide the improvement of the "actor's" control policy. Thus, we place AC methods in between VF and PG methods in the spectrum shown in Figure 2.

By "policy search" (PS) methods we refer to a generic class of optimization algorithms (for example, genetic and evolutionary algorithms [75], hill climbing, and CMA-ES [76]) that primarily rely on estimating the *ranks* among a population of policies with respect to their values. Ranks are typically easier to estimate than gradients; indeed PS methods can work perfectly well on policies that do not have analytically computable gradients, which PG methods usually require. Their disregard for everything but the relative order of the values among the current set of policies makes PS methods the least dependent on atomic state transitions for the purpose of learning.

It is unlikely that across so broad a scope as practical RL applications, different learning algorithms will fall neatly into the pockets presented in Figure 2 and exhibit characteristics consistent with the order predicted. The proposed order is merely intended as a pivot around which a broad experimental study can be undertaken. A lack of space prevents us from providing a detailed account of existing studies comparing algorithms for RL [77], [78], [79], [80]. By and large, such studies have been confined to isolated tasks such as pole balancing, Mountain Car and Keepaway soccer, and typically, different learning methods have been paired with different representations. Nonetheless, the trends reported in the literature predominantly concur with our supposition about the representational bias of different classes of methods, which is directly correlated with sample efficiency, and inversely with robustness. Essentially, we call for further studies of this nature, in particular with an emphasis on (a) carefully controlling representational aspects, and (b) comparing methods from the entire spectrum we have presented. A general theory on representation and learning is essential for developing automated RL techniques, since in practice, representation is imperfect to varying degrees.

B. *Method Selection and Integration*

As we hone our understanding of which methods perform best under different representations, the next step would be to automate the selection of the method to apply to the task at hand, assuming the quality of its representation can be estimated. Today "meta-learning" [81] and "algorithm portfolio" techniques [82] are being applied quite successfully to problems such as search [83]. Successfully replicating these strategies for RL is surely easier said than done, given the sheer complexity and the number of dimensions underpinning RL problems. We intend the ideas presented in this paper to serve as a guide for a more concrete research program.

In moving forward, another important strategy to actively explore is the development of more hybrid learning architectures, which can combine the strengths of existing methods. A growing line of research already addresses this need. Under the "Value and Policy Search" algorithm introduced by Baird and Moore [84], VF and PG methods can be combined naturally in a specified ratio. The "NEAT+Q" algorithm [35] undertakes Q-learning during fitness evaluations of members of a neural network population whose topologies are being evolved. Menache *et al.* [85] employ an optimization method for tuning the widths and positions of basis functions, which are linearly combined to approximate the value function. Several approaches have considered quickly learning a reasonable policy using VF methods, and then refining it through policy search [78], [86]. The famous helicopter control application due to Ng *et al.* [19] is an instance of policy search performed within a model inferred from grounded experiences. All these examples illustrate the merits of combining qualitatively different learning methods to obtain both sample efficiency and resistance to deficient representations. It would indeed be a worthwhile pursuit to translate the lessons learned from such isolated instances into general design principles.

VI. Summary

At the root of the reasoning that motivates a majority of RL methods is the formulation of sequential decision making tasks as finite, discrete MDPs. When these methods are applied to tasks in practice, they lose many of their desirable properties, such as achieving optimality or even converging. The chief reason for this disparity between theory and practice is the insufficiency of representation: specifically the inability to overcome state aliasing and inevitable errors in generalization and function approximation. Several existing lines of research address the important question of automatically improving representation. In this paper we argue for the relevance of the complementary perspective: to pick and tailor learning methods to work with imperfect representations. In this context, we specify that the ultimate objective of learning is to achieve high long-term reward after a minimal amount of training. It is incidental that this objective is exactly achieved using model-based and value function-based methods when the representation is perfect. When representations are imperfect, approximating the model or value function is to be taken as a means, and not the end.

Recognizing the need to embrace learning and representation in conjunction, we propose to undertake an extensive experimental study to order qualitatively different classes of RL methods as a function of their representational bias. We advance a plausible conjecture for such a study, which could lay the seeds for automatically selecting the best method for a given task and representation, and subsequently combining the strengths of different methods through hybrid architectures. Ultimately, the methodology we propose is an early and necessary step in the direction of realizing robust and automated learning methods for sequential decision making in practice.

Acknowledgments

The authors thank Matthew Hausknecht, Todd Hester, and anonymous reviewers of this paper for their suggestions towards improving it. This work has taken place in the Learning Agents Research Group (LARG) at the University of Texas at Austin. LARG is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

References

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

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

[3] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992.

[4] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Mach. Learn.*, vol. 38, no. 3, pp. 287–308, 2000.

[5] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, no. 2–3, pp. 209–232, 2002.

[6] R. I. Brafman and M. Tennenholtz, "R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning," *J. Mach. Learn. Res.*, vol. 3, pp. 213–231, 2003.

[7] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proc. ICML 2006*. ACM, 2006, pp. 881–888.

[8] P. Langley and M. Pendrith, "Symposium on applications of reinforcement learning: Final report for NSF Grant IIS-9810208," Institute for the Study of Learning and Expertise, Tech. Rep., March 1998. [Online]. Available: http://www.isle.org/symposia/reinf/final.pdf

[9] G. Tesauro, "Practical issues in temporal difference learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 257–277, May 1992.

[10] W. Zhang and T. G. Dietterich, "A reinforcement learning approach to job-shop scheduling," in *Proc. IJCAI 1995*. Morgan Kauffman, 1995, pp. 1114–1120.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[12] C. Thierry and B. Scherrer, "Building controllers for Tetris," *Int. Comp. Games Assoc. J.*, vol. 32, no. 1, pp. 3–11, March 2010.

[13] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 1017–1023.

[14] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996, pp. 1038–1044.

[15] S. Singh and D. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," in *Advances in Neural Information Processing Systems 9*. MIT Press, 1997, pp. 974–980.

[16] F. J. Gomez and R. Miikkulainen, "Active guidance for a finless rocket using neuroevolution," in *Proc. GECCO 2003*. Springer, 2003, pp. 2084–2095.

[17] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *Proc. AAAI 2004*. AAAI Press, 2004, pp. 611–616.

[18] C. Kwok and D. Fox, "Reinforcement learning for sensing strategies," in *Proc. ICRA 2004*. IEEE, 2004, pp. 3158–3163.

[19] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry, "Autonomous helicopter flight via reinforcement learning," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[20] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *Proc. IROS 2004*. IEEE, 2004, pp. 2849–2854.

[21] H. Lee, Y. Shen, C.-H. Yu, G. Singh, and A. Y. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *Proc. ICRA 2006*. IEEE, 2006, pp. 3003–3010.

[22] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement learning for optimized trade execution," in *Proc. ICML 2006*. ACM, 2006, pp. 673–680.

[23] D. Silver, R. S. Sutton, and M. Müller, "Reinforcement learning of local shape in the game of Go," in *Proc. IJCAI 2007*, 2007, pp. 1053–1058.

[24] I. Szita and A. Lőrincz, "Learning to play using low-complexity rule-based policies: Illustrations through Ms. Pac-Man," *J. Art. Int. Res.*, vol. 30, pp. 659–684, 2007.

[25] H. Finnsson and Y. Björnsson, "Simulation-based approach to General Game Playing," in *Proc. AAAI 2008*. AAAI Press, 2008, pp. 259–264.

[26] A. Guez, R. D. Vincent, M. Avoli, and J. Pineau, "Adaptive treatment of epilepsy via batch-mode reinforcement learning," in *Proc. AAAI 2008*. AAAI Press, 2008, pp. 1671–1678.

[27] E. İpek, O. Mutlu, , J. Martínez, and R. Caruana, "Self-optimizing memory controllers: A reinforcement learning approach," in *Proc. ISCA 2008*. IEEE Press, 2008, pp. 39–50.

[28] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[29] N. Hansen, A. S. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Trans. Evo. Comp.*, vol. 13, no. 1, pp. 180–197, February 2009.

[30] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *Proc. AAAI 1994*. AAAI Press, 1994, pp. 1023–1028.

[31] J. Pineau, G. J. Gordon, and S. Thrun, "Anytime point-based approximations for large POMDPs," *J. Art. Int. Res.*, vol. 27, pp. 335–380, 2006.

[32] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Opt.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[33] T. J. Perkins and D. Precup, "A convergent form of approximate policy iteration," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2003, pp. 1595–1602.
[34] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *Proc. ICML 2010*. Omnipress, 2010, pp. 719–726.
[35] S. Whiteson and P. Stone, "Evolutionary function approximation for reinforcement learning," *J. Mach. Learn. Res.*, vol. 7, pp. 877–917, 2006.
[36] S. D. Whitehead and D. H. Ballard, "Learning to perceive and act by trial and error," *Mach. Learn.*, vol. 7, no. 1, pp. 45–83, 1991.
[37] K. J. Åström, "Optimal control of Markov Processes with incomplete state information," *J. Math. Analysis App.*, vol. 10, pp. 174–205, January 1965.
[38] G. E. Monahan, "A survey of Partially Observable Markov Decision Processes: Theory, models, and algorithms," *Management Sci.*, vol. 28, no. 1, pp. 1–16, January 1982.
[39] R. A. McCallum, "Overcoming incomplete perception with utile distinction memory," in *Proc. ICML 1993*. Morgan Kauffman, 1993, pp. 190–196.
[40] A. K. McCallum, "Reinforcement learning with selective perception and hidden state," Ph.D. dissertation, Comp. Sci. Dept., U. Rochester, 1996.
[41] M. P. Holmes and C. L. Isbell, Jr, "Looping suffix tree-based inference of partially observable hidden state," in *Proc. ICML 2006*. ACM, 2006, pp. 409–416.
[42] M. R. Glickman and K. Sycara, "Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state," in *Proc. ICML 2001*. Morgan Kauffman, 2001, pp. 194–201.
[43] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Art. Int. Res.*, vol. 13, pp. 227–303, 2000.
[44] B. L. Digney, "Learning hierarchical control structures for multiple tasks and changing environments," in *From animals to animats 5*. MIT Press, 1998, pp. 321–330.
[45] R. E. Parr, "Hierarchical control and learning for Markov Decision Processes," Ph.D. dissertation, U. California Berkeley, 1998.
[46] R. S. Sutton, D. Precup, and S. P. Singh, "Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Art. Int.*, vol. 112, no. 1–2, pp. 181–211, 1999.
[47] S. Mahadevan, "Learning representation and control in Markov decision processes: New frontiers," *Found. Trends Mach. Learn.*, vol. 1, no. 4, pp. 403–565, 2009.
[48] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Mach. Learn.*, vol. 49, no. 2–3, pp. 161–178, 2002.
[49] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Mach. Learn.*, vol. 49, no. 2–3, pp. 291–323, 2002.
[50] S. Girgin and P. Preux, "Feature discovery in reinforcement learning using genetic programming," in *Proc. EuroGP 2008*. Springer, 2008, pp. 218–229.
[51] T. Jung and P. Stone, "Feature selection for value function approximation using Bayesian model selection," in *Proc. ECML PKDD 2009*. Springer, 2009, pp. 660–675.
[52] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, August 1988.
[53] P. Dayan and T. J. Sejnowski, "TD(λ) converges with probability 1," *Mach. Learn.*, vol. 14, pp. 295–301, 1994.
[54] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Auto. Cont.*, vol. 42, no. 5, pp. 674–690, 1997.
[55] D. Precup, R. S. Sutton, and S. Dasgupta, "Off-policy temporal difference learning with function approximation," in *Proc. ICML 2001*. Morgan Kaufmann, 2001, pp. 417–424.
[56] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proc. ICML 2009*. ACM, 2009, pp. 993–1000.
[57] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proc. 1993 Connectionist Models Summer School*. Lawrence Erlbaum, 1993, pp. 255–263.
[58] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proc. ICML 2008*. ACM, 2008, pp. 664–671.
[59] R. S. Sutton, C. Szepesvári, A. Geramifard, and M. Bowling, "Dyna-style planning with linear function approximation and prioritized sweeping," in *Proc. UAI 2008*. AAAI Press, 2008, pp. 528–536.
[60] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Mach. Learn.*, vol. 22, pp. 33–57, 1996.
[61] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, 2003.
[62] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 369–376.
[63] S. Kalyanakrishnan and P. Stone, "Batch reinforcement learning in a complex domain," in *Proc. AAMAS 2007*. IFAAMAS, 2007, pp. 650–657.
[64] S. P. Singh and R. C. Yee, "An upper bound on the loss from approximate optimal-value functions," *Mach. Learn.*, vol. 16, no. 3, pp. 227–233, 1994.
[65] R. J. Williams and L. C. Baird III, "Tight performance bounds on greedy policies based on imperfect value functions," in *Proc. Tenth Yale Workshop on Adaptive and Learning Systems*. Center for Systems Science, Yale University, 1994.
[66] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Art. Int. Res.*, vol. 15, pp. 319–350, 2001.
[67] J. W. Tukey, "The future of data analysis," *Annals Math. Stat.*, vol. 33, no. 1, pp. 1–67, March 1962.
[68] H. G. Cobb, "Inductive biases in a reinforcement learner," Navy Center for Applied Research in Artificial Intelligence, Washington DC, USA, Tech. Rep. AIC-92-013, 1992.
[69] H. G. Cobb and P. Bock, "Using a genetic algorithm to search for the representational bias of a collective reinforcement learner," in *Proc. PPSN III*. Springer, 1994, pp. 576–587.
[70] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge U. Engg. Dept., CUED/F-INFENG/TR 166, September 1994.
[71] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3–4, pp. 293–321, 1992.
[72] J. A. Boyan, "Least-squares temporal difference learning," *Mach. Learn.*, vol. 49, no. 2–3, pp. 233–246, 2002.
[73] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems 12*. MIT Press, 2000, pp. 1057–1063.
[74] S. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 1531–1538.
[75] K. O. Stanley, "Efficient evolution of neural networks through complexification," Ph.D. dissertation, Dept. Comp. Sci., U. Texas Austin, 2004.
[76] N. Hansen, *The CMA Evolution Strategy: A Tutorial*, January 2009. [Online]. Available: http://www.lri.fr/~hansen/cmatutorial.pdf
[77] S. Whiteson, M. E. Taylor, and P. Stone, "Critical factors in the empirical performance of temporal difference and evolutionary methods for reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 1, pp. 1–35, 2010.
[78] S. Kalyanakrishnan and P. Stone, "An empirical analysis of value function-based and policy search reinforcement learning," in *Proc. AAMAS 2009*. IFAAMAS, 2009, pp. 749–756.
[79] V. Heidrich-Meisner and C. Igel, "Similarities and differences between policy gradient methods and evolution strategies," in *Proc. ESANN 2008*. D-side Publication, 2008, pp. 149–154.
[80] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.
[81] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Art. Int. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.
[82] C. P. Gomes and B. Selman, "Algorithm portfolios," *Art. Int.*, vol. 126, no. 1–2, pp. 43–62, 2001.
[83] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla: Portfolio-based algorithm selection for SAT," *J. Art. Int. Res.*, vol. 32, pp. 565–606, 2008.
[84] L. Baird and A. Moore, "Gradient descent for general reinforcement learning," in *Advances in Neural Information Processing Systems 11*. MIT Press, 1999, pp. 968–974.
[85] I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals Op. Res.*, vol. 134, no. 1, pp. 215–238, 2005.
[86] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proc. ICML 2002*. Morgan Kauffman, 2002, pp. 227–234.