Statistically Linearized Least-Squares Temporal Differences

Matthieu Geist and Olivier Pietquin

***Abstract*— A common drawback of standard reinforcement learning algorithms is their inability to scale-up to real-world problems. For this reason, a current important trend of research is (state-action) value function approximation. A prominent value function approximator is the least-squares temporal differences (LSTD) algorithm. However, for technical reasons, linearity is mandatory: the parameterization of the value function must be linear (compact nonlinear representations are not allowed) and only the Bellman evaluation operator can be considered (imposing policy-iteration-like schemes). In this paper, this restriction of LSTD is lifted thanks to a derivative-free statistical linearization approach. This way, nonlinear parameterizations and the Bellman optimality operator can be taken into account (this last point allows taking into account value-iteration-like schemes). The efficiency of the resulting algorithms are demonstrated using a linear parametrization and neural networks as well as on a Q-learning-like problem. A theoretical analysis is also provided.**

***Index Terms*— reinforcement learning, value function approximation, statistical linearization, neural networks.**

I. Introduction

OPTIMAL control of dynamic systems is a recurent problem in a wide range of domains. The main answer of machine learning to this problem is the Reinforcement Learning (RL) paragon [1]. In this paradigm, a system steps from *state* to state as a result of *actions* performed by an external controler and of its own dynamics. To each transition from one step to another is associated a scalar *reward* signal. The objective is thus to build a controler or *agent* that learns a control *policy* which maximizes the expected cumulative reward. An important topic of RL is to estimate the so-called *value function* which locally models the expected cumulative reward of a given policy, especially when the set of possible states is too large for a tabular representation, which implies to use some function approximator. The value function being observed only through immediate rewards and not directly, this is more than a regression problem.

More formally, the system to be controlled is standardly modeled as a Markov Decision Process (MDP) $\{S, A, P, R, \gamma\}$: S is the (finite) state space, A the (finite) action space, $P \in \mathcal{P}(S)^{S\times A}$ the set of transition probabilities, $R \in \mathbb{R}^{S\times A\times S}$ the deterministic reward function and γ the forgetting factor. Actions are selected according to a stochastic policy $\pi \in \mathcal{P}(A)^S$. The state-action value function is the expected discounted cumulative reward starting in a state s, taking an action a and then following the policy π: $Q^\pi(s,a) = E[\sum_{i=0}^{\infty}\gamma^i r_i|s_0 = s, a_0 = a, \pi]$. The state-action value function satisfies the (linear) Bellman evaluation equation: $Q^\pi(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma E_{a'|s'}[Q^\pi(s',a')]]$. It can be rewritten as the fixed-point of the Bellman evaluation operator: $Q^\pi = T^\pi Q^\pi$. The aim of RL is to find a policy which maximizes the value function $V^\pi(s) = E_{a|s}[Q^\pi(s,a)]$ for each state: $\pi^* \in \operatorname{argmax}_\pi V^\pi$. The optimal state-action value function Q^* satisfies the (nonlinear) Bellman optimality equation: $Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{a'\in A} Q^\pi(s',a')]$. It can be rewritten as the fixed-point of the Bellman optimality operator: $Q^* = T^* Q^*$.

If the state space is so large that the (state-action) value function cannot be represented as a vector, some approximation architecture should be adopted, such as radial-basis-function network (linear parametrization) or multilayer perceptron (nonlinear parametrization) for example. Notice that estimating such a function is more than a regression problem, as values are never directly observed, only rewards which can be linked to values through a Bellman equation. The perhaps most famous approach for value function approximation is bootstrapping (briefly described in Sec. II). However, it has been shown that combining a bootstrapping approach with a nonlinear parametrization can cause divergence of the algorithm [2]. Consequently, a common thought in the RL community is that a new algorithm should have at most a linear complexity, because it would be stuck to a linear parametrization, which generally implies learning many parameters. However, this problem is linked to the bootstrap approach, and there are other ways to estimate the value function (to be sketched in Sec. II). Notably, least-squares temporal differences (LSTD) [3] is a prominent RL algorithm which is very efficient and theoretically well founded (notably, there should not be theoretical problem with nonlinear parameterizations). However, it implies solving a difficult optimization problem (see Sec. II-A), which explains that it has been only very recently considered in a nonlinear setting [4] (the position of our contribution compared to this approach being discussed in Sec. IV). The main contribution of this paper is Sec. III in which we extend LSTD (using a statistical linearization approach) to nonlinear parameterizations as well as to the Bellman optimality operator (which is nonlinear, non-derivable, but which allows computing directly the optimal Q-function). This new algorithm is experimented and compared to the state of the art in Sec. V.

II. Background

There are many approaches to value function approximation and we will focus here on parametric online learning algorithms. The problem is thus to find the best set of parameters θ^* that can be used to approximate Q^π or Q^* by a (maybe non-linear) parametric function $\hat{Q}_\theta$ belonging

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

to an hypothesis space $\mathcal{H} = \{\hat{Q}_\theta \in \mathbb{R}^{S\times A} | \theta \in \mathbb{R}^p\}$. Bootstrapping approaches consist in treating value function approximation as a supervised learning problem. As the value function is not directly observable, it is replaced by an estimate computed from previous observations $\hat{Q}_\theta(s,a) \approx R(s,a,s') + \gamma \hat{Q}_\theta(s',a')$. TD with function approximation [1] and fixed-point Kalman filter [5] are such algorithms. The problem is that bootstrapping approaches may diverge when combined with a nonlinear parameterization [2]. Another approach is residual algorithms [6]. They consist in minimizing the square error between the estimated state-action value function and its image through the Bellman evaluation operator ($\theta^* = argmin_\theta \|\hat{Q}_\theta - T\hat{Q}_\theta\|^2$). However, they are biased when trained using sampled trajectories. Other approaches rely on a statistical model of value function approximation, based on Gaussian processes [7] or on Kalman filtering [8]. They can be linked to residual approaches and therefore share the bias drawback. For both methods, this problem can be corrected with the introduction of a colored noise model [9][10]. However, introduction of this noise implies some memory effects which prevent from learning in an off-policy way (that is learning a policy while following another one, typically the optimal policy while following a suboptimal one, in other words considering the Bellman optimality operator). A last approach, the projected fixed-point approach, consists in minimizing the square error between the estimated state-action value function and the projection of the image of this function under the Bellman operator onto the hypothesis space (see Fig. 1). This optimization problem is addressed by the least-squares temporal differences (LSTD) algorithm [3] (it is also addressed in a slightly different way by the least-squares policy evaluation algorithm [11]).

LSTD is a sample efficient algorithm which has been used successfully within policy iteration [12] and actor-critic architectures [13]. Its main drawback is that it is stuck to linearity (respectively to parameters). To our mind, handling nonlinearities is of major interest. First, it allows considering nonlinear parameterizations such as neural networks, with the advantage that they are potentially much more compact (scale-up). Moreover, if nonlinearities are handled in a derivative-free manner, the Bellman optimality operator can be considered and the optimal state-action value function directly estimated in an off-policy learning manner (informally, this would provide a Q-learning-like extension of LSTD). Notice that taking into account the Bellman optimality operator *does not* reduce to learning in an off-policy manner by taking the greedy action (in the transiting state) respectively to the current state-action value function representation; it is much more difficult. In this contribution, we propose to extend the LSTD algorithm to nonlinear parameterizations and Bellman optimality operator (for Q-learning-like algorithms) thanks to a derivative-free statistical linearization approach.

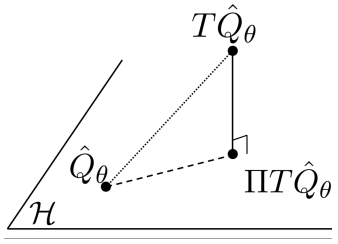

Fig. 1. Obtaining LSTD cost function.

A. Least-Squares Temporal Differences

An important subgoal of RL is to find a good approximation $\hat{Q}$ of either Q^π or Q^* when the state space is too large. In this paper, we focus on the case in which $\hat{Q}$ is a (not necessarily linear) parametric function approximator (e.g., linear features, neural network, etc.) belonging to an hypothesis space $\mathcal{H} = \{\hat{Q}_\theta \in \mathbb{R}^{S\times A} | \theta \in \mathbb{R}^p\}$. Formally and generally, LSTD aims at minimizing the square error between the approximate state-action value function and the projection of its image under a Bellman operator onto the hypothesis space $\mathcal{H}$:

$$\theta^* = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}} \|\hat{Q}_\theta - \Pi T\hat{Q}_\theta\|^2 \quad (1)$$

where T is one of the Bellman operators and Π the projection onto $\mathcal{H}$:

$$\Pi f = \underset{h\in\mathcal{H}}{\operatorname{argmin}} \|f - h\|^2 \quad (2)$$

This is illustrated on Fig. 1, the distance to be minimized being the dashed line.

Therefore, LSTD implies solving two nested optimization problems (respectively the plain and dashed lines on Fig. 1):

$$\omega_\theta = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \|\hat{Q}_\omega - T\hat{Q}_\theta\|^2, \theta^* = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}} \|\hat{Q}_\theta - \hat{Q}_{\omega_\theta}\|^2 \quad (3)$$

As we work with a parametric representation, this leads to the following fixed-point problem: $\theta^* = \omega_{\theta^*} = \operatorname{argmin}_{\omega\in\mathbb{R}^p} \|\hat{Q}_\omega - T\hat{Q}_{\theta^*}\|^2$.

Practically, learning is done from a set of i experimented transitions. An empirical cost function has thus to be considered: $\theta_i = \omega_{\theta_i} = \operatorname{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (\hat{Q}_\omega(s_j,a_j) - T\hat{Q}_{\theta_i}(s_j,a_j))^2$. However, the model (that is reward function and transition probabilities of the MDP) is not known, therefore the Bellman operator cannot be computed. Instead, a sampled operator has to be defined. By writing $\hat{T}$ the sampled Bellman operator (resp. $\hat{P}$ the sampled transition operator) defined as:

$$\hat{T}Q(s_j,a_j) = r_j + \gamma \hat{P}Q(s_j,a_j) \quad (4)$$

$$\hat{P}Q(s_j,a_j) = \begin{cases} Q(s_{j+1},a_{j+1}) \\ \max_a Q(s_{j+1},a) \end{cases} \quad (5)$$

and considering a set of i observed transitions, it leads to the following optimization problem (which is actually a fixed-point problem):

$$\theta_i = \omega_{\theta_i} = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\hat{Q}_\omega(s_j,a_j) - \hat{T}\hat{Q}_{\theta_i}(s_j,a_j)\right)^2 \quad (6)$$

Notice that θ_i appears in both sides of this equation. This is normal and due to the fact that this cost function translates the two nested optimization problems (3). Moreover, it is because of θ_i appearing in both sides that this optimization problem is difficult, and this explains that classical approaches such as stochastic gradient descent cannot be considered directly.

LSTD assumes a linear parameterization as well as the Bellman evaluation operator and provides the analytical solution to this optimization problem. We propose to extend LSTD to nonlinear parameterizations and to the Bellman optimality operator thanks to a statistical linearization approach.

B. Statistical Linearization

Let $f : \mathbf{x} \in \mathbb{R}^p \rightarrow \mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^q$. Assume that we want to linearize f around some point $\mathbf{x}_0$. A common approach is to use a Taylor expansion: $f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0)$. However, it requires f to admit a gradient. Statistical linearization [14] consists in sampling a set of m points around $\mathbf{x}_0$ (how to sample these points being the user's choice) and using them to compute a linear model minimizing the associated square error.

Let $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)} = f(\mathbf{x}^{(j)}))_{1\leq j\leq m}$ be a set of m sampled points and their images through f. We introduce some notations:

$$\begin{cases} \bar{\mathbf{x}} = \frac{1}{m}\sum_{j=1}^m \mathbf{x}^{(j)}, \quad \bar{\mathbf{y}} = \frac{1}{m}\sum_{j=1}^m \mathbf{y}^{(j)} \\ P_{\mathbf{xx}} = \frac{1}{m}\sum_{j=1}^m \left(\mathbf{x}^{(j)} - \bar{\mathbf{x}}\right)\left(\mathbf{x}^{(j)} - \bar{\mathbf{x}}\right)^T \\ P_{\mathbf{xy}} = \frac{1}{m}\sum_{j=1}^m \left(\mathbf{x}^{(j)} - \bar{\mathbf{x}}\right)\left(\mathbf{y}^{(j)} - \bar{\mathbf{y}}\right)^T = P_{\mathbf{yx}}^T \\ P_{\mathbf{yy}} = \frac{1}{m}\sum_{j=1}^m \left(\mathbf{y}^{(j)} - \bar{\mathbf{y}}\right)\left(\mathbf{y}^{(j)} - \bar{\mathbf{y}}\right)^T \end{cases} \tag{7}$$

As these points are sampled around $\mathbf{x}_0$, $\bar{\mathbf{x}} = \mathbf{x}_0$. Statistical linearization performs a linear regression of the form $\mathbf{y} = A^T\mathbf{x} + b + e$ where e is a centered noise and A and b minimize the sum of square errors: $\min_{A,b}\sum_{j=1}^m e_j^T e_j$ with $e_j = \mathbf{y}^{(j)} - A^T\mathbf{x}^{(j)} - b$. The solution to this optimization problem is given by:

$$A = P_{\mathbf{xx}}^{-1}P_{\mathbf{xy}} \text{ and } b = \bar{\mathbf{y}} - A^T\bar{\mathbf{x}} \tag{8}$$

The error variance is therefore given by: $P_{ee} = \sum_{j=1}^m e_j e_j^T = P_{\mathbf{yy}} - A^T P_{\mathbf{xx}} A$. How to practically sample these points is an open question which is addressed in section III-B.

III. Statistically linearized LSTD

In this section, we derive the statistically linearized LSTD (slLSTD) recursive algorithm in the general case before discussing a specific sampling method choice.

A. Application of the Statistical Linearization

From Eq. (4,6) the optimization problem is:

$$\theta_i = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^i \left(r_j + \gamma\hat{P}\hat{Q}_{\theta_i}(s_j,a_j) - \hat{Q}_\omega(s_j,a_j)\right)^2 \tag{9}$$

It can be linked to the following statistical observation equation:

$$r_j + \gamma\hat{P}\hat{Q}_{\theta_i}(s_j,a_j) = \hat{Q}_\omega(s_j,a_j) + n_j \tag{10}$$

where n_j is a white centered and unitary observation noise (using a non-unitary noise would lead to scale each square term of the cost function by the inverse of the associated variance P_{n_j}).

We statistically linearize $\hat{Q}_\omega(s_j,a_j)$ and $\hat{P}\hat{Q}_{\theta_i}(s_j,a_j)$ as explained in section II-B (recall that how to sample is addressed in section III-B):

$$q_j = \hat{Q}_\omega(s_j,a_j) = A_j^T\omega + b_j + e_j \tag{11}$$

$$\text{with } A_j = P_{\omega\omega}^{-1}P_{\omega q_j} \text{ and } b_j = \bar{q}_j - A_j^T\bar{\omega} \tag{12}$$

$$\text{and} \quad p_{q_j} = \hat{P}\hat{Q}_{\theta_i}(s_j,a_j) = C_j^T\theta_i + d_j + \epsilon_j \tag{13}$$

$$\text{with } C_j = P_{\theta_i\theta_i}^{-1}P_{\theta_i p_{q_j}} \text{ and } d_j = \bar{p}_{q_j} - C_j^T\bar{\theta}_i \tag{14}$$

where A_j, b_j, C_j and d_j are statistical linearization solutions (8) and e_j and ϵ_j are the associated error terms. Observation model (10) is thus equivalent to the following one:

$$\begin{aligned} &r_j + \gamma(C_j^T\theta_i + d_j + \epsilon_j) = A_j^T\omega + b_j + e_j + n_j \Leftrightarrow \\ &r_j + \gamma(C_j^T\theta_i + d_j) - b_j = A_j^T\omega + e_j - \gamma\epsilon_j + n_j \end{aligned} \tag{15}$$

The linearization error is taken into account in the centered noise u_j of variance P_{u_j}:

$$u_j = n_j + e_j - \gamma\epsilon_j, \quad P_{u_j} = E[u_j^2] \tag{16}$$

This equivalent observation model leads to the following optimization problem:

$$\theta_i = \omega_{\theta_i} = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^i \frac{1}{P_{u_j}}(r_j + \gamma(C_j^T\theta_i + d_j) - b_j - A_j^T\omega)^2 \tag{17}$$

It is important to notice that the linearization error is taken into account through the noise variance P_{u_j}: the higher the statistical linearization error, the lower the weighting of the corresponding square term in the sum. In other words, the optimization process takes naturally into account the quality of the approximation: if the linearization error is too high, the corresponding sample will be almost ignored.

The optimization problem being now linear, it can be solved analytically (thanks to the classical least-squares solution):

$$\begin{aligned} \theta_i &= \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^i \frac{1}{P_{u_j}}(r_j + \gamma(C_j^T\theta_i + d_j) - b_j - A_j^T\omega)^2 \\ &= \left(\sum_{j=1}^i \frac{A_jA_j^T}{P_{u_j}}\right)^{-1} \sum_{j=1}^i \frac{A_j(r_j + \gamma(C_j^T\theta_i + d_j) - b_j)}{P_{u_j}} \\ \Leftrightarrow \theta_i &= \left(\sum_{j=1}^i \frac{A_j(A_j - \gamma C_j)^T}{P_{u_j}}\right)^{-1} \sum_{j=1}^i \frac{A_j(r_j + \gamma d_j - b_j)}{P_{u_j}} \end{aligned} \tag{18}$$

The Sherman-Morrison formula allows computing directly the inverse of a rank-one perturbed matrix:

$$\left(M + \mathbf{x}\mathbf{y}^T\right)^{-1} = M^{-1} - \frac{M^{-1}\mathbf{x}\mathbf{y}^T M^{-1}}{1 + \mathbf{y}^T M^{-1}\mathbf{x}} \tag{19}$$

Applying formula (19) to (18) allows deriving a recursive estimation of θ_i. Assume that some priors θ_0 and M_0 are chosen, the slLSTD update is given by:

$$\begin{cases} K_i = \frac{M_{i-1}A_i}{P_{u_i}+(A_i-\gamma C_i)M_{i-1}A_i^T} \\ \theta_i = \theta_{i-1} + K_i(r_i + \gamma d_i - b_i - (A_i - \gamma C_i)^T\theta_{i-1}) \\ M_i = M_{i-1} - K_i \left(M_{i-1}^T(A_i - \gamma C_i)\right)^T \end{cases} \tag{20}$$

In these expressions, θ_i is the $p \times 1$ parameter vector, K_i the $p \times 1$ gain vector and M_i a $p \times p$ non-symmetric matrix. This is the more general form of the slLSTD algorithm. A remaining problem is to wisely sample points from which statistical linearization is performed, it is addressed in the following section.

B. Choice of the Statistical Linearization

As the algorithm is in a recursive form, quantities to be computed are A_i, b_i, C_i and d_i. Therefore, a sampling scheme for parameter vectors has to be chosen (how to sample and around what) in order to perform the corresponding statistical linearization. We sample according to the unscented transform, to be presented now.

1) Unscented Transform.: Let X be a random variable of mean $\bar{X}$ and variance P_X, and $Y = g(X)$ a nonlinear mapping of this random variable. The unscented transform (UT) [15] aims at approximating first and second order moments $\bar{Y}$ and P_Y of the random variable Y. The basic idea of the UT is that it is easier to approximate an arbitrary random vector (with samples) 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 g 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 less samples to be drawn, nonetheless allowing a given accuracy.

The original unscented transform is now described more formally. Let n be the dimension of X. A set of $2n+1$ so-called "sigma-points" $x^{(j)}$ is computed as follows:

$$\begin{cases} x^{(0)} = \bar{X} & j = 0 \\ x^{(j)} = \bar{X} + (\sqrt{(n+\kappa)P_X})_j & 1 \leq j \leq n \\ x^{(j)} = \bar{X} - (\sqrt{(n+\kappa)P_X})_{n-j} & n+1 \leq j \leq 2n \end{cases} \tag{21}$$

as well as associated weights

$$w_0 = \frac{\kappa}{n+\kappa} \text{ and } w_j = \frac{1}{2(n+\kappa)},\ j > 0 \tag{22}$$

where 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 g is computed for each of these sigma-points: $y^{(j)} = g(x^{(j)}),\quad 0 \leq j \leq 2n$. The set of sigma-points and their images can finally be used to approximate first and second order moments of Y, and even P_{XY}, the covariance matrix between X and Y: $\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)}$, $P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T$ and $P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T$. Other approximation schemes can be considered instead of the UT, such as the scaled unscented transform [15] or approximation schemes based on Sterling interpolation [16].

2) A_i and b_i statistical linearization.: Recall that A_i and b_i result from the statistical linearization of $\hat{Q}_\omega(s_i, a_i)$:

$$q_i = \hat{Q}_\omega(s_i, a_i) = A_i^T\omega + b_i + e_i \tag{23}$$

Points (that is parameter vectors) used to perform statistical linearization are sampled according to the UT (or to one of its variants). Associated mean and variance have still to be chosen.

The parameter vector ω to be considered is the solution of Eq. (17), it is thus the fixed-point solution $\omega_{\theta_i} = \theta_i$. As we work in a recursive estimation process, it is legitimate to linearize around the last estimate θ_{i-1}. The mean being chosen, the only remaining choice is the associated variance P_{i-1}. We use the same variance matrix as would have been provided by a statistically linearized recursive least-squares (slRLS) approach [17] used to perform supervised learning of the approximate state-action value function given true observations of the values. The fact that the (non-) observed Q-values are not used to update the variance matrix tends to legitimate this choice. Practically, this choice makes the matrix to decrease to zero as the number of observed transitions goes to infinity.

These choices being made, A_i and b_i can be computed. A first step is to compute the set of sigma-points, that is a set of parameter vectors (as well as associated weights (22)):

$$\{\omega_i^{(j)}, 0 \leq j \leq 2p\} = \begin{bmatrix} \theta_{i-1} & \theta_{i-1} \pm (\sqrt{(p+\kappa)P_{i-1}})_j \end{bmatrix} \tag{24}$$

The images of these sigma-points are also computed:

$$\{q_i^{(j)} = \hat{Q}_{\omega_i^{(j)}}(s_i, a_i), 0 \leq j \leq 2p\} \tag{25}$$

Sigma-points and their images are used to compute statistics of interest:

$$\bar{q}_i = \sum_{j=0}^{2p} w_j q_i^{(j)} \tag{26}$$

$$P_{\omega q_i} = \sum_{j=0}^{2p} w_j \left(\omega_i^{(j)} - \theta_{i-1}\right)(q_i^{(j)} - \bar{q}_i) \tag{27}$$

$$P_{qq_i} = \sum_{j=0}^{2p} w_j (q_i^{(j)} - \bar{q}_i)^2 \tag{28}$$

Given these statistics, A_i and b_i are the statistical linearization solution (see section II-B):

$$A_i = P_{i-1}^{-1} P_{\omega q_i} \text{ and } b_i = \bar{q}_i - A_i^T\theta_{i-1} \tag{29}$$

The variance matrix should be updated. It is done as explained before (slRLS approach and assuming an unitary observation noise):

$$P_i = P_{i-1} - P_{\omega q_i}\left(1 + P_{qq_i}\right)^{-1} P_{\omega q_i}^T \tag{30}$$

As the inverse of this matrix is necessary to compute A_i, we also maintain P_i^{-1} recursively thanks to the Sherman-Morrison formula:

$$P_i^{-1} = P_{i-1}^{-1} + \frac{P_{i-1}^{-1} P_{\omega q_i} P_{\omega q_i}^T P_{i-1}^{-1}}{1 + P_{qq_i} - P_{\omega q_i}^T P_{i-1}^{-1} P_{\omega q_i}} \tag{31}$$

3) C_i and d_i statistical linearization.: The same approach is used to compute C_i and d_i. Recall that these quantities come from the statistical linearization of $\hat{P}\hat{Q}_{\theta_i}(s_i, a_i)$:

$$p_{q_i} = \hat{P}\hat{Q}_{\theta_i}(s_i, a_i) = C_i^T \theta_i + d_i + \epsilon_i \tag{32}$$

As before, we linearize around the previous estimate, that is θ_{i-1} and consider the variance matrix Σ_{i-1} provided by an slRLS approach that would perform a supervised regression of $\hat{P}\hat{Q}_{\theta_i}$.

This being posed, quantities C_i and d_i can be computed. A first step is to compute the set of sigma-points (as well as associated weights (22)):

$$\{\theta_i^{(j)}, 0 \leq j \leq 2p\} = \begin{bmatrix} \theta_{i-1} & \theta_{i-1} \pm (\sqrt{(p+\kappa)\Sigma_{i-1}})_j \end{bmatrix} \tag{33}$$

The images of these sigma-points are also computed:

$$\{p_{q_i}^{(j)} = \hat{P}\hat{Q}_{\theta_i^{(j)}}(s_i, a_i), 0 \leq j \leq 2p\} \tag{34}$$

Sigma-points and their images are used to compute statistics of interest:

$$\bar{p}_{q_i} = \sum_{j=0}^{2p} w_j p_{q_i}^{(j)} \tag{35}$$

$$P_{\theta p_{q_i}} = \sum_{j=0}^{2p} w_j \left(\theta_i^{(j)} - \theta_{i-1}\right)\left(p_{q_i}^{(j)} - \bar{p}_{q_i}\right) \tag{36}$$

$$P_{p_q p_{q_i}} = \sum_{j=0}^{2p} w_j (p_{q_i}^{(j)} - \bar{p}_{q_i})^2 \tag{37}$$

Given these statistics, C_i and d_i are the statistical linearization solution (see section II-B):

$$C_i = \Sigma_{i-1}^{-1} P_{\theta p_{q_i}}, \quad d_i = \bar{p}_{q_i} - C_i^T \theta_{i-1} \tag{38}$$

The variance matrix is updated according to the slRLS scheme:

$$\Sigma_i = \Sigma_{i-1} - P_{\theta p_{q_i}} \left(1 + P_{p_q p_{q_i}}\right)^{-1} P_{\theta p_{q_i}}^T \tag{39}$$

Its inverse Σ_i^{-1} is also recursively computed:

$$\Sigma_i^{-1} = \Sigma_{i-1}^{-1} + \frac{\Sigma_{i-1}^{-1} P_{\theta p_{q_i}} P_{\theta p_{q_i}}^T \Sigma_{i-1}^{-1}}{1 + P_{p_q p_{q_i}} - P_{\theta p_{q_i}}^T \Sigma_{i-1}^{-1} P_{\theta p_{q_i}}} \tag{40}$$

4) Noise variance.: The last thing to do is to compute the variance P_{u_i} of the noise $u_i = n_i + e_i - \gamma \epsilon_i$. The noise n_i is independent (and centered and unitary), and e_i and ϵ_i are centered, thus:

$$P_{u_i} = 1 + E[(e_i - \gamma \epsilon_i)^2] \tag{41}$$

The term $E[(e_i - \gamma \epsilon_i)^2]$ should be computed. Recall that e_i and ϵ_i are noise terms linked to the statistical linearization:

$$e_i = q_i - A_i^T \omega - b_i \text{ and } \epsilon_i = p_{q_i} - C_i^T \theta_i - d_i \tag{42}$$

We use again the UT to compute this error term:

$$P_{u_i} = 1 + \sum_{j=0}^{2p} w_j (e_i^{(j)} - \gamma \epsilon_i^{(j)})^2 \tag{43}$$

with

$$e_i^{(j)} = q_i^{(j)} - A_i^T \omega_i^{(j)} - b_i = q_i^{(j)} - \bar{q}_i - A_i^T (\omega_i^{(j)} - \theta_{i-1}) \tag{44}$$

$$\epsilon_i^{(j)} = p_{q_i}^{(j)} - C_i^T \theta_i^{(j)} - d_i = p_{q_i}^{(j)} - \bar{p_{q_i}} - C_i^T (\theta_i^{(j)} - \theta_{i-1}) \tag{45}$$

C. Resulting Algorithm

All what is needed for a practical algorithm has been presented so far. In an initialization step, priors θ_0, M_0, P_0 and Σ_0 are chosen and P_0^{-1} and Σ_0^{-1} are set. At time step i, a transition and the associated reward are observed. The two sets of sigma-points are computed from θ_{i-1}, P_{i-1} and Σ_{i-1}. Using these sigma-points and their images, statistics of interest $\bar{q}_i$, $P_{\omega q_i}$, P_{qq_i}, $\bar{p}_{q_i}$, $P_{\theta p_{q_i}}$, $P_{p_q p_{q_i}}$ and P_{u_i} are computed and used to compute quantities linked to statistical linearization. Eq. (20) is then used to compute K_i and update θ_{i-1} and M_{i-1}. Recall that parameters are updated according to Eq. (20). However, under the proposed sampling scheme, b_i and d_i are given by Eq. (29) and (38). Parameters update thus simplifies as:

$$\theta_i = \theta_{i-1} + K_i (r_i + \gamma \bar{p}_{q_i} - \bar{q}_i) \tag{46}$$

Therefore, it is not necessary to compute b_i and d_i. Finally, matrices P_{i-1}, P_{i-1}^{-1}, Σ_{i-1} and Σ_{i-1}^{-1} are updated. The proposed slLSTD is summarized in Alg. 1.

1) Algorithmic Complexity.: We consider first the Bellman evaluation operator. All operations are in $O(p^2)$ (computational and memory complexity) except the Cholesky decomposition which is in $O(p^3)$. However, as variances updates (30) and (39) are rank one updates, the Cholesky decomposition can be perfomed in $O(p^2)$ (*e.g.*, see [18]). Therefore, memory and computational complexities are in $O(p^2)$. If the Bellman optimality operator is considered, a maximum over actions has to be computed for each of the sigma-points. In this case, memory complexity is still in $O(p^2)$ and computational complexity is in $O(|A|p^2)$.

2) Algorithm Analysis.: We argue that slLSTD generalizes LSTD. An important question is to know if slLSTD reduces to LSTD with a linear parameterization and the evaluation operator.

Theorem 1 (slLSTD generalizes LSTD): With a linear parameterization and the Bellman evaluation operator, slLSTD provides the LSTD estimator.

Proof: Assume that the parameterization is linear:

$$\hat{Q}_\omega(s_i, a_i) = \phi(s_i, a_i)^T \omega = \phi_i^T \omega \tag{47}$$

In the case of a linear mapping, the unscented transform is no longer an approximation [15] and the statistics of interest admit an analytical expression:

$$\bar{q}_i = \phi_i^T \theta_{i-1}, \; P_{\omega q_i} = P_{i-1}\phi_i, \; P_{qq_i} = \phi_i^T P_{i-1} \phi_i \tag{48}$$

Therefore $A_i = \phi_i$ and $b_i = 0$. The Bellman evaluation operator being considered, $\hat{P}\hat{Q}_\theta$ is also linear and:

$$\hat{P}\hat{Q}_\theta(s_i, a_i) = \phi(s_{i+1}, a_{i+1})^T\theta = \phi_{i+1}^T\theta \tag{49}$$

The same reasoning holds, and $C_i = \phi_{i+1}$ and $d_i = 0$. Therefore, there is no statistical linearization error, thus the noise variance is unitary: $P_{u_i} = 1$. After a few algebraic manipulations, slLSTD resumes to (under priors θ_0 and M_0):

$$\begin{cases} K_i = \frac{M_{i-1}\phi_i}{1+(\phi_i - \gamma\phi_{i+1})M_{i-1}\phi_i^T} \\ \theta_i = \theta_{i-1} + K_i(r_i + (\gamma\phi_{i+1} - \phi_i)^T\theta_{i-1}) \\ M_i = M_{i-1} - K_i\left(M_{i-1}^T(\phi_i - \gamma\phi_{i+1})\right)^T \end{cases} \tag{50}$$

This is exactly the original LSTD algorithm [3] (in its recursive form), which proves the result. ■

An important corollary is that any LSTD theoretical result also holds for slLSTD with a linear parameterization and the evaluation operator. To know if they still hold in the general case remains an open question.

IV. Relation to previous works

The slLSTD algorithm is linked to the so-called Kalman Temporal Differences (KTD) framework [8][10] because both methods rely on a statistically linearized recursive least-squares approach. However, slLSTD aims at solving the projected fixed-point problem described above whereas KTD minimizes a square Bellman residual which leads to biased estimates of the value function when transitions of the MDP are stochastic [19].

The closest algorithms to slLSTD (except LSTD which we actually generalize) are probably nonlinear GTD2 and nonlinear TDC [4] (nlGTD2 and nlTDC for short). They minimize the same cost function (the distance between the Q-function and the projection of its image through the Bellman operator). The difference lies in the fact that they use a stochastic gradient descent instead of the proposed statistically linearized recursive least-squares (they are therefore not compared to slLSTD in Q-learning-like settings in the following, because of the $\max$ operator which prevents from computing a gradient). Notice that minimizing this cost function with a stochastic gradient descent is far from being straightforward, it needs some prior work on how is expressed the cost and it requires a *weight duplication trick*. See [4] for details.

Advantages of nlGTD2 and nlTDC compared to slLSTD are the following ones: these algorithms are provably convergent and they are computationaly efficient (a $O(p)$ complexity with p the number of parameters). However, they are less sample efficient than slLSTD (this last one being a second order algorithm), they do not allow taking into account the Bellman optimality operator (because they rely on a gradient computation, and $\max_a \hat{Q}_\theta(s,a)$ does not generally admit an analytical gradient respectively to θ), and they also rely on the computation of the gradient and of parts of the hessian contrary to slLSTD which only requieres to evaluate the nonlinear parameterization for a given state-action pair and a given set of parameters. Consequently, it is more simple to consider a nonlinear parameterization such as a multilayer perceptron (MLP) with slLSTD.

V. Experiments

The LSTD algorithm has been experimented previously on complex tasks. We propose to show that the proposed generalization handles well nonlinearities on some toy problems (easy to understand and interpret) and that it is more sample efficient than other similar algorithms, and we argue that slLSTD should scale up as well as LSTD.

Notice that in these experiments we used the *scaled* unscented transform (which is really close to the presented UT, but alleviates some of its problems, see [15] for details) with the following (standard) parameters: $\alpha_{\text{UT}} = 0.1$, $\beta_{\text{UT}} = 2$ and $\kappa_{\text{UT}} = 0$.

A. Tsitsiklis Chain

This first experiment aims at illustrating the ability of slLSTD to handle nonlinear parameterization. It consists in a 3 states valued Markov chain first proposed by [2] to illustrate possible TD divergence (because of bootstrapping). State i transits to state i with probability 0.5 and to state $i-1$ with probability 0.5 too (state 1 transiting to state 1 or 3 with equi-probability). The reward is always zero, therefore the optimal value function is zero. This chain is very simple, however a nonlinear parameterization which causes TD with function approximation divergence is considered here. Let $\epsilon = 0.05$, let I be the 3×3 identity matrix and Q the 3×3 matrix defined as:

$$Q = \begin{pmatrix} 1 & \frac{1}{2} & \frac{3}{2} \\ \frac{3}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} & 1 \end{pmatrix} \tag{51}$$

The value function is parameterized by a single scalar θ, its parameterization is given as (notice that here $\hat{V}_\theta$ is a 3×1 vector):

$$\hat{V}_\theta = \exp\left((Q + \epsilon I)\,\theta\right) V_0 \text{ with } V_0 = \begin{pmatrix} 10 & -7 & -3 \end{pmatrix}^T \tag{52}$$

The optimal parameter is obviously $\theta^* = -\infty$.

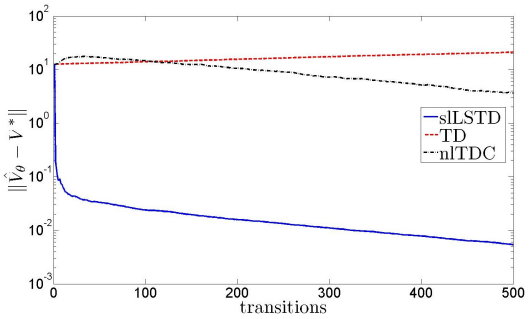

Fig. 2. Tsitsiklis chain.

TD, nlTDC [4] and slLSTD are compared, both initialized with $\theta_0 = 0$. For TD, the learning rate is chosen equal to $\alpha_i = 2.10^{-3}$, and for nlTDC $\alpha_i = 10^{-1}$ and $\beta_i = 10^{-2}$. For slLSTD, all priors (P_0, Σ_0 and M_0) are set to 10^4. Results

in Fig. 2 show the error $\|\hat{V} - V^*\|$ as a function of the number of observed transitions (notice the semi-log scale). TD estimates diverge (slowly), as expected, whereas nlTDC and slLSTD converge to the right estimate. However, from this experiment, slLSTD is much more sample efficient (this is due to the fact that slLSTD is a second order algorithm more than to the fact that parameters were not finely tuned).

B. Inverted Pendulum

This second experiment aims at illustrating the ability of slLSTD to handle the Bellman optimality operator. Recall that this operator implies strong nonlinearities (the $\max_a \hat{Q}_\theta(s,a)$ term) and that, as far as we know, we propose the first proper "Q-learning extension" of LSTD. Because induced nonlinearities do not admit an analytical gradient, nlTDC cannot be considered here. The inverted pendulum task 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 [12].

The slLSTD algorithm is considered with the Bellman optimality operator, and it is compared to the classic Q-learning with function approximation and the KTD-Q algorithm [8], which is another second-order value-iteration-like learning method also using the UT (see Sec. IV). All algorithms share the same 3×3 radial basis function (RBF) network per action (linear parameterization) of [12]. For all algorithms the initial parameter vector is set to zero. Training samples are collected online with random episodes (starting close to the equilibrium, which leads to an average length of 10). All algorithms learn from the same trajectories. Q-learning parameters are the same as for [12]. For slLSTD, priors (P_0, Σ_0 and M_0) are set to $10^2 I$. For KTD-Q, parameters are set to $P_{0|0} = 10^2 I$, $P_{n_i} = 1$ and $P_{v_i} = 0$ (see [8] for the meaning of these parameters). For each trial, learning is done over 500 episodes and the greedy policy tested every 50 episodes. Performance is measured as the number of steps in an episode (a maximum of 3000 being allowed, which corresponds to balancing the pendulum during 5 minutes). Results in Fig. 3 (semi-log scale) are averaged over 100 trials.

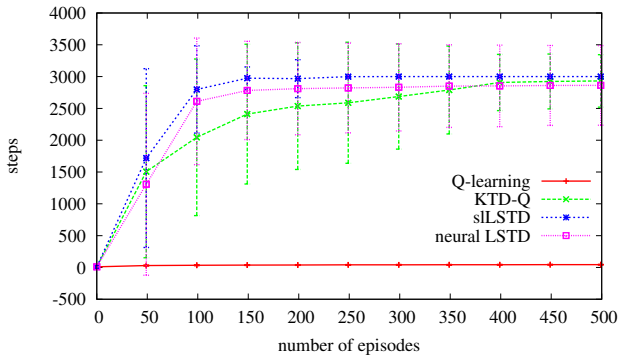

Fig. 3. Inverted pendulum.

KTD-Q and slLSTD learn an optimal policy (that is balancing the pole for the maximum number of steps) asymptotically and near-optimal policies are learned after only a few tens of episodes. The slLSTD learns faster and with less variance (however KTD-Q assumes a deterministic MDP, which is not the case here). With the same number of learning episodes, Q-learning with the same linear parameterization fails to learn a policy which balances the pole for more than a few tens of time steps.

We also consider the same experiment using a nonlinear parameterization for slLSTD, more precisely a multilayer perceptron (MLP), justifying the name neural LSTD. Notice that now both sources of nonlinearities (Bellman optimality operator and nonlinear parameterization) are considered at the same time. We use a 3-3-1 MLP (one input per state component and one for the action, three hidden neurons and one output) with $\tanh$ activation function. This involves much less parameters than the RBF network. Priors P_0 and Σ_0 are set to $10^{-2} I$ (in order to not sample sigma-points to far in the saturating zone) and M_0 to I (in order to have high enough first correction steps). Choosing these parameters requires some practice, however they were not finely tuned. Results are presented in figure 3. Neural LSTD performs slightly worst than the RBF-based slLSTD, but it requires much less parameters and the problem at hand involves more nonlinearities.

We also compare our results to the neural fitted-Q (NFQ) algorithm [20]. This is a batch value-iteration-like algorithm also using an MLP as the nonlinear parameterization. Recall that slLSTD is online and uses each sample only once. Results are reported on table I which compares the percentage of successful testing episodes (maintaining the pendulum for 3000 steps) as a function of the number of training random episodes. Our results are close to [20], although a little lower. There are two good reasons for this. First, we use a 3-3-1 MLP, whereas a 3-5-5-1 MLP (implying more parameters) is used in [20]. We have chosen such a parameterization because it involves less parameters than the RBF network, contrary to the 3-5-5-1 MLP. The second reason, which is more important, is that NFQ performs multiple passes through each sample (it is a batch algorithm), whereas neural LSTD uses each sample only one time. Therefore, this shows that neural LSTD is very sample efficient. This experiment also shows that the proposed slLSTD algorithm can be combined successfully with neural networks.

VI. Conclusion

In this paper, we have introduced and experimentally illustrated slLSTD, an efficient statistical linearization-based generalization of LSTD. The resulting online and second order algorithm handles well nonlinearities. This allows considering nonlinear parameterizations such as neural networks, as illustrated experimentally, and this also provides (thanks to the derivative-free nature of our contribution) a principled Q-learning-like extension of the LSTD algorithm. Being a second order algorithm, slLSTD is also more sample efficient than stochastic-gradient-descent-based algorithms such as nlTDC. We have also proved theoretically that slLSTD reduces to LSTD when considered with the Bellman evaluation operator and a linear parameterization.

Algorithm 1: Statistically Linearized LSTD algorithm (slLSTD)

Initialization;
Initialize vector θ_0, matrices M_0, P_0 and Σ_0 and compute matrices P_0^{-1} and Σ_0^{-1};

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

Observe current transition $(s_i, a_i, s_{i+1}, a_{i+1})$ and associated reward r_i;

Sigma-points computation ;
$\{\omega_i^{(j)},\, 0 \leq j \leq 2p\} =$
$\left[\theta_{i-1} \quad \theta_{i-1} \pm \left(\sqrt{(p+\kappa)P_{i-1}}\right)_j\right]$;
$\{\theta_i^{(j)},\, 0 \leq j \leq 2p\} =$
$\left[\theta_{i-1} \quad \theta_{i-1} \pm \left(\sqrt{(p+\kappa)\Sigma_{i-1}}\right)_j\right]$

Images of sigma-points computation ;
$\{q_i^{(j)} = \hat{Q}_{\omega_i^{(j)}}(s_i, a_i),\, 0 \leq j \leq 2p\}$;
$\{p_{q_i}^{(j)} = \hat{P}\hat{Q}_{\theta_i^{(j)}}(s_i, a_i),\, 0 \leq j \leq 2p\}$

Statistics of interest computation ;
$\bar{q}_i = \sum_{j=0}^{2p} w_j q_i^{(j)}$;
$P_{\omega q_i} = \sum_{j=0}^{2p} w_j(\omega_i^{(j)} - \theta_{i-1})(q_i^{(j)} - \bar{q}_i)$;
$P_{qq_i} = \sum_{j=0}^{2p} w_j(q_i^{(j)} - \bar{q}_i)^2$;
$\bar{p}_{q_i} = \sum_{j=0}^{2p} w_j p_{q_i}^{(j)}$;
$P_{\theta p_{q_i}} = \sum_{j=0}^{2p} w_j(\theta_i^{(j)} - \theta_{i-1})(p_{q_i}^{(j)} - \bar{p}_{q_i})$;
$P_{p_q p_{q_i}} = \sum_{j=0}^{2p} w_j(p_{q_i}^{(j)} - \bar{p}_{q_i})^2$;

Compute A_i **and** C_i ;
$A_i = P_{i-1}^{-1} P_{\omega q_i}$;
$C_i = \Sigma_{i-1}^{-1} P_{\theta p_{q_i}}$;

Compute noise variance ;
$P_{u_i} = 1 + \sum_{j=0}^{2p} w_j(q_i^{(j)} - \bar{q}_i - A_i^T(\omega_i^{(j)} - \theta_{i-1}) -$
$\gamma(p_{q_i}^{(j)} - \bar{p_{q_i}} - C_i^T(\theta_i^{(j)} - \theta_{i-1})))^2$;

Update parameters and M_{i-1} ;
$K_i = \frac{M_{i-1}A_i}{P_{u_i} + (A_i - \gamma C_i)M_{i-1}A_i^T}$;
$\theta_i = \theta_{i-1} + K_i(r_i + \gamma\bar{p}_{q_i} - \bar{q}_i)$;
$M_i = M_{i-1} - K_i(M_{i-1}^T(A_i - \gamma C_i))^T$;

Update P_{i-1} **and** P_{i-1}^{-1} ;
$P_i = P_{i-1} - P_{\omega q_i}(1 + P_{qq_i})^{-1} P_{\omega q_i}^T$;
$P_i^{-1} = P_{i-1}^{-1} + \frac{P_{i-1}^{-1} P_{\omega q_i} P_{\omega q_i}^T P_{i-1}^{-1}}{1 + P_{qq_i} - P_{\omega q_i}^T P_{i-1}^{-1} P_{\omega q_i}}$;

Update Σ_{i-1} **and** Σ_{i-1}^{-1} ;
$\Sigma_i = \Sigma_{i-1} - P_{\theta p_{q_i}}\left(1 + P_{p_q p_{q_i}}\right)^{-1} P_{\theta p_{q_i}}^T$;
$\Sigma_i^{-1} = \Sigma_{i-1}^{-1} + \frac{\Sigma_{i-1}^{-1} P_{\theta p_{q_i}} P_{\theta p_{q_i}}^T \Sigma_{i-1}^{-1}}{1 + P_{p_q p_{q_i}} - P_{\theta p_{q_i}}^T \Sigma_{i-1}^{-1} P_{\theta p_{q_i}}}$;

# random episodes	successful testing episodes: neural LSTD	NFQ
50	43 %	46 %
100	87 %	88 %
150	92 %	96 %
200	93 %	100 %
300	94 %	100 %
400	95 %	100 %

TABLE I

COMPARISON TO NFQ (RESULTS FOR NFQ ARE REPORTED FROM [20]).

Interesting perspectives would be to analyse theoretically the proposed algorithm and to study other sampling schemes. Combination of slLSTD with policy iteration (*e.g.*, neural LSPI) or actor-critics would also be of interest. Application of slLSTD (*e.g.*, combined with a neural network) to real-world problems is needed to assess its ultimate utility. Using such compact representations should allows addressing more complex tasks than the classic LSTD algorithm does.

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

[3] S. J. Bradtke and A. G. Barto, "Linear Least-Squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33–57, 1996.

[4] H. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. Sutton, "Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1204–1212.

[5] D. Choi and B. Van Roy, "A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning," *Discrete Event Dynamic Systems*, vol. 16, pp. 207–239, 2006.

[6] 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.

[7] 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.

[8] 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.

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

[10] M. Geist and O. Pietquin, "Eligibility Traces through Colored Noises," in *ICUMT 2010 (submitted)*, 2010.

[11] A. Nedić and D. P. Bertsekas, "Least Squares Policy Evaluation Algorithms with Linear Function Approximation," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 13, pp. 79–110, 2003.

[12] M. G. Lagoudakis and R. Parr, "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[13] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement Learning for Humanoid Robotics," in *Third IEEE-RAS International Conference on Humanoid Robots (Humanoids 2003)*, 2003.

[14] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*. Wiley, 1984.

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

[16] P. Nørgård, N. Poulsen, and O. Ravn, "New developments in state estimation for nonlinear systems," *Automatica*, vol. 36, no. 11, pp. 1627–1638, 2000.

[17] M. Geist and O. Pietquin, "Statistically Linearized Recursive Least Squares," in *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2010)*, Kittilä (Finland), August-September 2010, 5 pages, to appear.

[18] 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.

[19] 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, 2008.

[20] M. Riedmiller, "Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method," in *European Conference on Machine Learning*, 2005, pp. 317–328.