Reducing T-step Reinforcement Learning to Classification

John Langford
Mathematical Sciences Department
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
jcl@cs.cmu.edu

Bianca Zadrozny
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114
zadrozny@cs.ucsd.edu

Abstract

We discuss two reductions of reinforcement learning to classification: one in the presence of a generative model and the other in the presence of a (weaker) *trace* model. These reductions can be thought of as meta-algorithms for (approximately) solving reinforcement learning given a "strong learner" for classification.

Via the reductions, we provide new reinforcement learning algorithms, which apply to very large (and possibly infinite) state spaces. The structure of the reductions provides insight into the relationship between learning problems, and the trace model reduction opens up a new domain for reinforcement learning: *offline* reinforcement learning.

1 Introduction

Why should we be interested in reducing reinforcement learning to classification?

1. A reduction from reinforcement learning to classification implies new algorithms. The new algorithms are given by the composition of a classification algorithm with the reduction techniques outlined here. This automatic extension of the set of available techniques includes "support vector reinforcement learning", "RL with boosted decision trees", etc.
2. A reduction implies new theory. Suppose, for example, that we have some theorem bounding the true error rate of a learned classifier, c. Via the reduction, this also bounds the optimality of a learned policy.
3. Reductions are a method for understanding. Much of computational complexity theory and cryptology is a study of reductions. Knowing which problems reduce to other problems (and how efficiently they do so) provides understanding about the nature and difficulty of these problems.

The goal of reinforcement learning is finding a policy $\pi(a|s)$ which maps states to actions. Functionally, a policy is a (possibly probabilistic) mapping from states to actions. In the case of only two actions (which we limit consideration to here), the functional form of a policy $\pi : S \to A$ is exactly equivalent to a binary classifier $c : X \to \{0, 1\}$ where the "features" are the state space and the "class" (0 or 1) is the choice of action. Given this, it seems plausible that we can construct a policy π given a good learning algorithm for classifiers.

In this paper we turn that plausible conjecture into reality with a precise definition of a set of classification problems which can be used to construct a nonstationary policy π to optimize the T-step reward of reinforcement learning.

The study of the sample complexity of policy evaluation [6] is the inspiration of this paper. Implicit in naming "the VC dimension of a policy" is the notion of a reduction, which we make explicit here. The value of making the reduction explicit is a more exact understanding of the nature of the reinforcement learning problem and a new class of algorithms for solving these problems. A reduction from reinforcement learning to regression [4] in the presence of a "μ-reset" is similar.

As stated in the abstract, we present two reductions. The first reduction is from reinforcement learning with a generative model to classification. Good classification allows us to provide a *global*[1] guarantee on the optimality of the resulting policy. The drawback of this method is that it takes time exponential in the horizon, T.

The second method works on a much weaker *trace model* (defined later) and has a running time polynomial in T. The trace model is weak enough that the reduction can easily apply in *offline* settings, opening up new domains for reinforcement learning. The drawback of this method is that we can only state a local improvement guarantee.

2 Common Elements of Both Reductions

2.1 Definitions

An MDP is defined on a state set S and action set A. For every state s and action a, there is a probability $P(s'|a,s)$ over next states, s'. For every state s we enter, a reward $R(s) \in [0,1]$ is received and there is a distribution over initial states $P(s_0)$. Our goal in reinforcement learning on an MDP is choosing a (possibly nonstationary) mapping from states to actions which maximizes the expected T-step reward. The expected T-step reward is defined as:

$$\eta(\pi) = E_{s_0,s_1,\ldots,s_T \sim P(s_0),(P(s'|a,s),\pi(a|s))^T} \frac{1}{T}\sum_{i=1}^{T} R(s_i). \tag{2.1}$$

In other words, $\eta(\pi)$ is the average expected T-step reward given the MDP and policy.

The basic idea of the reductions is to use classification for action prediction with the following mapping $S = X$ (the input for classification is the state space) and $A = Y$ (the label space is the set of actions). Classification is optimized with respect to a distribution D on the space $X \times Y$ (or equivalently $S \times A$). What should D be?

2.2 From cost-sensitive classification to classification

One very important tool in both the reductions is a subreduction: from "cost-sensitive" (or "reward sensitive") classification to pure classification. In the cost sensitive setting, it is more important to classify some examples correctly. In particular, every example has an associated importance weight, w, and the goal is to minimize the expected importance of example mistakes:

$$E_{x,y,w\sim D} w I(c(x) \neq y) \tag{2.2}$$

here, $I(c(x) \neq y)$ is the indicator function which is 1 when the argument is true and 0 otherwise and the expectation is over draws of the triple, (x, y, w). cost-sensitive classification can model the fact that correct classification of some examples is more

[1] As opposed to local guarantees as provided by Policy Gradient techniques.

important than others. For example, a bank may be much more interested in the correct classification of large loans as "good" or "bad" rather than small loans.

To model the cost-sensitive setting, we assume there is a distribution D over (input, output, importance) triples (x, y, w). A basic folk theorem[2] from decision theory notes that if we have examples drawn from the altered distribution:

$$\hat{D}(x, y, w) \equiv \frac{w}{E_{x,y,w\sim D}[w]} D(x, y, w)$$

then optimal error rate classifiers for $\hat{D}$ are optimal loss minimizers for data drawn from D.

Minimizing the error rate on examples drawn according to $\hat{D}$ is slightly tricky[15], but feasible in a black-box manner for any classifier learning algorithm. With this tool in hand, we have half finished both the global and the local reduction.

3 From RL with a generative model to cost-sensitive classification

This first reduction works in the presence of a generative model. A generative model has two operations:

1. start_state() returns a state drawn from $P(s_0)$.
2. next_state(s, a) returns a state s and a reward $R(s)$ drawn from $P(s'|a, s)$ for an action a and state s.

A classification problem is defined by a range X and discrete output domain Y. Examples are drawn from a distribution D over $X \times Y$. The goal in classification is finding a function $c : X \to Y$ which minimizes errors under the distribution D. The error rate is defined as: $e_D(c) = \Pr_D(c(x) \neq y)$. In order to simplify the paper, we assume that $|A| = |Y| = 2$.

There are fundamental difficulties introduced in attempting to reduce T-step reinforcement learning to a single instance of classification, simply because the optimal T-step policy might be nonstationary. These difficulties are avoided when we reduce to T instances of classification, one for each timestep. These T learned classifiers are denoted c_i for $i \in \{1, ..., T\}$ and the *nonstationary* policy they create is $\pi(a|s, t) = c_t(s)$.

What are the distributions $D_1, ..., D_T$ that these T classifiers must work well on? The distribution on states at timestep i is dependent on the classifiers for previous timesteps. The rewards achieved based upon an action choice are dependent upon the classifiers for later timesteps. Thus, the "right" distribution is inherently circular—each learning problem influences the other learning problems. Somewhat surprisingly, it is possible to break this circularity in a manner that allows provable guarantees on global optimality.

We start (as in [6]) with the notion of a trajectory tree. At depth T, this requires $|A|^T = 2^T$ calls to the generative model per tree. A tree is constructed by first calling start_state(), and then using next_state(s, a) to recursively construct a binary tree of depth T. Every tree produces one sample for each of the T possible classifiers. This one sample is found by computing the empirical value function $\hat{Q}_t(s, a)$ on that tree. The empirical value function is defined recursively as: $\hat{Q}_t(s, a) = \frac{R(s')}{T} + \max_a \hat{Q}_{t+1}(s', a)$ where s' is the state reached in the tree after taking action a in state s. The recursion terminates at $\hat{Q}_T(s, a) = 0$. An example of this calculation is shown in figure 3.1.

[2] We say "folk theorem" here because the result appears to be well-known by some. It is a simplification of many results from decision theory.

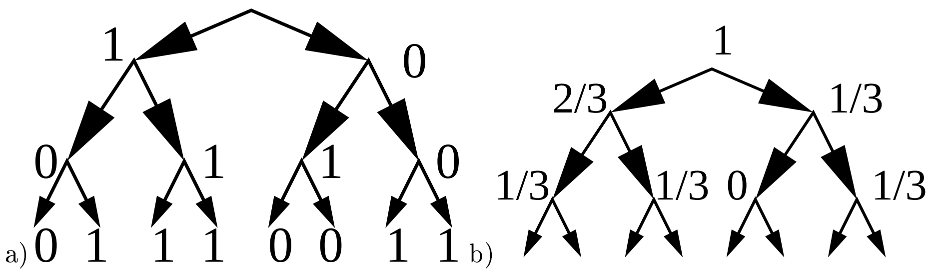

Figure 3.1: a) a possible trajectory tree annotated with the values of the states. b) the empirical values $\max_a \hat{Q}(s,a)$ of these states.

To extract the examples, consider a path through the tree which always takes a maximal value action. If s is the tth state on the path and a is a maximal action, then the (x,y,w) triple is $(s,a,\hat{Q}(s,a))$ for the tth classification problem. Note that our choice of importance is equivalent to implicitly assuming that the reward achieved after misclassification is 0, as it might be in the worst case. This pessimal assumption appears to be the only consistent choice, and violation of this assumption simply improves the theoretical guarantees we make later.

This set of T classification problems is a reduction of reinforcement learning to cost-sensitive classification. Coupled with the reduction of cost-sensitive classification to misclassification, we have a complete reduction from T-step reinforcement learning to error minimizing classification. The algorithm for solving T-step reinforcement learning is then very simple: For every classification problem, t, learn a classifier c_t and output the policy, $\pi(a|s,t) = c_t(a)$.

3.1 Analysis

In order to analyze the performance of this reduction, we make the strong learning assumption. The strong learning assumption states that we can approximately learn the concept. This hides a great deal of complexity, of course, but it allows us to focus on the efficiency of the reduction rather than on the classification problem. It is worth noting that various boosting algorithms can take any weak learning algorithm with error rate $\frac{1}{2} - \epsilon$ and boost it into a strong learning algorithm with error rate ϵ.

Definition 3.1 *(Strong Learning) A strong learning algorithm A, given sufficient samples S drawn i.i.d. from a distribution D produces a classifier c with small true error rate, $e(c) = \Pr_D(c(x) \neq y) \leq \epsilon$.*

An agnostic version of the strong learning assumption (with larger error rates) is usable instead. We use a "realizable" version here for clarity rather than necessity. Given the strong learning assumption we can state a performance guarantee.

Theorem 3.2 *(Learning Translation) An ϵ-strong classifier learning algorithm can be used to construct a policy π satisfying:*

$$\eta(\pi^*) - \eta(\pi) \leq \frac{(T+1)}{2}\epsilon$$

Proof

We analyze the construction outlined above. For learning problem t, let $N_t \leq \frac{T-t+1}{T}$ be the expected value of the importance weights, $N_t = E_{x,y,w \sim D_t} w$ in the

tth classification problem. The reduction from cost-sensitive to cost-insensitive classification implies that the optimal classifier has reward at least $N_t(1-\epsilon)$ if all other classifiers are perfect.

A policy is non-optimal if it takes some non-optimal action at some timestep. The expected loss due to errors in the first decision is bounded by $N_1\epsilon$. The expected loss due to errors in the second decision is $N_2\epsilon$, and so forth. The total expected loss in reward is at most:

$$\sum_{t=1}^{T} N_t\epsilon \leq \sum_{t=1}^{T} \frac{T-t+1}{T}\epsilon = \frac{T+1}{2}\epsilon$$

implying that the expected reward is at least $N_1 - \frac{T+1}{2}\epsilon$.

If π^* is the optimal policy and $\eta(\pi^*)$ is the reward of the optimal policy, then $N_1 \geq \eta(\pi^*)$ since the expected value of the path with largest value is greater than the value of the policy. □

The strong learning assumption hides issues of sample complexity. Roughly speaking, the necessary sample complexity is $2^T m$ where m is the number of samples required to realize the strong learning assumption for all of the classifiers.

A careful examination of the proof reveals that we are using the generalization in the classification learning algorithm to combat the optimism inherent in considering the path with maximal value. In contrast, we might have taken many samples from each state-action in the tree to achieve a better estimate of the expectation, as in [7]. The approach here is, of course, more tractable because each tree has size $2^T m$ rather than $(2m)^T$.

4 From RL with traces to cost-sensitive classification

This second reduction works in the presence of a trace model, which has the following operation:

- trace(π): returns a trace from the execution of a (probabilistic) policy π on the MDP, that is, a sequence of state-action-reward triples: $(\langle s_0, a_0, r_1\rangle, \langle s_1, a_1, r_2\rangle, \ldots, \langle s_{T-1}, a_{T-1}, r_T\rangle)$, where s_0 is drawn from $P(s_0)$, each a_i is drawn from $\pi(a|s_i)$, each s_i (for $0 < t < T$) is drawn from $P(s'|s_{i-1}, a_{i-1})$ and $r_i = R(s_i)$.

This model is weaker than the generative model since we can simulate the trace operation using the initial_state and next_state operations but the opposite is not true. The model is motivated by (for example) problems where a large company interacts with many individual (hopefully independent) people according to some policy. The company can change the policy, but it can never "reset" people and see what alternative outcomes might have arisen from an interaction.

Again we are going to reduce the problem of finding a good policy to T instances of classification, one for each timestep. The T learned classifiers are denoted c_t for $t \in \{1, ..., T\}$ and the nonstationary policy they create is $\pi(a|s,t) = c_t(s)$.

The optimization starts with some arbitrary initial policy, $c_1, ..., c_t$ and new classifiers c_t are learned iteratively. We start with $t = 1, 2, ...T, 1, 2, ..., T, 1, 2, ...$ For each optimization problem, we learn a new classifier c'_t, and then compare the two policies, $c_1, ..., c'_t, ...c_T$ and $c_1, ..., c_t, ..., c_T$ using Monte Carlo integration, as provided by the trace model. If the new policy has a provably larger expected value, we replace $c_t \leftarrow c'_t$, then move on to the next optimization problem. The process halts after we observe T nonupdates in a row.

The reduction for timestep t works by first calling the operation trace(π') m times, where the policy $\pi'(a|s,i)$ is given by $c_i(s)$ for $i < t$ and $i > t$. At timestep t,

$\pi'(a|s,i)$ is an arbitrary distribution. Each returned trace is made into an example of the form (x,y,w), where $x=s_t$, $y=a_t$ and $w=\frac{1}{T}\sum_{t=i+1}^{T} r_t$.

Let

$$W(s_t,a_t)=\frac{1}{T}\sum_{i=t+1}^{T} R(s_i)$$

where $s_{t+1}\sim P(s'|s_t,a_t)$ and $s_{i+1}\sim P(s'|s_i,\pi'(s_i))$ for $t<i\leq T$.

Then $E[W(x,c_t(x))]$ is the expected value of the reward obtained by following $c_t(x)$ in step t while following π' in all the subsequent steps.

The following theorem shows that this expected value can be rewritten in a way that allows us to use a classifier learner that maximizes (2.2) to learn the tth step classifier c_t that maximizes $E[W(x,c_t(x))]$ using the examples (x,y,w).

Theorem 4.1 *Let X be an arbitrary state space and $Y=\{0,1\}$ be a discrete action space. Assume we have examples (x,y) drawn from a distribution D over $X\times Y$. Let $W: X\times Y\to[0,1]$ be a stochastic function and let $c_t: X\to Y$ be a deterministic function. Assume that $\forall x,y:\ P(y|x)>0$. Then:*

$$E_D[W(x,c_t(x))]=E_D\left[\frac{W(x,y)}{P(y|x)}I(c_t(x)=y)\right]$$

Proof

$$\begin{aligned}
E_D\left[\tfrac{W(x,y)}{P(y|x)}I(c_t(x)=y)\right] &= E_D\left[\tfrac{W(x,y)}{P(y|x)}\,\middle|\, c_t(x)=y\right]P(c_t(x)=y)\\
&= E_D\left[\tfrac{W(x,c_t(x))}{P(c_t(x)=y|x)}\,\middle|\, c_t(x)=y\right]P(c_t(x)=y)\\
&= \textstyle\int_X \tfrac{W(x,c_t(x))}{P(c_t(x)=y|x)}P(x|c_t(x)=y)P(c_t(x)=y)dx\\
&= \textstyle\int_X \tfrac{W(x,c_t(x))}{P(c_t(x)=y|x)}P(x,c_t(x)=y)dx\\
&= \textstyle\int_X W(x,c_t(x))P(x)dx\\
&= E_D[W(x,c_t(x)]
\end{aligned}$$

In the first step of the proof, we use the fact $E[a\ I(b=c)]=E[a|b=c]P(b=c)$, which is easy to show. In the second step, we know that $c_t(x)=y$ because of the conditioning event, so we make substitutions accordingly. In the third step, we use the definition of conditional expectation. In the fourth step, we use Bayes' rule. The following steps are trivial. $\Box$

For each example (x,y,w), we have that $w=W(x,y)$. Thus, it follows from this theorem that

$$\frac{1}{m}\sum_{(x,y,w)}\frac{w}{P(y|x)}I(c_t(x)=y),$$

is an unbiased empirical estimate of $E[W(x,c_t(x))]$. Since we know $P(y|x)$, the probability that action y is executed in state x by the policy at step t, we can reweight each example (x,y,w) by $\frac{w}{P(y|x)}$ so that the learned classifier implicity minimizes equation 2.2 or, equivalently, maximizes

$$E_{x,y,w\sim D}wI(c(x)=y)$$

One crucial idea here is taking into account the probability that the arbitrary policy executes an action in a state to estimate the expected value of a different policy. This is inspired by approaches for solving sample selection bias and missing data problems in econometrics and statistics, which take into account the probability that an example is missing [5, 9] to estimate unbiased parameters for a model.

Note that the theorem holds only if $\forall x, y P(y|x) > 0$, that is, we require that the arbitrary policy chosen for timestep t have non-zero probability of executing each action in each state. However, the reduction degrades gracefully even when this is not the case if we define that $\frac{I(c_t(x)=y)}{\pi'(y|x)} = 0$ when $I(c_t(x) = y) = 0$ and $\pi'(y|x) = 0$. In this case, it is easy to see that the reduction will make the (approximately) optimal local improvement for an MDP that is identical to the original one, except that action y is not allowed in state x.

The policy obtained from this reduction is the (approximately) best possible local improvement over the arbitrary policy used for training. "Local" here refers to the fact that we optimize the classifier c_t for each step t in a greedy manner, assuming that the policy will remain the same for the other steps. However, we continually change the policy to improve its overall value. The only guarantee is that each classifier c_t added will increase the overall policy value.

Even though this is a local reduction it is particularly interesting because it opens up a new area for research in reinforcement learning: offline reinforcement learning, where we have only traces given to us acting under a known (stochastic) policy. This set of traces can be used to simulate the trace model by rejection sampling to produce a subset of traces consist with the policy π given to the trace model.

The offline setting is the most realistic for domains in which arbitrary interaction with the MDP is not feasible and a generative model is not known, but for which we have traces collected offline. Attempts to apply traditional reinforcement learning methods in some of these domains, such as direct marketing and recommender systems, have been made [12, 1] but the difficulty of direct interaction with the environment makes it hard to evaluate their success using current methods.

5 Discussion

5.1 Generalizations and Improved Analysis

There are several ways this work can be generalized. In general, we chose the simplest possible mathematical form rather than the most general form.

Application to POMDPs. The results are fully extensible to the POMDP setting. In the POMDP setting, instead of using the ith state s, you would use the sequence of observations, $o_1, ..., o_i$ as the input to your predictor.

A weak learning assumption. Various boosting algorithms ([14],[10]) can use the weak learning assumption (ability to predict correctly with probability $\frac{1}{2} - \epsilon$) to create a strong learning algorithm.

Much of reinforcement learning is analyzed with a discount factor γ rather than a hard horizon T. The analysis here could be applied to the discount factor setting instead by noting that any discount factor γ implies a horizon of $T = \frac{C}{1-\gamma}$ since truncation after $\frac{C}{1-\gamma}$ results in negligible loss of value for $C = 10$.

The $|A|^T$ scaling of the global-optimal approach in section 3 is not viable beyond $T = 20$ or so. The quadratic (in T) sample complexity makes the local approach in section 4 more attractive and might optimistically scale to $T = 10^4$. Like many situations with a local tractable solution and an intractable global solution, a hybrid technique which varies between these two approaches is likely to be useful.

In practical application on real-world reinforcement learning problems, individual classifiers probably do not differ much between timestep 100 and timestep 101. Taking advantage of this fact via multitask learning[2] or learning to learn[13] techniques may result in much reduced sample complexities.

5.2 Implications

The structure of the reductions suggests new directions of research. What is a good multitask learning algorithm when the target concept drifts? Can lifelong learning techniques be applied?

A reduction implies that theory about classification also applies to reinforcement learning. This was partially covered in [6], but there are more implications.

1. The PAC-Bayes bound [11] implies that stochastic policies have a *smaller* sample complexity than might otherwise be expected. In fact, application of PAC-Bayes bounds to policies is somewhat more natural than for classification because the use of stochastic policies has (historically) been more common than the use of stochastic classifiers. Noting that stochastic policies have a lower-than-expected sample complexity for learning is a direct improvement on the analysis of [6].
2. Shell bounds [8] could be applied to policy evaluation and might result in a much smaller sample complexity when many of the possible classifier chains/policies are "obviously bad".

There are other implications of course — the magic of a reduction is that it applies to *future* work on classification.

Many learning problems have now been reduced to binary classification. These include ranking, regression, reinforcement learning, multiclass classification, and cost-sensitive classification. Many learning problems such as clustering and unsupervised learning have also *not* been reduced to classification. Can we classify all learning problems into equivalence classes based upon the families of reductions they fall into?

References

[1] Guy Shani, Ronen Brafman and David Heckerman, "An MDP-Based Recommender System", UAI'02.

[2] Rich Caruana, "Multitask Learning", Thesis, Carnegie Mellon University, 1997.

[3] Sham Kakade, "On the Sample Complexity of Reinforcement Learning", Thesis, University College London, 2003.

[4] Sham Kakade and John Langford, "Approximately Optimal Approximate Reinforcement Learning", ICML 2002.

[5] James Heckman, "Sample selection bias as a specification error", Econometrica, 1977.

[6] Michael Kearns, Andrew Ng, Yishay Mansour, "Approximate Planning in Large POMDPs via Reusable Trajectories" Advances in Neural Information Processing Systems 12, MIT Press, 2000.

[7] Michael Kearns, Andrew Ng, Yishay Mansour, "A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes" Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence Morgan Kaufmann, 1999, pages 1324–1331.

[8] John Langford and David McAllester, "Computable Shell Decomposition Bounds" COLT 2000.

[9] Roderick Little and Donald Rubin, "Statistical Analysis with Missing Data", Chapter 3, Wiley, 2002.

[10] Yishay Mansour and David McAllester, "Boosting using branching programs", COLT2000.

[11] David McAllester, "PAC-Bayesian Model Averaging", COLT 1999.

[12] Edwin Pednault, Naoki Abe, Bianca Zadrozny and others, "Sequential cost-sensitive decision-making with reinforcement learning", KDD'02.

[13] Joseph O'Sullivan, "Transferring Learned Knowledge in a Lifelong Learning Mobile Robot Agent" In European Workshop on Learning Robots, 1998.

[14] Robert Schapire, "The Boosting Approach to Machine Learning", MSRI Workshop on Nonlinear Estimation and Classification, 2002.

[15] Bianca Zadrozny, John Langford, and Naoki Abe, "A Simple Method for Cost-Sensitive Learning", IBM Technical Report, 2003.