Eligibility Traces through Colored Noises

Matthieu Geist and Olivier Pietquin

Abstract— **The Gaussian Process Temporal Differences (GPTD) framework initiated statistical modeling of value function approximation. It was followed by the close Kalman Temporal Differences (KTD) approach. Both methods share the same drawback: they provide biased estimates of the value function when transitions of the system to be controlled are stochastic. A colored noise model has been introduced to cope with this problem in the GPTD framework, which actually leads to a Monte-Carlo estimate of the value function. In this paper, we generalize this colored noise model using ideas close to eligibility traces and apply it to the KTD framework. This allows removing the bias when the so-called eligibility factor is set to one, and decreasing it when this factor is strictly between zero and one. The proposed algorithm is experimented on the simple Boyan chain in order to study the effect of the eligibility factor. As KTD generalizes GPTD in the sense that it allows taking into account nonlinear parameterizations, we also propose an experiment combining the new algorithm with a neural network.**

Index Terms— **reinforcement learning, value function approximation, statistical modeling, colored noise, neural networks.**

I. Introduction

REINFORCEMENT learning (RL) [1] is generally considered as the machine learning answer to the control problem. In this paradigm, an agent learns to control optimally a dynamic system through interactions. At each time step i, the dynamic system is in a given state s_i and the agent applies an action a_i to it. According to its own dynamics, the system transits to a new state s_{i+1}, and a reward r_i is given to the agent. The objective is to learn a control policy which maximizes the expected cumulative reward. An important topic of RL is to estimate the value function which models the expected cumulative reward of a given policy, particularly when the state space is too big for a tabular representation, which implies to use some function approximator. However, it is more than a regression problem, as the value function is not directly observable.

In recent years, approaches based on statistical modelling of value function approximation have been introduced. The Gaussian Process Temporal Differences (GPTD) framework [2] consists in modeling the value function as a Gaussian Process and adopting a statistical generative model linking rewards to values through the (linear and sampled) Bellman evaluation equation plus a random observation noise. Related algorithms are derived by performing Bayesian inference. An interesting aspect of this framework is that it allows a non-parametric representation of the value function: using a linear dependency argument (in the feature space linked to a kernel representation), a kernel-based linear parameterization can be constructed online. Notice that a parametric representation can also be directly considered.

More recently, the related Kalman Temporal Differences (KTD) framework [3] has been introduced. A parametric representation is adopted, parameters are modeled as random variables and value function approximation is stated as a filtering problem. This approach allows handling non-stationarities through the specification of some evolution model for parameters. This is of interest for non-stationary systems and for generalized policy iteration schemes (in this setting, each policy improvement induces changes in the associated value function to be estimated, which is consequently non-stationary). It also handles nonlinearities thanks to a derivative-free approximation scheme. Consequently, nonlinear function approximator such as neural networks can be envisioned, as well as value-iteration-like algorithms.

Both frameworks are actually quite close. With a linear parameterization and if no evolution model is considered for KTD, they even reduce to the same algorithms. However significant differences exist: GPTD allows a non-parametric (but linear) kernel-based representation, whereas KTD allows handling non-stationarities and non-linearities. These approaches present the same drawback (shared by other methods such as residual algorithms [4]): they provide biased estimates of the value function when transitions of the system are stochastic. This is due to a common white observation noise assumption (a statistical approach suggests that errors are modeled as noises).

To handle this problem, a colored noise model is introduced in [5], to be presented in Section II-B. The resulting so-called Monte-Carlo GPTD (MC-GPTD) framework provides unbiased estimates of the value function. Moreover, it can be linked to Monte Carlo, in the sense that it performs actually a regression linking values to Monte-Carlo estimates of the discounted return. In this paper, we propose to generalize this noise model by using a principle close to eligibility traces [1]. We also extend the KTD framework with this new noise model. It can be used to extend GPTD too, however it is not done here due to a lack of place (it would take a few pages of algebraic manipulations; nevertheless, hints to do so are provided).

II. Background

This paper is placed in the framework of Markov decision processes (MDP). An MDP is a tuple $\{S, A, P, R, \gamma\}$, where S is the state space, A the action space, $P : s, a \in S \times A \rightarrow p(.|s, a) \in \mathcal{P}(S)$ a family of transition probabilities, $R : S \times A \times S \rightarrow \mathbb{R}$ the bounded reward function, and γ the discount factor. A policy π associates to each state a probability over actions, $\pi : s \in S \rightarrow \pi(.|s) \in \mathcal{P}(A)$. The value function of a

Matthieu Geist and Olivier Pietquin are with the IMS Research Group, Supélec, Metz, France (email: {matthieu.geist, olivier.pietquin}@supelec.fr).

given policy is defined as $V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi]$ where r_i is the immediate reward observed at time step i, and the expectation is done over all possible trajectories starting in s given the system dynamics and the followed policy. The Q-function allows a supplementary degree of freedom for the first action and is defined as $Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. RL aims at finding (through interactions) the policy π^* which maximises the value function for every state: $\pi^* = \text{argmax}_\pi(V^\pi)$. Two schemes among others can lead to the optimal policy. First, *policy iteration* implies learning the value function of a given policy and then improving the policy, the new one being greedy respectively to the learned value function. It requires solving the *Bellman evaluation equation*, which is given here for value and Q-functions: $V^\pi(s) = E_{s',a|\pi,s}[R(s,a,s') + \gamma V^\pi(s')]$, $\forall s$ and $Q^\pi(s,a) = E_{s',a'|\pi,s,a}[R(s,a,s') + \gamma Q^\pi(s',a')]$, $\forall s, a$. The second scheme, *value iteration*, aims directly at finding the optimal Q-function and to derive the optimal policy from it. It requires solving the *Bellman optimality equation*: $Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b \in A} Q^*(s',b)]$, $\forall s, a$. An important problem in RL is to find an approximate solution of one of the Bellman equations when state and/or action spaces are too large for classic dynamic programming or RL algorithms to hold. In the rest of this paper, we focus on the value function evaluation problem for a fixed policy. Consequently, corresponding subscripts are omitted. We assume that a parametric representation $\hat{V}_\theta(s)$ is adopted (*e.g.*, radial-basis-functions network, multilayered percpetron, *etc.*), with θ being the set of parameters. This structure is fixed beforehand, and we aim at learning these parameters through actual interactions with the system and such that $\hat{V}_\theta$ is a good approximation of the true value function $V^\pi(s)$. Extension to the Q-function evaluation is straightforward, but the Q-function direct optimization (that is solving the Bellman optimality equation) is not considered in this paper because of its off-policy aspect, which is further discussed in Section VI.

A. Kalman Temporal Differences

Originally, the Kalman filter paradigm [6] aims at online tracking the hidden state (modeled as a random variable) of a non-stationary dynamic system through indirect observations of this state. The idea behind KTD is to express value function approximation as a filtering problem: the parameters are the hidden state to be tracked, the observation being the reward linked to the parameters through a Bellman equation.

The KTD framework is briefly described here, further theoretical details are provided in [3]. As we focus on the value function evaluation problem, the policy is fixed, and corresponding subscripts are thus omitted. A statistical point of view is adopted, and the problem at hand is stated in a *state-space formulation* (that is the value function approximation problem is cast into a filtering paradigm):

$$\begin{cases} \theta_i = \theta_{i-1} + v_i & \text{(evolution)} \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) + n_i & \text{(observation)} \end{cases} \quad (1)$$

Using the vocabulary of Kalman filtering, the first equation is the evolution equation, it specifies that the parameter vector follows a random walk whose expectation corresponds to the optimal estimation of the value function. This is not an update equation. By definition, the evolution (or process) noise v_i is centered, white, independent and of variance matrix P_{v_i}. The second equation is the observation equation, it links the observed transition (and immediate reward) to the parameterized value function through the sampled Bellman evaluation equation. The observation noise n_i is supposed centered, white, independent and of variance P_{n_i}. This white noise assumption is necessary for KTD derivation. However it only holds for deterministic transitions, in which case the noise models the fact that the solution of the Bellman equation does not necessarily exist in the functional space spanned by the set of parameters. It cannot be white with a stochastic MDP because it would include some transition effects.

This state-space formulation states that the rewards are generated according to the observation equation. This equation is itself driven by the hidden and random parameters defining a family of functions whose expectation is the optimal approximated value function. It is a statistical model of the (sampled) Bellman equation. Moreover, if the problem at hand is formally stationary (that is stationary control policy and MDP transitions), there is no process noise. Nevertheless, the random walk model allows handling non-stationarity without harming the stationary case. An artificial process noise can also help avoiding local minima.

KTD aims at minimizing the mean squared error of the parameters estimates conditioned on past observed rewards: $J(\hat{\theta}_i) = E[\|\theta_i - \hat{\theta}_i\|^2 | r_{1:i}]$ with $r_{1:i} = r_1, \ldots, r_i$. Generally speaking, the corresponding estimator is $E[\theta_i | r_{1:i}]$, the conditional expectation of parameters given rewards. However, except in specific cases (*e.g.*, linear and Gaussian), no analytical solution can be found. Instead, the aim of the Kalman paradigm is to find the best *linear* estimator. Minimizing such a cost gives rise to the general KTD approach, which is summarized in Algorithm 1, and for which the following notations are used: $\hat{\theta}_{i|j} = E[\theta_i | r_{1:j}]$ is the prediction of the estimate according to past observed rewards $r_{1:j}$ and $P_{i|j} = \text{cov}(\theta_i - \hat{\theta}_{i|j} | r_{1:j})$ is the associated variance.

KTD breaks down into three steps. First, the prediction step consists in updating the parameters mean and covariance according to the evolution equation given the estimates at time $i-1$. Second some statistics of interest are computed. They are necessary for the correction step (the third one), which consists in correcting first and second order moments of the predicted parameter vector according to the Kalman gain K_i, the predicted reward $\hat{r}_{i|i-1}$ and the observed reward r_i. The difference between the observed and predicted reward, called innovation in the Kalman filtering paradigm, is a form of temporal difference (TD) error. Notice that being online KTD must be initialized with priors for the parameters mean and covariance. The unscented transform [7] can be used to derive practical algorithms [3]. This is done hereafter

in the framework of the KTD extension.

Notice that by setting $v_i = 0$ and assuming a linear parameterization, KTD provides the same algorithm as (parametric) GPTD. Compared to GPTD, KTD allows defining an evolution model for the parameter vector and handling a nonlinear parameterization.

Algorithm 1: General KTD algorithm

Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;

for $i \leftarrow 1, 2, \dots$ **do**

Observe transition (s_i, s_{i+1}) and reward r_i ;

Prediction step;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;

Compute statistics of interest;

$\hat{r}_{i|i-1} = E[\hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) | r_{1:i-1}]$;

$P_{\theta r_i} = E[(\theta_i - \hat{\theta}_{i|i-1})(\hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) - \hat{r}_{i|i-1}) | r_{1:i-1}]$;

$P_{r_i} = E[(\hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) - \hat{r}_{i|i-1})^2 | r_{1:i-1}] + P_{n_i}$;

Correction step;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i (r_i - \hat{r}_{i|i-1})$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

B. Engel's Colored Noise Model

The white observation noise assumption leads to a biased value function approximator if transitions are stochastic. Actually, it can be shown that KTD minimizes a (regularized) square Bellman residual [8], which classically leads to biased estimates [9]. The same problem arises in the GPTD framework which handle it with a colored noise model [5].

The value function can be defined as the expectation (over all possible trajectories) of the following discounted return random process:

$$D(s) = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i, s_{i+1}) | s_0 = s \tag{2}$$

with $a_i \sim \pi(.|s_i)$ and $s_{i+1} \sim p(.|s_i, a_i)$. This equation naturally leads to a Bellman-like equation:

$$D(s) = R(s, a, s') + \gamma D(s') \tag{3}$$

with $a \sim \pi(.|s)$ and $s' \sim p(.|s, a)$. By definition, the value function is the mean of this random process: $V(s) = E[D(s)]$. Let define $\Delta V(s)$, the related random zero-mean residual: $\Delta V(s) = D(s) - E[D(s)] = D(s) - V(s)$. The random process $D(s)$ can therefore be broken down into the deterministic value function plus a random zero-mean residual:

$$D(s) = V(s) + \Delta V(s) \tag{4}$$

Manipulating Equations (3) and (4), the reward can be expressed as a function of the value plus a noise:

$$R(s, a, s') = V(s) - \gamma V(s') + N(s, s') \tag{5}$$

$$\text{with } N(s, s') = \Delta V(s) - \gamma \Delta V(s') \tag{6}$$

In [5] residuals are assumed to be independent, which leads to a colored noise model. More precisely, it is a moving-average (MA) noise (as it is the sum of two white noises). Recall that the resulting MC-GPTD algorithm can be linked to Monte Carlo, in the sense that it performs actually a regression linking values to Monte-Carlo samples of the discounted return [5].

III. A New Colored Noise Model

GPTD and KTD perform local updates, in the sense that they use only the current temporal difference error. MC-GPTD performs global updates, in the sense that it (implicitly) uses the whole trajectory. The same holds for more classic TD algorithms, and eligibility traces [1] provide a theoretical bridge between local and global updates. Basically, TD updates the value function using a 1-step prediction and Monte-Carlo updates it using the whole trajectory. On the other hand, TD(λ), $\lambda \in [0, 1]$ being the eligibility factor, updates the value function using a λ-dependent mean of k-step predictions. With $\lambda = 0$ it reduces to TD and with $\lambda = 1$ it reduces to Monte-Carlo. We use the same principle to extend Engel's noise model.

A. The noise model

Engel's colored noise model is based on a 1-step prediction:

$$R(s, s') = V(s) - \gamma V(s') + N^{(1)}(s, s')$$

$$\text{with } N^{(1)}(s, s') = \Delta V(s) - \gamma \Delta V(s') \tag{7}$$

A k-step prediction can be similarly considered:

$$\sum_{i=0}^{k-1} \gamma^i R(s^{(i)}, s^{(i+1)}) = V(s) - \gamma^k V(s^{(k)}) + N^{(k)}(s, s^{(k)})$$

$$\text{with } N^{(k)}(s, s^{(k)}) = \Delta V(s) - \gamma^k \Delta V(s^{(k)}) \tag{8}$$

We define the new colored noise N^λ as the weighted mean of colored noises related to k-step predictions, with $\lambda \in [0, 1]$ the eligibility factor:

$$N^\lambda = \lambda \sum_{k=1}^{\infty} (1-\lambda)^{k-1} N^{(k)} \tag{9}$$

$$= \Delta V(s) - \gamma \lambda \sum_{k=1}^{\infty} \gamma^{k-1} (1-\lambda)^{k-1} \Delta V(s^{(k)})$$

The reader familiar with eligibility traces has probably noticed that in the above equation the role of λ and $1 - \lambda$ is flipped from standard works on TD methods. This is due to the fact that the noise terms have this quantity in the reversed role compared to the classically considered TD errors.

By definition, residuals are centered, and as in [5] we assume them to be independent. They can thus be modeled

as centered white noises u_i of theoretical variance $\sigma_i^2 = \mathrm{var}(D(s_{i+1}))$. This leads to the following noise model:

$$n_i^\lambda = u_i - \gamma\lambda \sum_{k=1}^{\infty} \gamma^{k-1}(1-\lambda)^{k-1} u_{i+k} \tag{10}$$

With $\lambda = 0$, n_i^λ reduces to a white observation noise: $n_i^{\lambda=0} = u_i$. This is the noise used by GPTD and KTD. With $\lambda = 1$, n_i^λ satisfies $n_i^{\lambda=1} = u_i - \gamma u_{i+1}$ which is the noise used by MC-GPTD. Therefore it generalizes previous noise models.

B. Handling the Noise in GPTD

GPTD can be extended using this new colored noise model in a way very similar to how GPTD is extended to MC-GPTD. The key to this is to express the covariance matrix of the random vector $N_i^\lambda = (n_1, \ldots, n_i)^T$ (which we actually do in section IV, see Eq. (32)) and to follow the same developments as in [10, Appendix A.2] (which notably use the partitioned matrix inversion formula). It is not so difficult, but it is not done here due to lack of place. Moreover, KTD extension proposed in the next section provides the same estimator, as long as the value function representation is parametric and linear and the evolution noise is set to zero.

An important remark should be made here. A GPTD(λ) algorithm has been proposed before [10]. However it is not the same algorithm as the method introduced above and using the new colored noise model. GPTD(λ) also uses a colored noise model, however it is derived in a different way. The LSTD(λ) algorithm [11] is analyzed as a maximum likelihood algorithm, which is used to derive a new colored noise model [10, Chapter 4.5.]. Colored noises being not the same, resulting algorithms are different. Notably, contrary to the proposed extension, GPTD(λ) does not reduce to GPTD when the eligibility factor is set to $\lambda = 0$. Moreover, as far as we know, GPTD(λ) is a pure batch algorithm, it cannot be expressed in a recursive form. This is not the case for the proposed extension.

C. Handling the Noise in KTD

The white observation noise assumption is mandatory for Kalman equations derivation. In the case of an autoregressive (AR) noise, a classical approach consists in extending the parameter vector with the noise (see [12] for example). However, noise (10) is not autoregressive, therefore extension of KTD is not so straightforward.

A first problem is that noise (10) is anti-causal, whereas it should be causal to be used in a Kalman filter. However, what we are interested in are the correlations induced by the noise. Yet, if residuals are assumed stationary (that is $\sigma_i^2 = \sigma_j^2 = \sigma^2$), causal and anti-causal noises provide the same correlations. Therefore, we consider the following noise model:

$$n_i^\lambda = u_i - \gamma\lambda(u_{i-1} + \cdots + \gamma^k(1-\lambda)^k u_{i-k-1} + \ldots) \tag{11}$$

This noise is actually the sum of the white noise u_i plus a weighted AR noise b_{i-1}^λ:

$$n_i^\lambda = u_i - \gamma\lambda b_{i-1}^\lambda \text{ with } b_i^\lambda = \gamma(1-\lambda)b_{i-1}^\lambda + u_i \tag{12}$$

The trick to handle this noise in the KTD framework is to rewrite the non-autoregressive noise n_i^λ as a vectorial AR noise $(b_i^\lambda, n_i^\lambda)^T$:

$$\begin{pmatrix} b_i^\lambda \\ n_i^\lambda \end{pmatrix} = \begin{pmatrix} \gamma(1-\lambda) & 0 \\ -\gamma\lambda & 0 \end{pmatrix} \begin{pmatrix} b_{i-1}^\lambda \\ n_{i-1}^\lambda \end{pmatrix} + \begin{pmatrix} u_i \\ u_i \end{pmatrix} \tag{13}$$

Given this autoregressive form of the noise of interest, it is easy to extend state-space model (1), the resulting algorithm being called KTD(λ). Let p be the number of parameters, I_p the $p \times p$ identity matrix and $\mathbf{0}$ the $p \times 1$ zero vector:

$$\begin{cases} \begin{pmatrix} \theta_i \\ b_i^\lambda \\ n_i^\lambda \end{pmatrix} = \begin{pmatrix} I_p & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \gamma(1-\lambda) & 0 \\ \mathbf{0}^T & -\gamma\lambda & 0 \end{pmatrix} \begin{pmatrix} \theta_{i-1} \\ b_{i-1}^\lambda \\ n_{i-1}^\lambda \end{pmatrix} + \begin{pmatrix} v_i \\ u_i \\ u_i \end{pmatrix} \\ r_i = \hat{V}_{\theta_i}(s_i) - \gamma\hat{V}_{\theta_i}(s_{i+1}) + n_i^\lambda \end{cases}$$
$$\Leftrightarrow \begin{cases} \mathbf{x}_i = F(\lambda)\mathbf{x}_{i-1} + v_i' \\ r_i = g_i(\mathbf{x}_i) \end{cases} \tag{14}$$

The parameters are now extended with the vectorial AR noise: $\mathbf{x}_i^T = (\theta_i^T, b_i^\lambda, n_i^\lambda)^T$. The evolution matrix $F(\lambda)$ takes into account the structure of the observation noise. The process noise is also extended to take it into account. The new noise v_i' is centered with variance $P_{v_i'}$:

$$P_{v_i'} = \begin{pmatrix} P_{v_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \\ \mathbf{0}^T & \sigma_i^2 & \sigma_i^2 \end{pmatrix} \tag{15}$$

The observation equation remains the same: $r_i = g_i(\mathbf{x}_i) = \hat{V}_{\theta_i}(s_i) - \gamma\hat{V}_{\theta_i}(s_{i+1}) + n_i$. However now the observation noise is a part of the evolution equation which is estimated along with parameters. Algorithm 1 applies to this equivalent state-space formulation, however with a slight modification: the evolution equation being no longer the identity , it has to be taken into account in the prediction step, as depicted in Algorithm 2.

D. Practical Algorithm

A last thing to do in order to obtain a practical algorithm is to compute the statistics of interest, which cannot generally be done analytically except for the linear case. This problem can be stated as the issue of computing first and second order moments of a nonlinearly transformed random variable. This problem is addressed by the Unscented Transform (UT) [7], the principle of which is briefly presented here.

Let X be a random vector and $Y = f(X)$ its nonlinear mapping. The basic idea of the UT is that it is easier to approximate an arbitrary random vector than an arbitrary nonlinear function. Its principle is to sample *deterministically* a set of so-called sigma-points from the expectation and the covariance of X. The images of these points through the nonlinear mapping f are then computed, and they are used to approximate statistics of interest. It shares similarities with Monte-Carlo methods, however here the sampling is deterministic and requires a number of samples linear in the dimension of the random vector X, nonetheless allowing a

given accuracy (up to 4^{th} order in terms of Taylor expansion [13]). Let n be the dimension of X. A set of $2n+1$ so-called sigma-points is computed as follows:

$$\begin{cases} x^{(0)} = \bar{X} & j = 0 \\ x^{(j)} = \bar{X} + (\sqrt{(n+\kappa)P_X})_j & 1 \leq j \leq n \\ x^{(j)} = \bar{X} - (\sqrt{(n+\kappa)P_X})_{n-j} & n+1 \leq j \leq 2n \end{cases} \tag{16}$$

as well as associated weights

$$w_0 = \frac{\kappa}{n+\kappa} \text{ and } w_j = \frac{1}{2(n+\kappa)},\ j > 0 \tag{17}$$

where $\bar{X}$ is the mean of X, P_X is its variance matrix, κ is a scaling factor which controls the sampling spread, and $(\sqrt{(n+\kappa)P_X})_j$ is the j^{th} column of the Cholesky decomposition of the matrix $(n+\kappa)P_X$. Then the image through the mapping f is computed for each of these sigma-points:

$$y^{(j)} = f(x^{(j)}),\ 0 \leq j \leq 2n \tag{18}$$

The set of sigma-points and their images can then be used to approximate first and second order moments of Y, $\bar{Y}$ and P_Y, as well as P_{XY}, the covariance between X and Y:

$$\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)} \tag{19}$$

$$P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T \tag{20}$$

$$P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T \tag{21}$$

Thanks to the UT, a practical algorithm can be derived. At each time step i, a sigma-point set is computed from $\hat{\mathbf{x}}_{i|i-1}$ and associated variance $P_{\mathbf{x}_{i|i-1}}$, as well as images of these sigma-points through the evolution equation. Both are used to compute the statistics of interest, which are then used to update the model. This is summarized in Algorithm 2.

Algorithm 2: KTD(λ): practical algorithm

Initialization: priors $\hat{\mathbf{x}}_{0|0} = \begin{pmatrix} \hat{\theta}_{0|0}^T & 0 & 0 \end{pmatrix}^T$ and $P_{\mathbf{x}_{0|0}}$;
for $i \leftarrow 1, 2, \ldots$ **do**
Observe transition (s_i, s_{i+1}) and reward r_i ;
Prediction Step;
$\hat{\mathbf{x}}_{i|i-1} = F(\lambda)\hat{\mathbf{x}}_{i-1|i-1}$;
$P_{\mathbf{x}_{i|i-1}} = F(\lambda)P_{\mathbf{x}_{i-1|i-1}}F(\lambda)^T + P_{v_i'}$;
Sigma-points computation ;
$\mathbf{X}_{i|i-1} = \{\hat{\mathbf{x}}_{i|i-1}^{(j)} = [(\hat{\theta}_{i|i-1}^{(j)})^T, \hat{b}_{i|i-1}^{(j)}, \hat{n}_{i|i-1}^{(j)}]^T, \quad 0 \leq j \leq 2(p+2)\}$ (from $\hat{\mathbf{x}}_{i|i-1}$ and $P_{\mathbf{x}_{i|i-1}}$ using the UT);
$\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2(p+2)\}$;
$\mathcal{R}_{i|i-1} = \{\hat{r}_{i|i-1}^{(j)} = \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_i) - \gamma \hat{V}_{\hat{\theta}_{i|i-1}^{(j)}}(s_{i+1}) + \hat{n}_{i|i-1}^{(j)},\ 0 \leq j \leq 2(p+2)\}$;
Compute statistics of interest;
$\hat{r}_{i|i-1} = \sum_{j=0}^{2(p+2)} w_j \hat{r}_{i|i-1}^{(j)}$;
$P_{\mathbf{x}r_i} = \sum_{j=0}^{2(p+2)} w_j (\hat{\mathbf{x}}_{i|i-1}^{(j)} - \hat{\mathbf{x}}_{i|i-1})(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1})$;
$P_{r_i} = \sum_{j=0}^{2(p+2)} w_j (\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1})^2$;
Correction step;
$K_i = P_{\mathbf{x}r_i} P_{r_i}^{-1}$;
$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + K_i(r_i - \hat{r}_{i|i-1})$;
$P_{\mathbf{x}_{i|i}} = P_{\mathbf{x}_{i|i-1}} - K_i P_{r_i} K_i^T$;

IV. Algorithm Analysis

A first thing to be assessed are computational and memory complexities of the proposed approach. Recall that p is the number of parameters. Vector (of size $(p+2) \times 1$) and matrices (of size $(p+2) \times (p+2)$) have to be stored, therefore memory complexity is in $O(p^2)$. All operations (basic linear algebra and evaluating the $2p+4$ sigma-points) are at most in $O(p^2)$, except the Cholesky decomposition which is in $O(p^3)$. However, as the variance update is a rank-one update, the Cholesky factorization can be performed in $O(p^2)$ [14]. Therefore computational complexity is also in $O(p^2)$. This is the same computational and memory complexity as GPTD [10] and LSTD [11].

We now state and prove the main theoretical result on KTD(λ):

Theorem 1: Assume that the posterior, the noise, and the prior (of mean $\theta_{0|0}$ and variance $P_{0|0}$) distributions are all Gaussian. Also assume that residuals have a constant variance σ^2. Then KTD(λ) provides the following estimate:

$$\begin{aligned} \hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \big(& \frac{1}{\sigma^2} \sum_{j=0}^{i} \big\{ r_j + \gamma \hat{V}_\theta(s_{j+1}) - \hat{V}_\theta(s_j) \\ & + \gamma\lambda \sum_{k=j+1}^{i} \gamma^{k-j-1} \Big(r_k + \gamma \hat{V}_\theta(s_{k+1}) - \hat{V}_\theta(s_k) \Big) \big\}^2 \\ & + \big(\theta - \theta_{0|0}\big)^T P_0^{-1} \big(\theta - \theta_{0|0}\big) \big) \end{aligned} \tag{22}$$

Proof: First notice that KTD is actually a special form of a Sigma-Point Kalman Filter (SPKF) with a linear evolution model. It is shown in [13, Chapter 4.5.] that under the hypothesis of Gaussian posterior and noises, such a filter produces a *maximum a posteriori* (MAP) estimator. The corresponding proof is made for a random walk evolution model (that is identity evolution matrix), however it can be easily extended to linear evolution model. It can thus be applied to state-space model (14):

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_i^{\text{MAP}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|r_{1:i}) \tag{23}$$

State-space model (14) being equivalent to state-space model (1) with the new observation noise n_i^λ instead of the white noise n_i, the same holds for the parameter vector, and we have:

$$\hat{\theta}_{i|i} = \hat{\theta}_i^{\text{MAP}} = \underset{\theta}{\operatorname{argmax}}\, p(\theta|r_{1:i}) \tag{24}$$

Using the Bayes rule, the posterior can be rewritten as the (normalized) product of likelihood and prior:

$$p(\theta|r_{1:i}) = \frac{p(r_{1:i}|\theta)p(\theta)}{p(r_{1:i})} \tag{25}$$

The denominator does not depend on parameters, so it has not to be taken into account in the maximization: maximum a posteriori reduces to maximum likelihood times prior. Maximizing a product of probability densities is equivalent to minimizing the sum of their negative logarithms:

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}}(-\ln(p(r_{1:i}|\theta)) - \ln(p(\theta))) \tag{26}$$

The prior is assumed Gaussian, therefore:

$$-\ln(p(\theta)) \propto (\theta - \theta_{0|0})^T P_{0|0}^{-1}(\theta - \theta_{0|0}) \tag{27}$$

It remains to compute the likelihood, which takes into account the structure of the proposed colored noise. Let $V_i(\theta) = (\hat{V}_\theta(s_1), \hat{V}_\theta(s_2), \ldots, \hat{V}_\theta(s_i))^T$, $R_i = (r_1, r_2, \ldots, r_i)^T$ and $N_i = (n_1^\lambda, n_2^\lambda, \ldots, n_i^\lambda)^T$ be the vectors containing histories of values, rewards and noises. Let the $i \times i$ bidiagonal H_i matrix be defined as:

$$H_i = \begin{pmatrix} 1 & -\gamma & 0 & \ldots \\ 0 & 1 & -\gamma & 0 \\ \vdots & \ddots & \ddots & -\gamma \\ 0 & \ldots & 0 & 1 \end{pmatrix} \tag{28}$$

Let also $\Sigma_{N_i} = E[N_i N_i^T]$ be the variance matrix of N_i. Under the current assumptions, the likelihood has a Gaussian distribution:

$$r_{1:i}|\theta \sim \mathcal{N}(R_i - H_i V_i(\theta), \Sigma_{N_i}) \tag{29}$$

The KTD(λ) estimate (which we recall to be the MAP estimate) thus satisfies:

$$\begin{aligned}\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}}\{&(R_i - H_i V_i(\theta))^T \Sigma_{N_i}^{-1}(R_i - H_i V_i(\theta)) \\ &+ (\theta - \theta_{0|0})^T P_{0|0}^{-1}(\theta - \theta_{0|0})\}\end{aligned} \tag{30}$$

Let $G_i(\lambda)$ be the $i \times i$ matrix which transforms the white noise u_i into the colored noise n_i^λ:

$$G_i(\lambda) = \begin{pmatrix} 1 & -\gamma\lambda & -\gamma^2(1-\lambda)\lambda & \ldots & -\gamma^{i-1}(1-\lambda)^{i-2}\lambda \\ 0 & 1 & -\gamma\lambda & \ldots & -\gamma^{i-2}(1-\lambda)^{i-3}\lambda \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \tag{31}$$

The variance matrix Σ_{N_i} is given by:

$$\Sigma_{N_i} = \sigma^2 G_i(\lambda) G_i(\lambda)^T \tag{32}$$

Its a simple algebraic exercice to check that the inverse of $G_i(\lambda)$ is given by:

$$G_i^{-1}(\lambda) = \begin{pmatrix} 1 & \gamma\lambda & \gamma^2\lambda & \ldots & \gamma^{i-1}\lambda \\ 0 & 1 & \gamma\lambda & \ldots & \gamma^{i-2}\lambda \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \tag{33}$$

Given this matrix, the KTD(λ) estimate can be made explicit:

$$\begin{aligned}\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}} \Big(&\frac{1}{\sigma^2} \left\| G_i^{-1}(\lambda)(R_i - H_i V_i(\theta)) \right\|^2 \\ &+ (\theta - \theta_0)^T P_0^{-1}(\theta - \theta_0) \Big)\end{aligned} \tag{34}$$

Given the form of $G_i^{-1}(\lambda)$, this proves the result. ■

This result strengthens the link to eligibility traces. With $\lambda = 0$ (white noise model), the minimized cost function (omitting the regularization term) is given by $\sum_{j=0}^{i}(r_i + \gamma\hat{V}_\theta(s_{j+1}) - \hat{V}_\theta(s_j))^2$. It is a square Bellman residual, which helps understanding the bias problem and which provides local updates. With $\lambda = 1$ (Engel's noise model), it is $\sum_{j=1}^{i}(\hat{V}_\theta(s_j) - \sum_{k=j}^{i}\gamma^{k-j}r_k)^2$. This cost function links states values to observed Monte Carlo returns. The assumptions used to obtain this result are rather strong. However, even if they are not satisfied, KTD(λ) still minimizes the mean squared error of parameters estimates conditioned on past observed rewards. Moreover, this result shows that KTD(1) is unbiased, and that KTD(λ) for $0 < \lambda < 1$ reduces the bias compared to KTD(0). The same holds for GPTD, which we recall to be (in its parametric form) a special case of KTD.

V. Experimental results

In this section we use the Boyan chain [11] to compare KTD(λ) with varying eligibility factors. LSTD(λ) is also considered (in its recursive form). We have chosen this simple benchmark to study the influence of λ over the mean error and the associated standard deviation (eligibility traces are known to influence variance of estimates when combined with other algorithms). Using eligibility traces does not harm the scaling property of GPTD or KTD (which have the same computational complexity, see Sec. IV).

A claimed advantage of KTD over GPTD is its ability to handle nonlinear parameterizations. To show it, we extend KTD(λ) to the control case and provide an experiment where the state-action value function is represented by a multilayer perceptron.

A. Influence of the eligibility factor

The Boyan chain is a 13-state Markov chain where state s^0 is an absorbing state, s^1 transits to s^0 with probability 1 and a reward of -2, and s^i transits to either s^{i-1} or s^{i-2}, $2 \leq i \leq 12$, each with probability 0.5 and reward -3. The feature vectors $\phi(s)$ for states s^{12}, s^8, s^4 and s^0 are respectively $[1,0,0,0]^T$, $[0,1,0,0]^T$, $[0,0,1,0]^T$ and $[0,0,0,1]^T$, and they are obtained by linear interpolation for other states. The approximated value function is thus $\hat{V}_\theta(s) = \theta^T\phi(s)$. The optimal value function is exactly linear in these features, and the corresponding optimal parameter vector is $\theta^* = [-24, -16, -8, 0]^T$. To measure the quality of each algorithm the Euclidian distance between the current parameter vector estimate and the optimal one $\|\theta - \theta^*\|$ is computed. The parameterization being linear, it is $\|\hat{V}_\theta - V^*\|$ up to a scaling factor.

For KTD(λ) noises are set to $P_{v_i} = 0$ and $\sigma^2 = 10^{-3}$ and priors to $\theta_{0|0} = 0$ and $P_{0|0} = I$. Same priors are used for the recursive LSTD(λ), which in fact means that both algorithms use the same regularization term. Notice that as the parameterization is linear and as the evolution noise is zero KTD(λ) results are the same as those provided by the sketched GPTD extension (as under these conditions both frameworks provide the same algorithm).

A first experiment consists in running KTD(λ) for various values of the eligibility factor. Results on Figure 1 show the mean error and the associated standard deviation after 50 and 500 learning episodes as a function of λ. These statistics are obtained using 1000 independent trials. After 50 episodes, there is no significant influence of the eligibility factor on mean errors or associated standard deviations, except for $\lambda = 0$ which corresponds to a biased estimate (Bellman residual minimization). After 500 episodes there is still no significant difference on mean errors. However, the value of λ affects the associated standard deviation, which is lower for λ close to 0.1. Notice that the mean error for $\lambda = 0$ is the same for 50 and 500 episodes. This means that the bias is reached quickly. Empirically, the bias is mainly removed as long as $\lambda > 0$.

In a second experiment KTD(λ) is compared to LSTD(λ). As for KTD(λ) the eligibility factor does not significantly affect the mean error, only KTD(1) is considered. Various values of the eligibility factor are considered for LSTD(λ). Learning is done for 40 episodes, and results presented on Figure 2 are averaged over 300 independent trials. Increasing the value of λ improves the convergence rate of LSTD(λ) while KTD(1) is a little bit faster. Tuning parameters (in fact, the prior) can probably improve the performance of LSTD(λ). However, we argue that KTD(λ) is at least as efficient as LSTD(λ) (except KTD(0) which provides a biased estimate contrary to LSTD(0)).

B. Handling nonlinearities: neural KTD(λ)

Here we show that KTD(λ) handles nonlinear parameterizations, which is how KTD extends GPTD. For this, we approximate the state-action value function with a multilayer percpetron (MLP). The considered experiment is the inverted pendulum task which requires balancing a pendulum of unknown length and mass at the upright position by applying forces to the cart it is attached to. It is fully described in [15].

A first problem is to extend KTD(λ) to control. Actually, this is quite easy. Algorithm 2 can be straightforwardly extended to the Q-function evaluation, and actions can be sampled according to any exploration/exploitation scheme (ϵ-greedy policy, Boltzman policy, *etc*.). Moreover, KTD(λ) provides an uncertainty information (encoded in the variance matrix $P_{i|i}$) which can be useful for the exploration/exploitation dilemma, as shown in the case of KTD(0) [16] (which we recall to be the original KTD algorithm). By the way, we consider here a simple ϵ-greedy policy. The ϵ factor is set to $\epsilon_i = \epsilon_0 \frac{n_0}{n_0+i}$ with $\epsilon_0 = 0.5$ and $n_0 = 600$ and where i is the number of transitions.

The Q-function is approximated with a 3-3-1 MLP (one input for each of the two state components, one for the action, three hidden neurons and on output for the state-action value) with $\tanh$ activation function. We call this algorithm neural KTD(λ). We used the *scaled* UT [7] instead of the presented UT (because it alleviates some of its problem, see [7] again) with the following (somehow standard) parameters: $\alpha_{\text{UT}} = 10^{-2}$, $\beta_{\text{UT}} = 2$ and $\kappa_{\text{UT}} = 0$. The parameter vector $\theta_{0|0}$ is initialized randomly close to zero, and other parameters are as follows: $\sigma = 10^{-2}$, $P_{0|0} = 10^{-1} I$ and $P_{v_i} = \eta P_{i-1|i-1}$ with $\eta = 10^{-6}$. We recognize that choosing these parameters can seem unnatural and need some practice. However, it is not really more difficult than choosing a learning rate for example. As a baseline we consider the classic SARSA algorithm parameterized with a 3×3 radial basis function (RBF) network per action (the same as in [15]). We consider a linear parameterization in this case because it can be improper to use SARSA with nonlinear function approximator [17]. The same ϵ factor is used, and the learning rate is set to $\alpha_i = \alpha_0 \frac{n_1}{n_1+i}$ with $\alpha_0 = 0.5$ and $n_1 = 100$. Results comparing KTD(λ) (for $\lambda = 0$ and $\lambda = 0.7$) and SARSA are presented in Fig. 3 (linear and log scales). Performance is measured as the number of steps in an episode, and results are averaged over 100 trials.

KTD(λ) performs much better than SARSA, which was to be expected as KTD is a second order algorithm. Actually, SARSA results are comparable to Q-learning results using the same parameterization reported in [15, Fig. 17]. Moreover, KTD(λ) shows performance comparable to the LSPI algorithm using the RBF parameterization [15, Fig. 16]. For example, after 250 learning episodes, KTD(0) maintains the pole for 2300 steps in average, KTD(0.7) for 2500 and LSPI for approximately 2400 [15]. Notice also that in this stochastic problem KTD(0.7) performs better than KTD(0). This can be linked to the bias removal, but also to the fact that using eligibility traces tends to speed up learning. This benchmark is quite simple, however it show that KTD(λ) can be efficiently combined with neural networks, which can be of interest for real world problems (scaling-up issue). Actually, the considered 3-3-1 MLP is quite small and involves twice less parameters than the RBF structure.

VI. Conclusion

In this paper we have introduced a new way (based on a colored noise model) to extend methods based on a statistical modeling of value function approximation (GPTD and KTD) to eligibility traces in the aim of treating stochastic MDP problems. We have introduced the related colored observation noise model, explained briefly how to handle it in the GPTD framework, shown how to take it into account in the KTD framework, analyzed the resulting algorithm and experimented it. Notably, we extended KTD(λ) to a control scheme and combined it with a multilayer percpetron, which shows its ability to handle nonlinearities.

Using eligibility traces in these framworks is of interest. First, it allows to obtain unbiased ($\lambda = 1$) or less biased ($0 < \lambda < 1$) value function approximators in the case

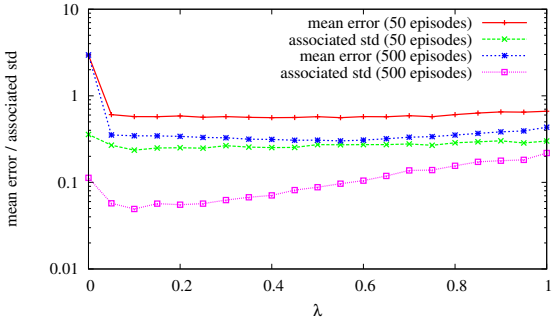


Fig. 1. Influence of the eligibility factor.

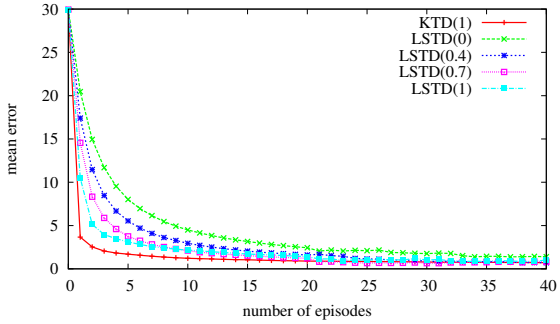
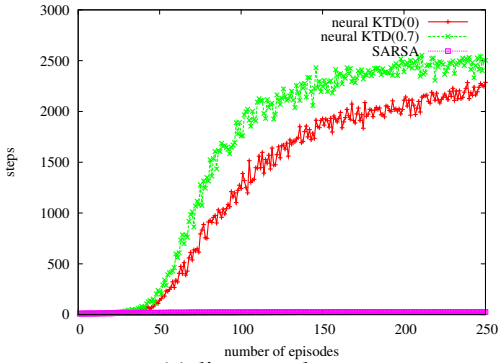
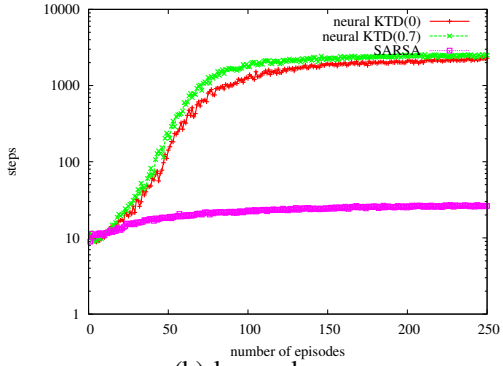


Fig. 2. Comparison to LSTD(λ).



(a) linear scale.



(b) log scale.

Fig. 3. neural KTD(λ).

of stochastic MDP. Compared to Engel's noise, it allows reducing the variance of estimates by wisely choosing the eligibility factor. Colored noises and eligibility traces induce a memory effect which can be harmful, for example in an off-policy learning scenario (like Q-learning [1]). This is why we have not considered the Bellman optimality equation (which can actually be done using the KTD framework, see the KTD-Q algorithm [3]), because of its off-policy aspect. However, decreasing λ diminishes this memory effect, which can be another advantage of the proposed noise model compared to Engel's one. Moreover, our noise provides a theoretical bridge between the white noise model and Engel's one.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, 3rd ed. The MIT Press, March 1998.

[2] Y. Engel, S. Mannor, and R. Meir, "Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning," in *Proceedings of the International Conference on Machine Learning (ICML 03)*, 2003, pp. 154–161.

[3] M. Geist, O. Pietquin, and G. Fricout, "Kalman Temporal Differences: the deterministic case," in *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, April 2009.

[4] L. C. Baird, "Residual Algorithms: Reinforcement Learning with Function Approximation," in *Proceedings of the International Conference on Machine Learning (ICML 95)*, 1995, pp. 30–37.

[5] Y. Engel, S. Mannor, and R. Meir, "Reinforcement Learning with Gaussian Processes," in *Proceedings of International Conference on Machine Learning (ICML-05)*, 2005.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[7] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[8] M. Geist, O. Pietquin, and G. Fricout, "Tracking in Reinforcement Learning," in *Proceedings of the 16th International Conference on Neural Information Processing (ICONIP 2009)*. Bangkok (Thailande): Springer, 2009.

[9] A. Antos, C. Szepesvári, and R. Munos, "Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, no. 1, pp. 89–129, April 2008.

[10] Y. Engel, "Algorithms and Representations for Reinforcement Learning," Ph.D. dissertation, Hebrew University, April 2005.

[11] J. A. Boyan, "Least-squares temporal difference learning," in *Proceedings of the 16th International Conference on Machine Learning (ICML 99)*. Morgan Kaufmann, San Francisco, CA, 1999, pp. 49–56.

[12] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, 1st ed. Wiley & Sons, August 2006.

[13] R. van der Merwe, "Sigma-point kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, OGI School of Science & Engineering, Oregon Health & Science University, Portland, OR, USA, April 2004.

[14] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for Modifying Matrix Factorization," *Mathematics of Computation*, vol. 28, no. 126, pp. 505–535, April 1974.

[15] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[16] M. Geist and O. Pietquin, "Managing Uncertainty within Value Function Approximation in Reinforcement Learning," in *Active Learning and Experimental Design workshop (collocated with AISTATS 2010)*, Sardinia, Italy, May 2010.

[17] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, pp. 674–690, 1997.