Bayesian reinforcement learning in Markovian and non-Markovian tasks

Adnane Ez-zizi
School of Experimental Psychology
University of Bristol
Bristol, UK
Email: a.ez-zizi@bristol.ac.uk

Simon Farrell
Department of Psychology
University of Western Australia
Perth, Australia
Email: simon.farrell@uwa.edu.au

David Leslie
Department of Statistics
University of Lancaster
Lancaster, UK
Email: d.leslie@lancaster.ac.uk

***Abstract*—We present a Bayesian reinforcement learning model with a working memory module which can solve some non-Markovian decision processes. The model is tested, and compared against SARSA(λ), on a standard working-memory task from the psychology literature. Our method uses the Kalman temporal difference framework, and its extension to stochastic state transitions, to give posterior distributions over state-action values. This framework provides a natural mechanism for using reward information to update more than the current state-action pair, and thus negates the use of eligibility traces. Furthermore, the existence of full posterior distributions allows the use of Thompson sampling for action selection, which in turn removes the need to choose an appropriately parameterised action-selection method.**

I. Introduction

Reinforcement learning (RL) provides a set of techniques to solve the learning problem of an agent that aims to maximise its long-term reward in an unknown environment. The agent is assumed to select actions depending on the current state of the environment, and depending on the state of the environment these actions may produce reward feedback. The reward feedback can be used by the agent to learn which actions in which states will maximise long-run reward [1].

There exists now a well-established theory for solving Markovian RL tasks. In fact, there are several algorithms that are assured to converge to an optimal policy provided a certain set of conditions [2], such as Q-learning [3], Actor-Critic [4] and SARSA [5]. But, these algorithms cannot be directly applied to tasks that do not satisfy the Markov property. The standard generalisation of Markovian processes is the partially observable Markov decision process (POMDP). One approach that has been proposed to deal with POMDPs is supplementing the agent with a memory device that can store past observations, allowing the agent to disambiguate the current state based on past observations [6], [7].

Models based on this approach have been successfully applied to working memory learning tasks from the psychology literature such as the 12-AX [8], T-maze [9] or n-back [10]. These working memory tasks are non-Markovian in that reward is contingent on both the current environmental state, and the state that was present in the recent past. The models that have been applied to such tasks are often slow to learn and require setting the learning parameters carefully in order to reasonably expect convergence to an optimal policy. In particular, it is crucial to allow the model to explore enough but not excessively (i.e., balancing the trade-off between exploration and exploitation). Solving the exploration-exploitation trade-off is especially critical in these models, as the addition of a working memory to the agent results in a sizeable state-action policy space.

Thompson sampling has received increasing interest in recent years as a method for balancing exploration and exploitation in bandit problems [11]. It is self-tuning and automatically anneals exploration–exploitation in response to the level of information in the value estimates. However it requires the estimation of posterior distributions of action values instead of simply action values. Thus a Bayesian approach to RL is required.

Despite its elegance, the Bayesian approach has

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.91

IEEE computer society

only been used occasionally in modern RL. Among the few model-free Bayesian RL models that have been proposed in the literature are the SARSA variant of the Kalman temporal difference model (KTD-SARSA) and its extended version (XKTD-SARSA), both of which use online value function approximation along with Kalman filtering to estimate the state-action value function of a given policy [12]. A related algorithm is the SARSA-based Gaussian process temporal difference (GPSARSA) [13], which can be seen as a special case of the KTD-SARSA model in stationary environments. These models consider state-action values to be random processes and compute their posterior distribution given the observed rewards and transitions.

In this paper, we show that KTD-SARSA has the same form of updates as SARSA(λ), but with time- and state-dependent step-sizes that are governed by the learned properties of the system instead of arbitrary learning rates and eligibility traces. We then show how to implement KTD-SARSA and XKTD-SARSA alongside a working memory module. This parallels the approach of [8], [9], [14] in which SARSA or Actor-Critic methods have been enhanced with working memory to solve non-Markovian tasks. The advantages of introducing the Bayesian approach are seen in the existence of a native mechanism for sharing information across state-action pairs, replacing eligibility traces, and a native mechanism for balancing exploration and exploitation, removing the need to use an appropriately parametrised action-selection method. The next section presents the RL framework that the Bayesian models are based on. Section III introduces the two Bayesian RL models and compares them with the classical SARSA(λ) model. It also analyses the updating scheme of the KTD-SARSA model in the simple case of a two-armed bandit task, and shows that it is equivalent to a simple Bayesian model with unknown mean and known variance. Section IV presents a modification of the two Bayesian models capable of solving non-Markovian tasks, and describes some initial experimental results.

II. Background and notation

A. Reinforcement learning framework

We consider RL tasks with discrete time and immediate rewards, where an agent is assumed to interact with its environment with the objective of maximising its expected future discounted rewards. Let $z_t = (s_t, a_t)$ and r_t be respectively the state-action pair and reward at time t. We consider that there are p possible state-action pairs denoted by z^1,...,z^p. The future discounted return at time t of an action-selection policy π is defined by:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+t}$$

where γ is a discount factor which determines how much future rewards are discounted. The expectation of this return R_t^π, given a current initial state-action pair $z = (s, a)$ is known as state-action value or Q-value of policy π, and is denoted by $Q_t^\pi(z)$. The higher the Q-value of an action, the more rewarding it is in the long term. Thus, the agent aims to find a policy that maximise these Q-values, which implicitly requires the agent to effectively estimate them. There are several algorithms that can perform this estimation on-line, one of them is SARSA, which will be introduced next.

B. SARSA model

One of the most well-known model-free RL algorithms is SARSA [5]. It works by keeping track of point estimates of state-action values of the current policy, which are updated at each time step t according to

$$Q_{t+1}(z_t) = Q_t(z_t) + \alpha\delta_t \tag{1}$$

where α is a learning rate that scales the magnitude of the updates, and δ_t is an error that represents the discrepancy between the expected reward and the observed reward at time t. δ-errors are computed using the temporal difference equation:

$$\delta_t = r_t - (Q_t(z_t) - \gamma Q_t(z_{t+1})) \tag{2}$$

Denote the vector of all state-action pairs by $\boldsymbol{Q_t}$:

$$\boldsymbol{Q_t} = \begin{pmatrix} Q_t(z^1) \\ \vdots \\ Q_t(z^p) \end{pmatrix}$$

Thus, (2) can be written into a vectorial form as:

$$r_t = \boldsymbol{F_t}\boldsymbol{Q_t} + \delta_t \tag{3}$$

where, $\boldsymbol{F_t} = \begin{pmatrix} 0 & \dots & 0 & 1 & \dots & -\gamma & 0 & \dots & 0 \end{pmatrix}$, which has all its elements equal to 0 except the j_1^{th} and j_2^{th} element such that $z^{j_1} = z_t$ (hence the value 1) and $z^{j_2} = z_{t+1}$ (hence the value $-\gamma$). This vectorial formulation will be useful when we present the Bayesian RL models.

For now we presented SARSA in its basic form, but there is a more general form of the algorithm, referred to as SARSA(λ), which uses eligibility traces [15] to allow the learner to modify state-action values other than the values of the currently visited state and chosen action. This is particularly useful when dealing with non-Markovian or partially observable Markovian tasks [16].

Under SARSA(λ), all state-action values are updated using a modified version of (1):

$$Q_{t+1}(z) = Q_t(z) + \alpha\delta_t e_t(z) \tag{4}$$

where $e_t(z)$ is the eligibility trace of the state-action pair z, which decays over time by a decay parameter $\lambda \in (0, 1)$.

Finally, there remains the problem of how to select actions based on the estimated Q-values. There are several widespread methods for this problem, such as ϵ-greedy or softmax [1]. These basic methods ensure that actions with the highest values in a given state are selected more often. However, we have found that the performance of any of these methods is extremely susceptible to the tuning parameters of the action-selection method. In this paper, we address the action selection problem by using Bayesian approaches to RL.

III. Bayesian Reinforcement learning for Markovian tasks

The two Bayesian RL models that we use in this paper are based on the Kalman temporal difference (KTD) framework [12]. The general idea behind these models is to assume that the Q-values are random variables, and then use the Kalman filter framework [17] to compute posterior distributions over the Q-values after each newly observed reward. More specifically, we assume that the relation between $\boldsymbol{Q_t}$ and r_t can be represented by a Kalman filter type system, where $\boldsymbol{Q_t}$ is the state vector and r_t is the observed variable:

$$\begin{cases} \boldsymbol{Q_t} &= \boldsymbol{Q_{t-1}} + \boldsymbol{V_t} \\ r_t &= \boldsymbol{F_t Q_t} + \delta_t \end{cases} \tag{5}$$

such that we assume that the state vector $\boldsymbol{Q_t}$, and the noises $\boldsymbol{V_t}$ and δ_t satisfy the conditions of a standard Kalman filter system (see for example [17]). However, when the underlying MDP is stochastic, we loosen the white noise assumption for δ_t, and consider it to be coloured instead (the reasons for this choice are given below). We denote the covariance matrix of $\boldsymbol{V_t}$ and variance of δ_t respectively by $\boldsymbol{P_{V_t}}$ and P_{δ_t}—that is, we have $\boldsymbol{V_t} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{P_{V_t}})$ and $\delta_t \sim \mathcal{N}(0, P_{\delta_t})$.

Next, we will present the original version of the KTD-SARSA model, which assumes that the transitions of the underlying MDP are deterministic. Then, we will present the extended version of the KTD-SARSA (XKTD-SARSA), which can handle MDPs with stochastic transitions.

A. KTD-SARSA: A Bayesian RL model for deterministic MDPs

1) Updating scheme: In a deterministic MDP, the noise δ_t can be simply assumed to be white [12], in which case the Kalman estimates of the Q-values vector and its uncertainty can be derived in a straightforward manner and are given by:

$$\begin{cases} \hat{\boldsymbol{Q}}_{\boldsymbol{t}} &= \hat{\boldsymbol{Q}}_{\boldsymbol{t-1}} + \boldsymbol{K_t}(r_t - \boldsymbol{F_t}\hat{\boldsymbol{Q}}_{\boldsymbol{t-1}}) \\ \hat{\boldsymbol{P}}_{\boldsymbol{t}} &= (\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P_{V_t}}) \\ & \quad - \boldsymbol{K_t}[\boldsymbol{F_t}(\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P_{V_t}})\boldsymbol{F_t^T} + P_{\delta_t}]\boldsymbol{K_t^T} \end{cases} \tag{6}$$

where:

$$\hat{\boldsymbol{Q}}_{\boldsymbol{t}} = \mathbb{E}(\boldsymbol{Q_t}|r_0, \cdots, r_t)$$

$$\boldsymbol{K_t} = \frac{(\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P_{V_t}})F_t^T}{\boldsymbol{F_t}(\hat{\boldsymbol{P}}_{\boldsymbol{t-1}} + \boldsymbol{P_{V_t}})\boldsymbol{F_t^T} + P_{\delta_t}}$$

$$\hat{\boldsymbol{P}}_{\boldsymbol{t}} = \mathbb{E}[(\boldsymbol{Q_t} - \hat{\boldsymbol{Q}}_{\boldsymbol{t}})(\boldsymbol{Q_t} - \hat{\boldsymbol{Q}}_{\boldsymbol{t}})^T]$$

$\boldsymbol{K_t}$ is the Kalman gain and $\hat{\boldsymbol{P}}_{\boldsymbol{t}}$ corresponds to the variance associated with the estimation $\hat{\boldsymbol{Q}}_{\boldsymbol{t}}$ of $\boldsymbol{Q_t}$.

2) Action-selection via Thomson sampling: To finish the specification of our model, we must determine how to select actions based on the estimated Q-values and their underlying uncertainty. Given that we have the whole distributions of the state-action values, one sensible approach, known as Thompson sampling [18] is to sample action a with the probability it has the highest Q-value in the current state. That is, at time t, if the state is

$s_t = s$, we compute for each action a the probability $\mathbb{P}(argmax_{a'} Q(s,a') = a \,|\, \text{observations to date})$. The resulting probability distribution can then be used to sample an action. This can be efficiently implemented by simply sampling Q-values from their posteriors for each of the possible actions in the current state, and then choose the action with the highest sampled Q-value.

3) Comparison with classical SARSA(λ) model: We have seen that KTD-SARSA and SARSA(λ) are both constructed based on the temporal difference equation (2), which is used in both cases, to relate the observed rewards to the Q-values to estimate. Here, we show that they are almost identical even at the algorithmic level—that is, the individual state-action values are updated similarly in both models. First, denote $k_t(z,z') = Cov(Q_t(z), Q_t(z'))$ for any state-actions pairs z, z'. Also, to simplify expressions, let us also assume that $\boldsymbol{P_{V_t}} = 0$ (this is a valid assumption if we are dealing with stationary environments as the ones captured by the tasks that we are using in this paper). By developing the equations in (6), we can write the posterior mean of each state-action value and its covariance with any other state-action pair as follows:

$$\hat{Q}_{t+1}(z) = \hat{Q}_t(z) + \alpha_t(z)\hat{\delta}_t^{TD} \tag{7}$$

$$\hat{k}_{t+1}(z,z') = \hat{k}_t(z,z') - \alpha_t(z)[\hat{k}_t(z',z_t) - \gamma \hat{k}_t(z',z_{t+1})] \tag{8}$$

where:

$$\hat{Q}_t(z) = \mathbb{E}(Q_t(z)|r_0,\cdots,r_t)$$
$$\hat{k}_t(z,z') = \mathbb{E}[(Q_t(z) - \hat{Q}_t(z))(Q_t(z') - \hat{Q}_t(z'))]$$
$$= \mathbb{E}[Cov(Q_t(z), Q_t(z')|r_0,\cdots,r_t)]$$
$$\hat{\delta}_t^{TD} = r_t + \gamma \hat{Q}_t(z_{t+1}) - \hat{Q}_t(z_t)$$
$$\alpha_t(z) = (\hat{k}_t(z,z_t) - \gamma \hat{k}_t(z,z_{t+1}))/$$
$$[\hat{k}_t(z_t,z_t) - 2\gamma \hat{k}_t(z_t,z_{t+1}) + \gamma^2 \hat{k}_t(z_{t+1},z_{t+1}) + P_{\delta_t}]$$

The updating equation for the posterior mean of a state-action value looks similar to that obtained using a standard SARSA(λ) (see (4)), but with time- and state-dependent learning rate $\alpha_t(z)$ (here the learning rate and eligibility trace terms have been replaced with a variable learning rate). Here also, all state-action values could potentially be updated depending on their covariance (-estimate) with the current expected reward. In fact, $\alpha_t(z)$ can be re-written in terms of variances and covariances as:

$$\alpha_t(z) = \frac{\mathbb{E}[Cov\{Q_t(z), \overline{r}_t | r_0,\cdots,r_t\}]}{\mathbb{E}[Var(\overline{r}_t|r_0,\cdots,r_t)] + P_{\delta_t}} = \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}$$

where $\overline{r}_t = Q_t(z_t) - \gamma Q_t(z_{t+1})$ is the expected reward at time t, and such that we denoted $\mathbb{E}[Cov(X,Y|r_0,\cdots,r_t)]$ as $\widehat{Cov_t}(X,Y)$ (estimator of the covariance between X and Y at time t), and $\mathbb{E}[Var(X|r_0,\cdots,r_t)]$ as $\widehat{Var_t}(X)$.

The formula for $\alpha_t(z)$ means that the more positively related $Q_t(z)$ with $\overline{r}_t$ is, the bigger is the step-size $\alpha_t(z)$ used to update $Q_t(z)$. This is more true when both the estimated variance of $\overline{r}_t$ and true variance of δ_t are small (i.e., when the estimation of the expected reward is more precise and the system is less noisy). This is comparable to how eligibility trace works—the more recently visited a state-action pair, the more updated its corresponding Q-value. Similarly, here, the more correlated a state-action value to the current expected reward, the more it is updated.

One striking finding here is that both the current and coming state-action value are updated in two opposite directions to move the expected reward toward the observed one, rather than updating only the value of the current state-action pair. In fact, from the definition of the expected reward, $\overline{r}_t$ will be mostly positively correlated with $Q_t(z_t)$ (thus, $\alpha_t(z_t) > 0$) and negatively correlated with $Q_t(z_{t+1})$ (thus, $\alpha_t(z_{t+1}) < 0$). For any other state-action values $Q_t(z)$, the direction of the update, and hence the sign of $\alpha_t(z)$, depends on the difference between the current covariance of $Q_t(z)$ with $Q_t(z_t)$ and the discounted covariance with $Q_t(z_{t+1})$. The fact that the learning rate here can be negative is not common in RL models, and needs to be explored even in non-Bayesian settings.

Likewise, (8) can be re-written into a more interpretable way as:

$$\hat{k}_{t+1}(z,z') = \widehat{Cov_{t+1}}(Q_{t+1}(z), Q_{t+1}(z')) = \widehat{Cov_t}(Q_t(z), Q_t(z')) - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)\widehat{Cov_t}(Q_t(z'), \overline{r}_t)}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}$$

which, in the case of variances, becomes:

$$\widehat{Var_{t+1}}(Q_{t+1}(z)) = \widehat{Var_t}(Q_t(z)) - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t)^2}{\widehat{Var_t}(\overline{r}_t) + P_{\delta_t}}$$

This means that the uncertainty in a state-action value estimate decreases as its covariance with the expected reward increases, and that this is more true when the estimate of the expected reward is more precise or when the system noise is small. The interesting thing is that this decrease in uncertainty does not depend directly on the observed reward r_t as in the updating equation of the Q-value means.

In the following, we will apply KTD-SARSA to a simple Markovian RL task—the two-armed bandit task. The purpose of this exercise is not to propose a new method for solving the bandit problem, but to understand the behaviour of the KTD approach and how it relates to standard bandit algorithms.

4) Application to the two-armed bandit task: To investigate the relationship between this model and more conventional frameworks, we briefly reduce it to the two-armed bandit problem, where an agent simply has to choose between two arms and maximise its instantaneous reward. Each arm provides random rewards which are independent of the other arm. Because there is only one state, z refers now to the selected action. We also assume that the discount rate γ is equal to 0. Hence, the true value of each arm is equal to its reward mean. Finally, we consider that the δ-errors have the same variance P_δ ($P_{\delta_t} = P_\delta$, for all t), and that the prior covariance between the Q-values of the two arms is equal to 0 (because rewards in both arms are independent). In this setting, (7) and (8) are simplified to

$$\hat{Q}_{t+1}(z_t) = \hat{Q}_t(z_t) + \alpha_t(z_t)(r_t - \hat{Q}_{t+1}(z_t)) \quad (9)$$

$$\hat{k}_{t+1}(z_t, z_t) = \hat{k}_t(z_t, z_t) - \alpha_t(z_t)\hat{k}_t(z_t, z_t) \quad (10)$$

where

$$\alpha_t(z_t) = \frac{\hat{k}_t(z_t, z_t)}{\hat{k}_t(z_t, z_t) + P_\delta}.$$

Here, only the mean and variance of the Q-value of the selected arm are updated (because the learning rate of the non-selected arm remains always equal to 0). These updating equations are similar to those obtained using a simple Bayesian updating model where reward $r(z)$ in each arm z is considered to be Gaussian with unknown mean $Q(z)$ and known variance P_δ, that is, $r(z)|Q(z) \sim \mathcal{N}(Q(z), P_\delta)$ [19].

We see both that the KTD framework results in conventional temporal differencing in the bandit problem, and that the standard temporal difference approach in bandit problems is in fact an efficient implementation of full Bayesian inference in a simple model.

B. XKTD-SARSA: An extension of KTD-SARSA for stochastic MDPs

The previous model is only suitable for tasks where the transitions of the underlying MDP are deterministic. In the stochastic case, the δ_t's can no longer be assumed independent. In fact, Geist and Pietquin [12] showed that ignoring this would result in biased Kalman estimates. To deal with this, they proposed to use a coloured noise as was previously suggested by Engel et al. [13]. Here, we present their extended model, where again we adapt it to be used without parametrised state-action value functions.

1) Updating scheme: In the extended version of the KTD-SARSA model, the δ-noise is assumed to be a moving average of two white noises (for a motivation for this choice, see [12] or [13]):

$$\delta_t = -\gamma u_t + u_{t-1} \quad \text{with} \quad u_t \sim \mathcal{N}(0, \sigma^2)$$

Under this new assumption, the Kalman estimates cannot be obtained directly as was done in the KTD case. A solution for extending Kalman filter to moving average noises is given in [20]. The basic idea of their solution is to transform the moving average noise into a vectorial autoregressive noise by adding an auxiliary random process w_t that stores the occurrences of u_t at each time step. More specifically, with $w_t = u_t$, we can write:

$$\begin{pmatrix} w_t \\ \delta_t \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} w_{t-1} \\ \delta_{t-1} \end{pmatrix} + \begin{pmatrix} 1 \\ -\gamma \end{pmatrix} u_t$$

The Kalman filter problem can then be easily solved using the resulting autoregressive noise, by rewriting (5) as:

$$\begin{cases} \boldsymbol{X_t} &= L\boldsymbol{X_{t-1}} + \boldsymbol{V'_t} \\ r_t &= \boldsymbol{F'_t}\boldsymbol{X_t} \end{cases} \quad (11)$$

where:

$$\boldsymbol{X_t} = \begin{pmatrix} \boldsymbol{Q_t^T} & w_t & \delta_t \end{pmatrix}^T$$
$$\boldsymbol{V'_t} = \begin{pmatrix} \boldsymbol{V_t^T} & u_t & -\gamma u_t \end{pmatrix}^T, \boldsymbol{F'_t} = \begin{pmatrix} \boldsymbol{F_t} & 0 & 1 \end{pmatrix}$$
$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I_p} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0^T} & 0 & 0 \\ \boldsymbol{0^T} & 1 & 0 \end{pmatrix}, \boldsymbol{P_{V_t}} = \begin{pmatrix} \boldsymbol{P_{V_t}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0^T} & \sigma^2 & -\gamma\sigma^2 \\ \boldsymbol{0^T} & -\gamma\sigma^2 & \gamma^2\sigma^2 \end{pmatrix}$$

The posterior mean and covariance of the augmented vector $\boldsymbol{X_t}$ are given by:

$$\begin{cases} \boldsymbol{\hat{X}_t} &= \boldsymbol{L\hat{X}_{t-1}} + \boldsymbol{K'_t}(r_t - \boldsymbol{F'_t\hat{X}_{t-1}}) \\ \boldsymbol{\hat{P}'_t} &= (\boldsymbol{L\hat{P}'_{t-1}L^T} + \boldsymbol{P'_{V_t}}) \\ & \quad - \boldsymbol{K'_t}[\boldsymbol{F'_t}(\boldsymbol{L\hat{P}'_{t-1}L^T} + \boldsymbol{P'_{V_t}})\boldsymbol{F'^T_t}]\boldsymbol{K'^T_t} \end{cases} \tag{12}$$

where $\boldsymbol{\hat{X}_t} = \begin{pmatrix} \boldsymbol{\hat{Q}_t^T} & \hat{w}_t & \hat{\delta}_t \end{pmatrix}^T$ and $\boldsymbol{K'_t}$ is the Kalman gain of the new system. The estimates of the mean and covariance matrix of the vector of state-action values $\boldsymbol{Q_t}$ can be obtained from $\boldsymbol{\hat{X}_t}$ and $\boldsymbol{\hat{P}'_t}$ by respectively extracting the first p elements of $\boldsymbol{\hat{X}_t}$ and the first $p \times p$ matrix block of $\boldsymbol{\hat{P}'_t}$. Notice also that δ_t and w_t are now part of the state to estimate.

2) Comparison with KTD-SARSA and SARSA(λ): Let us extract the scalar updating equations for the posterior mean and covariances of the Q-values from (12) as we did with KTD-SARSA:

$$\hat{Q}_{t+1}(z) = \hat{Q}_t(z) + \alpha_t(z)(\hat{\delta}_t^{TD} - \hat{\delta}_t) \tag{13}$$
$$\hat{k}_{t+1}(z, z') = \hat{k}_t(z, z') - \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t + w_t)\widehat{Cov_t}(Q_t(z'), \overline{r}_t + w_t)}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2} \tag{14}$$

where:

$$\hat{\delta}_{t+1} = \hat{\delta}_t + \alpha_t(\delta)(\hat{\delta}_t^{TD} - \hat{\delta}_t)$$
$$\alpha_t(z) = \frac{\widehat{Cov_t}(Q_t(z), \overline{r}_t + w_t)}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2}$$
$$\alpha_t(\delta) = \frac{\widehat{Cov_t}(w_t, \overline{r}_t + w_t) + \gamma^2\sigma^2}{\widehat{Var_t}(\overline{r}_t + w_t) + \gamma^2\sigma^2}$$

The updating equations (13) and (14) for computing the posterior mean and covariance of the Q-values are not as different from those obtained in the deterministic case (see (7) and (8)) as it might look like. In fact, we expect that the posterior mean $\hat{\delta}_t$ would converge toward 0 (the true mean of δ_t). Also, given that w_t is equal to the white noise u_t, the posterior variance $\widehat{Var_t}(w_t)$ and posterior covariance $\widehat{Cov_t}(w_t, Q_t(z))$ are expected to converge respectively towards σ^2 (the true variance of u_t) and 0 (the true covariance between $Q_t(z)$ and u_t, because u_t is *a priori* independent from all state-action values). Hence, (13) and (14) should become very similar to (7) and (8) in the long run, and the differences between XKTD-SARSA and KTD-SARSA are in the early phases of learning.

IV. Bayesian Reinforcement learning for non Markovian tasks

A. KTD-WM and XKTD-WM models

To allow KTD-SARSA and XKTD-SARSA to be applied effectively to non-Markovian tasks, we follow [8] and supplement the agent with a working memory (WM). The WM allows the agent to consider observations from the past, along with the current one, when selecting actions. We assume that it has C WM slots, and that each slot can store one of the previously encountered observations from the environment. The agent can modify the content of WM slots at each time step via memory updating actions called gating actions. The agent can choose to store the new observation in one of the slots while maintaining the content of the other ones; or maintain the content of all memory slots. These actions need to be chosen by the agent along with the motor actions (i.e, the actions that act on the environment to potentially produce a reward).

The inclusion of the additional WM state does not change the main dynamics of KTD-SARSA and XKTD-SARSA—that is, the updating equations in (6) and (12) remain the same. There are only two slight modifications: the first one consists of the current state s_t now being composed of the current observation o_t and the current memory content $m_t = (m_t^1, \ldots, m_t^C)$, where m_t^j is the observation held in the j^{th} WM slot at time t. The second modification is that action a_t now includes a motor action (a_t^M) and a gating action (a_t^G). Hence, each state-action pair z can be written $z = (m, o, a^M, a^G)$, where m, o, a^M, and a^G are respectively the current memory content, observation, motor action and gating action. The steps followed by the agent in the augmented KTD-SARSA (resp. XKTD-SARSA) model, which we refer to as KTD-WM (resp. XKTD-WM) can be summarised as follows:

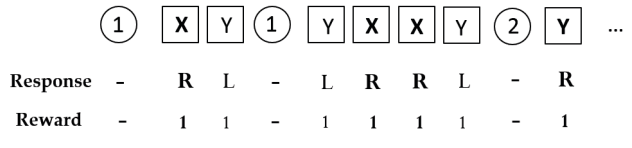

Figure 1. A sequence of observations from the 12-XY task. The correct response and the associated reward is given bellow each observation. Letters X and Y in bold are used to highlight when action 'R' is required.

1) Choose the prior mean and covariance of the Q-value vector.
2) Observe current state $s_t = (m_t, o_t)$.
3) Select a motor action a^M and a gating action a^G based on state-action values Q(s,.): $(a^M, a^G) \leftarrow Thomson(Q, s)$.
4) Observe reward r_t and go to next state s_{t+1}.
5) Update the mean and covariance matrix of the Q-values using (6) (resp., using (12) for XKTD-WM).
6) Repeat steps 2-5 until termination.

B. Example: 12-XY task

To evaluate the Bayesian WM-based RL models presented here, we used a very basic version of the 12-AX task [10], where we kept only the observations 1, 2, X and Y (Figure 1). In this task, digits are presented (1 vs 2), interspersed with letters (X or Y). The task is structured such that responses to the letters are rewarded depending on the last digit seen. The rules of the new task to be learned are as follows: if the last digit was 1 (resp. 2) and the current letter is X (resp. Y), then choose the 'R' response. In all other cases, choose 'L'. Reward is either 1 for correct responses or 0 for incorrect responses. (No reward was given in response to actions when the current observation was a digit). The task is structured such that a model without access to information from previous observations would perform at chance level, and hence we can easily assess the usefulness of our models.

We simulated the 12-XY using KTD-WM, XKTD-WM and SARSA(λ) with WM as presented in [14]. We simulated each model for 50 times, each run consisting of $N = 10000$ trials, and tracked the learning curves as shown in Figures 2. For all three models, we set the number of WM slots to $C = 1$ and the discount factor to $\gamma = 0.5$. The other parameters were chosen using an extensive grid search to find the best parameters for each model:

- for KTD-WM, the prior mean $\hat{\boldsymbol{Q}}_{\mathbf{0}} = \mathbf{0}$, the prior covariance $\hat{\boldsymbol{P}}_{\mathbf{0}} = 10 I_{64}$ (where I_{64} is the identity matrix of dimension 64—the number of possible state-action pairs), δ-errors variance $P_{\delta_t} = 1$ and the covariance of the state noise $\boldsymbol{P}_{\boldsymbol{V_t}} = \mathbf{0}$.
- for XKTD-WM, the prior mean $\hat{\boldsymbol{X}}_{\mathbf{0}} = \mathbf{0}$, the prior covariance $\hat{\boldsymbol{P}}'_{\mathbf{0}} = 10 I_{66}$, the variance of the white noise u_t, $\sigma^2 = 0.2$, which implies that $P_{\delta_t} = (1 + \gamma^2)\sigma^2 = 0.25$, and $\boldsymbol{P}_{\boldsymbol{V_t}} = 10^{-6} I_{64}$ (we did not use a null matrix to avoid numerical stability issues).
- for SARSA(λ)-WM, the learning rate $\alpha = 0.05$, the decay parameter $\lambda = 0.9$, and we used Boltzmann action selection [1] with temperature parameter $T = 0.2$.

Results show that XKTD-WM outperformed SARSA(λ)-WM both in terms of the final average accuracy and the time required to reach their top performance. For example, XKTD-WM could learn an optimal policy in all runs, resulting in an almost maximal average accuracy, whereas SARSA(λ)-WM sometimes failed to find an optimal policy or converged to a suboptimal policy, resulting in an average accuracy of about 92.5% over in the last block. KTD-WM could not solve the 12-XY in any run, but performed substantially better than chance, which is the level of accuracy expected to be achieved with a model without WM. This low performance is not very surprising given that the transitions are stochastic in the 12-XY, and hence XKTD-WM is more suitable than KTD-WM in this particular task.

V. Discussion and conclusion

We have investigated the Kalman temporal difference (KTD) approach to Bayesian reinforcement learning, and its extension XKTD. We have shown how they relate to the standard SARSA(λ) algorithm. We noted that the full Bayesian framework results in a natural replacement for eligibility traces, and allows the use of Thompson sampling, which also removes the need to parametrise the action-selection mechanism. We then tested the methods in a non-Markovian decision-problem, enhancing each method with a working memory module as in [8],

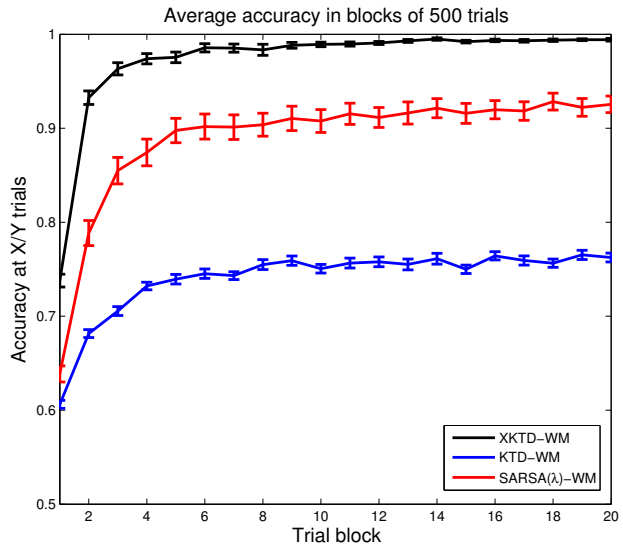

Figure 2. Learning curves for KTD-WM, XKTD-WM and SARSA(λ)-WM on the 12-XY task. Accuracy was averaged over blocks of 500 trials and over the 50 simulation runs. Error bars represent standard errors.

[9], [14]. Our preliminary results demonstrate the potential of the approach.

Perhaps one of the most interesting findings in this paper is the novel updating scheme used by these Bayesian models: Q-values of both the current and future state-action pair can potentially be updated so as to bring the expected reward closer to the observed one. This is different from the updating scheme in SARSA and other temporal difference-based methods, where the value of the next state-action pair is not updated until it is visited. This idea could be easily implemented in the non-Bayesian framework, for example, by assigning a positive learning rate for updating the value of the current state-action pair and a negative one for updating the value of the coming pair.

Simulations results showed that the proposed XKTD-WM model outperformed SARSA(λ) with WM on the 12-XY—a moderately challenging WM learning task. Future research should compare it with other standard WM-based RL models like the Actor-Critic on more challenging non-Markovian tasks. It also remains to be seen whether XKTD-SARSA with Thompson sampling achieves comparably performance to a carefully-tuned SARSA(λ) algorithm in a wider range of problems.

REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998.

[2] C. J. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[3] C. J. Watkins, "Learning from delayed rewards," Ph.D. dissertation, 1989.

[4] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-13, no. 5, pp. 834–846, Sept 1983.

[5] R. S. Sutton, "TD models: Modeling the world at a mixture of time scales," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 531–539.

[6] M. L. Littman, "Memoryless policies : theoretical limitations and practical results," *Proceedings of the third Conference on Simulation of Adaptive Behavior*, pp. 238–245, 1994.

[7] L. Peshkin, N. Meuleau, and L. Kaelbling, "Learning policies with external memory," *Machine Learning: Proceedings of the Sixteenth International Conference*, pp. 307–314, 1999.

[8] M. T. Todd, Y. Niv, and J. D. Cohen, "Learning to use working memory in partially observable environments through dopaminergic reinforcement," *Neural Inform Process Syst*, vol. 21, pp. 1689–1696, 2009.

[9] E. A. Zilli and M. E. Hasselmo, "Modeling the role of working memory and episodic memory in behavioral tasks," *Hippocampus*, vol. 18, no. 2, pp. 193–209, 2008.

[10] R. C. O'Reilly and M. J. Frank, "Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia," *Neural Computation*, vol. 18, pp. 283–328, 2006.

[11] B. C. May, N. Korda, A. Lee, and D. S. Leslie, "Optimistic bayesian sampling in contextual-bandit problems," *J. Mach. Learn. Res.*, vol. 13, pp. 2069–2106, Jun. 2012.

[12] M. Geist and O. Pietquin, "Kalman temporal differences," *Journal of Artificial Intelligence Research*, vol. 39, pp. 483–532, 2010.

[13] Y. Engel, S. Mannor, and R. Meir, "Reinforcement learning with Gaussian processes," *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208, 2005.

[14] K. Lloyd, N. Becker, M. W. Jones, and R. Bogacz, "Learning to use working memory: a reinforcement learning gating model of rule acquisition in rats," *Frontiers in Computational Neuroscience*, vol. 6, no. 87, 2012.

[15] R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. dissertation, 1984.

[16] J. Loch and S. Singh, "Using eligibility traces to find the best memoryless policy in partially observable markov decision processes," in *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, 1998, pp. 323–331.

[17] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Dept. Comput. Sci.,Univ. North Carolina, Chapel Hill, Tech. Rep., 2000.

[18] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. pp. 285–294, 1933.

[19] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, *Bayesian Data Analysis, 3rd Edition*, ser. Chapman & Hall/CRC Texts in Statistical Science, 2013.

[20] M. Geist and O. Pietquin, "Kalman filtering & colored noises: the (autoregressive) moving-average case," *Workshop Proceedings of ICMLA 2011*, no. 9, 2011.