Relating reinforcement learning performance to classification performance

Conference Paper · January 2005

DOI: 10.1145/1102351.1102411 · Source: DBLP

Bianca Zadrozny
IBM

Relating Reinforcement Learning Performance to Classification Performance

John Langford JL@HUNCH.NET
TTI-Chicago, 1427 E 60th Street, Chicago, IL 60637 USA

Bianca Zadrozny ZADROZNY@US.IBM.COM
IBM T.J. Watson, 1101 Kitchawan Road, Yorktown Heights, NY 10598 USA

Abstract

We prove a quantitative connection between the expected sum of rewards of a policy and binary classification performance on created subproblems. This connection holds without any unobservable assumptions (no assumption of independence, small mixing time, fully observable states, or even hidden states) and the resulting statement is independent of the number of states or actions. The statement is critically dependent on the size of the rewards and prediction performance of the created classifiers.

We also provide some general guidelines for obtaining good classification performance on the created subproblems. In particular, we discuss possible methods for generating training examples for a classifier learning algorithm.

1. Introduction

Motivation. Reinforcement learning on real-world problems requires coping with large (possibly infinite) state, observation, and action spaces. However, many reinforcement learning algorithms (such as Q-Learning (Watkins, 1989) and other algorithms (Sutton & Barto, 1998)) are only tractable for small environments with relatively few states and actions. This disparity is commonly handled by adding a function approximation step to the learning algorithms developed for small problems.

This approach is problematic. On the theoretical side, strong and unrealistic assumptions about function approximation accuracy (such as assuming a small estimation error for the entire space) are required to make weak statements such as "the performance of an iteration does not degrade much" for approximate policy iteration (Bertsekas & Tsitsiklis, 1996). The practical symptom of these weak performance guarantees is that the application of reinforcement learning to real-world problems typically requires the attention of experts, many samples of experience, a great deal of computational time, or a combination of all three for success.

In order to eliminate this reliance on unrealistic function approximation, reinforcement learning algorithms must be explicitly designed to be robust against prediction errors. The method used here relates reinforcement learning performance to classification performance through a reduction from reinforcement learning to classification. Note that we are *not* claiming that "reinforcement learning is classification" because knowing what to predict and predicting it well does not solve the temporal credit assignment problem—how are past actions in an interactive environment related to future reward?

A Simple Relationship. The goal of reinforcement learning is to find a policy

$$\pi : (O \times A \times R)^* \times O \to A$$

which maps a history of (observation,action,reward) sequences and a new observation to an action so as to maximize the expected sum of rewards in an unknown environment. The goal of (binary) classifier learning is to produce a classifier

$$c : X \to \{0, 1\}$$

which maps features to a binary label so as to minimize the error rate with respect to some unknown distribution.

A simple way to connect reinforcement learning to classifier learning is to notice that a policy can be represented by a binary classifier which, given observations as input, predicts the correct action. Since for any 2-action environment there is a sequence of correct actions that maximizes the rewards, choosing the

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

correct actions would be equivalent to choosing the correct labels and, consequently, minimizing the error rate of the classifier.

However, some important issues are left open:

1. What if the number of actions is greater than 2?
2. What if the classifier is sometimes wrong? Shouldn't later errors alter the correctness of earlier decisions?
3. If a classifier learning algorithm optimizes classification performance to some small error rate, does that imply a good policy? What if the environment is fundamentally noisy?

Results. In section 2, we address each of these issues and quantify the relationship between classification error rate and the expected sum of rewards performance of an induced policy in noisy environments with infinite states, multiple actions, and varying stochastic rewards. The analysis holds without any of the common assumptions used in past reinforcement learning work such as independence, Markovian state transitions and observable states. This is possible because we condition the performance of a policy on the performance of a binary classifier.

Note, however, that the existence of a relationship between policy performance and classifier error rate does not provide any guidance on how to obtain a classifier with a small error rate. In section 3 we provide some general guidelines for obtaining such a classifier. In particular, we discuss possible methods for generating representative examples from which to train a classifier learning algorithm.

1.1. Past Work

The RLGen analysis (Langford & Zadrozny, 2003) and algorithm relates the error rate of a classifier on certain examples created with a generative model to the performance of a policy. The work presented here is significantly more general in scope. In particular, it does not necessarily assume a generative model.

The Policy Search by Dynamic Programming (Bagnell et al., 2004) analysis is similar to the one presented here, so we make pointwise comparisons throughout the paper. A very rough summary is that we require fewer assumptions, achieve tighter bounds without a dependence on the distance between two measures, and the analysis motivates different design choices.

The Conservative Policy Iteration algorithm (Kakade & Langford, 2002) modifies approximate policy iteration so that policy updates result in improvement whenever a certain regression problem is solved well. This technique is complementary to the analysis presented here.

The algorithms of Lagoudakis and Parr (2003) and Fern et al. (2004) rely upon approximate policy iteration in order to construct a policy formed from a classifier. Thus, they inherit the non-robust guarantees of the approximate policy iteration framework.

Some earlier work (see theorem 3.1 of Khardon (1999)) can be understood as relating policy learning to classification.

2. A Reduction of Reinforcement Learning to Classification

In this section we connect the prediction ability of a classifier learning algorithm on certain problems to the performance of a reinforcement learning algorithm. This form of analysis[1], known as a reduction, has some subtle implications. In particular, assuming that the classifier learning algorithm succeeds implies that:

1. We do not need to consider exploration explicitly.
2. We do not need to generate training sets for classifier learning.
3. We do not need to make assumptions (such as smoothness, Markovian dynamics, observable states and fast mixing) about the environment which are typically introduced in order to create predictability.

What is the analysis good for if it hides crucial issues in reinforcement learning? It gives us insight into what are the critical prediction problems necessary for solving reinforcement learning and the relative difficulty of these problems. This may lead to algorithms which attempt to more directly answer these prediction problems, yielding better performance.

2.1. Definitions

We define the reinforcement learning problem in a very general manner that can capture most standard definitions as special or equivalent cases. (See the Appendix for a discussion). The reason for doing this is that the analysis becomes both more general and simpler.

[1] One way to think about this form of analysis is "20 questions with faults". I promise to answer any questions about the location of a hidden object but lie some number of times. Your goal (as the designer) is to come up with a good set of questions such that if I lie rarely, you still have good performance.

Inspired by predictive state representations (Singh et al., 2004) we avoid the use of "state" in the definition of reinforcement learning. The following definition is for a decision process (DP).

Definition 2.1 *(Reinforcement Learning Problem) A reinforcement learning problem D is defined as a conditional probability table $D(o', r|(o, a, r)^*, o, a)$ on a set of observations O and rewards $r \in [0, \infty)$ given any (possibly empty) history $(o, a, r)^*$ of past observations, actions (from action set A), and rewards.*

For simplicity, we avoid constraining a policy to be stationary and define the goal of reinforcement learning as optimizing the T-step sum of rewards.

Definition 2.2 *(Reinforcement Learning Goal) Given some horizon T, find a policy $\pi : (o, a)^* \rightarrow a$, optimizing the expected sum of rewards:*

$$\eta(D, \pi) = E_{(o,a,r)^T \sim D,\pi} \left[\sum_{t=1}^{T} r_t \right]$$

where r_t is the t-th observed reward. Here, the expectation is over the process which generates a history using D and choosing actions from π.

2.2. The Reduction

Preparation. Given the above definition of a reinforcement learning problem, there exists some optimal policy π^* which maximizes the expected sum of rewards for any valid history. We use π^* in defining a sequence of classification problems in the reduction.

The reduction initially assumes that we can solve multiclass reward-sensitive classification problems[2]. In reward-sensitive classification, the rewards for predicting a label correctly or incorrectly for an example are not necessarily fixed at 1 and 0 respectively (as in usual accuracy-maximizing classification) but can be any positive real number. We rely on other work (Langford & Beygelzimer, 2005) which states that any cost-sensitive (or equivalently, reward-sensitive) classification problem can be broken down into multiple binary classification problems.

Algorithm. The reduction consists of sequentially building reward-sensitive classification problems for each time step, always assuming that the optimal actions will be executed in the remaining steps. More concretely:

1. Given the empty history, predict the action which optimizes the expected sum of rewards assuming that π^* is followed for the next $T - 1$ steps.
2. Given that we use the first predictor, predict the action which optimizes the expected sum of rewards assuming that π^* is followed for the next $T - 2$ steps.
3. Given that we use the first and second predictors, predict the action which optimizes the expected sum of rewards assuming π^* is followed for the next $T - 3$ steps. And so forth.

Discussion. These problems are *not* "predict according to the measure over states induced by π^* at time step T" as in Bagnell et al. (2004). Instead, we take into account the policies that were constructed in previous steps. We chose to construct the problems in this manner because this allows stronger performance guarantees. Similarly, the prediction problems are ordered from first-to-last rather than last-to-first as in Bagnell et al. (2004). With the last-to-first approach a wrong action choice made in the first step renders later prediction performance irrelevant. With the first-to-last approach, a wrong action choice made in the last step given the first steps merely degrades performance. This observation allows us to eliminate a dependence on the "variation distance" between the measures induced by π^* and π. This dependence is inherent to the algorithm and setting analyzed. From the reduction's viewpoint, the dependence is difficult to accept because it can be large even for π of equivalent performance to π^*. (In experiments not reported here the ordering also appears to be important.)

Note that this way of constructing the prediction problems is *not* optimal in general. (It is, however, the best for which we know how to prove a performance guarantee.) The first classifier is learned assuming a perfect policy for the later steps. But because the later policy may be imperfect, the first classifier should be adjusted to take later mistakes into accounts. This can be done safely (without harming performance) using either an iteration-over-T classifier update mechanism (as in Bagnell et al. (2004)) or a mixture update mechanism (as in Langford and Zadrozny (2003)) similar to the conservative policy iteration algorithm (Kakade & Langford, 2002).

2.3. Analysis

Definitions. The reduction algorithm gives us a sequence of reward-sensitive classifiers $c_1, ..., c_T$, one for each time step. We can consider them together as one policy π defined by:

$$\pi((o, a, r)^{t-1}, o_t) = c_t((o, a, r)^{t-1}, o_t)$$

where o_t is the tth observation and $(o, a, r)^{t-1}$ is the first $t - 1$ observed (observation, action, reward)

[2] These problems can also be thought of as "do regression and pick the best".

triples. Each classifier has a reward rate:

$$l_t(D,\pi,c_t) = E_{(a,o,r)^T \sim D,(\pi,c_t,\pi^*)}\left[\sum_{t'=t}^{T} r_{t'}\right]$$

where the composite policy (π, c_t, π^*) uses π for $t-1$ steps then c_t for one step and π^* for the remaining $T-t$ steps.

The regret of a policy is defined as the difference between the expected reward of the policy and the expected reward of the optimal policy:

$$\rho(D,\pi) = \eta(D,\pi^*) - \eta(D,\pi)$$

Each classifier has a regret:

$$\rho_t(D,\pi,c_t) = l_t(D,\pi,\pi_t^*) - l_t(D,\pi,c_t)$$

Lemma 2.3 *(Relative RL Regret) For every reinforcement learning problem D and every policy $\pi = c_1, ..., c_T$, we have:*

$$\rho(D,\pi) = \sum_{t=1}^{T} \rho_t(D,\pi,c_t)$$

This is the core technical result. The most similar previous result was the PSDP analysis (Bagnell et al., 2004). The PSDP analysis makes an explicit assumption that the distribution over states of a near optimal policy is known and characterizes the performance in terms of the distance between the distribution over states of the learned policy and the near optimal policy. The analysis here is more basic. In particular, the same assumption could be added here yielding a first-to-last PSDP-like analysis and explicit algorithm.

Proof.

$$\begin{aligned}
\rho(D,\pi) &= \eta(D,\pi^*) - \eta(D,\pi) \\
&= E_{(a,o,r)^T \sim D,\pi^*}\left[\textstyle\sum_{t=1}^{T} r_t\right] \\
&\quad - E_{(a,o,r)^T \sim D,\pi}\left[\textstyle\sum_{t=1}^{T} r_t\right] \\
&= \textstyle\sum_{t=1}^{T} E_{(a,o,r)^T \sim D,(\pi,\pi_t^*,\pi^*)}\left[\textstyle\sum_{t'=1}^{T} r_{t'}\right] \\
&\quad - E_{(a,o,r)^T \sim D,(\pi,c_t,\pi^*)}\left[\textstyle\sum_{t'=1}^{T} r_{t'}\right] \\
&= \textstyle\sum_{t=1}^{T} E_{(a,o,r)^T \sim D,(\pi,\pi_t^*,\pi^*)}\left[\textstyle\sum_{t'=t}^{T} r_{t'}\right] \\
&\quad - E_{(a,o,r)^T \sim D,(\pi,c_t,\pi^*)}\left[\textstyle\sum_{t'=t}^{T} r_{t'}\right] \\
&= \textstyle\sum_{t=1}^{T} \left[l_t(D,\pi,\pi_t^*) - l_t(D,\pi,c_t)\right] \\
&= \textstyle\sum_{t=1}^{T} \rho_t(D,\pi,c_t)
\end{aligned}$$

The first, second, fifth and sixth equalities are definitions. The third equality follows from the "telescoping property" of sums($\sum_{i=1}^{T} a_{i-1} - a_i = a_0 - a_T$). The fourth equality follows because $\sum_{t'=1}^{t-1} r_t$ has an identical expectation for both processes. ▌

This lemma shows that the regret of the composite policy created by the reduction can be decomposed into the regrets of the component classifiers. This has the following implications:

1. Small regret relative to the optimal policy can be achieved by solving *only* these T reward-sensitive classification problems. This is in contrast to methods like Q-learning(Watkins, 1989) which attempt to predict the value function of states and actions at many locations. Similarly, the algorithm E^3 (Kearns & Singh, 1998) can be thought of as attempting to learn a model sufficiently powerful to approximately predict the value of all policies.

2. In the presence of errors, the optimal reward-sensitive classification problems that need to be solved are partially associated with the optimal policy and partially associated with earlier suboptimal choices.

Note again that this result does not tell us how to solve these prediction problems because it does not tell us how to construct training sets for reward-sensitive classifier learning at each step. The next section discusses some methods for solving these prediction problems.

To Binary Classification. This reduction can be easily combined with the SECOC (Sensitive Error Correcting Output Code) reduction (Langford & Beygelzimer, 2005) which reduces (multi-class) cost-sensitive classification to binary (accuracy-maximizing) classification. The relevant details of this reduction are that it maps maps cost-sensitive examples to binary classification examples and (hence) measures D on cost-sensitive examples to measures SECOC(D) over binary classification examples. For any binary classifier distribution D', the error rate of a binary classifier c is defined as:

$$e(D',c) = \Pr_{(x,y)\sim D'}(c(x) \neq y)$$

And the binary classification regret is defined as:

$$\rho_b(D',c) = e(D',c) - \min_{c'} e(D',c')$$

The above lemma and the SECOC reduction give the following result.

Theorem 2.4 *(Reinforcement Learning To Binary Classification) For every reinforcement learning problem D and policy π defined by binary classifiers c_t obtained through the SECOC reduction, we have:*

$$\rho(D,\pi) \leq 4\sum_{t=1}^{T}\sqrt{\rho_b(SECOC(D_t),c_t)\sum_{a}\rho_t(D_t,\pi,c_a)}$$

where c_a is a classifier that always takes action a and D_t is the distribution over sequences, $(o, a, r)^{t-1}, o_t$.

Proof. The proof is a conjunction of an earlier theorem (Langford & Beygelzimer, 2005) that reduces cost-sensitive classification to binary classification and the Relative RL regret lemma 2.3.

Cost-sensitive classification is defined by a distribution D over examples $(x, l_1, ..., l_k)$ where $l_i \in [0, \infty)$ is the cost of predicting class i (without loss of generality, we assume that there exists an i with $l_i = 0$). The cost-sensitive loss is defined as:

$$l_{cs}(D, h) = E_{(x,l_1,...,l_k)\sim D}\left[l_{h(x)}\right]$$

which is the expected cost of the choices made. Similarly the cost-sensitive regret is defined as:

$$\rho_{cs}(D, h) = l_{cs}(D, h) - \min_{h'} l_{cs}(D, h').$$

which is the expected cost of the choices made minus the expected cost of the optimal choices.

The SECOC reduction naturally maps D to a new distribution SECOC(D) over binary examples $(x', 0)$ or $(x', 1)$ and has the following guarantee for a multiclass classifier h_c formed out of the binary classifier c:

$$\rho_{cs}(D, h_c) \leq 4\sqrt{\rho_b(\text{SECOC}(D), c) E_{(x,l_1,...,l_k)\sim D} \sum_i l_i}.$$

For our reinforcement learning reduction, the number of classes is equal to the number of available actions ($k = |A|$) and the loss l_a is the difference in reward between acting according to the optimal action (then following π^*) and acting according to a (then following π^*). The expected value of this difference is the regret $\rho_t(D_t, \pi, c_a)$.

Applying this analysis to each time step t and combining the results with lemma 2.3 gives the theorem. ▌

This theorem shows that the regret of the policy (on the left-hand side) is bounded by the regret of the binary classifiers $\rho_b(c_t, \text{SECOC}(D_t))$ and a particular sum. The sum is over actions (implying that a problem with many actions is inherently more difficult) and the term is the regret of following action a and then acting according to the optimal policy rather than acting according to the optimal policy at timestep t. This last term is a time-dependent form of advantage (Baird, 1993) which implies that achieving small policy regret is harder when there is a large difference in the value of actions.

3. Obtaining Good Prediction Performance

Good classifier prediction performance is usually achieved through a combination of several factors including expert knowledge, the use of learning algorithms that are known to perform well for similar tasks and, perhaps most importantly, the availability of training examples drawn from the same distribution as the examples about which the classifier is expected to make predictions. In this section, we explain each of these factors and discuss how they affect the prediction performance needed for solving reinforcement learning problems.

3.1. Expert knowledge

In many practical machine learning situations, a human expert knows (at least implicitly) how to make correct predictions and learning is used to transfer this knowledge to a machine. This is true, for example, in text classification or face recognition. In these situations, it is often possible to facilitate learning by embedding the expert's knowledge into the feature representation. In other words, we can represent the examples using features that are known by the expert to be predictive for the particular problem in hand.

Another way of using expert's knowledge to improve predictive performance is to use a prior (in a Bayesian setting) that favors solutions which seem more reasonable to the expert. This makes the task of learning easier since it reduces the space of possible solutions and, consequently, the amount of data that is needed for achieving a certain performance level.

3.2. Past performance in other problems

Traditionally, in applied machine learning research, we test classifier learning algorithms on many different problems and prefer algorithms that exhibit better performance across a wide variety of problems. This methodology implies that learning algorithms that are considered successful in practice have over time implicitly incorporated information about the characteristics of typical prediction problems.

One significant outstanding question in the context of reductions to classification is whether or not the prediction problems created through reductions share the characteristics of typical classification problems. If this is the case, classifier learners that have been developed using natural problems as benchmarks will tend to also be successful for problems created through reductions. Experimental results for some simple reductions (Zadrozny et al., 2003; Beygelzimer et al., 2004; Langford & Zadrozny, 2005) suggest that this is true, but more work needs to be done.

3.3. Training examples

The main source of information for a classifier learning algorithm is a set of labeled training examples that inform the learner what are the correct labels for a set of observations. Most classifier learning algorithms are designed with the assumption that the training examples are independently drawn from the same distribution as the examples for which the classifier is expected to make predictions. This assumption cannot always be fulfilled for reinforcement learning problems. For example, suppose that we have an agent in the world who travels to a conference in Bonn, Germany. If this is the first trip abroad for the agent, it is difficult to think of any past experiences as being drawn from the same distribution as the agent is being tested on.

Nevertheless, there are several mechanisms available for providing examples from the (approximate) test distribution that rely on some piece of extra information (such as a human expert or a generative model).

1. **RL as Information Transfer**. There are a number of applications of reinforcement learning where a robot can be trained by a human (for example assembly line manufacture, game playing, etc.). If we think of the human as an oracle to the optimal policy, then a small set of questions can be directly asked and answered. This requires an amount of work proportional to $O(mT^2|A|)$ where m is the number of examples per time step, $|A|$ is the number of actions and T is the number of time steps. The complexity is proportional to $O(T^2)$ because the human must guide the policy for $O(T)$ timesteps for each of the T prediction problems.

2. **Homing**. A "homing" policy is a policy which allows an inexact reset to an initial state (or initial state distribution). This mechanism has been recently analyzed for reinforcement learning (Even-Dar et al., 2005) with the result that it is possible to learn to act near optimally by analyzing the history of a process of intermittent homing, exploration and exploitation.

   The internal representation of this algorithm (which, intuitively, consists of probability tables over observations given belief states and actions) can be used to choose near-optimal actions given past history. Consequently, the method here could be used to find a more compact collection-of-classifiers representation for the policy.

3. **Resetting**. Many reinforcement learning algorithms are applied in a setting where a "reset to the start" action is available. This might naturally apply, for example, to a vacuuming robot that always starts from the same location and is easy to implement in simulators. The homing analysis applies here as well, and can be tightened by the removal of the need for homing.

4. **Generative Model**. A generative model answers questions of the form "Given the state s and action a, what is a draw for the next reward, observation and state?" This form of information is typically only available in the form of a simulator for the actual reinforcement learning problem. With this simulator, it is possible to behave according to a near optimal policy using a sparse sampling tree (Kearns et al., 1999) repeatedly. However, the computational complexity of this process is extreme, so solving the sequence of prediction problems discussed here may compile the information about what is the correct policy.

   Somewhat surprisingly, it is possible to prove that small error rate classification performance with respect to samples extracted from a (exponentially less expensive) trajectory tree (Kearns et al., 2000) implies a near optimal policy (Langford & Zadrozny, 2003). This approach works in stochastic environments where it is nevertheless possible to predict a near optimal action and fails (via an inherently large error rate) in other stochastic environments.

4. Discussion

A common complaint about learning theory in general is that it provides analysis for worst case scenarios. This can lead to unreasonably weak statements based on unreasonably strong assumptions. However, in the context of reinforcement learning, it is very difficult to justify the assumptions behind an average case analysis because we cannot easily assert that reinforcement learning problems are drawn from some fixed known probability distribution.

There is an alternative to the worst case/average case debate. We can do analysis subject to a new assumption. The assumption of prediction ability is a natural choice because we know simultaneously that in the worst case we cannot predict, yet for the cases that we care about we can often make successful predictions.

The result presented here can be thought about either narrowly or broadly. The narrow view is that we have made a quantitative connection between the ability of a classifier to perform well on certain problems and the performance of a policy. This statement provides intuition relating the minimum[3] difficulty of solving reinforcement learning to the difficulty of solving classification.

The broader view is that we have created a new form of analysis, even in the context of machine learning reductions. Other machine learning reductions relate the performance of one task to the performance of another by a mapping from one task to another inherent in the learning process (as discussed in Beygelzimer et al. (2004)). The essential difference here is that we can analyze and discuss the quantitative relationship between learning problems *without* explicit reference to a training set. This change is undesirable because the analysis does not lead directly to a reinforcement learning algorithm. The change may also be a necessity in reinforcement learning because of the sometimes unavoidable need to predict without prior representative experience as discussed in section 3.3.

References

Bagnell, D. (2004). Personal communication.

Bagnell, D., Kakade, S., Ng, A., & Schneider, J. (2004). Policy search by dynamic programming. *Advances in Neural Information Processing Systems 16 (NIPS*2003).*

Baird, L. C. (1993). *Advantage updating* (Technical Report). Wright Laboratory.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neurodynamic programming.* Athena Scientific.

Beygelzimer, A., Dani, V., Hayes, T., Langford, J., & Zadrozny, B. (2004). Reductions between classification tasks. Preprint at `http://hunch.net/~jl/projects/reductions/mc_to_c/mc_submission.ps`.

Evan-Dar, E., Kakade, S., & Mansour, Y. (2005). Reinforcement learning in POMDPs without resets. Preprint at `http://www.cis.upenn.edu/~skakade/papers/rl/learn_pomdp.ps`.

Fern, A., Yoon, S., & Givan, R. (2004). Approximate policy iteration with a policy language bias. *Advances in Neural Information Processing Systems 16 (NIPS*2003).*

Kakade, S., & Langford, J. (2002). Approximately optimal approximate reinforcement learning. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002)* (pp. 267–274).

Kearns, M., Ng, A., & Mansour, Y. (1999). A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-1999)* (pp. 1324–1331).

Kearns, M., Ng, A., & Mansour, Y. (2000). Approximate planning in large pomdps via reusable trajectories. *Advances in Neural Information Processing Systems 12 (NIPS*1999).*

Kearns, M., & Singh, S. (1998). Near optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998)* (pp. 260–268).

Khardon, R. (1999). Learning to take actions. *Machine Learning*, *35*, 57–90.

Lagoudakis, M., & Parr, R. (2003). Reinforcement learning as classification: Leveraging modern classifiers. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (pp. 424–431).

Langford, J., & Beygelzimer, A. (2005). Sensitive error correcting output codes. *Proceedings of the Eighteenth Annual Conference on Learning Theory (COLT-2005).* To appear.

Langford, J., & Zadrozny, B. (2003). Reducing T-step reinforcement learning to classification. Preprint at `http://hunch.net/~jl/projects/reductions/RL_to_class/colt_submission.ps%`.

Langford, J., & Zadrozny, B. (2005). Estimating class membership probabilities using classifier learners. *Proceedings of the Tenth International Workshop on AI and Statistics (AISTATS-2005)* (pp. 198–205).

Singh, S., James, M. R., & Rudary, M. R. (2004). Predictive state representations: A new theory for modeling dynamical systems. *Proceedings of the Twentieth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2004)* (pp. 512–519).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* MIT Press.

Watkins, C. (1989). *Learning from delayed rewards.* Doctoral dissertation, Cambridge University.

[3] It is only a minimum of course because we did not fully address how to gain the information necessary to learn these predictions. In other words we did not solve the temporal credit assignment problem.

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive classification by cost-proportionate example weighting. *Proceedings of the International Conference on Data Mining (ICDM-2003)* (pp. 435–442).

Appendix

A. Relationships between different reinforcement learning problem and goal definitions

The definition of reinforcement learning on a decision process (DP) used in this paper is mathematically elegant (both general and simple). It is important, however, to understand how this definition relates to the definitions that have been used in most previous reinforcement learning work. There are two standard definitions: the MDP and the POMDP definitions.

Definition A.1 *(MDP Reinforcement Learning) The reinforcement learning problem is defined by a set of states S, a set of actions A, an initial state distribution $p(s)$, and two conditional probability tables: $p(s'|s,a)$, giving the probability of state s' when action a is executed in state s, and $p(r|s')$ giving the reward distribution for each state s'.*

An MDP is a DP with the Markov assumption (the next state and reward distributions only depend on the current state and action). Thus, an MDP can be encoded by a DP through the mappings: $s \rightarrow o$, $p(o'|a,o), p(r|o') \rightarrow D(o', r|(o,a,r)^*, o, a) = D(o', r|(o,a))$. The last mapping is always feasible because D can express an arbitrary conditional distribution between r and o', as is necessary. Note that because a history can be empty, the initial state distribution is $p(o) = D(o', r)$.

Another commonly used definition is the POMDP definition.

Definition A.2 *(POMDP Reinforcement Learning) The reinforcement learning problem is defined by a set of states S, a set of actions A, a set of observations O, and two conditional probability tables: $p(o,r|s)$, giving the joint probability of observation o and reward r for each state s, and $p(s'|s,a)$, giving the probability of state s' when action a is executed in state s.*

An unconstrained POMDP and a DP are equally general. In particular, for any sequence of (observation, action, reward) triples, there exists a DP and a POMDP which predict the sequence with probability 1. This property means that POMDPs and DPs are "fully general" and assume no structure about the world. The advantage of DP is simplicity—there is no need to mention or use states.

For reinforcement learning goals, one common alternative is to consider the γ discounted sum of rewards.

Definition A.3 *(Reinforcement Learning Goal) For some $\gamma \in (0,1)$, find a policy $\pi : (o,a)^* \rightarrow a$ optimizing the expected sum of rewards:*

$$\eta_\gamma(D,\pi) = E_{(o,a,r)^* \sim \pi, D} \sum_{t=1}^{\infty} \gamma^t r_t$$

where r_t is the t-th observed reward. Here, the expectation is over the process which generates a history using D and choosing actions from π.

One reason why this definition is convenient is that there exists a stationary (rather than nonstationary) policy that is optimal with respect to this goal for any MDP.

A basic statement can be made (Bagnell, 2004) relating this goal to the T step goal used in this paper whenever rewards are bounded. In particular, for every reinforcement learning problem D, every γ, and every tolerance $\epsilon > 0$, there exists a hard horizon T and a modified reinforcement learning problem $D'(D)$ such that for all policies π we have $|\eta_\gamma(D,\pi) - \eta(D'(D),\pi)| \leq \epsilon$. (Here, we add the specification of the reinforcement learning problem to the performance measure.)

To do this, we first define[4] $T = \frac{1}{1-\gamma} \ln \frac{1-\gamma}{\epsilon}$. Then, we let $D'(D)$ be the reinforcement learning problem which for any valid history of D with probability γ uses D to draw the next observation and reward and with probability $1-\gamma$ draws the a special "end world" observation and 0 reward. Any history with an "end world" always results in an "end world" observation and 0 reward. The proof that this works is simple—the accumulated "end world" probability decreases the expected value of rewards at timestep t by γ^t and truncating the sum at T changes the expectation by at most ϵ.

A similar transformation from T-step reward to γ-discounted reward can be defined by choosing γ very near to 1, then altering the reinforcement learning problem so all rewards after T timesteps are 0.

Together these transformation imply that there is little mathematical difference between the expressiveness of these two performance goals. We prefer the T-step formulation because of the simplicity attained by eliminating γ and avoiding infinite sums.

[4] This is for the case where $r \in [0,1]$. The size of the reward interval rescales ϵ in general.