Protecting Against Evaluation Overfitting in Empirical Reinforcement Learning

Shimon Whiteson*, Brian Tanner†, Matthew E. Taylor‡, and Peter Stone§
*Informatics Institute, University of Amsterdam
†Department of Computing Science, University of Alberta
‡Department of Computer Science, Lafayette College
§Department of Computer Science, University of Texas at Austin

***Abstract*—Empirical evaluations play an important role in machine learning. However, the usefulness of any evaluation depends on the *empirical methodology* employed. Designing good empirical methodologies is difficult in part because agents can *overfit* test evaluations and thereby obtain misleadingly high scores. We argue that reinforcement learning is particularly vulnerable to *environment overfitting* and propose as a remedy *generalized methodologies*, in which evaluations are based on multiple environments sampled from a distribution. In addition, we consider how to summarize performance when scores from different environments may not have commensurate values. Finally, we present proof-of-concept results demonstrating how these methodologies can validate an intuitively useful range-adaptive tile coding method.**

I. Introduction

In machine learning, algorithms are typically evaluated subjectively, theoretically, and/or empirically. While subjective evaluations leverage intuition to identify novel methods, they may fail to validate counterintuitive approaches or expose fallacious assumptions. Theoretical results are more rigorous but cannot always be obtained and may not consistently predict performance in practice, e.g., when there are large constant factors or the requisite assumptions do not hold.

Empirical evaluations are thus an important complement to subjective and theoretical evaluations. They can illuminate salient underlying algorithmic properties and spur the development of increasingly practical algorithms [1], [2], [3]. For example, the UCI repository [4] has helped machine learning gain a critical role in real-world applications.

However, the usefulness of any empirical evaluation depends on the *empirical methodology* employed. If not carefully designed, the methodology may fail to guard against *evaluation overfitting*, in which the system under evaluation obtains misleadingly high scores by overspecializing to the evaluation setting. Many types of overfitting are possible, e.g., in *data overfitting*, a well-known concern in supervised learning, a learned function is overfit to a specific data set.

The primary aim of this paper is to devise empirical methodologies that guard against overfitting in on-line reinforcement learning.[1] We argue that, in this setting, data overfitting is less of a concern than *environment overfitting*, wherein the learning algorithm itself is overspecialized to the environment, i.e., Markov decision process (MDP), in which it is evaluated.

For example, consider one evaluation methodology in common use: measuring the average cumulative reward accrued by an algorithm on a set of independent learning trials on a single benchmark environment. Through various design choices, e.g., about the state representation, initial value function, learning rate, etc., the algorithm designer can construct an algorithm that excels at this evaluation but performs poorly in other environments. In the extreme case, a familiar benchmark such as Mountain Car [5] can be trivially overfit by an algorithm that uses the optimal policy throughout and thus outperforms any competitor that actually learns.

The unfairness of this comparison highlights the fundamental limitation of methodologies based on single environments: they do not control for the designer's prior knowledge about the environment. Such concerns have led some to question whether performance improvements on benchmark environments reflect real progress in machine learning or simply an increase in prior knowledge about those environments [6], [7], [8].

While environment overfitting can occur in any learning algorithm, it is particularly pernicious in reinforcement learning. Unlike supervised learning, in which evaluations are typically based on fixed data sets, reinforcement learning often relies on publicly available simulators for evaluation. Since designers can generate unlimited data for tuning, there may be little uncertainty about the environment left when the algorithm is tested and thus little protection against environment overfitting.

To address this problem, we advocate departing from methodologies based on single environments. While the use of multi-environment evaluation has been suggested before, it is not common practice in reinforcement learning. Furthermore, to the best of our knowledge, specific methodologies for multi-environment evaluation in reinforcement learning have not been formulated.

We propose a simple solution based on *generalized methodologies* in which evaluations are based on multiple environments sampled from a distribution. By creating uncertainty about the evaluation conditions, these methodologies place a consistent limit on the prior knowledge available to any designer. Furthermore, they make explicit what generality across environments is desired from agents, as the distribution

[1] While the methodologies we propose may also be useful in batch reinforcement learning or other related settings, we do not consider them here.

specifies both a target class and the likelihood of encountering environments in that class.

We propose two specific generalized methodologies, one suitable for comparing algorithms across time and another for one-shot settings such as competitions, as well as two ways of summarizing performance across environments. In addition, we illustrate the potential of generalized methodologies by comparing Sarsa(λ) with tile coding to a range-adaptive variant on several generalized environments. These proof-of-concept results demonstrate that generalized methodologies can empirically validate an intuitively useful form of adaptation that single-environment methodologies cannot.

II. Evaluation Overfitting

The purpose of an empirical evaluation in machine learning is to gather data that helps characterize a learning algorithm's behavior. In this paper, we model the evaluation process as an interaction between a *designer*, who designs an *agent*, and an *evaluator*, who evaluates the agent. We formulate the evaluation as a series of independent *trials* and the behavior of interest as a scalar performance metric or *score*. The purpose of the evaluation is thus to estimate some statistic, e.g., the expected value, of the score distribution based on the data gathered during the trials.

We assume a self-interested designer: one who seeks the agent that will obtain the best score, even if that agent will not generalize well. This does not imply that actual designers aim to misrepresent their algorithms in empirical results. On the contrary, bias is typically implicit and unintentional. However, by seeking methodologies that cannot be manipulated even by self-interested designers, we strive to free designers from having to worry about their own biases, because the goals of finding good agents and doing well on the evaluations are closely aligned.

In some cases, the score that the evaluation produces may be of inherent importance, such as when the evaluation setting exactly corresponds to a real-world problem. However, in most research settings, the experimental results do not have inherent value, e.g., Mountain Car policies and classifiers for UCI datasets have little practical use. Instead, such experiments are intended to represent, implicitly or explicitly, performance on some *target distribution* of conditions. For example, based on Sutton's well-known tile-coding paper [5], no reader would infer that tile coding is useful only for the particular environments to which he applied it or the particular random number seeds he used in his experiments. Instead, those experiments suggest tile coding is a useful approach in a variety of conditions.

We define *evaluation overfitting* as a general phenomenon in which an algorithm performs well in an empirical evaluation without performing well in the target distribution. The AI community has identified various problems that can be broadly classified as evaluation overfitting, including data overfitting, method overfitting [7], and reliance on fixed benchmarks [9] or unrealistic prior knowledge of utility functions [10].

A. Data Overfitting

Perhaps the most familiar form of evaluation overfitting is *data overfitting*, which occurs when the function produced by the agent, e.g., a classifier or policy, is so customized to a particular data set that it fails to generalize to independently sampled data from the same environment. In this case, the target distribution corresponds to the distribution over labeled examples that defines the environment, from which the data to which the function is customized is merely a sample.

Data overfitting is a familiar concern in supervised learning, in which evaluations are typically conducted on fixed data sets. In such a setting, separating the data into training and test sets, e.g., via cross-validation, is essential for ensuring that the evaluation scores are unbiased estimates of performance on the target distribution.

In contrast, data overfitting is typically not a concern in reinforcement learning. The primary reason is that reinforcement learning, in an on-line setting, is *interactive*, i.e., the actions chosen by the agent affect what data it sees. As a result, evaluations cannot be conducted on fixed data sets. Instead, evaluators typically have access to simulators with which they can generate unlimited data. Thus, data overfitting can be avoided simply by using fresh data for each evaluation, without the need for cross-validation or similar mechanisms.

B. Environment Overfitting

Environment overfitting occurs when the learning algorithm is so customized to the environment on which it is evaluated that it performs poorly on other environments for which it was intended. In environment overfitting, the learning algorithm itself, rather than the function it produces, is overfit. Furthermore, the target distribution is a distribution over environments, rather than a distribution over data in a single environment.

Whereas data overfitting is more problematic in supervised learning, environment overfitting is more problematic in reinforcement learning. In supervised learning, the limited availability of data restricts the designer's ability to find an algorithm that overfits the environment because he or she is uncertain from what distribution the data was drawn. Hence, to maximize the expected score on new data, the designer must submit an agent that performs well across the range of environments that could have generated the data. In contrast, in reinforcement learning, designers typically also have access to a simulator. Since they can generate unlimited data, they can try out unlimited state representations, initial value functions, learning rates, etc. during the design process. In doing so, they iteratively acquire more and more knowledge about the future evaluation conditions. As a result, during testing there may be little environment uncertainty and thus little protection against environment overfitting.

C. Fitting versus Overfitting

The notion of environment overfitting invites the question: how broad should the target distribution be? Theoretical work often aims to discover algorithms whose performance can

be guaranteed across a wide range of environments, such as the set of all discrete MDPs. However, many researchers are focused on more practical problems, such as how to learn effectively in specific settings. In such cases, customizing an algorithm to a particular setting at the expense of performance in other settings is often an effective algorithmic design strategy. In fact, when No Free Lunch theorems [11] apply, it may be the only option.

While it may seem that environment overfitting can actually be desirable because it allows more specialization, this is not the case. On the contrary, the target distribution is simply smaller in such cases. The confusion is eliminated if we distinguish between *fitting* and *overfitting*. The former means customizing an algorithm to the target distribution at the expense of environments outside that distribution. The latter means customizing an algorithm to the evaluation setting at the expense of the target distribution. Thus, fitting is always desirable and overfitting is always undesirable.

While the target distribution may be large or small, it typicaly contains more than one environment in a reinforcement-learning problem. Intuitively, reinforcement-learning methods use data to reduce their uncertainty about the agent's environment. This occurs explicitly in model-based methods and implicitly in model-free ones. If the target distribution contains only one environment, then such uncertainty does not exist, and planning methods e.g., dynamic programming [12], could be used instead.

Although a target distribution with only one environment may still involve other forms of uncertainty, e.g., via a stochastic transition function, such uncertainty does not necessitate learning because it is not reducible. For example, once the transition function is known, no amount of data can further help predict a state transition before it occurs.

III. Generalized Environments

In this section, we consider how to construct an empirical methodology that protects against environment overfitting in reinforcement learning. Clearly, the single-environment methodologies in common use are not ideal in this regard. If the target distribution contains multiple environments, a methodology that uses only one environment invites the designer to overfit it at the expense of performance on the target distribution's other environments.

However, this does not imply that single-environment methodologies are useless. On the contrary, even when the target distribution contains multiple environments, single-environment methodologies can still produce meaningful results if the designers make a good-faith effort not to overfit the environment. For example, though the optimal policy for Mountain Car is well known, researchers still regularly use that environment to publish results that show evidence of real learning. Such results are meaningful if designers ignore their prior knowledge about the environment when constructing agents.

Nonetheless, such solutions are not completely satisfying. Even well-intentioned researchers may implicitly incorporate prior knowledge about the environment into their design decisions. As a result, there is no way to be sure that performance improvements are due to the development of better learning algorithms rather than simply an increase in prior knowledge about the environment.

To avoid this problem and ensure protection against environment overfitting, we advocate departing from single-environment evaluations. In their place, we propose methodologies based on *generalized environments*. The main idea is simple: to evaluate agents on multiple environments drawn from the target distribution.

We define a generalized environment $\mathcal{G} = \langle \Theta, \mu \rangle$ as a distribution μ over a set of environments Θ. In a generalized methodology, the agent's score is summarized across experimental trials in different environments sampled from Θ according to μ. Each $\theta \in \Theta$ is an MDP.

For example, consider the Generalized Helicopter Hovering environment [13], one of several generalized environments used in the recent Reinforcement Learning Competitions [14]. The agent's goal is to hover a helicopter as close as possible to a fixed position. However, in each trial, the agent faces a different θ with an unknown wind velocity, forcing the agent to adapt on-line to maximize performance.

Because the agent does not know in advance what environment it will face in each trial, a generalized environment based on MDPs can be alternatively viewed as a *partially observable* MDP (POMDP) in which θ is a hidden state factor and $\mathcal{G}$ is the distribution over this factor in the initial state. This view corresponds to how the problem of learning in an MDP is often formulated in Bayesian reinforcement learning [15], [16].

However, regardless of whether we view the generalized environment as a POMDP, the critical point is that an empirical methodology that evaluates an agent's ability to learn in an MDP should model, not just a single MDP, but the agent's uncertainty about what MDP it faces. By doing so, generalized environments control for the uncertainty the agent faces at the start of each trial. This enables fair comparisons between agents because no amount of tuning by designers can eliminate the uncertainty represented by $\mathcal{G}$. The generalized environment also makes explicit what generality is desired, as the choice of $\mathcal{G}$ specifies both a target class and the relative importance of environments in that class.

A. Open Generalized Methodology

Figure 1 depicts one way to use a generalized environment for empirical evaluations, which we call an *open generalized* methodology because $\mathcal{G}$ is known to the designer. In the *tuning phase*, the designer samples environments freely from $\mathcal{G}$. For each sampled θ, the designer can generate unlimited data, i.e., by trying out various agents that select actions, receive rewards, and observe state transitions. In the *test phase*, the designer submits an agent for evaluation. The evaluator conducts a set of trials to evaluate the agent, selecting for each trial a new θ sampled independently from $\mathcal{G}$.

Since the experimental trials used for testing are interactive and generate data independent of the tuning phase, this

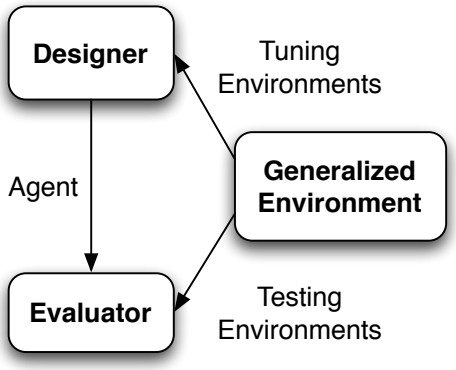

Fig. 1. Generalized evaluation methodology for reinforcement learning.

methodology protects against data overfitting, like conventional reinforcement-learning experiments. Additionally, because each θ is not fixed but sampled independently from $\mathcal{G}$, it also protects against environment overfitting. Even though unlimited tuning is permitted, the designer does not know in advance what environments the agent will face. Thus, overspecializing the agent to any particular θ will yield poor test performance. Of course, agents that excel within $\mathcal{G}$ may perform poorly on environments that are unlikely according to μ. However, since $\mathcal{G}$ is a designation of what generality is desirable, such an agent is fit, not overfit, to $\mathcal{G}$.

B. Secret Generalized Methodology

In the open generalized methodology, $\mathcal{G}$ creates uncertainty about the evaluation conditions. However, there is no uncertainty about $\mathcal{G}$ itself. Instead, $\mathcal{G}$ is fixed and public, enabling the designer to construct agents customized to $\mathcal{G}$. In some cases, when the robustness desired from the agent can be accurately formalized, such customization is appropriate, since $\mathcal{G}$ corresponds exactly to the target distribution. However, in many cases, the target distribution is not known exactly. For example, when training a helicopter to hover, only limited information about typical wind conditions may be available. Thus, $\mathcal{G}$ may only approximate the target distribution. In such cases, the open methodology allows designers to create agents that perform well in $\mathcal{G}$ but poorly in the true target distribution, a phenomenon we call *uncertainty overfitting*.

To protect against uncertainty overfitting, we propose a *secret generalized* methodology. Unlike the open variant, $\mathcal{G}$ is hidden from the designer, who receives access only to a fixed set of tuning environments. Agents are tested on a different sample of environments from $\mathcal{G}$, as in the open generalized methodology. A limited selection of tuning environments creates uncertainty about $\mathcal{G}$ and agents that are inappropriately specialized to the tuning environments will perform poorly when evaluated on new environments from $\mathcal{G}$. Consequently, the agent must not overfit the tuning environments in order to perform well across the hypothetical distribution of $\mathcal{G}$'s that could have generated the tuning environments.

In addition, the secret generalized methodology obviates the need to explicitly define Θ and μ. When the target distribution is not understood well enough to be formalized, generalized evaluations can still be conducted if sample environments can be generated for tuning and testing. This is analogous to common practice in supervised learning, in which labeled data sets are generated without making explicit the distribution over all labeled examples.

Furthermore, the secret generalized methodology retains good protection against data and environment overfitting, for the same reasons as the open methodology. However, unlike the open methodology, the secret generalized methodology requires secrecy, as $\mathcal{G}$ must remain hidden from the designer.

The need for secrecy has significant practical implications, as it means that the evaluation cannot be conducted by the designer. Instead, a neutral, trusted third party is required. While arranging such a setup is not trivial, it is feasible in one-shot settings, such as competitions, in which all agents are evaluated at approximately the same time. One person or group can oversee the evaluations and guard the secret information during the competition. However, using such a methodology to compare agents across time is probably infeasible, as the hidden information must stay hidden during the whole intervening period.

Therefore, the choice between the open and secret generalized methodologies involves a trade-off between protection against overfitting and ease of use. The secret version gives the best protection but involves constraints that in practice probably limit its applicability to one-shot settings. The open variation abandons protection against uncertainty overfitting but can be used to compare agents evaluated at different times and thus assess the progress of the field.

C. Meta-Generalized Methodology

In principle, we can avoid the trade-off between the open and secret methodologies. In fact, we can protect against uncertainty overfitting without requiring secrecy by redefining each element of Θ to be an entire generalized environment instead of an MDP. For convenience, we denote this *meta-generalized* environment as $\mathcal{H} = \langle \Gamma, \tau \rangle$, where τ is a distribution over a set of generalized environments Γ. Variations between generalized environments in Γ can capture the evaluator's own uncertainty about the true target distribution of environments.

In an open version of this *meta-generalized* methodology, the designer is allowed to sample an unlimited number of generalized environments from Γ for tuning. When evaluated, the agent's performance is aggregated across multiple *meta-trials*. At the beginning of each meta-trial i, a generalized environment $\mathcal{G}_i$ is sampled from Γ according to τ. $\mathcal{G}_i$ remains fixed throughout the meta-trial, during which the agent interacts with a series of MDPs sampled from $\mathcal{G}_i$.

Note that, though τ is a distribution over distributions, it cannot be 'flattened' into a single distribution because the MDPs within a meta-trial are not sampled independently. Instead, they are conditional on $\mathcal{G}_i$. Thus, within each meta-trial, the agent can learn, not only about each MDP it faces, but about $\mathcal{G}_i$. To perform well across meta-trials, an agent's

learning algorithm must be robust to different possible values of $\mathcal{G}$, though it can be fit to $\mathcal{H}$.

Just as a generalized methodology controls for uncertainty about the environment by defining a distribution over Θ, a meta-generalized methodology controls for uncertainty about the target distribution by defining a distribution over $\mathcal{G}$. While this approach may seem like overkill, it is no more elaborate than meta-learning models proposed for supervised learning [17], [18] and reinforcement learning [15]. However, in many empirical settings, the number of trials required could be prohibitive, in which case the secret generalized methodology may be a more practical approach to protecting against uncertainty overfitting.

IV. Generalized Performance Measures

When using generalized environments, an agent's scores must be summarized across trials conducted on different environments. A straightforward approach is to simply average the scores: estimating the expected score across $\mathcal{G}$. Abusing notation a bit, we can treat $\mathcal{G}$ as a random variable with sample space Θ and distribution μ, let S_x be a random variable for the score of some agent x, and then write the expected score as $\mathbb{E}[\mathbb{E}[S_x|\mathcal{G}]] = \mathbb{E}[S_x]$. However, averaging may not be a safe function for summarizing scores when the MDPs that generate those scores are on different scales. According to Roberts, a statement about a summary is *meaningful* only "if its truth value is unchanged whenever every scale is replaced by another acceptable scale" [19]. For example, consider two pairs of measurements on incommensurate scales, $\langle -32°\mathrm{C}, 130°\mathrm{F}\rangle$ and $\langle -10°\mathrm{C}, 100°\mathrm{F}\rangle$. The statement "the average of the first pair is larger than the second ($49 > 45$)" is true but not meaningful: if we convert the °F measurements to °C, the first average is smaller than the second ($4 \not> 7$).

Issues with scales can arise with generalized domains because the rewards (and therefore the scores) from different environments often do not have commensurate value due to differences in difficulty, length of trials, range of rewards, etc. In fact, the scale of rewards in an MDP is arbitrary in the sense that the partial ordering over policies is unchanged for all positive scalings of the reward function. In practice, the scale is often selected arbitrarily and does not directly correspond to a standard unit such as dollars, time, or utility. In Mountain Car, for example, the agent receives a reward of -1 per step but could just as easily receive -1000 or -0.001. Therefore, if agents are compared using average scores, their relative performance can be changed abruptly simply by scaling the rewards for some environments up or down. Related issues have been noted in supervised machine learning [20], [21] and may explain why average accuracy across multiple data sets is rarely used to summarize performance in that community [21].

However, other summarization functions are possible, each with their own advantages and limitations. For example, to better balance the impact of each environment and to obviate any need for commensurate scales between environments, the agents can be evaluated by the probability that one agent would outperform the other in a random trial on the same environment: $\mathbb{E}[\mathbb{P}(S_x > S_y|\mathcal{G})] = \mathbb{P}(S_x > S_y)$. Using the *sign test* [22], we can evaluate the hypothesis that $\mathbb{P}(S_x > S_y) > 0.5$ by counting how many times x outscores y in a series of matched trials. The sign test is useful when pairs of observations arise under similar conditions and different pairs arise under different conditions [23], a situation also identified for comparing classifiers across environments [21].

V. Results

To illustrate the potential of generalized methodologies, we present proof-of-concept results that demonstrate that a novel, range-adaptive tile-coding function approximator is *not inferior* to a conventional approach that requires prior knowledge about the range of state variable values. Our aim is to evaluate the empirical methodologies, not the learning methods themselves. Hence, we are not advocating the range-adaptive method but rather illustrating how generalized methodologies can validate an intuitively useful form of adaptation in a way that single-environment methodologies cannot.

One disadvantage of a simple tile-coding implementation is that it requires environment-specific prior knowledge of the range of possible states.[2] To address this, the range-adaptive tile coder dynamically spreads its fixed memory resources over the range of values observed so far. Each time the agent observes values outside of this range, the value function is *transplanted* to a new, larger range.

In particular, the transplant operation, shown in Algorithm 1, is called once for each tiling. It iterates over every tile, calculating the range of values that activate each tile under the old range and what tile will be activated under the new range. Then, the old feature weights are summed in the new tile positions and divided by the number of old tiles that were transplanted there. Given a state, the getTileForState operation determines which tile in the current tiling is activated by that state. The getCenterOfTile operation is one way to approximately invert this operation: given a tile, it returns one state, the center of that tile, which would activate it.

Algorithm 1 Transplant

```
for i := 0 ... numTiles do
    c := getCenterOfTile(i,oldInputRanges)
    k := getTileForState(c,newInputRanges)
    newWeights[k] := newWeights[k] + oldWeights[i]
    newWeightCounts[k] := newWeightCounts[k] + 1
end for
for i := 0 ... numTiles do
    newWeights[i] := newWeights[i]/newWeightCounts[i]
end for
```

By automatically finding an appropriate resolution for the function approximator, the range-adaptive approach has a clear potential performance advantage. However, demonstrating that

[2]Some variations use *infinite planes* with hashing and require information about tile width instead of range: see http://webdocs.cs.ualberta.ca/~sutton/tiles2.html.

advantage experimentally is not straightforward. In a single-environment methodology, the designer can eliminate uncertainty about the observation ranges through repeated experimentation and thus there is no need for range adaptability. On the contrary, the usefulness of range adaptability lies in its robustness across multiple environments with different ranges. Convincingly demonstrating this usefulness requires a methodology that ensures the designer cannot know the observation ranges a priori. In other words, it requires a methodology that uses more than one environment.

Therefore, we evaluate the range-adaptive tile coder by comparing it to a fixed-range alternative in a generalized environment. In particular, we use a generalized environment comprised of variations of three standard benchmark environments: Mountain Car, Acrobot, and Puddle World [5]. First we created three generalized environments: Θ_M based on Mountain Car, Θ_A based on Acrobot, and Θ_P based on Puddle World. Each particular environment in each Θ has three distinct and independent generalizations $\theta_i \in \Theta = \langle \mathcal{N}, \mathcal{T}, \mathcal{I} \rangle$. First, at each time step, the change in the state caused by the agent's action is modified by a random perturbation sampled uniformly from a range $[-\mathcal{N}, \mathcal{N}]$, with $\mathcal{N}$ itself selected uniformly from $[0, 0.5]$ for each environment. Second, the observation values are transformed by a randomly selected series of environment-specific transformation functions $\mathcal{T}$ that scale, translate, invert, and apply trigonometric nonlinearities. These transformations are reversible, preserving the Markov property. Finally, with equal probability, $\mathcal{I}$ specifies either fixed initial conditions or a randomized starting state for each episode. These transformations create ranges of observations that vary dramatically between environments.

Our experiments use three families of agents, which we call A, B, and C. The *adaptive* approach (A) receives no prior knowledge about the range of the state variables in each environment and uses Algorithm 1 to automatically distribute its memory resources over the range of observed values in each trial. The *baseline* approach (B), which is expected to perform poorly, uses fixed ranges: the smallest range for each variable that covers the values possible across all environments in the experiment. Finally, the *cheater* approach (C) uses a fixed-range tile coding but receives perfect environment-specific information about the range of the state variable values for the environment used in each trial.

We first conducted a set of three experiments, one for each of Θ_M, Θ_A, and Θ_P. All algorithms were debugged using the canonical, non-generalized versions of these environments. To create the *tuned* agents for the final evaluation, we ran a large number of candidate agent configurations for 1 trial each on a fixed *tuning set* of 25 environments sampled from the generalized environment. The candidate set of agents was comprised of all combinations of the following settings for each of the three agent families: $\alpha \in \{2.0, 1.0, .75, .5, .25, .125, 0.06125\}$, $\lambda \in \{.99, .95, .9, .75, .5, .25, .125, .06125, 0\}$, tilings $\in \{4, 8, 16, 32\}$, cells per dimension $\in \{4, 8, 16, 32\}$, and tile configurations (stripes over each variable, joint tiling over all variables, or all factorial variable combinations), totalling 3024 candidate agent configurations for each family. Each experimental trial lasted 70,000 time steps. The tuned agent for each family was the candidate agent with the highest average reward per completed episode across the tuning set of MDPs.

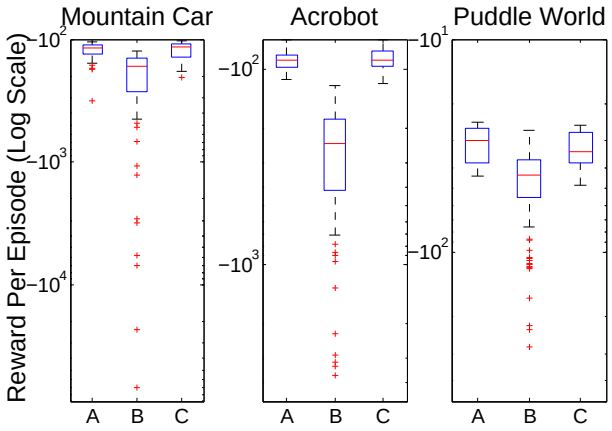

Fig. 2. Generalized methodology results for the adaptive (A), baseline (B), and cheater (C) approaches on three generalized environments.

For each agent family and generalized environment, the tuned agent was run for one trial in each of the 100 additional environments in the *test set*. Box plots of the results are shown in Figure 2.

Not surprisingly, B performs quite poorly. Because it does not know the correct range for each environment, it must use a range that is often much too large, sacrificing resolution in its function approximation. In contrast, A performs much better (Student's t-tests confirm that the performance difference between A and B is statistically significant for all three generalized environments at $p < 0.05$). In fact, by adapting online, A roughly matches the performance of C, even without oracle range information. Because the generalized environments create uncertainty about the correct range, range adaptability becomes advantageous, which no amount of overfitting by B can change. Hence, this methodology, unlike the single-environment alternative, makes it possible to empirically validate the intuitive value of this adaptability.

We also analyzed our experimental results on all three generalized environments with the probabilistic summarization described in Section IV. The probabilistic summarization is a pairwise measure, so we treated the tuning-selection problem as a voting problem, where each experimental trial was a voter and each agent was a candidate. We used Copeland's method [24] to select as our tuned agent the agent that performed best in all pairwise comparisons with other agents. In the Mountain Car and Acrobot environments, this approach selected the same agent as the average score approach. In Puddle World, a different agent was selected by the probabilistic approach, but experimentation showed that the differences between those agents were not significantly different.

We also ran the tuned agents selected by the probabilistic summarization on the test set of 100 environments. The results

are analogous to the expected score approach: $\mathbb{P}(A > B) > 0.5$ with $p < 0.05$ in all three environments, but a similar statement about A and C cannot be made with confidence. Thus, in these three generalized environments the score distributions are sufficiently similar that the expected score approach is not misleading. However, note that the score axis in the figure shows that the scores in each of the generalized environments are from different ranges.

Of course, this experimental setup is artificial because the distribution over environments does not reflect some real-world uncertainty, but was designed to yield a comparison favoring the adaptive approach. However, the point is that generalized methodologies make such a comparison possible. The distribution over environments indicates what type of task generality is desirable and thus enables us to formalize the problem that the adaptive approach solves. Using single-task methodologies, this and many other forms of adaptability will not show an empirical advantage, as simpler task-overfit methods will perform as well or better.

To explore this variation between score distributions, we also combined the three generalized environments to make new *union environments* $\Theta_U = \Theta_M \cup \Theta_A \cup \Theta_P$. In Θ_U, we compared A to C using the probabilistic summary. As before, we could not determine with confidence whether A and C were different. However, within the adaptive approach, the agent with the highest expected score *is* statistically significantly worse than the agent selected by Copeland's method when they are compared using the probabilistic criteria. Thus, the expected score can be a misleading summarization of performance across the union environment.

VI. Discussion

Overall, we believe that generalized methodologies are a promising way to improve empirical evaluations in reinforcement learning. They protect against environment overfitting and enable fairer comparisons between agents. By specifying a distribution over environments, they make explicit what environment generality is desired and thus incentivize the kinds of adaptability that can make reinforcement-learning algorithms more useful in practice.

What form of generalized methodology is most appropriate depends on the purpose of the empirical evaluation. In a competitive setting, the secret generalized methodology is appealing because it protects against both environment and uncertainty overfitting. However, when comparing agents across time, the open variant is more practical, as it obviates the need for secrecy.

What performance measure is most appropriate depends on the type of generalized environment, its scores, and the goals of the evaluator. Encouraging the design of all-purpose methods requires generalized environments like the union environment that contain qualitatively different environments, in which case probabilistic measures like the sign test are preferable. However, when the environments are similar and the distribution captures some real-world uncertainty, e.g., over wind conditions when flying helicopters, maximizing expected cumulative reward may better reflect the evaluator's goals.

VII. Related Work

Many researchers have argued for the importance of empirical evaluations in artificial intelligence [2], [3] and machine learning in particular [1]. In addition, competitions based on empirical evaluations have long been popular. For instance, the well-publicized Netflix competition used a single-environment methodology with holdout data, encouraging innovation by many teams over a multi-year period [25]. There have also been three recent RL Competitions [14], which used generalized benchmarks to evaluate participating agents.

However, several researchers have also raised concerns about overemphasizing empirical results. To our knowledge, Falkenauer [7] was the first to identify the problem of environment overfitting (which he called "method overfitting"). Similarly, Ponce *et al*. [8] point out that public data sets can become stale and Langford [9] enumerates many types of overfitting, some of which are special cases of environment overfitting. Others have pointed out statistical problems in typical evaluations [22] or bemoaned the emphasis they create on software engineering instead of research innovation [26]. Drummond and Japkowicz [6] liken statistical benchmarking to an addiction and argue it is time to "kick the habit."

Furthermore, some researchers argue that an overemphasis on generality has limited the empirical success of reinforcement-learning methods. For example, Lane and Smart [27] claim that "(PO)MDPs are so general as to be nearly useless in many cases of practical interest." Focusing on specialized methods is also consistent with recent findings in the study of learning in natural systems, which has led Gallistel to conclude that assuming there is a general purpose learning process in the brain "is equivalent to assuming that there is a general purpose sensory organ, which solves the problem of sensing" [28].

In addition to generalized environments, several other models of multi-environment learning have been proposed, including learning to learn [17], inductive bias learning [18], multi-task reinforcement learning [15], [29], [30], and transfer learning [31]. However, until now, multi-environment models have not been proposed as a means to combat environment overfitting. Nonetheless, many of the methods developed for such settings may, like Bayesian approaches that explicitly reason about environment distributions [15], [16], be particularly well suited to generalized environments.

Nouri *et al*. [32] propose an empirical methodology for evaluating performance in policy evaluation, a subproblem of reinforcement learning. By using fixed data sets, they avoid the complications of interactivity. The scarcity of data available to the designer in this approach provides some protection against environment overfitting without the need for multiple environments. However, this methodology cannot evaluate an agent's ability to explore efficiently, a critical component of the on-line reinforcement-learning problem.

Many benchmarking resources designed to facilitate experiments in reinforcement learning are currently available. For instance, the RL-Repository[3] and RL-Library[4] provide a number of RL environments and RL-Glue [33] provides a way of allowing agents to easily interact with different environments. While both help enable experiments on many environments, they do not help protect against environment or uncertainty overfitting.

VIII. Future Work

Although this paper focuses on reinforcement learning, generalized methodologies could also be used in other areas of machine learning. In supervised learning, the need for such methodologies may not be as acute, since the limited availability of data already provides some protection against environment overfitting. However, generalized methodologies could still be used to eliminate the need for secrecy or to add protection against uncertainty overfitting.

Generalized methodologies may prove most useful in other interactive learning settings such as *active learning* [34], in which the agent can select which training examples it sees. Since the requirements in terms of empirical evaluation are similar to those of reinforcement learning, the potential advantages of generalized methodologies are similar too.

In the future, we will investigate the application of generalized methodologies to such settings. Also, we hope to develop a formal framework for empirical methodologies and use it to prove properties of generalized methodologies. Finally, we intend to develop specific benchmark generalized methodologies with which the community can assess its progress.

We hope that one consequence of using such generalized benchmarks will be the development and refinement of more specialized reinforcement-learning methods. As Lane and Smart put it,"a profitable approach for the future is to cleave RL into a number of sub-disciplines, each studying important 'special cases'. By doing so, we will be able to take advantage of the properties of these cases in ways that our current (PO)MDP frameworks are unable to" [27].

References

[1] P. Langley, "Machine learning as an experimental science," *Machine Learning*, vol. 3, no. 1, pp. 5–8, 1988.

[2] P. Cohen and A. Howe, "How evaluation guides AI research," *AI Magazine*, vol. 9, no. 4, pp. 35–43, 1988.

[3] H. Simon, "Artificial intelligence: an empirical science," *Artificial Intelligence*, vol. 77, no. 1, pp. 95–127, 1995.

[4] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

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

[6] C. Drummond and N. Japkowicz, "Warning: Statistical benchmarking is addictive. Kicking the habit in machine learning," *Journal of Experimental and Theoretical Artificial Intelligence*, 2009.

[7] E. Falkenauer, "On method overfitting," *Journal of Heuristics*, vol. 4, pp. 281–287, 1998.

[8] J. Ponce, T. Berg, M. Everingham, D. Forsyth, M. Herbert, S. Lazebnik, M. Marszalek, C. Schmid, B. Russel, A. Torralba, C. Williams, J. Zhang, and A. Zisserman, *Dataset Issues in Object Recognition*. Springer, 2007.

[9] J. Langford, "Clever methods of overfitting," 2005. [Online]. Available: http://hunch.net/?p=22

[10] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, no. 3, pp. 203–231, 2001.

[11] D. H. Wolpert, "The lack of *a priori* distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[12] R. E. Bellman, "A Markov decision process," *Journal of Mathematical Mechanics*, vol. 6, pp. 679–684, 1957.

[13] R. Koppejan and S. Whiteson, "Neuroevolutionary reinforcement learning for generalized helicopter control," in *GECCO 2009: Proceedings of the Genetic and Evolutionary Computation Conference*, 2009, pp. 145–152.

[14] S. Whiteson, B. Tanner, and A. White, "The reinforcement learning competitions," *AI Magazine*, vol. 31, no. 2, pp. 81–94, 2010.

[15] A. Wilson, A. Fern, S. Ray, and P. Tadepalli, "Multi-task reinforcement learning: a hierarchical Bayesian approach," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 1015–1022.

[16] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

[17] S. Thrun and L. Pratt, Eds., *Learning to learn*. Norwell, MA, USA: Kluwer, 1998.

[18] J. Baxter, "A model of inductive bias learning," *J. of AI Research*, vol. 12, pp. 149–198, 2000.

[19] F. S. Roberts, "Limitations on conclusions using scales of measurement," in *Operations Research and The Public Sector*, 1994, vol. 6, ch. 18, pp. 621–671.

[20] G. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine learning*, vol. 40, no. 2, pp. 159–196, 2000.

[21] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, p. 30, 2006.

[22] S. L. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining and Knowledge Discovery*, vol. 1, pp. 317–327, 1997.

[23] W. J. Dixon and A. M. Mood, "The statistical sign test," *Journal of the American Statistical Association*, vol. 41, no. 236, pp. 557–566, 1946.

[24] D. Saari and V. Merlin, "The Copeland method," *Economic Theory*, vol. 8, no. 1, pp. 51–76, 1996.

[25] Y. Koren, "The BellKor solution to the Netflix grand prize," August 2009. [Online]. Available: http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

[26] J. N. Hooker, "Testing heuristics: We have it all wrong," *Journal of Heuristics*, vol. 1, pp. 33–42, 1995.

[27] T. Lane and W. Smart, "Why (PO)MDPs lose for spatial tasks and what to do about it," in *Proceedings of the ICML 2005 Workshop on Rich Representations for Reinforcement Learning*, 2005.

[28] C. R. Gallistel, "The replacement of general-purpose learning models with adaptively specialized learning modules," in *The Cognitive Neurosciences*. Cambridge, MA: MIT Press, 2000, pp. 1179–1191.

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

[30] M. Snel and S. Whiteson, "Multi-task evolutionary shaping without pre-specified representations," in *GECCO 2010: Proceedings of the Genetic and Evolutionary Computation Conference*, July 2010, pp. 1031–1038.

[31] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *JMLR*, vol. 10, no. 1, pp. 1633–1685, 2009.

[32] A. Nouri, M. L. Littman, L. Li, R. Parr, C. Painter-Wakefield, and G. Taylor, "A novel benchmark methodology and data repository for real-life reinforcement learning," 2009, poster at the Multidisciplinary Symposium on Reinforcement Learning.

[33] B. Tanner and A. White, "RL-Glue : Language-independent software for reinforcement-learning experiments," *Journal of Machine Learning Research*, vol. 10, pp. 2133–2136, September 2009.

[34] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[3] http://www-anw.cs.umass.edu/rlr

[4] http://library.rl-community.org