Temporal Difference Bayesian Model Averaging: A Bayesian Perspective on Adapting Lambda

Carlton Downey CARLTON.DOWNEY@ECS.VUW.AC.NZ
Victoria University of Wellington, Wellington, New Zealand

Scott Sanner SSANNER@NICTA.COM.AU
Statistical Machine Learning Group & AI Group, NICTA & ANU, Canberra, Australia

Abstract

Temporal difference (TD) algorithms are attractive for reinforcement learning due to their ease-of-implementation and use of "bootstrapped" return estimates to make efficient use of sampled data. In particular, TD(λ) methods comprise a family of reinforcement learning algorithms that often yield fast convergence by averaging multiple estimators of the expected return. However, TD(λ) chooses a very specific way of averaging these estimators based on the fixed parameter λ, which may not lead to optimal convergence rates in all settings. In this paper, we derive an automated Bayesian approach to setting λ that we call temporal difference Bayesian model averaging (TD-BMA). Empirically, TD-BMA always performs as well and often *much* better than the best fixed λ for TD(λ) (even when performance for different values of λ varies across problems) without requiring that λ or any analogous parameter be manually tuned.

1. Introduction

In reinforcement learning (RL) it can be crucial to minimize sample complexity (Kakade, 2003) — a bound on the number of samples required to achieve a given quality policy. Fewer samples may lead to faster algorithms, but perhaps even more crucial to real-world applications, fewer samples require fewer expensive interactions with the environment. This is especially important when these samples involve the risk of mistakes that have real costs (monetary or otherwise).

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

Temporal difference (TD) learning methods have proved to be an empirically effective way to reduce the sample complexity of RL algorithms (Sutton & Barto, 1998). The primary insight of these methods is that rather than taking independent Monte Carlo (MC) sample estimates of the value being learned, one may achieve faster learning rates by reusing "bootstrapped" value estimates from immediate successor states. One popular and particularly effective TD method — TD(λ) — goes even one step further in reducing sample complexity by *averaging* over different n-step lookahead estimators of the same value, weighting by a function of λ that decays exponentially in n.

In this work, we build on the TD(λ) approach of computing a weighted average of n-step return value estimates. However, our weighted averaging approach is intended to directly exploit one of the reasons why TD(λ) methods are often successful in practice at reducing sample complexity — weighted averages can reduce the variance of the value estimate, leading to less noise in the update process, and ultimately faster convergence. Based on this view, we adopt a statistically principled method of weighting the n-step return value estimates in a TD(λ)-like approach we call Temporal Difference Bayesian Model Averaging (TD-BMA).

Empirically, we show that TD-BMA generally performs *much* better than the best fixed λ for TD(λ) and does so automatically *without* requiring λ or any analogous parameter be manually tuned for a problem. Given the promise of TD-BMA methods, we conclude with a discussion of important future generalizations.

2. Preliminaries

2.1. Markov Decision Processes

We assume the decision-making environment to be a *Markov decision process* (MDP) (Puterman, 1994) in

which an agent interacts by repeatedly executing an action in the currently observed state, receiving a reward signal and then stochastically transitioning to a successor state. Formally, an MDP can be defined as a tuple $\langle S, A, T, R\rangle$. $S = \{s_1, \ldots, s_n\}$ is a finite set of fully observable states. $A = \{a_1, \ldots, a_m\}$ is a finite set of actions. $T : S \times A \times S \to [0, 1]$ is a stationary, Markovian transition function, where we write $T(s, a, s') = P(s'|s, a)$. We will assume that a stochastic reward function $R : S \times A \times \mathbb{R} \to [0, 1]$ is associated with every state and action, where we write $R(s, a, r) = P(r|s, a)$ for $r \in \mathbb{R}$. γ $(0 \leq \gamma \leq 1)$ is a discount factor used to specify that a reward obtained t timesteps into the future is discounted by γ^t. $\gamma = 1$ is permitted if total accumulated reward is finite.

A stochastic exploration policy $\pi : S \times A \to [0, 1]$ specifies a probability distribution $\pi(s, a) = P(a|s)$ over actions a to take in each state s. The value $Q^\pi(s, a)$ of taking an action a in state s and then following the policy π thereafter can be defined using the infinite horizon, expected discounted reward criterion:

$$Q_\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot r_{t+1} \middle| s_0 = s, a_0 = a \right] \quad (1)$$

where r_{t+1} is the reward obtained after taking an action a_t in state s_t at time t (assuming s_0 and a_0 respectively represent the state and action at $t = 0$). Then we can define a value function $V^\pi(s) = Q_\pi(s, \pi(s))$ that represents the expected value obtained by starting in state s and acting according to π.

The objective in an MDP is to find a policy π^* such that $\forall \pi, s.\ V^{\pi^*}(s) \geq V^\pi(s)$; at least one such optimal policy is guaranteed to exist (Puterman, 1994).

In the *reinforcement learning* (RL) setting, the transition probabilities $P(s'|s, a)$ and reward probabilities $P(r|s, a)$ may not be explicitly known to the agent although both can be sampled from experience. In order to learn an optimal policy, we can adopt the *generalized policy iteration* (GPI) framework shown in **Algorithm 1** that is known to capture many reinforcement learning approaches (Sutton & Barto, 1998).

GPI interleaves policy evaluation and greedy policy update steps (steps 2 and 3, respectively). Since we require stochastic policies to ensure all states and actions are explored in RL, we use the simple ϵ-greedy exploration policy shown in step 3, where exploratory actions are chosen uniformly randomly with ϵ probability. Because step 3 is given, the remainder of this paper will focus on efficient policy evaluation in step 2 of GPI via *temporal difference* reinforcement learning. For the policy evaluation task, we assume a standard

Algorithm 1 GENERALIZED POLICY ITERATION

begin

1. Start with arbitrary initial policy π_0 & $i = 0$.
2. Estimate $Q^{\pi_i}(s, a)$
3. $\pi_{i+1}(s, a) = P(a|s) = \begin{cases} 1 - \epsilon & \arg\max_a Q^{\pi_i}(s, a) \\ \frac{\epsilon}{|A|} & a_1 \\ \cdots & \cdots \\ \frac{\epsilon}{|A|} & a_{|A|} \end{cases}$
4. If termination criteria not met, $i = i + 1$ & goto 2.

end

episodic RL setting consisting of multiple trials. Each trial terminates at some (different) time T, after which all rewards r_t for $t > T + 1$ are assumed to be 0.

2.2. Temporal Difference Learning

Temporal difference (TD) RL methods (Sutton & Barto, 1998) were introduced in order to allow efficient reuse of all sampled rewards during policy evaluation. The basic idea begins with the observation that for time t, a 1-step *sample return* $R_t^{(1)}$ of $Q_\pi(s_t, a_t)$ given in (1) can be "bootstrapped" from the estimate for $Q_\pi(s_{t+1}, a_{t+1})$

$$R_t^{(1)} = r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) \quad (2)$$

where s_{t+1} and a_{t+1} are sampled according to π and r_{t+1} sampled according to $P(r_{t+1}|s_t, a_t)$. Then given $R_t^{(1)}$, we arrive at a simple 1-step TD update rule that allows us to adjust $Q_\pi(s_{t+1}, a_{t+1})$ according to the TD error $\Delta Q_t^{(1)}$:

$$Q_\pi(s_t, a_t) = Q_\pi(s_t, a_t) + \alpha \underbrace{\left[R_t^{(1)} - Q_\pi(s_t, a_t) \right]}_{\Delta Q_t^{(1)}} \quad (3)$$

Here, $\alpha > 0$ is a learning rate. In the literature, this Q-value version of the TD update is known as SARSA for *state, action, reward, state, action* (Sutton & Barto, 1998).

In general, rather than just a 1-step return, we can define arbitrary n-step $(n > 0)$ bootstrapped returns:

$$R_t^{(n)} = \sum_{i=1}^{n} \gamma^{i-1} r_{t+i} + \gamma^n Q_\pi(s_{t+n}, a_{t+n}) \quad (4)$$

where all actions $a_1, \ldots, a_{t+n}$ are sampled from $\pi(s, a) = P(a|s)$, all states $s_1, \ldots, s_{t+n}$ are sampled from $T(s, a, s') = P(s'|a, s)$, and all rewards $r_1, \ldots, r_{t+n}$ are sampled according to $P(r_{t+i}|s_{t+i-1}, a_{t+i-1})$.

All $R_t^{(n)}$ turn out to be estimators for $Q_\pi(s_t, a_t)$. Since it is well-known that averaging samples of multiple estimators yields reduced variance over the use of any

Algorithm 2 SARSA(λ)-OFFLINE-UPDATE(T)

begin
  $R_T^\lambda = r_{T+1}$
  for $(t = T \dots 0)$ **do**
    if $(t < T)$ **then**
      // Compute λ-average of returns
      $R_t^\lambda = r_{t+1} + \gamma\left[(1-\lambda)Q_\pi(s_{t+1}, a_{t+1}) + \lambda R_{t+1}^\lambda\right]$
    $Q_\pi(s_t, a_t) = Q_\pi(s_t, a_t) + \alpha\left[R_t^\lambda - Q_\pi(s_t, a_t)\right]$
end

single sample, it is clearly advantageous to use some form of averaged return in place of the single 1-step return $R_t^{(1)}$ to reduce variance. One particularly elegant and computationally efficient weighted averaging approach is given by TD(λ) ($0 \leq \lambda \leq 1$), which defines the return (Sutton & Barto, 1998):

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} \tag{5}$$

One can verify that the geometric series sum of weights $(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1} = 1$. Then instead of the 1-step TD update, we can substitute the 1-step TD error $\Delta Q_t^{(1)}$ with the n-step averaged TD(λ) error

$$\Delta Q_t^\lambda = \left[R_t^\lambda - Q_\pi(s_t, a_t)\right] \tag{6}$$

in (3) and thus obtain the the TD(λ) update rule for Q-values known as SARSA(λ).

Looking ahead to a form that will be useful for our derivation of temporal difference Bayesian model averaging, we will derive a simple offline[1] algorithm for TD(λ) following Section 7.4 of (Sutton & Barto, 1998) to reorganize the summation and express the return R_t^λ recursively in terms of R_{t+1}^λ:

$$\begin{aligned}
R_t^\lambda &= \sum_{k=t}^{T-1}(\gamma\lambda)^{k-t}\left[R_k^{(1)} - \gamma\lambda Q_\pi(s_{k+1}, a_{k+1})\right] \\
&= R_t^{(1)} - \gamma\lambda Q_\pi(s_{t+1}, a_{t+1}) + \gamma\lambda R_{t+1}^\lambda \\
&= r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) - \gamma\lambda Q_\pi(s_{t+1}, a_{t+1}) + \gamma\lambda R_{t+1}^\lambda \\
&= r_{t+1} + \gamma\left[(1-\lambda)Q_\pi(s_{t+1}, a_{t+1}) + \lambda R_{t+1}^\lambda\right] \qquad (7)
\end{aligned}$$

Combining the update in (3) with the error ΔQ_t^λ in (6) and the recursive definition of R_t^λ in (7), we obtain the offline SARSA(λ) algorithm shown in **Algorithm 2** that is called once at the end of every trial of policy evaluation of T time steps.

[1] Meaning that in a learning task consisting of trials, the TD(λ) update is performed once for all encountered states at the end of a trial.

3. Temporal Difference Bayesian Model Averaging

If we revisit the TD(λ) return in (5), we note this expression chooses a particular fixed function of λ to weight each bootstrapped n-step return estimate. While there are excellent computational reasons for choosing this particular weighting scheme, we set aside such issues for now and set out to answer a simple question: is there a potentially better way to set the weights of each t-step return, e.g., with an objective to reduce variance in the value estimator?

One commonly used technique for reducing estimator variance is Bayesian model averaging (BMA). In particular, BMA provides an intuitive data-dependent way to adjust the weights of multiple estimators according to the likelihood that they might have generated the observed data. But how do we combine BMA with TD estimators? We begin with the general case and then proceed to develop a specific model that is effective in practice and computationally efficient — having the same time and space complexity as SARSA(λ), but adapting λ in a Bayesian manner.

3.1. General Case

In the general Bayesian model averaging setting, we will maintain Q-values $Q_\pi(s,a)$, not as constants, but as a multivariate probability distribution $P(\vec{q}|D)$ where D represents the set of all returns observed for a state and action and $\vec{q} = (q_{s_1,a_1}, \dots, q_{s_1,a_{|A|}}, \dots, q_{s_{|S|},a_1}, \dots, q_{s_{|S|},a_{|A|}}) \in \mathbb{R}^{|S||A|}$. In Section 3.2, we will discuss concrete ways to maintain this Q-value distribution, but for now we assume it is given and focus on how to use it for TD learning.

Given $P(q_{s,a}|D)$, we derive an expected return model R_t^{BMA} for state $s = s_t$ and action $a = a_t$ based on Bayesian model averaging:

$$\begin{aligned}
R_t^{BMA} &= \mathbb{E}_{P(\vec{q}|D)}\left[q_{s,a}\right] \\
&= \int_{q_{s,a}\in\mathbb{R}} q_{s,a} P(q_{s,a}|D)\, dq_{s,a} \\
&= \int_{q_{s,a}\in\mathbb{R}} q_{s,a} \sum_{m\in MC,TD} P(q_{s,a}|m)P(m|D)\, dq_{s,a} \\
&= \sum_{m\in MC,TD} \underbrace{\mathbb{E}_{P(q_{s,a}|m)}\left[q_{s,a}\right]}_{\text{prediction of } m} \underbrace{P(m|D)}_{\text{weight of } m} \qquad (8)
\end{aligned}$$

The first step expands the definition of expectation and marginalizes out irrelevant random variables. The second step introduces Bayesian model averaging by introducing and marginalizing over an additional model parameter m specifying either the *MC* (1-step *local* Monte Carlo sample return $R_{t+1}^{(1)}$) or

TD (1-step temporal difference bootstrapped return $Q_\pi(s_{t+1}, a_{t+1})$) models, assuming each model of $q_{s,a}$ is independent of D given m. Finally, distributing $q_{s,a}$, swapping the $\sum$ and $\int$, and exploiting the independence of $P(m|D)$ from $q_{s,a}$, the integral is expressed in expectation form. This yields the intuitive Bayesian model averaging result that we should weight (i.e., trust) the expected predictions of the *MC* and *TD* models by the conditional probability of these models given the data D; this fundamental observation underlies the premise of the paper:

If we arbitrarily *set* $P(MC|D) = \lambda$ *and* $P(TD|D) = (1-\lambda)$ *then* $R_t^{BMA} = R_t^\lambda$ *and we have exactly re-derived the offline view of SARSA(*λ*). But given this Bayesian model averaging perspective, we ask whether* $P(MC|D) = \lambda$ *and* $P(TD|D) = (1-\lambda)$ *with fixed* λ *are really sensible model weightings? In fact, these weights are not data dependent as the form of* $P(MC|D)$ *and* $P(TD|D)$ *suggest they should be. In the next section we derive a novel model-weighting for* $P(MC|D)$ *and* $P(TD|D)$ *that is both data-dependent and as efficient to compute as the non-data-dependent SARSA(*λ*).*

3.2. Gaussian Model

We assume Q-values of each state-action pair are independently Gaussian distributed (Dearden et al., 1998), thus we write local models m over each $q_{s,a}$ as follows:[2]

$$P(q_{s,a}|m) = \mathcal{N}(q_{s,a}; \mu^m_{s,a}, (\sigma^m_{s,a})^2). \tag{9}$$

Each model m is summarized by sufficient statistics $(\mu^m_{s,a}, \sigma^m_{s,a})$ representing the mean and standard deviation for the normal distribution $\mathcal{N}$ over each $q_{s,a}$.

For models $m \in \{MC, TD\}$, the model means are respectively $\mu^{MC}_{s,a} = q^{MC}_{s,a}$ and $\mu^{TD}_{s,a} = q^{TD}_{s,a}$ where $q^{MC}_{s,a}$ and $q^{TD}_{s,a}$ are defined w.r.t. $s = s_t$ and $a = a_t$:

$$q^{TD}_{s,a} = r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) \tag{10}$$
$$q^{MC}_{s,a} = r_{t+1} + \gamma R^{BMA}_{t+1} \tag{11}$$

We note some crucial details: (a) $q^{MC}_{s,a}$ and $q^{TD}_{s,a}$ are *samples* of random variable $q_{s,a}$ — we construct Bayesian models of $q_{s,a}$ using these *MC* and *TD* samples as means $\mu^{MC}_{s,a}$ and $\mu^{TD}_{s,a}$; (b) looking ahead we provide the recursive definition of R^{BMA}_{t+1} in (15); (c) technically, we note the MC return is only *locally* Monte Carlo (not a true MC return to the end of the trial).

Given the means $\mu^{MC}_{s,a}$ and $\mu^{TD}_{s,a}$, we *could* explicitly compute $\sigma^{MC}_{s,a}$ and $\sigma^{TD}_{s,a}$ from the data set $D = \bigcup_{a\in A, s\in S} D_{s,a}$ where $D_{s,a} = \{R^{BMA}_t | s = s_t, a = a_t\}$ is the set of observed returns. But this would require caching all data and recomputing $\sigma^{MC}_{s,a}$ and $\sigma^{TD}_{s,a}$ from the new means $\mu^{MC}_{s,a}$ and $\mu^{TD}_{s,a}$ on *each* state-action visit. As this is prohibitively expensive, we must avoid caching D. However, we note that an excellent surrogate standard deviation that captures the general deviation characteristics of the observed returns $D_{s,a}$ is simply the standard deviation $\sigma_{s,a}$ of $D_{s,a}$:

$$\mu_{s,a} = \frac{\sum_{d\in D_{s,a}} d}{|D_{s,a}|} \qquad \sigma_{s,a} = \left(\frac{\sum_{d\in D_{s,a}} (d-\mu_{s,a})^2}{|D_{s,a}|}\right)^{\frac{1}{2}}$$

Hence we set *both* $\sigma^{MC}_{s,a} = \sigma^{TD}_{s,a} = \sigma_{s,a}$. Later in the context of TD-BMA in **Algorithm 3**, we will discuss how the $\mu_{s,a}$ and $\sigma_{s,a}$ can be updated efficiently online.

Now that we've specified $\mu^m_{s,a}$ and $\sigma^m_{s,a}$ to define $P(q_{s,a}|m)$ for both $m \in \{MC, TD\}$ in (9), we return to (8). For (8), we need to derive two quantities to compute R^{BMA}_t: (a) the model prediction which we can trivially derive from the known properties of the normal distribution: $\mathbb{E}_{P(q_{s,a}|m)}[q_{s,a}] = \mu^m_{s,a} = q^m_{s,a}$; and (b) the model weight $P(m|D)$, which we derive as follows where $m = m_{s,a}$ and $C = (2\pi\sigma^2_{s,a})^{-1/2}$:

$$\begin{aligned}
P(m|D) &\overset{\text{Indep.}}{=} P(m|D_{s,a}) \overset{\text{Bayes rule}}{\propto} P(D_{s,a}|m)P(m) \\
&\overset{\text{Unif. prior}}{\propto} P(D_{s,a}|m) \\
&\overset{\text{i.i.d.}}{=} \textstyle\prod_{d\in D_{s,a}} P(d|m) \\
&= \textstyle\prod_{d\in D_{s,a}} \mathcal{N}(d; q^m_{s,a}, \sigma^2_{s,a}) \\
&= \textstyle\prod_{d\in D_{s,a}} C e^{-\frac{(d-q^m_{s,a})^2}{2\sigma^2_{s,a}}} \\
&= C^{|D_{s,a}|} e^{\left(\frac{1}{2\sigma^2_{s,a}}\right)\boxed{\sum_{d\in D_{s,a}} \left(-d^2 - (q^m_{s,a})^2 + 2q^m_{s,a} d\right)}}
\end{aligned} \tag{12}$$

In order, we exploited independence of the model m (for s, a) from data other than $D_{s,a}$, Bayes rule, an assumption of a uniform prior $P(m)$ over models m, the i.i.d. assumption, the definition of the normal distribution $\mathcal{N}(d; q^m_{s,a}, \sigma^2_{s,a})$ and from there, simple exponential and algebraic identities. Next we simplify $\square$:

$$\begin{aligned}
\square &= -\sum_{d\in D_{s,a}} d^2 - \sum_{d\in D_{s,a}} (q^m_{s,a})^2 + 2q^m_{s,a} \sum_{d\in D_{s,a}} d \\
&= -|D_{s,a}|(\sigma^2_{s,a} + \mu^2_{s,a}) + -|D_{s,a}|(q^m_{s,a})^2 + 2q^m_{s,a}|D_{s,a}|\mu_{s,a} \\
&= -|D_{s,a}|\left(\sigma^2_{s,a} + \mu^2_{s,a} + (q^m_{s,a})^2 - 2q^m_{s,a}\mu_{s,a}\right) \\
&= -|D_{s,a}|\sigma^2_{s,a} + -|D_{s,a}|(q^m_{s,a} - \mu_{s,a})^2
\end{aligned} \tag{13}$$

Here we rewrote the first $\sum$ using the identity:

$$\begin{aligned}
\sigma^2_{s,a} &= \frac{\sum_{d\in D_{s,a}} d^2}{|D_{s,a}|} - \mu^2_{s,a} \\
\implies \textstyle\sum_{d\in D_{s,a}} d^2 &= |D_{s,a}|(\sigma^2_{s,a} + \mu^2_{s,a})
\end{aligned}$$

[2] For each $m \in \{MC, TD\}$, there is actually a set of models $\{m_{s,a}\}$ for all state-action pairs; we drop the subscript and write $m = m_{s,a}$ when s, a are clear from context.

We simplified the second $\sum$ noting that the summand is independent of the variable being summed over. And for the third $\sum$, we have simply noted this is the definition of the mean $\mu_{s,a}$ of $D_{s,a}$ (modulo $|D_{s,a}|$).

Now, substituting (13) into $\square$ of (12), we obtain:

$$\begin{aligned} P(m|D) &\propto C^{|D_{s,a}|} e^{-\frac{|D_{s,a}|\sigma^2_{s,a}}{2\sigma^2_{s,a}} - \frac{|D_{s,a}|}{2\sigma^2_{s,a}}\left(q^m_{s,a}-\mu_{s,a}\right)^2} \\ &= e^{-\frac{|D_{s,a}|}{2}} C^{|D_{s,a}|} \left(e^{-\frac{\left(q^m_{s,a}-\mu_{s,a}\right)^2}{2\sigma^2_{s,a}}} \right)^{|D_{s,a}|} \\ &= e^{-\frac{|D_{s,a}|}{2}} \mathcal{N}(q^m_{s,a}; \mu_{s,a}, \sigma^2_{s,a})^{|D_{s,a}|} \end{aligned} \tag{14}$$

This is both a pleasing and important result. We began with $P(m|D)$ where the model m had a mean $q^m_{s,a}$ likely to change on every update (it is itself a sample). This raised the possibility that we might have needed to recompute the entire model posterior $P(m|D)$ *every* time the data set D was updated. However, by various manipulations and identities, we showed in the final result that in fact we need only know $|D_{s,a}|$ and the first and second moments ($\mu_{s,a}$ and $\sigma^2_{s,a}$) of $D_{s,a}$ in order to compute $P(m|D)$. This is important since we will show next that all of these statistics can be updated online in constant time without caching D!

Finally we derive the exact individual probabilities $P(m|D)$ for $m \in \{MC, TD\}$ where (a) $N_{s,a} = |D_{s,a}|$ and (b) we note the proportionality constants (call them C_{MC} and C_{TD}) for respective models $P(MC|D)$ and $P(TD|D)$ are *independent* of the model and hence factor and cancel from $P(m|D)$, leaving just $\mathcal{N}(\cdot)$:

$$\begin{aligned} P(m|D) &= \frac{P(m|D)}{\sum_{m\in\{MC,TD\}} P(m|D)} \\ &= \frac{[\mathcal{N}(q^m_{s,a}; \mu_{s,a}, \sigma^2_{s,a})]^{|N_{s,a}|}}{[\mathcal{N}(q^{MC}_{s,a}; \mu_{s,a}, \sigma^2_{s,a})]^{|N_{s,a}|} + [\mathcal{N}(q^{TD}_{s,a}; \mu_{s,a}, \sigma^2_{s,a})]^{|N_{s,a}|}} \end{aligned}$$

With this result, we have derived all components required to compute R^{BMA}_t from (8) for the Gaussian BMA model. Recalling definitions of $q^{TD}_{s,a}$ in (10) and $q^{MC}_{s,a}$ in (11) and defining $\lambda = P(MC_{s,a}|D)$ (then $1-\lambda = 1 - P(MC_{s,a}|D) = P(TD_{s,a}|D)$), we obtain:

$$\begin{aligned} R^{BMA}_t &= \sum_{m\in MC_{s,a}, TD_{s,a}} \mathbb{E}_{P(q_{s,a}|m)}\left[q_{s,a}\right] P(m|D) \\ &= q^{TD}_{s,a} P(TD_{s,a}|D) + q^{MC}_{s,a} P(MC_{s,a}|D) \\ &= (r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}))(1-\lambda) + (r_{t+1} + \gamma R^{BMA}_{t+1})\lambda \\ &= (1-\lambda) r_{t+1} + \lambda(r_{t+1}) + \gamma(1-\lambda) Q_\pi(s_{t+1}, a_{t+1}) + \gamma\lambda R^{BMA}_{t+1} \\ &= r_{t+1} + \gamma\left[(1-\lambda) Q_\pi(s_{t+1}, a_{t+1}) + \lambda R^{BMA}_{t+1}\right] \end{aligned} \tag{15}$$

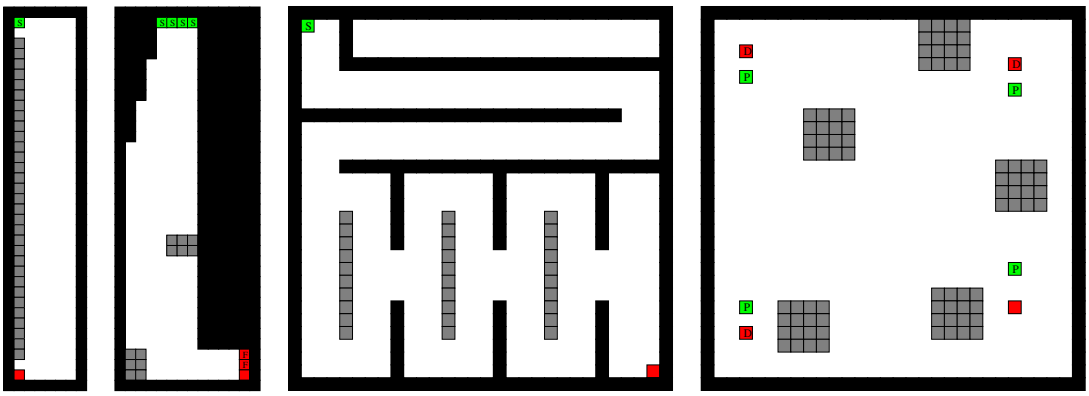
This final result is exactly the same form as (7) *except* that here we have re-derived the SARSA(λ) TD return estimate in a Bayesian framework where λ is *automatically* adapted to the data using the Bayesian model averaging perspective.

Algorithm 3 TD-BMA-Offline-Update(T)

begin
  $R^{BMA}_T = r_{T+1}$
  for *($t = T \ldots 0$)* **do**
    // For readability, let $s = s_t,\ a = a_t$
    if *($t < T$)* **then**
      // Compute model expectations
      $q^{TD}_{s,a} = r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})$
      $q^{MC}_{s,a} = r_{t+1} + \gamma R^{BMA}_{t+1}$
      // Compute BMA model weighting λ
      // $\lambda = P(MC_{s,a}|D)$
      // $1-\lambda = 1 - P(MC_{s,a}|D) = P(TD_{s,a}|D)$
      $\lambda = \frac{\mathcal{N}(q^{MC}_{s,a}; \mu_{s,a}, \sigma^2_{s,a})^{N_{s,a}}}{\mathcal{N}(q^{MC}_{s,a}; \mu_{s,a}, \sigma^2_{s,a})^{N_{s,a}} + \mathcal{N}(q^{TD}_{s,a}; \mu_{s,a}, \sigma^2_{s,a})^{N_{s,a}}}$
      // Compute BMA return; note the
      // equivalence: $R^{BMA}_t = q^{TD}_{s,a}(1-\lambda) + q^{MC}_{s,a}\lambda$
      $R^{BMA}_t = r_{t+1} + \gamma\left[(1-\lambda) Q_\pi(s_{t+1}, a_{t+1}) + \lambda R^{BMA}_{t+1}\right]$
    $Q_\pi(s,a) = Q_\pi(s,a) + \alpha\left[R^{BMA}_t - Q_\pi(s,a)\right]$
    // Update μ, σ sufficient statistics (Knuth, 1998)
    $N_{s,a} = N_{s,a} + 1$
    $\Delta = R^{BMA}_t - \mu_{s,a}$
    $\mu_{s,a} = \mu_{s,a} + \frac{\Delta}{N_{s,a}}$
    $M^2_{s,a} = M^2_{s,a} + \Delta \cdot (R^{BMA}_t - \mu_{s,a})$
    if $N_{s,a} > 1$ **then**
      $\sigma_{s,a} = \sqrt{M^2_{s,a}/N_{s,a}}$
end

From this we note that we need only modify the SARSA(λ) update in **Algorithm 2** in three ways to obtain TD-BMA in **Algorithm 3**: (a) in place of a fixed λ calculation, we compute the BMA adaptive version of λ; (b) from this λ, we showed in (15) we can compute R^{BMA}_t analogously to R^{λ}_t, then in Q-value update (3) we replace the error ΔQ^{λ}_t with $\Delta Q^{BMA}_t = \left[R^{BMA}_t - Q_\pi(s_t, a_t)\right]$; (c) using the new return R^{BMA}_t, we do an online update of the sufficient statistics $N_{s,a}, \mu_{s,a}, \sigma_{s,a}$ of $D_{s,a}$ required by TD-BMA.

Constant time and space online algorithms for maintaining the mean $\mu_{s,a}$ and standard deviation $\sigma_{s,a}$ as new data is accumulated in $D_{s,a}$ are given in (Knuth, 1998) and are shown in **Algorithm 3**; these updates also maintain $N_{s,a}$, which trivially counts the number of updates for $Q_\pi(s,a)$, and $M^2_{s,a}$, which is required for the numerically stable, efficient online computation of $\sigma^2_{s,a}$. For initialization, we assume $\mu_{s,a} = 0$, $\sigma_{s,a} = \infty$, $N_{s,a} = 0$ and $M^2_{s,a} = 0$.



(a) Cliff (b) Corner (c) Curves (d) Mail

Figure 1. (a,b,c) Various RACETRACK domains evaluated in this paper. Initial states are labeled 'S', terminal states are labeled 'F'. Black squares delineate walls and whitespace indicate legal car coordinates. Gray indicates obstacles that reset the car to the initial state and result in a penalty of -100. (d) The MAIL ROBOT domain evaluated in this paper. The robot can pick up mail at blocks labeled P that is intended for different dropoff points labeled D. Gray squares in this domain incur a penalty of -100, but do not reset the robot state.

Complexity Analysis: One can easily verify that both SARSA(λ) and TD-BMA require constant time per s_t, a_t update and hence each trial of T steps takes $O(T)$ time for both algorithms. While TD-BMA requires four more constants per state-action pair $(N_{s,a}, \mu_{s,a}, \sigma^2_{s,a}, M^2_{s,a})$ than SARSA(λ), both algorithms clearly still have $O(|S||A|)$ space complexity. Hence TD-BMA and SARSA(λ) have the same time and space complexity, even though TD-BMA manages to adapt λ in a Bayesian model averaging framework.

4. Empirical Results

Having worked through the derivation of TD-BMA, we now reach the moment of truth: does TD-BMA really provide a better way of adapting λ than using any fixed λ across different problem domains?

To answer this question, we examine the SARSA(λ) and TD-BMA policy evaluation algorithms respectively defined in **Algorithms 2** and **3** within the same Generalized Policy Iteration framework of Algorithm 1 where $\epsilon = 0.05$ and we update the policy π after every trial. We evaluate SARSA(λ) and TD-BMA on two very different reinforcement learning domains: a variant of the RACETRACK domain (Barto et al., 1993) with the objective to reach the goal as fast as possible with fewest penalties, and a large state-space variant of the MAIL ROBOT domain with repeating multiple objectives to collect mail that randomly arrives at pickup points and deliver it to the dropoff points that the mail is addressed to.

Racetrack Formally, the state in RACETRACK is a combination of a car's coordinate position $\langle x, y \rangle$ and velocity $\langle x', y' \rangle$ in each coordinate direction. A car begins at one of the initial start states (chosen uniformly randomly) with velocity $\langle x', y' \rangle = \langle 0, 0 \rangle$, receives -1 for every action taken, except for 0 in the absorbing goal states. Actions $\langle x'', y'' \rangle$ available to the car are integer accelerations $\{-1, 0, 1\}$ in each coordinate direction. If the car hits a wall, then its velocity $\langle x', y' \rangle$ is reset to $\langle 0, 0 \rangle$. We use discount $\gamma = 1$ and terminate trials when $T > 200$ steps. Every action costs -1 to execute, except for grayed areas which incur penalty -100 and cause a reset to the initial state.

The three RACETRACK domains we use in this paper are shown in Figure 1(a,b,c).

Mail Robot Formally, the state in MAIL ROBOT is a combination of a robot's coordinate position $\langle x, y \rangle$, velocity $\langle x', y' \rangle$, the boolean status variables $\langle p_1, p_2, p_3, p_4 \rangle$ for each pickup point p_i indicating whether mail has arrived, holders on the robot for up to four pieces of mail $\langle m_1, m_2, m_3, m_4 \rangle$ where each piece of mail m_i (if it exists) is assigned to one of the four dropoff points d_j, i.e., $m_i \in \{d_1, d_2, d_3, d_4\}$. Locomotion in this problem is identical to the RACETRACK domain. Mail randomly arrives continuously with a mean interval of 30 time steps at each pickup point. The reward is zero for all actions except encountering a gray square (-100) or successfully delivering a piece of mail ($+4$). We use discount factor 0.9 and terminate trials after $T = 300$ time steps.

Figure 1(d) shows the topology of the mail robot domain that we evaluate in this paper. We note that taking into account all state variables, the robot could potentially explore millions of reachable states in this problem, making it very difficult for RL algorithms.

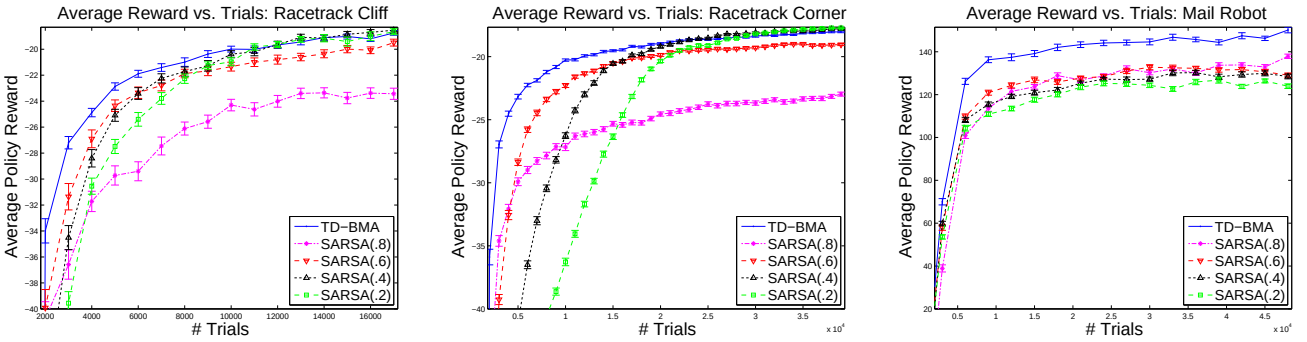


Figure 2. Average reward for policy vs. the # of learning trials on two configurations of the RACETRACK domain and the single configuration of the MAIL ROBOT domain. 95% confidence intervals obtained from averaging policy performance over 100 separate training runs are shown for each algorithm.

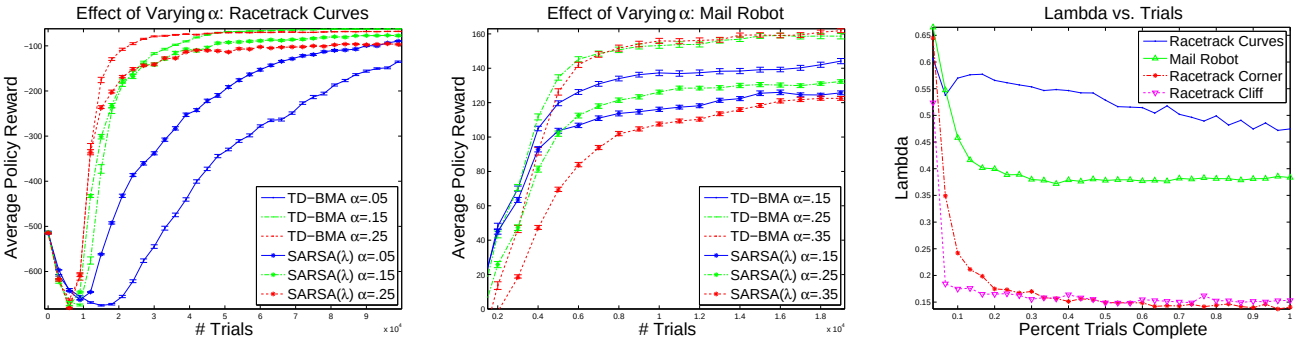


Figure 3. (left,center) Average reward for policy vs. the # of learning trials for various α settings of SARSA(λ) and TD-BMA on one RACETRACK and one MAIL ROBOT domain with 95% confidence intervals obtained from averaging policy performance over 100 separate training runs. (right) Average value of λ per trial vs. number of trials for various domains.

4.1. Results Summary

In Figure 2, we show the average policy reward for the policy vs. the number of learning trials with a limit on T as defined above for each respective problem and $\alpha = .15$ for all problems. We do not show $\lambda = 0$ (pure TD) or $\lambda = 1.0$ (pure Monte Carlo) as these were always outperformed by one of the other λ values for SARSA(λ). Most importantly, we note that the adaptive λ of TD-BMA *always* outperforms all other fixed values of λ in SARSA(λ) at *all* stages of learning.

In Figure 3 (left,center), we show the performance of TD-BMA and the best SARSA(λ) for different learning rates α. For MAIL ROBOT we see that TD-BMA for *any* α outperforms the best SARSA(λ) over *all* α. For RACETRACK Curves, we see that TD-BMA outperforms SARSA(λ) for the same α, except $\alpha = 0.05$. TD-BMA performs best with more aggressive learning rates since it reduces variance in the return estimates.

For Figure 3 (right), we see the average λ per trial vs. the percent trials completed (different domains had differing numbers of trials). We see that λ generally decreases over time as the value function stabilizes and TD-BMA learns it can trust the TD return. The exception is MAIL ROBOT where the state is techically partially observed since mail arrival is time-dependent, yet time is not encoded in the state; since the MC return provides useful time-dependent information not given by the time-independent TD return, TD-BMA settles on $\lambda \approx 0.5$ to maintain the best value estimate.

5. Related Work

While the mention of Bayesian reinforcement learning combined with a Gaussian value belief model may suggest a similarity to *Gaussian process temporal difference learning* (GPTD) (Engel et al., 2005), the two frameworks are very different in intent. GPTD learning makes use of the non-parametric Gaussian Process in its Bayesian model, which is very different in spirit and computation to the more standard independent Gaussian distribution per state-action pair used here. Most importantly, we note that to the best of our knowledge the GPTD approach has *not* been used to explicitly adapt the λ parameter in TD methods,

which is the purpose of our Bayesian model and derivation in TD-BMA.

In addition to GPTD, there are numerous other Bayesian RL approaches, e.g. (Dearden et al., 1998; 1999; Wang et al., 2005; Poupart et al., 2006). However, the focus of these other Bayesian RL approaches is typically to exploit the value distribution for the purpose of balancing the *exploration vs. exploitation tradeoff*. Again, our present work is concerned simply with automatically adapting the λ parameter in a Bayesian setting although we note that many of the ideas proposed in these papers that exploit the Bayesian value model for the exploration-exploitation tradeoff may be adapted to use the value model and efficient update algorithm presented here.

(Sutton & Singh, 1994) propose three alternate methods for adaptive weighting schemes in TD algorithms. However, their first two proposals are intended for the restricted subclass of acyclic state spaces only (the domains evaluated here were cyclic) and the third proposal is a model-based RL algorithm that requires maintaining an estimate of the model (n.b., we did not incur the time and memory expense of maintaining a model estimate of $T(s, a, s')$ for TD-BMA), which for a fixed policy could incur up to $O(|S|^2)$ time and space overhead not required by TD-BMA.

6. Concluding Remarks

This work proposed a Bayesian model averaging approach to adapting the λ parameter in temporal difference reinforcement learning methods. We began by deriving a novel Bayesian model averaging perspective of the temporal difference update intended to reduce variance in the average of the different n-step returns of TD-based SARSA(λ) and showed that while the Bayesian perspective suggests adapting λ in a data-dependent way, standard SARSA(λ) approaches simply fix this parameter and hand-tune it to a particular problem. Using our novel Bayesian perspective, we contributed the efficient Gaussian-based TD-BMA algorithm to compute a temporal difference Bayesian model average of returns that has exactly the *same time and space complexity* as SARSA(λ) while *automatically adapting* λ in light of all previously seen data. Empirically, TD-BMA performed as well and generally *much* better than SARSA(λ) for all fixed values of λ *without* requiring manual tuning of the λ parameter.

Important steps for future work are to determine whether TD-BMA can be extended to accommodate changing λ using an extension of the eligibility trace used in TD/SARSA(λ) (Sutton & Barto, 1998). This would enable an online update permitting within-trial TD-BMA learning. A side benefit of an eligibility trace-like approach to TD-BMA is that it provides one way to introduce function approximation techniques to TD-BMA similarly to the way eligibility traces facilitate function approximation in TD/SARSA(λ). Together these ideas hold promise for a novel class of data-dependent, Bayesian λ-adaptive TD/SARSA(λ) algorithms with better performance than any fixed λ.

Acknowledgements

We thank the anonymous reviewers and David Silver for their comments. NICTA is funded by the Australian Government's Backing Australia's Ability initiative, and the Australian Research Council's ICT Centre of Excellence program.

References

Barto, Andrew G., Bradtke, Steven J., and Singh, Satinder P. Learning to act using real-time dynamic programming. Technical Report UM-CS-1993-002, U. Mass. Amherst, 1993.

Dearden, Richard, Friedman, Nir, and Russell, Stuart J. Bayesian q-learning. In *AAAI/IAAI*, pp. 761–768, 1998.

Dearden, Richard, Friedman, Nir, and Andre, David. Model based bayesian exploration. In *UAI*, 1999.

Engel, Yaakov, Mannor, Shie, and Meir, Ron. Reinforcement learning with Gaussian processes. In *ICML-05*, pp. 201–208, New York, NY, USA, 2005.

Kakade, Sham. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, London, UK, March 2003.

Knuth, Donald E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Boston, 1998.

Poupart, Pascal, Vlassis, Nikos, Hoey, Jesse, and Regan, Kevin. An analytic solution to discrete bayesian reinforcement learning. In *ICML '06*, pp. 697–704, New York, NY, USA, 2006. ACM.

Puterman, Martin L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

Sutton, R. S. and Singh, S. P. On bias and step size in temporal-difference learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pp. 91–96, New Haven, CT, 1994.

Sutton, Richard and Barto, Andrew. *Reinforcement Learning*. MIT Press, 1998.

Wang, Tao, Lizotte, Daniel, Bowling, Michael, and Schuurmans, Dale. Bayesian sparse sampling for online reward optimization. In *ICML-05*, 2005.