# “PURE” REINFORCEMENT LEARNING WITH CLASSIFICATION

JOHN LANGFORD

## 1. Introduction

Problem This paper addresses the purest reinforcement learning problem, which intuitively is “choose actions in the world so as to optimize rewards given observations”. We make *no* assumptions about the world other than the ability to approximately solve a certain collection of classification problems. In particular, we rely on no ability to “reset” to a start state, no generative model to generate outcomes given states and actions [3], and no “mixing time” [1]. A “homing policy” [2] is (in some sense) not an assumption, but we do not need it in order to state the basic result (it may be useful in the process of acquiring good classifications).

Solution Motivation The method of reduction in learning allows us to ask questions of the sort “How does the ability to (approximately) solve simple prediction problems allow us to (approximately) solve complex prediction problems?” Aside from the theoretical motivations of this question, this style of analysis often results in practical new algorithms (such as [4], [5]) for solving learning problems in diverse domains. Here, we show that, for the right definition of “learning”, *all* learning problems can be (approximately) solved by (approximately) solving certain classification problems. Here the “all” is due to the ability to embed a very large set of learning problems into reinforcement learning.

Caveats The method we present is the first connection of this sort, but it may not be the last because the reduction is “loose”—small error rates must be achieved in order to guarantee that the created policy is anywhere near optimal. Consequently, very significant tightenings of these results may be possible. It is also important to understand that the algorithm presented here is quite computationally expensive, so again room for improvement exists.

## 2. Solution Sketch

### 2.1. Definitions.
It is possible to think of the general reinforcement learning problem as learning in a POMDP (Partially Observable Markov Decision Process) which is defined by a set of states, actions, observations, and rewards together with conditional probability tables on these sets. However, this is unnecessarily complicated because we need never use (or explicitly consider) states. Instead, consider the following definition:

**Definition 2.1.** (Reinforcement Learning Problem) A reinforcement learning problem $\mathrm{RL} = \{A, O, D\}$ is defined by a set of actions $A$, a set of observations $O$, and conditional probability tables $D(o', r|(a, o, r)^*)$ where $(a, o, r)^*$ is any history of past actions, observations, and rewards. Here $r \in [0, 1]$ is some reward.

There are a variety of goals considered in reinforcement learning. We consider the following one:

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

$$\max_{\pi} E_{(a,o,r)^T \sim \pi, D} \sum_{t=1}^{T} r_t$$

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 $\pi$.

This definition is essentially equivalent to several other definitions. For example, the commonly used $\gamma$ discounting definition which optimizes the $\gamma$-discounted sum of rewards can be simulated by modifying the reinforcement learning problem so that at each timestep with probability $1-\gamma$, a "termination" observation is made and no future rewards are observed. The truncation of the infinite sum at $T$ introduces an error which decreases exponentially in $T(1-\gamma)$.

2.2. **The Algorithm.** The basic idea of reductions in learning is to reduce more complicated learning problems to simpler ones. One way to think about classification problems in particular is in the form of a decision question with a yes/no answer (which is sometimes answered incorrectly). For example "If I take action $a_1$ is the reward greater than 0.5?" Not all binary questions are allowed—only those for which we can derive a coherent classification problem (and hence talk about classificaton error rates) make sense.

The set of questions used here have the form "What is the expected reward of doing action $a_t$ after some prefix $a_1, ..., a_{t-1}$?" This is a reduction to a form of regression rather than classification. However, we can further reduce the regression problem to binary classification using a simple reduction from cost estimation (which is another term for regression) to classification [6]. The method chooses the action $a_t$ at timestep $t$ leading to the largest sum of expected future rewards.

## 3. Analysis

Before stating the analysis, we need to make a basic reductions-related sanity check. Any reinforcement learning problem $D$ and prefix of actions $a_1, ...., a_t$ creates a measure $D(o, r|a_1, ..., a_t) = E_{o_1,r_1,...,o_t,r_t \sim a_1,...,a_t,D} D(o, r|(a, o, r)^t)$. Consequently, there exists a well defined distribution over immediate rewards given any prefix of actions. This implies that the regression problems defined each correspond to an induced measure $D_{a_1,...,a_t}$ over immediate rewards.

A basic theorem can be proved about this algorithm. For this theorem, let the performance of any policy $\pi$be defined by:

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

**Lemma 3.1.** *(Reduction Statement) Let $\pi_T$ be the policy created by the regressors above. Let the predictions (given any past observations) be given by $\hat{r}_{a_1,...,a_t}$. If, for all action prefixes $a_1, ..., a_t$ with $t \in \{1, ..., T\}$:*

$$E_{(o,r) \sim D_{a_1,...,a_t}} |r - \hat{r}_{a_1,...,a_t}| \leq \epsilon$$

*then*

$$\eta(\pi^*) - \eta(\pi_T) \leq f(\epsilon)$$

*Proof.* (very sketchy) First note that if the predictions are perfect, then $\eta(\pi^*) = \eta(\pi_T)$. Then, note that in the worst case (with adversarial regressors constrained by the predictability assumption), the sum of rewards for any $T$-length prefix is altered by at most $\epsilon T$. (details still to be filled in.) □

## References

[1] Michael Kearns and Satinder Singh, Near-Optimal Reinforcement Learning in Polynomial Time. Proceedings of the 15th International Conference on Machine Learning, pp. 260-268, 1998, Morgan Kaufmann.

[2] Sham Kakade (and others?) "Reinforcement Learning in POMDPs Without Resets" http://www.cis.upenn.edu/~skakade/papers/rl/learn_pomdp.ps

[3] Michael Kearns, Yishay Mansour, and Andrew Ng, "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.

[4] example

[5] algorithms

[6] John Langford and Alina Beygelzimer, "Sensitive Error Correcting Output Codes", http://hunch.net/~jl/projects/reductions/pecoc/v7/pecoc.ps