Journal of Universal Computer Science, vol. 13, no. 9 (2007), 1246-1269
submitted: 30/11/06, accepted: 27/9/07, appeared: 28/9/07 © *J.UCS*

Focus of Attention in Reinforcement Learning

Lihong Li
(Rutgers University, Piscataway, NJ, USA
lihong@cs.rutgers.edu)

Vadim Bulitko
(University of Alberta, Edmonton, AB, Canada
bulitko@ualberta.ca)

Russell Greiner
(University of Alberta, Edmonton, AB, Canada
greiner@cs.ualberta.ca)

Abstract: Classification-based reinforcement learning (RL) methods have recently been proposed as an alternative to the traditional value-function based methods. These methods use a classifier to represent a policy, where the input (features) to the classifier is the state and the output (class label) for that state is the desired action. The reinforcement-learning community knows that focusing on more important states can lead to improved performance. In this paper, we investigate the idea of *focused learning* in the context of classification-based RL. Specifically, we define a useful notation of state importance, which we use to prove rigorous bounds on policy loss. Furthermore, we show that a classification-based RL agent may behave arbitrarily poorly if it treats all states as equally important.

Key Words: reinforcement learning, function approximation, generalization, attention.
Category: I.2.6 [Artificial Intelligence]: Learning

1 Introduction

Many real-world tasks—such as flying a helicopter, playing a game, and managing inventory—cannot be accomplished in a single step but instead require an agent to perform a sequence of actions. Such problems are usually formulated as *sequential decision making* problems, and have become one of the key challenges in artificial intelligence. We can model these problems using the agent-environment interface (Figure 1), where the agent is a (software or hardware) implementation that has a certain predefined goal and can perceive the environment and take actions according to a *policy*. Roughly speaking, a policy answers the question: "Given what I have observed so far, what shall I do now?". Everything outside the agent is considered a part of the environment, which the agent may not be able to control, nor even necessarily observe. The agent maintains knowledge about the environment, and interacts with the environment as follows: on each cycle, the agent perceives the environment (either completely or partially), and takes an action that can change the environment; the environment evolves according to this action and at the same time provides the agent with some feedback measuring the

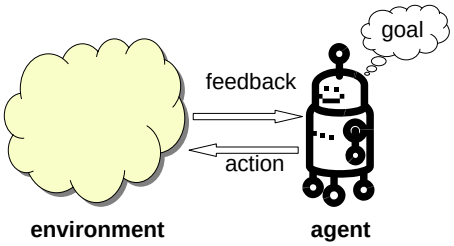

Figure 1: The agent-environment interface.

goodness of the action; then the next cycle begins. An environment is called *deterministic* if its evolution is deterministic; otherwise, it is called *stochastic*.[1]

Consider the task of balancing on a bicycle [Randløv and Alstrøm, 1998], for instance, where the agent (either a person or a computer program) tries to balance on the bicycle for as long as possible. She is able to perceive the *state* of the bicycle (e.g., the angle of the handlebar and the vertical angle of the bicycle, both of which are relevant to the balancing task), and is able to perform an action, which has two components—the torque applied to the handlebar and the displacement of the rider. When an action is applied, the bicycle's state changes according to a set of nonlinear equations. If the rider is balanced for one time step, she receives an immediate reward of $+1$; otherwise, the task terminates and no further rewards are received. We can model this task as a sequential decision-making problem, where the rider tries to find a policy that maps each bicycle state to an action, such that a rider who acts according to the policy will maximize the sum of immediate rewards she receives before the bicycle falls.

Reinforcement learning (RL) [Sutton and Barto, 1998], or *neuro-dynamic programming* [Bertsekas and Tsitsiklis, 1996], is a well-studied general framework for learning to act optimally in sequential decision making problems, through interaction with a possibly unknown and stochastic environment. Having succeeded in a number of challenging applications, such as the game of backgammon [Tesauro, 1992, Tesauro, 1995], job-shop scheduling [Zhang and Dietterich, 1995], elevator dispatching [Crites and Barto, 1996], dialogue policy learning [Singh et al., 2002], power control in wireless transmitters [Berenji and Vengerov, 2003], and helicopter control [Ng et al., 2004], reinforcement learning has attracted a great deal of research interest and became one of the central topics in machine learning and artificial intelligence.

There are two main classes of reinforcement-learning approaches. One approach is based on "value functions", which map each state to a numeric value such that an optimal policy involves simply applying the action that produces the optimal expected value over the resulting states. Policy-search methods, on the other hand, seek an ap-

[1] The next section provides a more formal description of these terms and concepts.

proximation to the optimal policy *directly* in the policy space.

Recent developments in the latter group of methods include applications of high performance classifiers. In this framework, the agent acquires a policy by learning a classifier that labels each state with the appropriate action. E.g., in the bicycle example, it might map the state $\langle \text{handlebar} = 0.5^o, \text{ bikeAngle} = -2.1^o \rangle$ to the action $\langle \text{torque} = 2.3, \text{ displacement} = 0.01 \rangle$. One of the challenges is to find an appropriate collection of "labelled databases"—here a set of $\langle$state, appropriate action$\rangle$ pairs. Recent implementations of this idea have demonstrated promising performance in several domains by learning high-quality policies through high-accuracy classifiers [Yoon et al., 2002, Lagoudakis and Parr, 2003, Fern et al., 2004]. This is different from traditional supervised learning, in that its goal of maximizing expect reward is subtly different from minimizing expected classification error. This is basically because here, some decisions are more important that others. In riding a bicycle, for example, making mistakes when the bicycle is well balanced can often be recovered from, but taking a wrong action when the bicycle is barely balanced can force the bicycle to enter into a "death spiral" from which crash is unavoidable. This effect can be amplified, as one decision has impacts on the agent's subsequent decisions [Scheffer et al., 1997].

Hence, a simple average over classification errors is not appropriate, as a system with higher classification accuracy may actually have *lower* policy quality [Li, 2004]. We demonstrate that classification-based reinforcement learning methods should instead *focus* the learning process on *more important* states. In particular, we will consider two classes of RL problems; for each of them, we define state importance based on quantities that are measurable by an agent, and then prove performance bounds for classification-based methods that take importance into account. In contrast, we also show examples where importance-insensitive agents can behave arbitrarily poorly.

The rest of the paper is organized as follows. Section 2 defines the notation and formulates the RL problem. Section 3 reviews the existing work on classification-based RL, and then motivates our work. Section 4 contains our main results: focused-learning methods for two classes of problems—batch and online RL with a generative model. Finally, Section 5 concludes the paper and discusses future directions.

As there are a number of terms used throughout this paper, we include two short glossaries: Table 1 summarizes terms from the reinforcement-learning literature, and Table 2 summarizes terms specific to our results.

2 Foundations

In reinforcement learning, sequential decision-making problems are usually modelled as finite Markov decision processes (MDPs) [Puterman, 1994]. An MDP is a six-tuple: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, D, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the finite state and action spaces, respectively. The agent starts in some initial state $s_0 \in \mathcal{S}$ generated from a start-state distribution D. At each time step t, it perceives the current state $s_t \in \mathcal{S}$, takes an action

Table 1: Glossary of standard reinforcement learning terms used in this paper

$\mathcal{A}$	action space
D	start state distribution
γ	discount factor in the range of $(0, 1)$
$\mathcal{L}(\pi)$	policy loss (5)
π	policy mapping $\mathcal{S}$ to $\mathcal{A}$
$\pi_Q(s)$	greedy policy w.r.t. Q (7)
$\pi_V(s)$	greedy policy w.r.t. V (6)
π^*	optimal policy
Π	policy space
$Q(s, a)$	action-value function estimate
$Q^\pi(s, a)$	action-value function (3)
$Q^*(s, a)$	optimal action-value function
r_t	reward received at time t
R_0	cumulative discounted return (1)
$\mathcal{S}$	state space
t	time step index
$V(s)$	state-value function estimate
$V^\pi(s)$	state-value function (2)
$V^*(s)$	optimal state-value function
$\mathcal{V}(\pi)$	policy value (4)

$a_t \in \mathcal{A}$, receives an immediate reward signal $r_{t+1} \in \mathbb{R}$, and reaches the next state $s_{t+1} \in \mathcal{S}$ according to the transition probability $\mathcal{P}^{a_t}_{s_t s_{t+1}} = \Pr(s_{t+1}|s_t, a_t)$; then it repeats the same process from state s_{t+1}. We assume the immediate rewards are bounded random variables, i.e., there exists $R_{\max} > 0$ such that $|r_t| \leq R_{\max}$ for all t, and $\mathbf{E}\{r_t|s_{t-1} = s, a_{t-1} = a\} = \mathcal{R}^a_s$. The goal of the agent is to select actions sequentially to maximize the discounted return:

$$R_0 = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^k r_{k+1} + \cdots \tag{1}$$

where the *discount factor* $\gamma \in (0, 1)$ has the effect of valuing future rewards less than the current reward. (This also ensures that R_0 is well-defined.)

A policy π is the set of state-action rules adopted by the agent to select actions at each state: $\pi : \mathcal{S} \to \mathcal{A}$. Given a policy π, we define $V^\pi(s)$, the state-value function, as the expected return received by following π from state s:

$$V^\pi(s) = \mathbf{E}_\pi \left\{ r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots | s_0 = s \right\}, \tag{2}$$

where the expectation is taken over all possible state-action sequences generated by following π. Similarly, the action-value function $Q^\pi(s, a)$ is defined as the expected

Table 2: Glossary of terms specific to our results

$d^{\tau,D}$	discounted cumulative visitation probability (13)
$G^{\pi}(s)$ and $G^{\pi}(s,\tau)$	importance in the online setting (29,30)
$G^{\nu}(s,\nu\rightarrow\tau)$	advantage of switching to τ from ν (14)
$G^{*}(s)$ and $G^{*}(s,\pi)$	importance in the batch setting (18,19)
$\mu_t^{\tau,D}$	state-visitation probability (12)
π^{*}_{CI}	classification-based policy, based on T_{CI} (10)
π^{*}_{CS}	classification-based policy, based on T_{CS} (23)
$\pi'_{\mathsf{CI\text{-}cRL}}$	cost-*insensitive* greedy policy (28)
$\pi'_{\mathsf{CS\text{-}cRL}}$	cost-*sensitive* greedy policy (33)
T_{CI}	cost-*insensitive* training data (9)
$T_{\mathsf{CI\text{-}cRL}}$	cost-*insensitive* training data (*c.f.,* Algorithm 1)
T_{CS}	cost-*sensitive* training data (22)
$T_{\mathsf{CS\text{-}cRL}}$	cost-*sensitive* training data (*c.f.,* Algorithm 4)
T_{Q^*}	training data (8)
$\mathcal{T}$	sampled states, subset of $\mathcal{S}$
ω	distribution over state space

return received by taking action a in state s and following π thereafter:

$$Q^{\pi}(s,a)=\mathbf{E}_{\pi}\left\{r_1+\gamma r_2+\gamma^2 r_3+\cdots|s_0=s,a_0=a\right\}. \tag{3}$$

We use *policy value*

$$\mathcal{V}(\pi)=\mathbf{E}_{s_0\sim D}\left\{V^{\pi}(s_0)\right\} \tag{4}$$

as the measure of a policy's performance, which is defined as the expected return obtained by following π from the start state s_0 drawn according to distribution D [Ng and Jordan, 2000]. A reinforcement-learning agent tries to *learn* the optimal policy, i.e., the one with the maximum value: $\pi^{*}=\arg\max_{\pi}\mathcal{V}(\pi)$. The corresponding optimal state- and action-value functions are denoted by $V^{*}(s)$ and $Q^{*}(s,a)$, respectively. Another equivalent way to measure the quality of a policy is through *policy loss*:

$$\mathcal{L}(\pi)=\mathcal{V}(\pi^{*})-\mathcal{V}(\pi). \tag{5}$$

Since $\mathcal{V}(\pi^{*})$ is a constant for a given problem, maximizing $\mathcal{V}(\pi)$ is equivalent to minimizing $\mathcal{L}(\pi)$.

After a value function (V or Q) is obtained, the associated deterministic, *greedy* policy is given by:

$$\pi_V(s_t)=\arg\max_{a}\mathbf{E}\left\{r_{t+1}+\gamma V(s_{t+1})|a_t=a\right\} \tag{6}$$

or

$$\pi_Q(s_t) = \arg\max_a Q(s_t, a). \tag{7}$$

When $V(\cdot)$ or $Q(\cdot,\cdot)$ is close to the optimal value functions $V^*(\cdot)$ or $Q^*(\cdot,\cdot)$, we anticipate the value of the greedy policy defined above will be near optimal [Singh and Yee, 1994]. Clearly, using action-value functions $Q(\cdot,\cdot)$ allows the agent to make greedy decisions without knowledge about the rewards and state transitions of the environment. For this reason, action-value functions are more convenient to work with than state-value functions.

This work explores the quality of the policies produced by reinforcement learners that differ in how they collect and weigh their training data. These learners will typically have to solve some optimization task (e.g., (23)); here we assume that they can find the exact solution.

3 Classification-based Reinforcement Learning

As noted earlier, there are two primary approaches to obtaining near-optimal policies. Value-function based methods approximate the optimal value function under the assumption that the greedy policy π_V with respect to the approximation V will be close to π^* when V is close to V^* [Singh and Yee, 1994]. In particular, temporal difference learning [Sutton, 1988, Watkins, 1989] implements this approach by updating V iteratively online. On the other hand, policy-search methods work directly in a policy space Π and build an approximation $\hat{\pi}^* \in \Pi$ to the optimal policy π^*. Examples include [Williams, 1992, Baxter and Bartlett, 1999, Baxter et al., 1999, Kearns et al., 2000, Ng and Jordan, 2000, Sutton et al., 2000, Fern et al., 2004]. In this paper we will focus on *classification-based* policy-search RL methods where the policy is represented as a classifier mapping states to actions.

3.1 Classification-based Batch Reinforcement Learning

So far we have introduced the *online* version of reinforcement learning, wherein the agent learns and acts at the same time. In this section, we will briefly discuss a simpler formulation called *batch* reinforcement learning, where learning of the policy or value function occurs *offline*. Such a batch/offline reinforcement learning framework is important wherever online learning is not feasible (e.g., when the reward data do not come in continuously, or when acting online may be too expensive as in robotics), and therefore a fixed set of experiences has to be acquired and used for learning policies offline [Draper et al., 2000, Dietterich and Wang, 2002, Levner and Bulitko, 2004].

A related technique called *experience replay* has been employed in robotics and was shown to speed up reinforcement learning and reduce possible damage to the

learning robot [Lin, 1992]. An experience-replay RL agent remembers its past online experiences and then repeatedly updates its value function or policy using these experiences offline. Another similar idea has been adopted in the Dyna-Q architecture [Sutton, 1990], in which the agent improves its policy or value function from both the "real" online experiences and the "imaginary" experience generated by the agent's model about the environment. Thus, both experience replay and Dyna-Q can be viewed as combinations of online and offline/batch reinforcement learning. Another advantage of batch reinforcement learning is that sometimes it facilitates theoretical analysis such as the sample complexity analysis [Kearns et al., 2000], as well as applications of advanced supervised learning algorithms [Dietterich and Wang, 2002].

In one specific setting that can be reduced to a supervised learning problem, we assume that the state space is *sampled* (independently) according to some distribution $\omega(\cdot)$, and the optimal action values for these sampled states are computed or estimated. The sampled states together with their optimal action values form the training set for supervised learning. For this reason, these sampled states are called *training states*. Formally, the training data are provided in the form of

$$T_{Q^*} = \{\langle s, a, Q^*(s,a)\rangle \mid s \in \mathcal{T}, a \in \mathcal{A}\}, \tag{8}$$

where $\mathcal{T} \subset \mathcal{S}$ is a sampled state space with s drawn randomly from ω.

The assumption of knowing the optimal action values may at first seem unrealistic. In practice, however, a technique called *sparse sampling* [Kearns et al., 2002, Kocsis and Szepesvári, 2006] may be used to compute or estimate such values. In it, the optimal action values are computed by applying all possible action sequences to each training state. Note that in the infinite-horizon case where the action sequences can be infinitely long, the discount factor γ has to be strictly less than one, which implies that the optimal action values can be estimated to any desired precision by considering action sequences up to a limited depth [Kearns et al., 2000].

Sparse sampling is especially useful for domains where good control policies generalize well across problems of different sizes. The agent can start with problems of tractable state spaces and apply the expansion efficiently to obtain the information needed for batch reinforcement learning. Once a good policy is computed, it may generalize to problems with larger state spaces. There have been several successful applications of this technique [Draper et al., 2000, Dietterich and Wang, 2002, Levner and Bulitko, 2004, Wang and Dietterich, 1999].

When the optimal action values are acquired for the sampled space $\mathcal{T}$, optimal actions $a^*(s) = \arg\max_a Q^*(s,a)$ can be computed, and the training data for learning a classifier-based policy can be formed:

$$T_{\mathsf{CI}} = \{\langle s, a^*(s)\rangle \mid s \in \mathcal{T}\}, \tag{9}$$

where the subscript CI (*cost-insensitive*) is in contrast to CS (*cost-sensitive*) that will be introduced later in the paper. The RL agent can then induce a classifier π^*_{CI} from the

training set T_{CI} by minimizing the classification error[2]:

$$\pi^*_{\mathsf{CI}} = \arg\min_{\pi \in \Pi} \Pr_{s \sim \omega}(\pi(s) \neq a^*(s)). \tag{10}$$

3.2 Classification-based Approximate Policy Iteration

When the state space is small, a policy can be represented by a lookup table with one entry per state. In such a case, *policy iteration* [Howard, 1960] has been found efficient in solving MDPs. Two steps are involved in each policy iteration. In the *policy-evaluation* step, the agent computes the value function of its current policy π, such as $Q^\pi(s,a)$; then in the *policy-improvement* step, it computes the *greedy policy* π' from π:

$$\pi'(s) = \arg\max_a Q^\pi(s,a), \quad \forall s \in \mathcal{S}. \tag{11}$$

It can be shown that $\mathcal{V}(\pi') > \mathcal{V}(\pi)$, unless π is optimal [Puterman, 1994].

If the state space is prohibitively large, however, function approximation has to be used to represent the policy or the value function. A general framework of approximation within policy iteration is known as *approximate policy iteration* (API) [Bertsekas and Tsitsiklis, 1996], which is similar to policy iteration, except (i) the policy and value functions are represented by function approximators such as neural networks (in contrast to lookup tables in policy iteration), and/or (ii) the value function is not computed exactly but is estimated. Consequently, the greedy policy $\hat{\pi}'$, computed in the inner loop of API, is an approximation to π'. Unlike π', $\hat{\pi}'$ may be worse than the original policy π, due to approximation errors.

With policies represented by classifiers, API has been successfully applied to several domains [Lagoudakis and Parr, 2003]. We refer to these methods as "cost-insensitive" as *equal* importance is assigned to all training states when learning a classifier. Algorithm 1 outlines the algorithm CI-cRL (Cost-Insensitive classification based Reinforcement Learning), which is due to Lagoudakis and Parr [Lagoudakis and Parr, 2003]. The algorithm makes calls to a subroutine Rollout (Algorithm 2) for estimating action values[Tesauro and Galperin, 1997]. Note that with the condition in line 10 of CI-cRL, only actions that are strictly better than others are included in the training set; otherwise, there is a tie in Q-values and thus no reason to prefer one action over another.

3.3 Classifier Accuracy versus Policy Performance

Before going into technical details, we establish some intuition by looking at a simple car-shopping problem. Figure 2 models the problem as a three-step (finite-horizon), non-discounted MDP. Non-terminal states are labeled with the decisions the user is

[2] In here and the rest of the paper, we always define the objective function (such as (10)) with states s drawn from the same distribution as the sampling distribution ω when constructing $\mathcal{T}$. For notational simplicity, we will assume this implicitly without explicitly including $s \sim \omega$ in the objective functions.

Algorithm 1 CI-cRL: Cost-insensitive classification-based RL. Learn is a sub-routine that induces a classifier from the input training data.

Input:
- M: generative model
- $\mathcal{T}$: set of training states
- γ: discount factor
- K: number of trajectories used to estimate the Q-function
- H: maximum length of trajectories

$\hat{\pi}' \leftarrow$ random policy
repeat
  $\pi \leftarrow \hat{\pi}'$
  $T_{\text{CI-cRL}} \leftarrow \emptyset$
  for all $s \in \mathcal{T}$ **do**
    for all $a \in A$ **do**
      $\hat{Q}^{\pi}(s,a) \leftarrow \text{Rollout}(M, s, a, \gamma, \pi, K, H)$
    end for
    $\hat{a}^*_{\pi} \leftarrow \arg\max_{a \in A} \hat{Q}^{\pi}(s,a)$
    if $\hat{Q}^{\pi}(s, \hat{a}^*_{\pi}) > \hat{Q}^{\pi}(s,a)$ for all $a \neq \hat{a}^*_{\pi}$ **then**
      $T_{\text{CI-cRL}} \leftarrow T_{\text{CI-cRL}} \cup \{\langle s, \hat{a}^*_{\pi} \rangle\}$
    end if
  end for
  $\hat{\pi}' \leftarrow \text{Learn}(T_{\text{CI-cRL}})$
until $\pi \approx \hat{\pi}'$
return π

to make. The edges are labeled with the possible actions and the associated immediate rewards. The agent starts by choosing the engine condition and finishes with the action of re-sell the car. Starting with the state engine condition, she has two choices: good and poor. Then she decides on the size of the car (small/large), and finally on its color (black/white). After all three choices are made, the agent buys the car and collects the final rewards by reselling the car shortly thereafter.

The optimal policy π^* is shown in Table 3. Suppose the agent is using batch reinforcement learning, and is considering two policies π_1 and π_2 (shown in the table). If the seven possible states are selected uniformly, policy π_1 has the classification accuracy of 86% and the policy value $\mathcal{V}(\pi_1)$ of 30, while policy π_2 is considerably less accurate (14%) but has a much higher policy value of 1000.

This phenomenon takes place because the correlation between the policy value and the classification accuracy is not monotonic. This is because in sequential decision-making the classification accuracy (i.e., the probability that a policy π outputs the optimal action) is *not* the target performance measure of a reward-maximizing agent. It

Algorithm 2 An implementation of the rollout technique. Simulate is a sub-routine that generates the next state and immediate reward using a generative model.

Input:
- M: generative model
- (s_0, a_0): the state-action pair to evaluate
- π: the policy to follow
- γ: discount factor
- K: number of trajectories
- H: maximum length of trajectories

for $k = 1, 2, \cdots, K$ **do**
  $(s, a) \leftarrow (s_0, a_0)$
  $(s', r) \leftarrow \mathsf{Simulate}(M, s, a)$
  $\hat{Q}_k^\pi \leftarrow r$
  $s \leftarrow s'$
  for $h = 1, 2, \cdots, H-1$ **do**
    $a \leftarrow \pi(s)$
    $(s', r) \leftarrow \mathsf{Simulate}(M, s, a)$
    $\hat{Q}_k^\pi \leftarrow \hat{Q}_k^\pi + \gamma^h r$
    $s \leftarrow s'$
  end for
end for
$\hat{Q}^\pi \leftarrow \frac{1}{K} \sum_{k=1}^{K} \hat{Q}_k^\pi$
return $\hat{Q}^\pi$

Table 3: Optimal and two approximate policies for the car-shopping problem.

	engine condition	size	color	accuracy	policy value
optimal policy π^*	good	large	white	100%	1030
policy π_1	poor	large	white	86%	30
policy π_2	good	small	black	14%	1000

is tempting for the agent to increase the classification accuracy and agree with the optimal policy in more states. However, it can be more crucial to agree with the optimal policy in states that are more *important* in affecting the policy value [Li, 2004, Li et al., 2004a, Li et al., 2004b]. Therefore, it would be helpful for the agent to *focus* the learning process on more critical states. This important observation motivates our work in the subsequent sections.

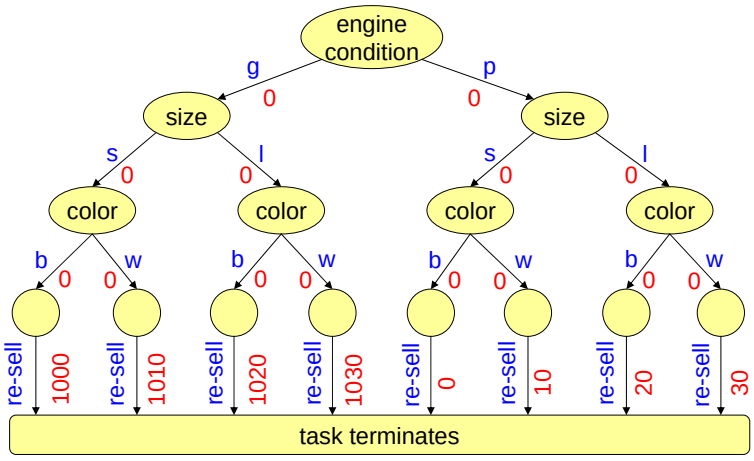

Figure 2: The car-shopping problem. Non-terminal states are labeled with the factors on which the user is to make decisions. The edges are labeled with the actions (g for good, p for poor, s for small, l for large, b for black, and w for white) and the immediate rewards. The user starts by choosing the engine condition and finishes with the need to re-sell the car.

3.4 Related Work

Some prior RL works have also explored the idea of focusing on important states. For example, prioritized sweeping [Moore and Atkeson, 1993] attempts to update only the more important states, based on the magnitudes of their *Bellman errors*. Real-time dynamic programming (RTDP) [Barto et al., 1995] considers a state to be more relevant if it is visited more often. Later in the paper, we will derive policy loss bounds based on state visitation probabilities and quantities known as *advantage* [Baird, 1993]. As state visitation probabilities are often unknown and not easy to estimate, we proposed approximate solutions that only depends on advantages. In this sense, our notion of state importance is orthogonal to RTDP's. It is of interest to combine the two notions of state importance in future research.

People in machine-learning reduction also emphasizes the use of cost in classification-based RL (*e.g.*, [Kim et al., 2003, Langford and Zadrony, 2003, Langford and Zadrony, 2005]). Those systems, however, consider a different RL setting, where an episode terminates after T steps and $\gamma = 1$, where the optimal policy is usually *nonstationary*—it consists of T policies that specify the rule of acting for horizons $1, 2, \cdots, T$. Consequently, they make use of T individual classifiers to represent this T-step nonstationary policy. In contrast, we consider infinite-horizon problems where only a *single* policy/classifier is involved. This makes the algorithm and analysis

trickier because there are interventions between decisions made in different time steps: that is, the single classifier being used has to work "consistently" over time. This is why we adopt the approximate policy iteration framework in the online RL setting (Section 4.3), while in the T-step RL setting, the T policies are usually learned *backwards* one by one, with the policy at horizon t depending only on the policy at horizon $t+1$. This paper thus extends previous results in T-step problems to a different setting, which requires a different analysis.

4 Focusing Attention in Reinforcement Learning

In this section, we formalize the idea of focused learning by introducing the concept of *state importance*. The policy-switching theorem will be presented first, followed by an investigation of the batch and online RL problems. The soundness of our approach is guaranteed by several theorems, presented and proven below. More details are found in a longer thesis [Li, 2004].

4.1 The Policy-Switching Theorem

Theorem 1 below, which establishes how the policy value is changed when a policy is switched to another, forms the basis of our later developments.

Let ν and τ be two arbitrary policies for a discounted, infinite-horizon MDP. The state-visitation distribution, $\mu_t^{\tau,D}(s)$, is the probability that state s is visited at time t by following policy τ with start states drawn according to D, that is,

$$\mu_t^{\tau,D}(s) = \Pr\{s_t = s|\tau, D\}. \tag{12}$$

Define

$$d^{\tau,D}(s) = \sum_{t=0}^{\infty} \gamma^t \cdot \mu_t^{\tau,D}(s) \tag{13}$$

as the "discounted cumulative visitation probability", and

$$G^{\nu}(s, \nu \to \tau) = Q^{\nu}(s, \tau(s)) - Q^{\nu}(s, \nu(s)) \tag{14}$$

as the advantage of action values between the "original" action, $\nu(s)$, and the "new" action, $\tau(s)$.

Theorem 1. *Given the definitions above,*

$$\mathcal{V}(\tau) - \mathcal{V}(\nu) = \sum_{s \in \mathcal{S}} \left(G^{\nu}(s, \nu \to \tau) \cdot d^{\tau,D}(s) \right). \tag{15}$$

Proof. Suppose the agent follows the policy τ from any start state $s_0 \sim D$, producing the trajectory $s_0, a_0, r_1, s_1, a_1, \cdots, s_t, a_t, r_{t+1}, s_{t+1}, \cdots$. Then

$$\begin{aligned} V^\nu(s_0) = Q^\nu(s_0, \nu(s_0)) \;&=\; Q^\nu(s_0, \tau(s_0)) - G^\nu(s_0, \nu \to \tau) \\ &= \mathop{\mathbf{E}}_{\tau}\{r_1 + \gamma V^\nu(s_1) - G^\nu(s_0, \nu \to \tau)\} \end{aligned} \tag{16}$$

Note that equation (16) is recurrent. In a similar fashion we derive:

$$\begin{aligned} V^\nu(s_0) &= \mathop{\mathbf{E}}_{\tau}\{r_1 + \gamma V^\nu(s_1) - G^\nu(s_0, \nu \to \tau)\} \\ &= \mathop{\mathbf{E}}_{\tau}\left\{r_1 + \gamma r_2 + \gamma^2 V^\nu(s_2) - G^\nu(s_0, \nu \to \tau) - \gamma G^\nu(s_1, \nu \to \tau)\right\} \\ &= \mathop{\mathbf{E}}_{\tau}\{r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\nu(s_3) - G^\nu(s_0, \nu \to \tau) - \gamma G^\nu(s_1, \nu \to \tau) \\ &\qquad -\gamma^2 G^\nu(s_2, \nu \to \tau)\} \\ &= \cdots \\ &= \mathop{\mathbf{E}}_{\tau}\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\} - \mathop{\mathbf{E}}_{\tau}\left\{\sum_{t=0}^{\infty} \gamma^t G^\nu(s_t, \nu \to \tau)\right\} \\ &= V^\tau(s_0) - \mathop{\mathbf{E}}_{\tau}\left\{\sum_{t=0}^{\infty} \gamma^t G^\nu(s_t, \nu \to \tau)\right\}. \end{aligned}$$

Therefore,

$$V^\tau(s_0) - V^\nu(s_0) = \mathop{\mathbf{E}}_{\tau}\left\{\sum_{t=0}^{\infty} \gamma^t G^\nu(s_t, \nu \to \tau)\right\}. \tag{17}$$

Taking the expectation of (17) over all start states $s_0 \sim D$, we obtain

$$\begin{aligned} \mathcal{V}(\tau) - \mathcal{V}(\nu) &= \mathop{\mathbf{E}}_{s_0 \sim D}\{V^\tau(s_0) - V^\nu(s_0)\} \\ &= \mathop{\mathbf{E}}_{s_0 \sim D}\left\{\mathop{\mathbf{E}}_{\tau}\left\{\sum_{t=0}^{\infty} \gamma^t G^\nu(s_t, \nu \to \tau)\right\}\right\} \\ &= \sum_{t=0}^{\infty} \gamma^t \mathop{\mathbf{E}}_{s_0 \sim D, \tau}\{G^\nu(s_t, \nu \to \tau)\} \\ &= \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \Big(G^\nu(s, \nu \to \tau) \cdot \mu_t^{\tau, D}(s)\Big) \\ &= \sum_{t=0}^{\infty} \sum_{s \in \mathcal{S}} \Big(G^\nu(s, \nu \to \tau) \cdot \gamma^t \, \mu_t^{\tau, D}(s)\Big) \\ &= \sum_{s \in \mathcal{S}} \left(G^\nu(s, \nu \to \tau) \cdot \sum_{t=0}^{\infty} \gamma^t \, \mu_t^{\tau, D}(s)\right) \\ &= \sum_{s \in \mathcal{S}} \big(G^\nu(s, \nu \to \tau) \cdot d^{\tau, D}(s)\big). \end{aligned}$$

□

A concept similar to $G^{\nu}(s, \nu \to \tau)$, defined above, was introduced in [Baird, 1993] and called *advantage*. Our Theorem 1 parallels a result by Kakade and Langford [Kakade and Langford, 2002].

4.2 Focused Batch Reinforcement Learning

Recall from Section 3.1 that, in batch RL, a set of training data T_{Q^*} is provided in the form (8) and the learning task is to approximate the optimal policy π^*.

We now introduce an appropriate measure of state importance for this task. Intuitively, a state s is important from a decision-theoretic perspective if making a wrong decision in s can have significant repercussions. Formally, it is defined as the difference in the optimal values of $a^*(s)$ and the other action $\bar{a}(s)$:[3]

$$G^*(s) = Q^*(s, a^*(s)) - Q^*(s, \bar{a}(s)). \tag{18}$$

Similarly, $G^*(s, \pi)$ is defined as:

$$\begin{aligned} G^*(s, \pi) &= Q^*(s, a^*(s)) - Q^*(s, \pi(s)) \\ &= G^*(s) \cdot \mathcal{I}(\pi(s) \neq \pi^*(s)), \end{aligned} \tag{19}$$

where $\mathcal{I}(e)$ is the indicator function that equals 1 when e is true and 0 otherwise. The next corollary follows from Theorem 1.

Corollary 2. *For any policy π,*

$$\mathcal{L}(\pi) = \sum_{s \in \mathcal{S}} G^*(s, \pi) \cdot d^{\pi, D}(s) \; . \tag{20}$$

Proof. In Theorem 1, let $\tau = \pi$ and $\nu = \pi^*$, then

$$G^{\nu}(s, \nu \to \tau) = Q^*(s, \pi(s)) - Q^*(s, \pi^*(s)) = -G^*(s, \pi).$$

Then the policy loss can be computed by Theorem 1,

$$\begin{aligned} \mathcal{L}(\pi) &= -\big(\mathcal{V}(\pi) - \mathcal{V}(\pi^*)\big) \\ &= -\sum_{s \in \mathcal{S}} \left(-G^*(s, \pi) \cdot \sum_{t=0}^{\infty} \gamma^t \mu_t^{\tau, D}(s) \right) \\ &= \sum_{s \in \mathcal{S}} \big(G^*(s, \pi) \cdot d^{\tau, D}(s) \big) \; . \end{aligned}$$

□

[3] For simplicity, we focus on the binary-action case in this paper. To extend to multiple-action cases, we could simply define the state importance as the difference between the *best* and the *worst* state values, but then the bounds may be too loose to be useful. A future study will involve a cost matrix rather than a real-valued cost, which makes the problem significantly more challenging.

This result is still, however, of a limited practical use since the quantity $d^{\pi,D}$, which is related to the state-visitation distribution $\mu_t^{\pi,D}$, is usually unavailable to the agent. Here, we instead minimize the following upper bound of $\mathcal{L}(\pi)$:[4]

$$\mathcal{L}(\pi) \;\leq\; \sum_{s\in\mathcal{S}} \left(G^*(s,\pi) \cdot \sum_{t=0}^{\infty} \gamma^t \right) \;=\; \frac{1}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s,\pi).$$

This suggests the following practical approximation approach to policy loss:

$$\arg\min_{\pi\in\Pi} \mathcal{L}(\pi) \;\approx\; \arg\min_{\pi\in\Pi} \frac{1}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s,\pi) \;=\; \arg\min_{\pi\in\Pi} \sum_{s\in\mathcal{S}} G^*(s,\pi). \quad (21)$$

Thus, given a set of training data T_{Q^*} described in (8), the agent can first compute $G^*(s)$ for all training states $s \in \mathcal{T}$ by (18), build a training set with state importance:

$$T_{\mathsf{CS}} = \{\langle s, a^*(s), G^*(s)\rangle \mid s \in \mathcal{T}\} \quad (22)$$

and then solve the optimization problem:[5]

$$\pi^*_{\mathsf{CS}} = \arg\min_{\pi\in\Pi} \sum_{s\in\mathcal{S}} G^*(s,\pi). \quad (23)$$

By replacing $G^*(s,\pi)$ in (23) with (19), solving for π^*_{CS} is turned precisely into the cost-sensitive classification problem with the *misclassification costs* conditioned on individual cases [Turney, 2000]. Indeed, in this setting, s is the attribute, $a^*(s)$ is the desired class label, and $G^*(s)$ is the misclassification cost.

This importance-sensitive (or cost-sensitive) algorithm is called CS (as opposed to CI in Section 3.1) and is summarized in Algorithm 3. It calls a subroutine, CS-Learn, which is a cost-sensitive classification algorithm.

A problem of both theoretical and practical interest is whether it is preferable to find π^*_{CS} as opposed to π^*_{CI}. We see below that the answer is Yes. Theorem 3 below provides an upper bound of the policy loss of π^*_{CS}. In contrast, Theorem 4 establishes that π^*_{CI} can be arbitrarily poor in the sense that the policy loss can be arbitrarily close to its upper bound, $\sum_{s\in\mathcal{S}} G^*(s)/(1-\gamma)$, even if the policy it produces agrees with the optimal policy in almost all states.

Theorem 3. *If π^*_{CS} has a sufficiently high quality, that is, there exists $\epsilon > 0$ such that*

$$\frac{\sum_{s\in\mathcal{S}} G^*(s,\pi^*_{\mathsf{CS}})}{\sum_{s\in\mathcal{S}} G^*(s)} < \epsilon, \quad (24)$$

then

$$\mathcal{L}(\pi^*_{\mathsf{CS}}) \leq \frac{\epsilon}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s).$$

[4] In practice, we may use density estimation techniques (e.g., [Duda et al., 2000]) to estimate $\mu^{\pi,D}$. This can lead to tighter bounds than the general results we provide in this paper.

[5] Note the summation is over the whole state space rather than the set of training states.

Algorithm 3 CS: Cost-Sensitive batch RL based on classification. CS-Learn is a subroutine that induces a cost-sensitive classifier from the input training data.

0: **Input:**
  – T_{Q^*}: set of training data
1: $T_{\mathsf{CS}} \leftarrow \emptyset$
2: **for all** $s \in \mathcal{T}$ **do**
3: $a^* \leftarrow \arg\max_{a \in A} Q^*(s,a)$
4: $\bar{a} \leftarrow$ the other (suboptimal) action
5: $g \leftarrow Q^*(s,a^*) - Q^*(s,\bar{a})$
6: $T_{\mathsf{CS}} \leftarrow T_{\mathsf{CS}} \cup \{\langle s, a^*, g\rangle\}$
7: **end for**
8: $\hat{\pi}^* \leftarrow$ CS-Learn(T_{CS})
9: return $\hat{\pi}^*$

Proof. According to Corollary 2,

$$\begin{aligned}
\mathcal{L}(\pi^*_{\mathsf{CS}}) &= \sum_{s\in\mathcal{S}} \left(G^*(s,\pi^*_{\mathsf{CS}}) \cdot \sum_{t=0}^{\infty} \gamma^t \, \mu_t^{\pi^*_{\mathsf{CS}},D}(s) \right) \\
&\leq \sum_{s\in\mathcal{S}} \left(G^*(s,\pi^*_{\mathsf{CS}}) \cdot \sum_{t=0}^{\infty} \gamma^t \right) \\
&= \frac{1}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s,\pi^*_{\mathsf{CS}}) \\
&\leq \frac{\epsilon}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s).
\end{aligned}$$

The last step above is due to condition (24). □

Theorem 4. *Let ϵ be the classification error of π^*_{CI} defined in (10), then for any $\epsilon, \xi \in (0,1)$, there exists an MDP and π^*_{CI} such that:*

$$\mathcal{L}(\pi^*_{\mathsf{CI}}) > \frac{1-\xi}{1-\gamma} \sum_{s\in\mathcal{S}} G^*(s). \tag{25}$$

Note that the quantity $\sum_{s\in\mathcal{S}} G^*(s)/(1-\gamma)$ is the upper bound of $\mathcal{L}(\pi^*_{\mathsf{CI}})$.

Proof. We prove this theorem by providing an MDP and a policy that has an arbitrary low classification error (bounded by ϵ), but has an arbitrarily poor policy value. For that MDP, we show that its parameters can always be tuned to satisfy (25), with the classification error no greater than ϵ.

As depicted in Figure 3, the MDP has N states: $\mathcal{S} = \{s^1, s^2, \cdots, s^N\}$ and two actions for each state: one is the optimal action a^*, the other is the suboptimal action

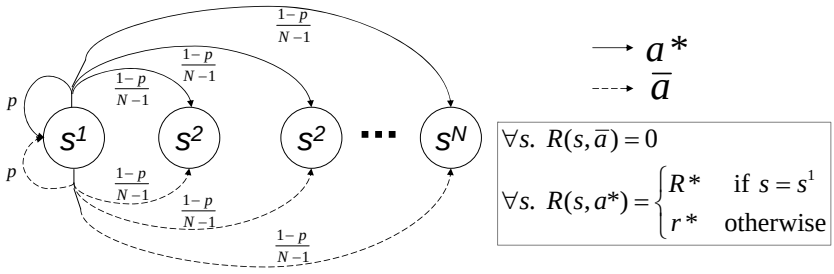

Figure 3: The MDP used by the proof of Theorem 4. Since next-state transition probabilities of all states are identical, only the probabilities for state s^1 are depicted.

$\bar{a}$. Taking a^* in state s^1 leads to a positive reward R^* while the reward is r^* in all other states; taking $\bar{a}$ always results in a zero reward. The next-state distribution is independent of the current state and actions taken: there is a probability p to go to s^1 and a probability $\frac{1-p}{N-1}$ to transition to any other state. The start state distribution is:

$$D(s) = \begin{cases} p & \text{if } s = s^1 \\ \frac{1-p}{N-1} & \text{otherwise .} \end{cases}$$

We let $R^* \gg r^* > 0$ and p be close to 1. Intuitively, s^1 is much more important than any other state from the sequential-decision-making point of view. Note that this is an ergodic, infinite-horizon MDP, and the optimal policy always chooses action a^*.

Consider the following policy π:

$$\pi(s) = \begin{cases} \bar{a} & \text{if } s = s^1 \\ a^* & \text{otherwise.} \end{cases} \tag{26}$$

Clearly,

$$G^*(s) = \begin{cases} R^* & \text{if } s = s^1 \\ r^* & \text{otherwise} \end{cases} \quad \text{and} \quad G^*(s,\pi) = \begin{cases} R^* & \text{if } s = s^1 \\ 0 & \text{otherwise.} \end{cases}$$

Observe that for all π, t:

$$\mu_t^{\pi,D}(s) = \begin{cases} p & \text{if } s = s^1 \\ \frac{1-p}{N-1} & \text{otherwise} \end{cases}$$

$$d^{\pi,D}(s) = \begin{cases} \frac{p}{1-\gamma} & \text{if} s = s^1 \\ \frac{1-p}{(N-1)(1-\gamma)} & \text{otherwise} \end{cases}.$$

By Corollary 2, the policy loss of π is:

$$\mathcal{L}(\pi) \ = \ \sum_{s\in\mathcal{S}} \left(G^*(s,\pi) \sum_{t=0}^{\infty} \gamma^t \mu_t^{\pi,D}(s) \right) \ = \ \frac{pR^*}{1-\gamma} + \frac{(1-p)r^*}{1-\gamma}. \tag{27}$$

Note that the classification error of π is $\frac{1}{N}$. By letting $\frac{1}{N} \leq \epsilon$, or equivalently, $N \geq \lceil \frac{1}{\epsilon} \rceil$, the policy has a classification error of at most ϵ. We want to find a positive $\xi \in (0,1)$ such as

$$\mathcal{L}(\pi) \;\geq\; \frac{1-\xi}{1-\gamma} \sum_{s \in \mathcal{S}} G^*(s) \;=\; \frac{1-\xi}{1-\gamma} \big(R^* + (N-1)r^*\big).$$

Using (27), this holds if

$$\begin{aligned} \frac{pR^*}{1-\gamma} + \frac{(1-p)r^*}{1-\gamma} &\geq \frac{1-\xi}{1-\gamma} \big(R^* + (N-1)r^*\big) \\ (p-1+\xi)R^* &\geq \big((1-\xi)(N-1) - (1-p)\big)r^*, \end{aligned}$$

which is true if $p - 1 + \xi > 0$, or $p > 1 - \xi$. This can always be achieved with a large enough R^*:

$$R^* \geq \frac{\big((1-\xi)(N-1) - (1-p)\big)}{p-1+\xi}\, r^*.$$

□

4.3 Focused Online Reinforcement Learning

In this section, we extend the idea of focused learning to the online RL problem, where the agent interacts with the environment online to obtain experiences in the form of $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$, but the $Q^*(s,a)$ values are *not* available for any (s,a)-pair. Our work is based on the API implementation CI-cRL (Algorithm 1). Note that this algorithm does not compute the exact greedy policy π' in (11). Instead, it uses a learning algorithm in each iteration to construct a classifier $\pi'_{\text{CI-cRL}}$ to approximate π' by solving the following cost-insensitive classification problem:

$$\pi'_{\text{CI-cRL}} = \arg \min_{\hat{\pi}' \in \Pi} \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s)). \tag{28}$$

In other words, $\pi'_{\text{CI-cRL}}$ tries to agree with π' in as many states as possible, hoping that this will mean that $\mathcal{V}(\pi'_{\text{CI-cRL}})$ is close to $\mathcal{V}(\pi')$.

Below, we develop the cost-sensitive counterpart of this algorithm following steps similar to the ones in the previous subsection. Similar to the batch-RL case, we define the importance of a state as:

$$G^\pi(s) = Q^\pi(s, a^*_\pi(s)) - Q^\pi(s, \bar{a}_\pi(s)). \tag{29}$$

where $a^*_\pi(s)$ and $\bar{a}_\pi(s)$ are the greedy and non-greedy actions in s with respect to π. Intuitively, $G^\pi(s)$ measures how much additional reward can be obtained by switching the action from $\bar{a}_\pi(s)$ to $a^*_\pi(s)$ in state s, and then following policy π thereafter. Likewise, we define

$$\begin{aligned} G^\pi(s,\tau) &= Q^\pi(s, a^*_\pi(s)) - Q^\pi(s, \tau(s)) \\ &= G^\pi(s) \cdot \mathcal{I}(\tau(s) \neq a^*_\pi(s)). \end{aligned} \tag{30}$$

Corollary 5. *If during policy improvement a policy π is changed to $\hat{\pi}'$, which is an approximation to the greedy policy π', and*

$$\forall s \in \mathcal{S},\ |d^{\pi',D}(s) - d^{\hat{\pi}',D}(s)| < \varepsilon, \tag{31}$$

then

$$\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}') \leq \sum_{s\in\mathcal{S}} G^{\pi}(s,\hat{\pi}') \cdot d^{\hat{\pi}',D}(s) + \varepsilon \sum_{s\in\mathcal{S}} G^{\pi}(s). \tag{32}$$

Proof. In Theorem 1, fix $\nu = \pi$; let τ be π' and $\hat{\pi}'$, respectively. Then we have

$$\mathcal{V}(\pi') - \mathcal{V}(\pi) = \sum_{s\in\mathcal{S}} \left(G^{\pi}(s, \pi \to \pi') \cdot d^{\pi',D}(s) \right),$$
$$\mathcal{V}(\hat{\pi}') - \mathcal{V}(\pi) = \sum_{s\in\mathcal{S}} \left(G^{\pi}(s, \pi \to \hat{\pi}') \cdot d^{\hat{\pi}',D}(s) \right).$$

By subtracting these two equations we obtain

$$\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}') = \sum_{s\in\mathcal{S}} g(s)\ ,$$

where

$$g(s) = G^{\pi}(s, \pi \to \pi') \cdot d^{\pi',D}(s) - G^{\pi}(s, \pi \to \hat{\pi}') \cdot d^{\hat{\pi}',D}(s).$$

If $\pi(s) = a^*_{\pi}(s)$, then

$$G^{\pi}(s, \pi \to \pi') = 0$$
$$G^{\pi}(s, \pi \to \hat{\pi}') = -G^{\pi}(s) \cdot \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s))$$
$$\begin{aligned} g(s) &= G^{\pi}(s) \cdot \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s)) \cdot d^{\hat{\pi}',D}(s) \\ &= G^{\pi}(s, \hat{\pi}') \cdot d^{\hat{\pi}',D}(s). \end{aligned}$$

If $\pi(s) \neq a^*_{\pi}(s)$, then

$$\begin{aligned} G^{\pi}(s, \pi \to \pi') &= G^{\pi}(s) \\ G^{\pi}(s, \pi \to \hat{\pi}') &= G^{\pi}(s) \cdot \mathcal{I}(\hat{\pi}'(s) = \pi'(s)) \\ &= G^{\pi}(s) \cdot \big(1 - \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s))\big) \end{aligned}$$

$$\begin{aligned} g(s) &= G^{\pi}(s) \cdot d^{\pi',D}(s) - G^{\pi}(s) \cdot \big(1 - \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s))\big) \cdot d^{\hat{\pi}',D}(s) \\ &= G^{\pi}(s)\big(d^{\pi',D}(s) - d^{\hat{\pi}',D}(s) + \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s)) \cdot d^{\hat{\pi}',D}(s)\big) \\ &= G^{\pi}(s)\big(d^{\pi',D}(s) - d^{\hat{\pi}',D}(s)\big) + G^{\pi}(s) \cdot \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s)) \cdot d^{\hat{\pi}',D}(s) \\ &\leq \varepsilon \cdot G^{\pi}(s) + G^{\pi}(s) \cdot \mathcal{I}(\hat{\pi}'(s) \neq \pi'(s)) \cdot d^{\hat{\pi}',D}(s) \\ &= \varepsilon \cdot G^{\pi}(s) + G^{\pi}(s, \hat{\pi}') \cdot d^{\hat{\pi}',D}(s) \end{aligned}$$

In either case, $g(s) \leq \varepsilon \cdot G^{\pi}(s) + G^{\pi}(s, \hat{\pi}') \cdot d^{\hat{\pi}',D}(s)$. The corollary follows immediately since $\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}') = \sum_s g(s)$. □

This corollary bounds $\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}')$ via $G^{\pi}(s, \hat{\pi}')$ in each individual state. Note that π' and $\mathcal{V}(\pi')$ are constant if π is fixed. Hence,

$$\arg\max_{\hat{\pi}' \in \Pi} \mathcal{V}(\hat{\pi}') \;=\; \arg\min_{\pi' \in \Pi} \left[\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}')\right].$$

In other words, we can find the greedy policy $\hat{\pi}'$ to maximize $\mathcal{V}(\hat{\pi}')$ by maximizing $\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}')$. The representation (32) is, however, still of a limited practical value since $d^{\hat{\pi}', D}(s)$ and ε are usually unavailable to the agent. Thus, we propose a similar practical approximation:

$$\pi'_{\mathsf{CS\text{-}cRL}} = \arg\min_{\hat{\pi}' \in \Pi} \sum_{s \in \mathcal{S}} G^{\pi}(s, \hat{\pi}') \tag{33}$$

$$\approx \arg\min_{\hat{\pi}' \in \Pi} \left[\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}')\right]. \tag{34}$$

We prefer $\pi'_{\mathsf{CS\text{-}cRL}}$ as opposed to $\pi'_{\mathsf{CI\text{-}cRL}}$ since it takes state importance into account, which is highly related to the policy value according to Corollary 5. Again, by replacing $G^{\pi}(s, \pi)$ in (33) with (30), solving $\pi'_{\mathsf{CS\text{-}cRL}}$ is turned precisely into a cost-sensitive classification problem, where s is the attribute, $a^*_{\pi}(s)$ is the desired class label, and $G^{\pi}(s)$ is the misclassification cost.

The analysis above becomes the basis of our new algorithm called CS-cRL (Cost-Sensitive classification-based RL), which is depicted in Algorithm 4. It is a cost-sensitive version of CI-cRL (cf. Algorithm 1). The algorithm relies on the subroutine CS-Learn that returns a cost-sensitive classifier based on the presented training data.

The following two theorems answer the questions: (i) whether the approximation in (34) is sound, and (ii) whether it is preferable to solve $\pi'_{\mathsf{CS\text{-}cRL}}$ as opposed to $\pi'_{\mathsf{CI\text{-}cRL}}$. Theorem 6 below states that if ε is small and if the classifier is of a sufficiently high quality, then $\pi'_{\mathsf{CS\text{-}cRL}}$ will be close to π' in terms of policy value. In contrast, Theorem 7 asserts that as long as $\pi'_{\mathsf{CI\text{-}cRL}}$ has a non-zero classification error, the difference $\mathcal{V}(\pi') - \mathcal{V}(\pi'_{\mathsf{CI\text{-}cRL}})$ can be arbitrarily close to its upper bound, $\sum_{s \in \mathcal{S}} G^{\pi}(s)/(1-\gamma)$.

Theorem 6. *If $\pi'_{\mathsf{CS\text{-}cRL}}$ has a sufficiently high quality, i.e., there exists $\epsilon > 0$ such that*

$$\frac{\sum_{s \in \mathcal{S}} G^{\pi}(s, \pi'_{\mathsf{CS\text{-}cRL}})}{\sum_{s \in \mathcal{S}} G^{\pi}(s)} < \epsilon,$$

then, using the notation in (31), we have

$$\mathcal{V}(\pi') - \mathcal{V}(\pi'_{\mathsf{CS\text{-}cRL}}) \leq \left(\frac{\epsilon}{1-\gamma} + \varepsilon\right) \sum_{s \in \mathcal{S}} G^{\pi}(s).$$

Algorithm 4 CS-cRL: Cost-Sensitive classification-based RL. CS-Learn is a subroutine that induces a cost-sensitive classifier from the input training data.

Input:
- M: generative model
- $\mathcal{T}$: set of training states
- γ: discount factor
- K: number of trajectories
- H: maximum length of trajectories

$\hat{\pi}' \leftarrow$ random policy
repeat
  $\pi \leftarrow \hat{\pi}'$
  $T_{\mathsf{CS\text{-}cRL}} \leftarrow \emptyset$
  for all $s \in \mathcal{T}$ **do**
    for all $a \in A$ **do**
      $\hat{Q}^{\pi}(s,a) \leftarrow \mathsf{Rollout}(M, s, a, \gamma, \pi, K, H)$
    end for
    $\hat{a}^*_{\pi} \leftarrow \arg\max_{a \in A} \hat{Q}^{\pi}(s,a)$
    if $\hat{Q}^{\pi}(s,\hat{a}^*_{\pi}) > \hat{Q}^{\pi}(s,a)$ for all $a \neq \hat{a}^*_{\pi}$ **then**
      $g \leftarrow \hat{Q}^{\pi}(s,\hat{a}^*_{\pi}) - \hat{Q}^{\pi}(s,a)$
      $T_{\mathsf{CS\text{-}cRL}} \leftarrow T_{\mathsf{CS\text{-}cRL}} \cup \{\langle s, \hat{a}^*_{\pi}, g\rangle\}$
    end if
  end for
  $\hat{\pi}' \leftarrow \mathsf{CS\text{-}Learn}(T_{\mathsf{CS\text{-}cRL}})$
until $\pi \approx \hat{\pi}'$
return π

Proof. According to Corollary 5,

$$\begin{aligned}
\mathcal{V}(\pi') - \mathcal{V}(\pi'_{\mathsf{CS\text{-}cRL}}) \;&\leq\; \sum_{s\in\mathcal{S}} \left(G^{\pi}(s, \pi'_{\mathsf{CS\text{-}cRL}}) \cdot \sum_{t=0}^{\infty} \gamma^t \right) + \varepsilon \cdot \sum_{s} G^{\pi}(s) \\
&= \frac{1}{1-\gamma} \sum_{s\in\mathcal{S}} G^{\pi}(s, \pi'_{\mathsf{CS\text{-}cRL}}) + \varepsilon \cdot \sum_{s} G^{\pi}(s) \\
&< \frac{\epsilon}{1-\gamma} \sum_{s\in\mathcal{S}} G^{\pi}(s) + \varepsilon \cdot \sum_{s} G^{\pi}(s) \;=\; \left(\frac{\epsilon}{1-\gamma} + \varepsilon \right) \sum_{s\in\mathcal{S}} G^{\pi}(s).
\end{aligned}$$

The second inequality above is due to the condition stated in the theorem. □

Theorem 7. *Let ϵ be the classification error of $\pi'_{\mathsf{CI\text{-}cRL}}$ (i.e., the objective function in (28)), then for any $\epsilon, \xi \in (0,1)$, there exists an MDP and $\pi'_{\mathsf{CI\text{-}cRL}}$ such that:*

$$\mathcal{V}(\pi') - \mathcal{V}(\pi'_{\mathsf{CI\text{-}cRL}}) > \frac{1-\xi}{1-\gamma} \sum_{s\in\mathcal{S}} G^{\pi}(s).$$

Proof. This existential proof is similar to the proof of Theorem 4, using the same example, except setting $\pi = \pi' = \pi^*$. Let $\hat{\pi}'$ be defined by (26). Then following the same calculations in the proof with the fact that $\varepsilon = 0$, we can always find N, p, r^*, and R^* such that π' has an ϵ-classification error but $\mathcal{V}(\pi') - \mathcal{V}(\hat{\pi}')$ can be arbitrarily close to its upper bound, $\sum_{s \in \mathcal{S}} G^{\pi}(s)/(1-\gamma)$. □

5 Future Work and Conclusions

Our results open several avenues for future research. First, by placing a threshold on the importance of a training state we can prune down the training data set. An immediate advantage is a reduction of the training time. Another direction is to investigate alternative ways of defining state importance. Note that Corollary 5 depends on the factor ε, and the approximation steps were taken in (21) and (34). Thus, it may be possible to find better measures of state importance and/or better approximations (e.g., estimating $d^{\pi,D}(s)$ via density estimation [Duda et al., 2000] or online sampling [Barto et al., 1995]) leading to better RL algorithms. Finally, it is interesting to extend our results to multiple-action cases as well as to situations without generative models.

In this paper, we have investigated the problem of focusing attention in classification-based RL, considering two settings using two suitable state importance measures. We showed theoretically that focused learning leads to reasonable performance guarantees, while their non-focusing counterparts do not.

Acknowledgements

We gratefully appreciate the helpful discussions with Rich Sutton, Doina Precup, Ron Parr, Ilya Levner, and Greg Lee. This research was supported by the Alberta Ingenuity Center for Machine Learning (AICML), the University of Alberta, and the National Science and Engineering Research Council.

References

[Baird, 1993] Baird, L. (1993). Advantage updating. Technical Report WL-TR-93-1146, Wright-Patterson Air Force Base, OH. Available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145.

[Barto et al., 1995] Barto, A. G., Bradtke, S. J., and Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1–2):81–138.

[Baxter and Bartlett, 1999] Baxter, J. and Bartlett, P. (1999). Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, Research School of Information Sciences and Engineering, Australian National University, July.

[Baxter et al., 1999] Baxter, J., Weaver, L., and Bartlett, P. (1999). Direct gradient-based reinforcement learning: II. Gradient ascent algorithms and experiments. Technical report, Research School of Information Sciences and Engineering, Australian National University, September.

[Berenji and Vengerov, 2003] Berenji, H. R. and Vengerov, D. (2003). A convergent actor critic based fuzzy reinforcement learning algorithm with application to power management of wireless transmitters. *IEEE Transactions on Fuzzy Systems*, 11(4):478–485, August.

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

[Crites and Barto, 1996] Crites, R. H. and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8 (NIPS-95)*, pages 1017–1023.

[Dietterich and Wang, 2002] Dietterich, T. G. and Wang, X. (2002). Batch value function approximation via support vectors. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*, pages 1491–1498.

[Draper et al., 2000] Draper, B., Bins, J., and Baek, K. (2000). ADORE: Adaptive object recognition. In *International Conference on Vision Systems*, Spain.

[Duda et al., 2000] Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience, 2nd edition.

[Fern et al., 2004] Fern, A., Yoon, S. W., and Givan, R. (2004). Approximate policy iteration with a policy language bias. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

[Howard, 1960] Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

[Kakade and Langford, 2002] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-02)*, pages 267–274.

[Kearns et al., 2000] Kearns, M. J., Mansour, Y., and Ng, A. Y. (2000). Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 1001–1007.

[Kearns et al., 2002] Kearns, M. J., Mansour, Y., and Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49:193–208.

[Kim et al., 2003] Kim, H. J., Langford, J., and Ng, A. Y. (2003), Policy search via supervised learning. In *Proceedings of the Machine Learning Reductions Workshop*, Chicago, IL.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári C. (2006). Bandit based Monte-Carlo planning. In *Proceedings of the Seventeenth European Conference on Machine Learning (ECML06)*, pages 282–293.

[Lagoudakis and Parr, 2003] Lagoudakis, M. G. and Parr, R. (2003). Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pages 424–431.

[Langford and Zadrony, 2003] Langford, J. and Zadrozny, B. (2003). Reducing T-step reinforcement learning to classification. In *Proceedings of the Machine Learning Reductions Workshop*, Chicago, IL.

[Langford and Zadrony, 2005] Langford, J. and Zadrozny, B. (2005). Relating reinforcement learning performance to classification performance. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML-05)*, pages 473–480.

[Levner and Bulitko, 2004] Levner, I. and Bulitko, V. (2004). Machine learning for adaptive image interpretation. In *Proceedings of the Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04)*, pages 870–876.

[Li, 2004] Li, L. (2004). Focus of attention in reinforcement learning. Master's thesis, University of Alberta, Edmonton, AB, Canada. Tech report TR07-12, Department of Computing Science, University of Alberta. URL: http://www.cs.ualberta.ca/research/techreports/.

[Li et al., 2004a] Li, L., Bulitko, V., and Greiner, G. (2004). Batch reinforcement learning with state importance. In *Proceedings of the Fifteenth European Conference on Machine Learning (ECML-04)*, pages 566–568.

[Li et al., 2004b] Li, L., Bulitko, V., and Greiner, G. (2004). Focusing attention in reinforcement learning. In Daniela Pucci de Farias, Shie Mannor, Doina Precup, and Georgios Theocharous, editors, *AAAI-04 Workshop on Learning and Planning in Markov Processes: Advances and Challenges*, pages 43–48. AAAI Technical Report WS-04-08.

[Lin, 1992] Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321.

[Moore and Atkeson, 1993] Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130.

[Ng and Jordan, 2000] Ng, A. Y. and Jordan, M. I. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 406–415.

[Ng et al., 2004] Ng, A. Y., Kim, H. J., Jordan, M. I., and Sastry, S. (2004). Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, New York.

[Randløv and Alstrøm, 1998] Randløv, J. and Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth Interational Conference on Machine Learning (ICML-98)*, pages 463–471.

[Scheffer et al., 1997] Scheffer, T., Greiner, R., and Darken, C. (1997) Why experimentation can be better than perfect guidance. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)*, pages 331–339.

[Singh et al., 2002] Singh, S. P., Litman, D., Kearns, M. J., and Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133.

[Singh and Yee, 1994] Singh, S. P. and Yee, R. C. (1994). An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233.

[Sutton, 1988] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

[Sutton, 1990] Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-90)*, pages 216–224.

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

[Sutton et al., 2000] Sutton, R. S., McAllester, D., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NIPS-99)*, pages 1057–1063.

[Tesauro, 1992] Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

[Tesauro, 1995] Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March.

[Tesauro and Galperin, 1997] Tesauro, G. and Galperin, G. R. (1997). On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing Systems 9 (NIPS-96)*, pages 1068–1074.

[Turney, 2000] Turney, P. (2000). Types of cost in inductive concept learning. In *Proceedings of the ICML-00 Workshop on Cost-Sensitive Learning*, pages 15–21.

[Wang and Dietterich, 1999] Wang, X. and Dietterich, T. G. (1999). Efficient value function approximation using regression trees. In *Proceedings of the IJCAI-99 Workshop on Statistical Machine Learning for Large-Scale Optimization*.

[Watkins, 1989] Watkins, C J.C.H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambridge, UK.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[Yoon et al., 2002] Yoon, S. W., Fern, A., and Givan, R. (2002). Inductive policy selection for first-order MDPs. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 568–576.

[Zhang and Dietterich, 1995] Zhang, W. and Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1114–1120.