

Filtered Reinforcement Learning

Douglas Aberdeen

National ICT Australia, Canberra, Australia
douglas.aberdeen@nicta.com.au,
WWW home page: <http://csl.anu.edu.au/~daa/>

Abstract. Reinforcement learning (RL) algorithms attempt to assign the credit for rewards to the actions that contributed to the reward. Thus far, credit assignment has been done in one of two ways: uniformly, or using a discounting model that assigns exponentially more credit to recent actions. This paper demonstrates an alternative approach to temporal credit assignment, taking advantage of exact or approximate prior information about correct credit assignment. Infinite impulse response (IIR) filters are used to model credit assignment information. IIR filters generalise exponentially discounting eligibility traces to arbitrary credit assignment models. This approach can be applied to *any* RL algorithm that employs an eligibility trace. The use of IIR credit assignment filters is explored using both the GPOMDP policy-gradient algorithm and the Sarsa(λ) temporal-difference algorithm. A drop in bias and variance of value or gradient estimates is demonstrated, resulting in faster convergence to better policies.

1 Introduction

A reinforcement learning (RL) agent performs actions in a world according to a parameterised policy. The agent seeks to adjust the policy in order to maximise a long-term measure of reward. A core difficulty is correctly distinguishing which actions caused high long-term pay offs. This *temporal credit assignment* problem is usually solved in one of two ways: (1) if the task ends in finite time then credit can sometimes be assigned uniformly; (2) if the task has an infinite horizon a discounted model assigns exponentially more credit to recent actions.

This paper shows how to use exact or approximate prior information about reward delays to build a tailored credit assignment model. Taking advantage of such information leads to a reduction in the bias and variance of estimates; either value estimates or gradient estimates depending on the algorithm employed. We demonstrate that the drop in bias and variance can be orders of magnitude given good prior information about the correct way to assign credit to actions.

For example, the action of adding chemicals to a bio-reactor, or the injection of a drug into a patient, will not have an immediate impact. Rather, the response of the system to the input ramps up with time to a maximum, then decays slowly. In this case exponential credit assignment is inappropriate because the majority of the credit goes to the most recent input (or action), rather than the input that is currently dominating the system response.

This paper introduces the use of infinite impulse response (IIR) filters to shape the temporal credit assignment. The IIR filter generalises *eligibility traces* to non-exponential models of credit assignment. IIR models of credit assignment can be applied to any RL algorithm that employs an eligibility trace. This paper uses the GPOMDP policy-gradient algorithm [1], and the Sarsa(λ) algorithm [2].

This rest of this paper is organised as follows. Section 2 describes the algorithms modified in this paper: GPOMDP and Sarsa(λ). Section 3 describes how these algorithms can be extended to arbitrary temporal credit assignment. Section 4 presents experiments that demonstrate the bias and variance improvements that can be gained using IIR filters. Section 5 presents a less trivial example to motivate the use of trace filtering in real world problems.

2 Background

Markov decision processes (MDPs) are a natural framework for RL. Specifically, the GPOMDP and Sarsa(λ) algorithms are described. Both employ exponential credit assignment models that will be generalised to IIR assignment models. The algorithms are described in the fully observable MDP framework for simplicity, but GPOMDP extends to partially observable environments without loss of *local* convergence guarantees [1]. Some experiments in this paper are only partially observable, where an observation replaces the exact state.

2.1 MDPs

A MDP consists of states $\mathcal{S} = \{1, \dots, |\mathcal{S}|\}$ of the world, actions $\mathcal{A} = \{1, \dots, |\mathcal{A}|\}$ available to the agent in each state,¹ and a (possibly stochastic) reward $r(i)$ for each state $i \in \mathcal{S}$.

Each action $a \in \mathcal{A}$ determines a stochastic matrix $P(a) = [\Pr(j|i, a)]$ where $\Pr(j|i, a)$ denotes the probability of making a transition from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ given action $a \in \mathcal{A}$. The GPOMDP and Sarsa(λ) algorithms are *model-free* methods that do not assume explicit knowledge of $P(a)$, however, such knowledge might contribute to the temporal credit assignment modelling.

All policies are stochastic. The probability of choosing action a given state i , and parameters $\theta \in \mathbb{R}^{|\mathcal{A}|}$ is $\Pr(a|\theta, i)$. The evolution of the world state i is Markov, with an $|\mathcal{S}| \times |\mathcal{S}|$ transition probability matrix $P(\theta) = [\Pr(j|\theta, i)]$ whose entries indexed by i, j are given by

$$\Pr(j|\theta, i) = \sum_{a \in \mathcal{A}} \Pr(a|\theta, i) \Pr(j|i, a). \quad (1)$$

In this paper, the *soft-max* function is used to generate $\Pr(a|\theta, i)$ from the real valued output of a parameterised function. Given a vector $y \in \mathbb{R}^{|\mathcal{A}|}$, the

¹ Generally, state spaces may be uncountably infinite. However, the maths becomes more complex without altering the final algorithms.

probability of action $a \in \{1, \dots, \mathcal{A}\}$ according to the soft-max distribution generated by the vector y is

$$\Pr(a|y) := \frac{\exp(y_a)}{\sum_{m=1}^{|\mathcal{A}|} \exp(y_m)}. \quad (2)$$

The vector y represents the real valued output of a function representing action likelihoods for policy-gradient algorithms, or state/action values for value algorithms. All but the final experiment use parameters $\theta_{a,i}$ representing each likelihood, or value, of action a given state i . I.e., a table lookup representation. Thus, given state i , we have $y = [\theta_{1,i}, \dots, \theta_{|\mathcal{A}|,i}]$.

2.2 GPOMDP

GPOMDP is an infinite-horizon policy-gradient algorithm [1]. It computes the gradient of the *long-term average reward*

$$\eta(\theta) := \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}_\theta \left[\sum_{t=1}^T r_t \right], \quad (3)$$

with respect the policy parameters θ . The expectation \mathbb{E}_θ is over the distribution of states $\{i_0, i_1, \dots\}$ induced by $P(\theta)$. Updating the policy parameters in the direction of the gradient locally maximises the average reward. Under a standard ergodicity assumption $\eta(\theta)$ is independent of the starting state and is equal to

$$\eta(\theta) = \sum_{i=1}^{|\mathcal{S}|} \pi_i(\theta) r(i) = \pi(\theta)^\top r, \quad (4)$$

where $\pi(\theta)$ is the stationary distribution of states induced by the current θ , and $r := [r(1), \dots, r(|\mathcal{S}|)]^\top$.

Theorem 1 (From [1]). *Let I be the identity matrix and let e be a column vector of 1's. Further, drop the explicit dependence of η , P , and π on θ . The true gradient of the long-term average reward with respect to θ is*

$$\nabla \eta = \pi^\top (\nabla P) [I - P + e\pi^\top]^{-1} r. \quad (5)$$

This theorem, proved in [1], establishes the true gradient of the long-term average reward with respect to the policy parameters, but requires known $\Pr(j|i, a)$ and also requires inverting a potentially large matrix.

The GPOMDP algorithm computes a Monte-Carlo approximation of (5). The agent interacts with the environment, producing a state, action, reward sequence $\{i_1, a_1, r_1, \dots, i_T, a_T, r_T\}$. Under mild technical assumptions, including ergodicity and bounding all the terms involved, the approximation is:

$$\widehat{\nabla} \eta = \frac{1}{T} \sum_{t=0}^{T-1} \nabla \log \Pr(a_t | \theta, i_t) \sum_{s=t+1}^T \beta^{s-t-1} r_s, \quad (6)$$

where the discount factor $\beta \in [0, 1)$ has been introduced to control the variance of the gradient estimate. The discount factor implicitly assumes that rewards are exponentially more likely to be due to recent actions. Without such an assumption, rewards must be assigned over a potentially infinite horizon, resulting in estimates with infinite variance. As β decreases, the variance decreases, but the bias of the gradient estimate increases [1]. In practice, (6) is implemented efficiently using an exponentially discounted eligibility trace, described in Section 2.4. The algorithmic form of GPOMDP is described in Section 3.1.

2.3 Sarsa(λ)

Sarsa(λ) [2] is introduced only briefly to highlight the general applicability of IIR credit assignment. Sarsa(λ) estimates

$$Q(i, a, \theta) := \lim_{T \rightarrow \infty} \mathbb{E}_\theta \left[\sum_{t=1}^T \delta^t r_t | i_0 = i, a_0 = a \right],$$

which is the expected discounted sum of rewards over time. Policies in this paper are derived from $Q(i, a, \theta)$ using the soft-max function (2), preferring actions with high value. Updates to θ are driven by temporal differences: the error between the current value estimate and the actual return

$$\begin{aligned} d(i_t, i_{t+1}) &= [r_t + \delta Q(i_{t+1}, a_{t+1}, \theta_t)] - Q(i_t, a_t, \theta_t), \\ \theta_{t+1} &= \theta_t + \alpha_t d(i_t, i_{t+1}) e_t. \end{aligned}$$

The discount factor δ determines the importance of long-term rewards compared to instant rewards, α_t is a possibly variable step size, and e_t is the current eligibility trace that stores how eligible each parameter is for an update in the direction of $d(i_t, i_{t+1})$. Temporal errors can also be computed over multiple time steps, resulting in n -step temporal differences $d(i_t, i_{t+n})$. The Sarsa(λ) algorithm implicitly calculates all such n -step temporal differences to the end of the episode via the use of the eligibility trace. Sarsa(λ) places a weight of λ^{n-1} on the n -step temporal error when updating parameters [2]. Thus, increasing the λ parameter places more weight on temporal differences computed over many steps. As $\lambda \rightarrow 1$ we approach a Monte Carlo method that uses the actual sum of returns received from the target state to the terminal state, providing low bias, but high variance. Reducing λ reduces variance at the cost of increased bias [3]. If domain knowledge such as “the rewards received for the first τ steps after actions, are random with mean 0” is available, the exponential weighting model λ^{n-1} is not appropriate. The n -step temporal difference weight should be 0 for the first τ steps, then decay exponentially with $\lambda^{n-\tau}$. For Sarsa(λ), this paper generalises eligibility traces to allow *arbitrary* weights for each n -step temporal difference. Computing the eligibility trace e_t is described in the next section. The algorithmic form of Sarsa(λ) is described in Section 3.2.

2.4 Computing Eligibility Traces

A standard mechanism for implementing temporal credit assignment is the *eligibility trace*. The eligibility trace e is a vector of length $|\theta|$, one element for each parameter $\theta_p \in \theta$. The trace stores how eligible each state/action pair is for receiving the credit for a reward or temporal difference. If function approximation is used the trace stores how eligible each parameter is for being updated.

Eligibility traces work in a similar way for most RL algorithms. After receiving a state i_t and choosing an action a_t , the gradient of the parameterised policy, or Q -function, is computed for action a_t , and added to the trace. At each step the trace is also multiplied by the trace discount. The GPOMDP trace update is

$$e_{t+1} = \beta e_t + \nabla \log \Pr(a_t | \theta, i_t), \quad (7)$$

where $\nabla \log \Pr(a_t | \theta, i_t)$ is the log derivative of the soft-max function (2) for the chosen action. The Sarsa(λ) eligibility trace update is

$$e_{t+1} = \delta \lambda e_t + \nabla Q(i_{t+1}, a_{t+1}, \theta_{t+1}). \quad (8)$$

The use of λ in this equation implements the weighting by λ^{n-1} of the n -step return (see [2] for details). The additional factor of δ in the eligibility trace update is needed to maintain consistency with the discount factor selected for the domain. The value discount factor δ — which changes the quantity being estimated — should not be confused with the GPOMDP discount factor of β , which is introduced to control the variance of estimates of $\nabla \eta$.²

The parameterisation of $Q(i, a, \theta)$ in this paper is a lookup table, with one parameter for each combination of state and action. The gradient is 1 for the parameter indexed by the current state and action, and 0 otherwise.

Eligibility traces are used in many RL algorithms including TD(λ), Sarsa(λ), Williams' REINFORCE [4], and GPOMDP. See [2] for a good introduction to eligibility traces.

3 Filtering Traces

Equations (7) and (8) show the usual method of updating the eligibility trace for GPOMDP and Sarsa(λ). Implicit in this update is the idea of assigning exponentially more credit to recent actions. Figure 1 shows how an instantaneous reward of 1 at time t , would be credited to the action chosen τ steps ago with a discount factor of 0.9. Control theorists would view Figure 1 the impulse response of a first order infinite impulse response (IIR) filter.

In this section first order eligibility traces are extended to arbitrary IIR filters, allowing domain specific knowledge about credit assignment to be used. The aim is to subtract a zero-mean, non-zero variance process from the gradient or value estimates, effectively reducing the variance of these estimates. This paper extends the author's early trace filtering work [5] to value-methods, and

² However, the two quantities are related to each other [1].

provides empirical evaluations of the methods. Other relevant work includes [6], which discusses *replacing* traces, and [1] which mentions the possibility of higher-order trace filters.³

IIR filters are a common signal processing tool for producing an output at each time step from a weighted combination of the $|b|$ most recent inputs and the $|a| - 1$ past outputs [8]. The term “infinite” impulse response arises because filters may have an infinitely non-zero response to an impulse input at time 0. IIR filters have uses in many digital signal processing applications. IIR filters allow efficient eligibility traces that can allow arbitrary assignment of credit.

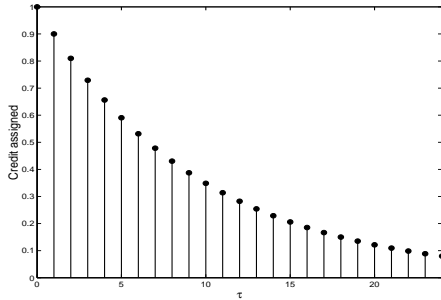


Fig. 1. The first 25 points of the infinite impulse response generated by the $\beta = 0.9$ model.

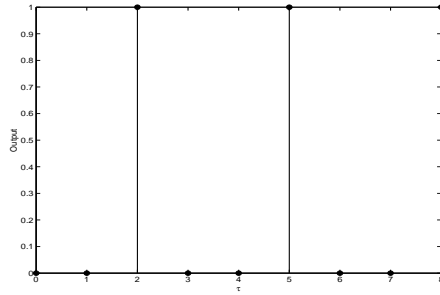


Fig. 2. The optimal FIR filter for Test I, with parameters $a = [1]$, $b = [0, 0, 1, 0, 0, 1, 0, 0, 1]$.

The general form of an IIR filter is

$$e_{t+1} = - \left(\sum_{n=1}^{|a|-1} a_n e_{t+1-n} \right) + \left(\sum_{n=0}^{|b|-1} b_n x_{t-n} \right), \quad (9)$$

where $a = [a_0, a_1, \dots, a_{|a|-1}]$ and $b = [b_0, b_1, \dots, b_{|b|-1}]$ are vectors of filter coefficients. The x 's represent the filter input at each time step. For eligibility traces, the filter input x_t is the gradient of the policy with respect to the chosen action and current parameters. The coefficient a_0 is assumed to be 1. To implement the filter, eligibility traces from the last $|a| - 1$ steps, and gradients x from the last $|b|$ action choices, must be stored, requiring $O(|\theta|(|b| + |a| - 1))$ units of memory.

Let τ denote the delay between an action and a possible reward. Impulse response plots, such as Figure 1, show the filter outputs e_τ after a scalar impulse input of $x_0 = 1$ and $x_\tau = 0$ for all $\tau > 0$. The response at delay τ represents the likelihood of receiving a reward for an action chosen at $\tau = 0$. The filter coefficients determine the poles and zeros of the filter's discrete-time frequency response. Given a desired impulse response, i.e., a desired temporal credit assignment model, filter coefficients can be chosen by a least squares fit of the filter

³ The author is also trying to locate [7], which may be relevant.

coefficients to the Fourier transform of the impulse response.⁴ The complexity of implementing the filter increases linearly with the number of coefficients. Hence it can be useful to choose a filter that only approximates the desired credit assignment, but has few coefficients. Unsurprisingly, experiments demonstrate it is better to award credit mistakenly than to not award credit where it is due.

A special case of IIR filters are finite impulse response (FIR) filters. FIR filters set $a = [1]$ and the maximum τ for which the impulse response is non-zero is given by $|b|$. FIR filters are useful if actions stop affecting rewards after a finite time. Finally, to recover the standard exponential credit assignment model with discount factor β , we set $a = [1.0, -\beta]$, and $b = [1]$.

3.1 GPOMDP(iir)

Algorithm 3 shows the GPOMDP(iir) algorithm for approximating $\nabla\eta$. The IIR filter is implemented by line 6. To recover the original GPOMDP algorithm of [1] line 6 is replaced with (7).

The gradient estimate returned by GPOMDP(iir) is passed to a Polak-Ribière conjugate-gradient ascent routine to update the parameters. The whole process repeats until the magnitude of Δ_T drops below 10^{-10} , indicating a local maximum has been reached. Convergence guarantees for GPOMDP [1] tell us that as $\beta \rightarrow 1$, and as the number of estimation steps $T \rightarrow \infty$, the GPOMDP estimate $\Delta_T = \nabla\eta$ exactly. Here, a similar result is presented for GPOMDP(iir). Without loss of generality the following proposition assumes an FIR filter with an unbounded number of coefficients. This includes all possible IIR filters. In practice IIR filters with a finite number of coefficients are more efficient.

Proposition 1. *Let the GPOMDP(iir) filter coefficients be $b_n = 1 \forall n = 0, \dots, |b| - 1$. Let Δ_T be the estimate produced by GPOMDP(iir) after T steps. Then under ergodicity assumptions, and if all absolute quantities in Algorithm 3 can be bounded by constants,⁵ then $\lim_{T \rightarrow \infty} \lim_{|b| \rightarrow \infty} \Delta_T = \nabla\eta$ w.p.1.*

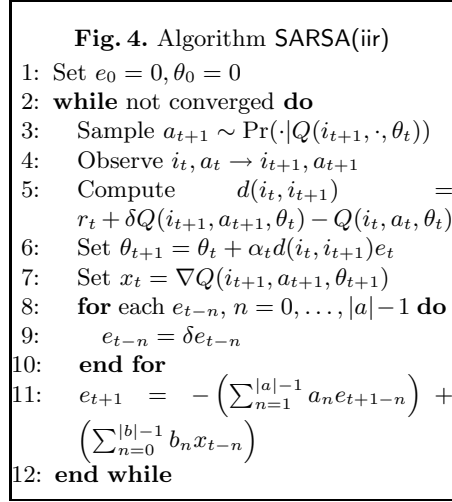
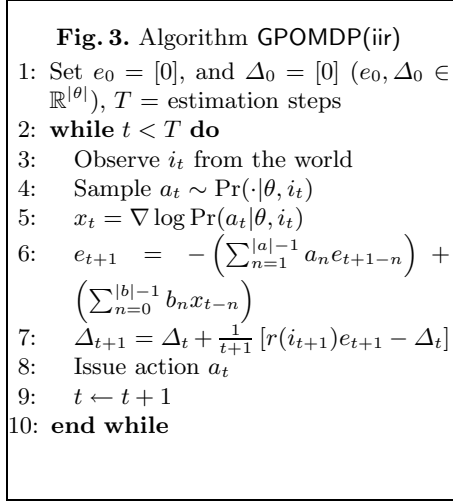
It states that as the number of estimation steps $T \rightarrow \infty$, and as the filter impulse response approaches a step function, the GPOMDP(iir) estimate converges to $\nabla\eta$. The proof can be seen by observing that approaching the step response filter is the same credit assignment model as $\beta \rightarrow 1$, then applying the proofs of [1]. If a tailored filter correctly assigns all credit the estimation will be unbiased without needing an infinite step response filter. Compare this with GPOMDP, which cannot produce unbiased estimate unless credit assignment is truly exponential. However, errors in the filter design will introduce bias into gradient estimates.

3.2 SARSA(iir)

Algorithm 4 shows the SARSA(iir) algorithm for approximating $Q(i, a)$. The IIR filter is implemented by lines 8–11. Lines 8 to 10 pre-discount the currently

⁴ The Matlab function `invfreqz` performs this task.

⁵ See [1] for the detailed assumptions.



stored eligibility traces before applying the IIR filter. This is needed to maintain the consistency of discounted value estimates, set by the temporal difference calculation in line 3.

Unlike GPOMDP, SARSA(iir) is an online algorithm that updates parameters at each step. Various termination conditions can be used, including stopping when values change by less than a threshold over a fixed number of iterations.

4 Experimental Bias and Variance

Four simple partially observable (PO)MDPs were contrived as initial test cases: (I) the POMDP of Figure 5 with $p = 0$; (II) the POMDP of Figure 5 with $p = 0.5$; (III) the POMDP of Figure 6 when completely observable; (IV) the POMDP of Figure 6 when only the tree depth is observable.

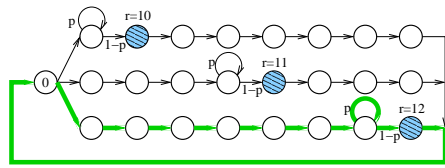


Fig. 5. The completely unobservable MDP used to test IIR trace filtering in Tests I and II. Test I sets $p = 0.0$; Test II sets $p = 0.5$ so that rewards may occur an arbitrarily long time after the action at state 0.

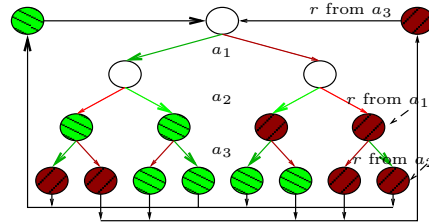


Fig. 6. In tests III and IV rewards have a delay of 1 step. In test IV, only the depth down the tree is observable. In light states $r = 1$, in dark states $r = -1$.

Tests I and II are examples in which normal GPOMDP performs poorly but trace filtering performs well. The optimal policy is to follow the lower path. The 3 actions only have an effect in state 0, deterministically leading to one of the 3 outward transitions. This POMDP is harder than it looks. For GPOMDP with $\beta < 0.97$ the reward discounted back to the action at state 0 appears higher for the upper two paths than for the lower optimal path. Thus for $\beta \leq 0.97$, the gradient will drive parameters away from the optimal policy.

In Tests III and IV the agent must fall down the correct branch of the tree to maximise its reward. The agent can move left or right at each level. The reward is always delayed by 1 step, that is, when the agent makes a decision leaving the top node, level 0, it gets the relevant reward when it reaches level 2. The test is interesting because rewards overlap; the reward received immediately after executing an action is actually due to the previous action.

GPOMDP(iir) was applied to all tests. The number of gradient estimate steps used for each test is shown in Tables 1 and 2. The bias optimal filter for Test I, where $p = 0$ in Figure 5, has a finite response with impulses corresponding to the 3 possible reward delays $\tau = 2, 5$ and 8 . This filter is shown in Figure 2. Two filters were applied to Test II. The first is a conservative FIR filter that assumes that rewards must be received between 2–12 steps after the relevant action. It makes no other assumption so all impulses between $\tau = 2$ and $\tau = 12$ have equal value, defining a rectangular filter. A good filter for Test II, where $p = 0.5$, should have an infinite response since rewards can be received an arbitrarily long time into the future. An IIR filter with impulses at the same places as Figure 2 was also tested, but the impulses were allowed to decay exponentially by setting the a_1 weight to -0.75 . This filter is shown in Figure 8. It might be suspected that we should decay the impulses by a factor of p , however it was found empirically that this produced a bias and variance worse than the FIR filter. This indicates that it is important to over-estimate credit assignment if bias needs to be minimised. The optimal filter for Tests III and IV is $a = [1]$ and $b = [0, 1]$, a single impulse at $\tau = 1$, accounting for the 1 step delay in rewards.

Because the test POMDPs are small, we can compute true gradient by evaluating Equation (5) directly. The true gradient was compared to 50 GPOMDP(iir) gradient estimates for each test and filter. Comparisons were also drawn with normal GPOMDP exponentially discounting at $\beta = 0.9$ and $\beta = 0.99$.

4.1 Results

The bias and variance of the estimates are shown in Tables 1 and 2. For Test I, $\beta = 0.9$ produced a gradient pointing in the wrong direction; $\beta = 0.99$ is in the correct direction but the high variance meant a total of around 1,000,000 estimation steps were required to achieve convergence to the correct policy. The simple FIR filter required only around 10,000 estimation steps. In Test II, the FIR filter only marginally improved the bias, however the variance was greatly reduced. The IIR filter improved the bias further because it does not arbitrarily cut off credit after 12 steps, but introducing an infinite response increased the variance. This demonstrates that the bias/variance trade-off in the choice of

discount factor is still evident when designing arbitrary filters. Proposition 1 tells us that one choice of unbiased filter for any POMDP is an infinite step response. A large class of POMDPs have unbiased filters that are not infinite step responses. For example, the POMDP of Test I and any POMDP that visits a recurrent state after at most T steps. Tests III and IV also show an order of magnitude improvement in bias and variance.

Table 1. Results for Tests I and II.

T	I: 10^6		II: 10^6	
Trace type	Bias	Var.	Bias	Var.
$\beta = 0.9$	176°	12.3	176°	18.4
$\beta = 0.99$	14.7°	2090	14.7°	2140
FIR	0.107°	7.72	13.9°	10.71
IIR			4.35°	59.5

Table 2. Results of Tests III and IV.

T	III: 1000		IV: 400	
Trace type	Bias	Var.	Bias	Var.
$\beta = 0.9$	0.610°	0.560	1.11°	111
$\beta = 0.99$	1.15°	2.88	2.36°	655
FIR	0.0450°	0.278	0.394°	16.7

4.2 SARSA(iir) Results

The SARSA(iir) algorithm was run on a completely observable version of Test I, comparing it to the performance of Sarsa(λ) with different values of λ . For these experiments, δ was fixed at 0.99 and the parameter step size α was fixed at 0.01. Three filters were used, with 100 training runs of each. The first using the optimal FIR filter of Figure 2, the second using $\lambda = 0.9$, and the third $\lambda = 0.99$.

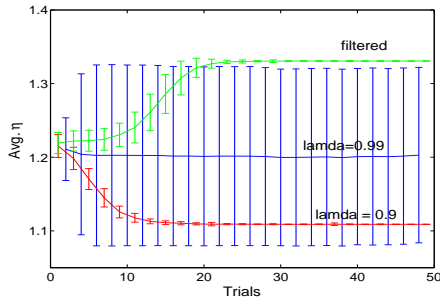


Fig. 7. Results of applying the SARSA(iir) to Test I. The error bars show one standard deviation.

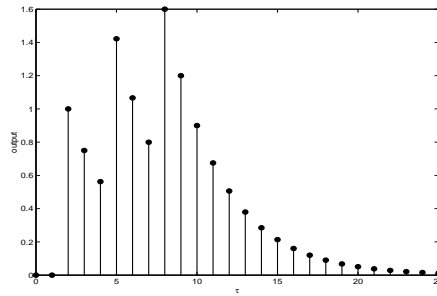


Fig. 8. A good IIR filter for Test II, with parameters $a = [1, -0.75]$, $b = [0, 0, 1, 0, 0, 1, 0, 0, 1]$.

Figure 7 plots the mean reward η received over trials of 1000 steps each, averaged over the 100 optimisation runs. The top line uses the filter of Figure 2, consistently finding the optimal policy in around 25,000 steps. The policy is effectively random when $\lambda = 0.99$ and the worst policy, with $\eta = 1.11$ is found

when $\lambda = 0.9$. This is consistent with the results using GPOMDP(iir), where the discount factor must be greater than 0.97 to avoid the worst policy.

5 Drug Dosage Control

When drugs are injected into the body they take a few tens of seconds to reach maximum effect. The effect then decreases with time. Figure 9 shows an approximate drug concentration curve for the large class of drugs that follow first order *kinetics of elimination* [9]. The challenge is to choose dosages at each time step to maintain a desired drug concentration. Using Matlab to approximate a filter for Figure 9 results in an IIR filter with coefficients $a = [1.0, -1.30, 0.096, 0.24]$ and $b = [0.0074, 0.039, -0.023, 0.18, 0.092, 0.21]$. These coefficients were used to construct a simulator for the reaction of a person to drug injection actions. The simulator implements (9), allowing an action to be chosen at each discrete time step and outputting the current total concentration from all doses. The state space is a subspace of $\mathbb{R}^{|a|-1+|b|}$, consisting of a vector containing the last 3 total concentrations, and the last 6 actions.

At each time step the agent receives a measurement of the current drug concentration in the patient, i.e., the last output of the simulator’s implementation of (9), corrupted by $\mathcal{N}(0, 0.05)$ noise. A neural network is used to parameterise the policy for this continuous state space. The 2 inputs are the desired concentration level and the observed concentration level, normalised to a $[0, 1]$ range. The 3 hidden units are squashed using the tanh function. The agent chooses from 3 actions: 0 dose, 0.5 dose, or 1.0 dose. The 3 network outputs form the y vector, passed to a soft-max function to produce a stochastic policy. The network weights were randomly initialised in $[-0.01, 0.01]$. The desired concentration level was defined as $14.8 \sin(2\pi t/1000)$, forcing the agent to track a moving target.

Five filters were tried over 100 training runs each: standard exponential discounting with $\beta = \{0.8, 0.9, 0.99\}$, a rectangular FIR filter that cuts off credit assignment after $\tau = 40$, and an IIR filter with the same IIR filter coefficients used in the simulator. Each gradient was estimated with $T = 2 \times 10^4$ steps.

5.1 Results

Figure 10 shows the convergence of the two *best* filters. Using the IIR filter instead of discounting with $\beta = 0.9$ improved the average η from 0.621 to 0.675. A single sided t-test indicates this result is significant with 99.9% confidence. The variance of the final IIR solutions was 0.010 compared to 0.014 for $\beta = 0.9$. Random policies achieved $\eta = 0.47$ on average. It took an average of 19.0 seconds to obtain a value of $\eta = 0.8$ (achieved 6/100 times) using a 3GHz Pentium 4, or 1.28 times longer than the exponential discounting took to achieve the same η (which it did 4/100 times). Discounting with $\beta = 0.9$ was the second best filter because of the roughly 0.9 decay rate in Figure 9 after the initial peak. The rectangular filter was the next best with average $\eta = 0.599$, $\beta = 0.8$ returned $\eta = 0.591$, and $\beta = 0.99$ returned $\eta = .520$. Increasing the estimation times for the IIR filter allowed $\eta = 0.812$ to be found consistently.

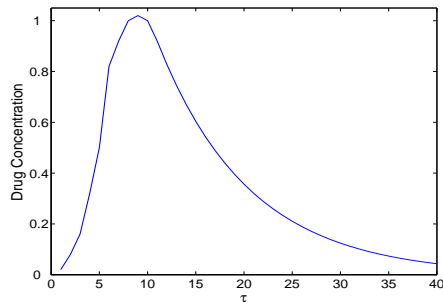


Fig. 9. A first-order kinetics drug-concentration vs. time curve. The drug is injected at $\tau = 0$.

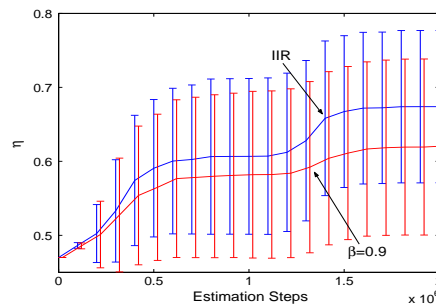


Fig. 10. Drug dosage task results averaged over 100 runs. Error bars indicate one standard deviation.

6 Conclusions and Further Work

Higher-order models of temporal credit assignment can result in greatly reduced bias and variance, both for value estimates and policy-gradient estimates. Trace filtering can be applied to any RL algorithm that uses an eligibility trace. Better theoretical characterisations of the bias-variance trade-offs of IIR filters are needed to aid the choice of robust filters. The preliminary experiments are being extended to problems from non-linear control literature.

Acknowledgements

National ICT Australia is funded by the Australian Government, the Australian Research Council, and the ICT Centre of Excellence program.

References

1. Baxter, J., Bartlett, P.L.: Infinite-horizon policy-gradient estimation. *JAIR* **15** (2001) 319–350
2. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. (1998)
3. Watkins, C.J.C.H.: *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, England (1989)
4. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8** (1992) 229–256
5. Aberdeen, D.: *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Australian National University (2003)
6. Singh, S.P., Sutton, R.S.: Reinforcement learning with replacing eligibility traces. *Machine Learning* **22** (1996) 123–158
7. Gurvits, L., Lin, L.J., Hanson, S.J.: Incremental learning of evaluation functions for absorbing markov chains: New methods and theorems. pre-print (1994)
8. Elliott, S.: *Signal Processing for Active Control*. Academic Press (2001)
9. Boroujerdi, M.: *Pharmacokinetics: Principles and Application*. McGraw-Hill, New York, NY (2002)