

Parametric Value Function Approximation: a Unified View

Matthieu Geist*

*IMS Research Group

Supélec

Metz, France

Email: matthieu.geist@supelec.fr

Olivier Pietquin*†

†UMI 2958

Georgia-Tech - CNRS

Metz, France

Email: olivier.pietquin@supelec.fr

Abstract—Reinforcement learning (RL) is a machine learning answer to the optimal control problem. It consists of learning an optimal control policy through interactions with the system to be controlled, the quality of this policy being quantified by the so-called value function. An important RL subtopic is to approximate this function when the system is too large for an exact representation. This survey reviews and unifies state of the art methods for parametric value function approximation by grouping them into three main categories: bootstrapping, residuals and projected fixed-point approaches. Related algorithms are derived by considering one of the associated cost functions and a specific way to minimize it, almost always a stochastic gradient descent or a recursive least-squares approach.

Index Terms—Reinforcement learning, value function approximation, survey.

I. INTRODUCTION

Optimal control of stochastic dynamic systems is a trend of research with a long history. Several points of view can be adopted according to the information available on the system such as a model of the physics ruling the system (automation) or a stochastic model of its dynamic (dynamic programming). The machine learning response to this recurrent problem is the Reinforcement Learning (RL) paradigm [1][2][3].

The system to be controlled is usually modeled as a Markov Decision Process [4] (MDP) $\{S, A, P, R, \gamma\}$. An MDP is made up of a set S of states (the different configurations of the system), a set A of actions (which cause a change of the system's state), a set $P : s, a \in S \times A \rightarrow p(\cdot|s, a) \in \mathcal{P}(S)$ of Markovian transition probabilities (the probability to transit from one state to another under a given action), a bounded reward function $R : s, a, s' \in S \times A \times S \rightarrow r = R(s, a, s') \in \mathbb{R}$ associating a scalar to each transition and a discounting factor γ which decreases long-term rewards' influence. How the agent acts with the system is modeled by a so-called policy, $\pi : s \in S \rightarrow \pi(\cdot|s) \in \mathcal{P}(A)$, which associates to each state a probability distribution over actions. The quality of such a policy is quantified by a so-called value function, $V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi]$, which associates to each state the expected cumulative discounted reward from starting in the considered state and then following the given policy. An optimal policy is one of those which maximize the associated value function for each state: $\pi^* \in \operatorname{argmax}_\pi V^\pi$.

Thanks to the Markovian property, value functions can be (more or less simply) computed using so-called Bellman equations. The value function of a given policy satisfies the (linear) Bellman evaluation equation and the optimal value function (which is linked to one of the optimal policies)

satisfies the (nonlinear) Bellman optimality equation. Alternatively, these value functions can be seen as being the fixed-points of associated Bellman operators (see Section II). These Bellman equations are very important for dynamic programming and reinforcement learning, as they allow computing the value function.

If the model (that is transition probabilities and the reward function) is known and if state and action spaces are small enough, the optimal policy can be computed using dynamic programming [5]. A first scheme, called policy iteration, consists in evaluating an initial policy (that is computing the associated value function using the linear Bellman evaluation equation) and then improving this policy, the new one being greedy respectively to the computed value function (it associates to each state the action which maximizes the expected cumulative reward obtained from starting in this state, applying this action and then following the initial policy). Evaluation and improvement are iterated until convergence (which occurs in a finite number of iterations). A second scheme, called value iteration, consists in computing directly the optimal value function (using the nonlinear Bellman optimality equation and an iterative scheme based on the fact that the value function is the unique fixed-point of the associated Bellman operator). The optimal policy is greedy respectively to the optimal value function. There is a third scheme, based on linear programming; however, it is not considered in this article.

RL aims at estimating the optimal policy without knowing the model and from interactions with the system. Value functions can no longer be computed exactly, they have to be estimated, which is the main scope of this paper. RL heavily relies on dynamic programming, in the sense that most of approaches are some sort of generalizations of value or policy iteration. A first problem is that computing a greedy policy (required for both schemes) from a value function requires the model to be known. The state-action value function (also named Q -function) alleviates this problem by providing an additional degree of freedom on the first action to be chosen: $Q^\pi(s, a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. A greedy policy can thus be obtained by maximizing the Q -function over actions, for any given state. The state-action value function generalizes the value function in the sense that $V^\pi(s) = E_{a|s, \pi}[Q^\pi(s, a)]$. Therefore, the rest of this paper focuses uniquely on the Q -function.

There are two main approaches to estimate an optimal policy. The first one, based on value iteration, consists in estimating directly the optimal state-action value function which is then used to derive an estimate of the optimal policy.

The second one, based on policy iteration, consists in mixing the estimation of the Q -function of the current policy (policy evaluation) with policy improvement in a generalized policy iteration scheme (generalized in the sense that evaluation and improvement processes interact, independently of the granularity and other details). This scheme presents many variations. Generally, the Q -function is not perfectly estimated when the improvement step occurs. Each change in the policy implies a change in the associated Q -function; therefore, the estimation process can be non-stationary. The policy can be derived from the estimated state-action value function (for example, using a Boltzmann distribution or an ϵ -greedy policy): this implies an underlying dilemma between exploration and exploitation. The policy can also have its own representation, which leads to actor-critic architectures.

All these approaches share a common subproblem: estimating the (state-action) value function (of a given policy or the optimal one directly). The aim of this paper is to review the more classic related algorithms by adopting an unifying view (an extended and more detailed version being provided in [6]). Before this, the underlying formalism is shortly presented.

II. PRELIMINARIES

Thanks to the Markovian property of transitions, the state-action value function Q^π of a given policy π satisfies the linear Bellman evaluation equation:

$$Q^\pi(s, a) = E_{s', a' | s, a, \pi} [R(s, a, s') + \gamma Q^\pi(s', a')].$$

A Bellman evaluation operator related to the Q -function, $T^\pi : Q \in \mathbb{R}^{S \times A} \rightarrow T^\pi Q \in \mathbb{R}^{S \times A}$, can be defined: $T^\pi Q(s, a) = E_{s', a' | s, \pi} [R(s, a, s') + \gamma Q(s', a')]$. This operator is a contraction and Q^π is its unique fixed-point: $Q^\pi = T^\pi Q^\pi$. Using it is not practical, as it requires knowing the model. Therefore, a sampled Bellman evaluation is introduced. For a transition $(s_i, a_i, s_{i+1}, a_{i+1})$, a_{i+1} being sampled according to π , and the associated reward r_i , it is defined as:

$$\hat{T}^\pi Q(s_i, a_i) = r_i + \gamma Q(s_{i+1}, a_{i+1}).$$

The optimal state-action value function Q^* satisfies the non-linear Bellman optimality equation:

$$Q^*(s, a) = E_{s' | s, a} [R(s, a, s') + \gamma \max_{a' \in A} Q^*(s', a')].$$

The associated Bellman optimality operator, $T^* : Q \in \mathbb{R}^{S \times A} \rightarrow T^* Q \in \mathbb{R}^{S \times A}$, is defined as: $T^* Q(s, a) = E_{s' | s, a} [R(s, a, s') + \gamma \max_{a' \in A} Q(s', a')]$. A more practical sampled Bellman optimality operator is also defined. Assume that a transition (s_i, a_i, s_{i+1}) and associated reward r_i are observed, it is given by:

$$\hat{T}^* Q(s_i, a_i) = r_i + \gamma \max_{a \in A} Q(s_{i+1}, a).$$

An important RL subtopic is to estimate the (state-action) value function of a given policy or directly the Q -function of the optimal policy from samples, that is observed trajectories of actual interactions. This article focuses on parametric approximation: the estimated state-action value function is of the form \hat{Q}_θ , where θ is the parameter vector; this estimate belongs to an hypothesis space $\mathcal{H} = \{\hat{Q}_\theta | \theta \in \mathbb{R}^p\}$ which specifies the architecture of the approximation. For example, if the state space is sufficiently small an exact tabular representation can be chosen for the value function. The estimate is thus of the form $\hat{Q}_\theta(s, a) = e_{s, a}^T \theta$ with $e_{s, a}$ being a unitary

TABLE I
SUMMARY.

| | bootstrap | residual | projected fixed-point direct / iterated | |
|-----------------------------------|---------------------------|-------------|--|---------------|
| stochastic gradient descent | TD SARSA Q-learning | R-SGD | (nl)GTD2 (nl)TDC GQ(λ) | |
| (recursive) least-squares | FPKF | GPTD KTD | LSTD sLSTD | LSPE Q-OSP |
| other | | | | fitted-Q |

vector which is equal to 1 in the component corresponding to the state-action couple (s, a) and 0 elsewhere. More complex hypothesis spaces can be envisioned, such as those generated by neural networks. Estimating a function from samples is a common topic of supervised learning. However, estimating a (state-action) value function is a more difficult problem because values are never directly observed, but just rewards which define them. Actually, value function approximation consists in estimating a fixed-point of the T operator (either T^π or T^*) from sampled trajectories (that is using the sampled \hat{T} operator): $\hat{Q}_\theta \approx T\hat{Q}_\theta$. This article reviews state of the art value function approximators by grouping them into three categories. We will show that all approaches minimize the following empirical cost function:

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

and they differs on how $\xi \in \mathbb{R}^p$ is instantiated. First, bootstrapping approaches ($\xi = \theta_{j-1}$, see Equation (3) in Section III) consist in treating value function approximation as a supervised learning problem. As values are not directly observable, they are replaced by an estimate computed using a sampled Bellman operator (bootstrapping refers to replacing an unobserved value by an estimate). Second, residual approaches ($\xi = \omega$, see Equation (4) in Section IV) consist in minimizing the square error between the (state-action) value function and its image through a Bellman operator. Practically, a sampled operator is used, which leads to biased estimates. Third, projected fixed-point approaches ($\xi = \theta_i$ or $\xi = \theta_{i-1}$ depending on direct or iterated resolution, see Equations (5,6) in Section V) minimize the squared error between the (state-action) value function and the projection of the image of this function under the (sampled) Bellman operator onto the hypothesis space (see Figure 1). Therefore, there are basically three cost function, which can be minimized either using a stochastic gradient descent or a recursive least-squares approaches. All algorithms discussed in this survey enter in this categorization, as summarized in Table I.

Moreover, we will show that most of them can be written in the form of a Widrow-Hoff update. Generally speaking, this means that parameters are updated proportionally to the prediction error made on the last observation. In the case of value function approximation, this prediction error takes the specific form of a so-called temporal difference error δ_i , which is the difference between right and left sides of a Bellman equation when the sampled operator is considered:

$$\delta_i = \hat{T} \hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i).$$

Therefore, most of presented algorithms can be written in the

following form:

$$\theta_i = \theta_{i-1} + K_i \delta_i, \quad (2)$$

where K_i is an algorithm-dependent gain which indicates in which direction the parameters are corrected (the temporal difference error quantifying the magnitude of this correction).

Notice that if this paper focuses on how to learn the (state-action) value function from samples, it is not concerned with how these samples are generated. Otherwise speaking, the control problem is not addressed. Nevertheless, this does not mean that we restrict ourselves to the on-policy learning setting (consider the Bellman optimality operator). Extensions of the reviewed algorithms to eligibility traces (see Section VI for a definition) are only mentioned.

III. BOOTSTRAPPING APPROACHES

Bootstrapping approaches cast the (state-action) value function approximation into a supervised learning problem. The (state-action) value of interest is assumed to be observed, and corresponding theoretical cost functions are considered, given that the (state-action) value function of a given policy π is evaluated or that the optimal Q -function is directly estimated. The corresponding theoretical cost function is written generically as: $J(\theta) = \|Q - \hat{Q}_\theta\|^2$. The corresponding empirical optimization problem for i observed state-action couples (s_j, a_j) and associated unobserved Q -values q_j is: $\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i (q_j - \hat{Q}_\omega(s_j, a_j))^2$. The principle of bootstrapping algorithms is to derive an online algorithm which minimizes the above cost function. Generally, such an algorithm can be written in the form of a Widrow-Hoff update which corrects the parameter vector in the direction provided by the algorithm-dependent gain K_i and with a magnitude proportional to the prediction error: $\theta_i = \theta_{i-1} + K_i (q_i - \hat{Q}_{\theta_{i-1}}(s_i, a_i))$. However, the Q -value q_i is never observed. It is where the bootstrapping principle applies: the unobserved q_i value is replaced by some estimate. Given that we are looking for a fixed-point of one of the Bellman operators, this estimate is provided by $\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i)$: $\theta_i = \theta_{i-1} + K_i (\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i))$. Alternatively, this can be seen as solving the following optimization problem [7]:

$$\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i \left(\hat{T}\hat{Q}_{\theta_{j-1}}(s_j, a_j) - \hat{Q}_\omega(s_j, a_j) \right)^2. \quad (3)$$

Practical algorithms are then derived given that this cost function is solved using a stochastic gradient descent or a recursive least-squares approach, and given what Bellman operator is considered.

A. Bootstrapped Stochastic Gradient Descent

Stochastic gradient descent consists in adjusting parameters by an amount proportional to an approximation of the gradient of the cost function, only evaluated on the last training example. Let α_i be a learning rate satisfying the classical stochastic approximation criterion, $\sum_{i=1}^{\infty} \alpha_i = \infty$ and $\sum_{i=1}^{\infty} \alpha_i^2 < \infty$. Applied to the bootstrapped cost-function (3), the stochastic gradient descent approach provides the following update rule [2] (assuming an initial parameter vector θ_0):

$$\begin{cases} K_i &= \alpha_i \left(\nabla_{\theta_{i-1}} \hat{Q}_\theta(s_i, a_i) \right) \\ \theta_i &= \theta_{i-1} + K_i \left(\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \right) \end{cases}.$$

Here the gain is the gradient, and the prediction error $\delta_i = \hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)$ is the temporal difference error announced in Section II.

With the Bellman evaluation operator and if the Q -function is replaced by the value function, it is the TD algorithm with function approximation; with the Bellman evaluation operator, this provides the SARSA algorithm; with the Bellman optimality operator, this is the Q-learning algorithm [2]. Under some assumptions, notably a linear parameterization hypothesis, TD with function approximation can be shown to be convergent [8]. However, this is no longer the case when it is combined with a nonlinear function approximator [8]. Despite this, one of the important successes of reinforcement learning is based on TD with neural network-based function approximation [9]. Convergence results holding for TD with function approximation apply rather directly to SARSA with function approximation. Under some strong assumptions, Q-learning with function approximation can be shown to be convergent too [10]. All these algorithms have been extended to eligibility traces [2]. However, notice that extending the Q-learning algorithm to eligibility traces should be done with caution, because of its off-policy aspect. This problem of combining off-policy learning with eligibility traces is discussed further later.

B. Bootstrapped Recursive Least-Squares

The fixed-point Kalman Filter (FPKF) [7] also seeks at minimizing the empirical cost function (3). However, the parameterization is assumed to be linear and a (recursive) least-squares approach is adopted instead of the stochastic gradient descent used for preceding algorithms. The considered hypothesis space is of the form $\mathcal{H} = \{\hat{Q}_\theta : (s, a) \in S \times A \rightarrow \phi(s, a)^T \theta \in \mathbb{R} | \theta \in \mathbb{R}^p\}$, where $\phi(s, a)$ is a feature vector (to be chosen beforehand). For a given state-action couple (s_j, a_j) , $\phi(s_j, a_j)$ is shortened as ϕ_j . Thanks to linearity in parameters, cost function (3) is convex and has a unique minimum. This optimization problem can be solved analytically by zeroing the gradient respectively to ω : this is the principle of the least-squares method. Moreover, thanks to the Sherman-Morrison formula¹, this estimation process can be made recursive (therefore recursive least-squares). This provides the following algorithm (assuming an initial parameter vector θ_0 as well as the associated $p \times p$ matrix P_0):

$$\begin{cases} K_i &= \frac{P_{i-1} \phi_i}{1 + \phi_i^T P_{i-1} \phi_i} \\ \theta_i &= \theta_{i-1} + K_i \left(\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \phi_i^T \theta_{i-1} \right) \\ P_i &= P_{i-1} - \frac{P_{i-1} \phi_i \phi_i^T P_{i-1}}{1 + \phi_i^T P_{i-1} \phi_i} \end{cases}.$$

This update is still in the form of a Widrow-Hoff update, with the aforementioned temporal difference error and with a gain K_i depending on the P_i matrix. It can be shown to be convergent under some assumptions, for both sampled operators (evaluation and optimality) [7]. As far as we know, it has not been extended to eligibility traces.

IV. RESIDUAL APPROACHES

Residual approaches aim at finding an approximation of the fixed-point of one of the Bellman operators by minimizing the distance between the (state-action) value function and its image through one of the Bellman operators. The

¹It allows inverting a rank-one perturbed matrix efficiently.

associated theoretical cost function is: $J(\theta) = \|\hat{Q}_\theta - T\hat{Q}_\theta\|^2$. Practically, learning is done using samples and the Bellman operator is replaced by a sampled Bellman operator, the model (particularly transition probabilities) being unknown. The associated empirical cost function is therefore:

$$\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i \left(\hat{T}\hat{Q}_\omega(s_j, a_j) - \hat{Q}_\omega(s_j, a_j) \right)^2 \quad (4)$$

A common drawback of all approaches aiming at minimizing this cost function is that they produce biased estimates of the (state-action) value function: minimizing the empirical cost function (4) does not lead to minimize the theoretical cost function $\|\hat{Q}_\theta - T\hat{Q}_\theta\|^2$ asymptotically. Basically, this is due to the fact that the expectation of a square is not the square of the expectation: $E[\|\hat{Q}_\theta - T\hat{Q}_\theta\|^2] = \|\hat{Q}_\theta - T\hat{Q}_\theta\|^2 + \operatorname{Var}(\hat{T}\hat{Q}_\theta)$. There is an unwanted variance term acting as a penalty factor which favors smooth functions. If such penalties are commonly used for regularization, this one is harmful as it cannot be controlled [11]. All methods presented below can be modified so as to handle the problem; this will be shortly discussed. However, it is important to note that any algorithm aiming at minimizing this cost presents this bias problem.

A. Residual Stochastic Gradient Descent

The so-called residual algorithms (R-SGD for residual stochastic gradient descent) [12] minimize the empirical cost function (4) using a stochastic gradient descent. The corresponding update rule is therefore (assuming an initial θ_0):

$$\begin{cases} K_i = \nabla_{\theta_{i-1}} \left(\hat{Q}_\theta(s_i, a_i) - \hat{T}\hat{Q}_\theta(s_i, a_i) \right) \\ \theta_i = \theta_{i-1} + \alpha_i K_i (\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)) \end{cases}$$

Once again, we obtain a Widrow-Hoff update. A first problem arises when the sampled Bellman optimality operator is considered. In this case, the gradient of the \max operator must be computed²: $\nabla_{\theta_{i-1}} (\max_{a \in A} \hat{Q}_\theta(s_{i+1}, a))$. Another problem is that these algorithms compute biased estimates of the (state-action) value function, as explained above. In order to handle this problem, a double sampling scheme can be used [12]. Let us consider the Bellman evaluation operator. Two transitions are independently generated from the current state-action couple. One of them is used to compute the gain K_i and the other one to compute the TD error δ_i . These two transitions being sampled independently, taking the expectation of $K_i \delta_i$ leads to the use of the true (that is unsampled) Bellman operator, without variance term contrary to the use of the same transition in both gain and TD error. However, this suggests that transitions can be sampled on demand (for example using a simulator), which can be a strong assumption. As far as we know, this algorithm has not been extended to eligibility traces.

B. Residual Least-Squares

The Gaussian Process Temporal Differences (GPTD) framework [13] models the value function as a Gaussian process [14]. A generative model linking rewards to values through the sampled Bellman evaluation operator and an additive noise is set, the Gaussian distribution of a state's value conditioned on past observed rewards is computed by

²Actually, $\max_{a \in A} \hat{Q}_\theta(s_{i+1}, a)$ is generally non-differentiable respectively to θ . A solution could be to rely on Fréchet sub-gradients.

performing Bayesian inference, and the value of this state is estimated as the mean of this Gaussian distribution. The associated variance quantifies the uncertainty of this estimate. Notice that the optimality operator cannot be considered in this framework because of a mandatory linearity assumption (linearity of the generative model linking). A problem is that the considered Gaussian process is actually a vector with as many components as the number of states encountered during learning. To alleviate this scaling problem, an online sparsification scheme is used [15], which actually constructs online a kernel-based linear parametric representation. If sparsification is done in a preprocessing step or if the representation is considered asymptotically (after an infinite number of interactions), the GPTD value function representation can be seen as a parametric one. Moreover, a parametric version of the GPTD framework exists [16]. It is the view adopted here: feature selection is an important topic, however this paper focuses on learning the parameters of a representation, not on learning the representation itself.

The (parametric) GPTD algorithm assumes a linear parameterization and the Bellman evaluation operator and minimizes cost function (4) using a classical recursive linear least-squares approach³. Using the same notations as before, Equation (4) can be rewritten as $\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i (r_j + \gamma \phi_{j+1}^T \omega - \phi_j^T \omega)^2$. Let note $\Delta \phi_j = \phi_j - \gamma \phi_{j+1}$. The unique parameter vector minimizing the above convex cost can be computed analytically and recursively (thanks to Sherman-Morrison), which provides the GPTD update (assuming initial θ_0 and P_0):

$$\begin{cases} K_i = \frac{P_{i-1} \Delta \phi_i}{1 + \Delta \phi_i^T P_{i-1} \Delta \phi_i} \\ \theta_i = \theta_{i-1} + K_i (r_i - \Delta \phi_i^T \theta_{i-1}) \\ P_i = P_{i-1} - \frac{P_{i-1} \Delta \phi_i \Delta \phi_i^T P_{i-1}}{P_{i-1} + \Delta \phi_i^T P_{i-1} \Delta \phi_i} \end{cases}$$

One can recognize the temporal difference error $\delta_i = r_i - \Delta \phi_i^T \theta_{i-1}$ and a gain K_i , to be linked again to the discussed Widrow-Hoff update. Notice that P_i is actually a variance matrix quantifying the uncertainty over current parameters estimation (it is the variance of the parameter vector conditioned on past i observed rewards). This is not clear from the proposed least-squares-based derivation, however it is direct by adopting a Bayesian (or even pure Kalmanian) perspective: under linear and Gaussian assumptions, recursive least-squares, Kalman filtering and Bayesian filtering provide all the same solution [17]. As all other residual methods, GPTD produces biased estimates of the value function when transitions are stochastic. Using a proper colored noise model (instead of the classical white noise assumption) leads to an unbiased estimate [18]. The corresponding algorithm is named MC-GPTD. However, this noise also induces a memory effect which prevents from learning in an off-policy manner, much like eligibility traces does [19]. Notice also that this leads to minimize another cost function, linking states' estimates to Monte Carlo samples of the discounted return [18].

The so-called Kalman Temporal Differences (KTD) framework [20][21] generalizes the GPTD framework: it handles nonlinear parameterizations (*e.g.*, multilayered perceptron [22]), it handles the Bellman optimality operator (which provide a sort of "Q-learning" extension of GPTD), and it

³This derivation differs from [13], its validity relies strongly on the link between Bayesian inference, Kalman filtering and recursive least-squares under Gaussian and linear assumptions [17].

allows learning in non-stationary conditions (the system to be controlled can be non-stationary, but also generalized policy iteration induces a non-stationary followed policy [23][24]). The idea behind the original derivation of KTD is to cast (state-action) value function approximation into the Kalman filtering paradigm [25], as developed in [21]. However, unscented Kalman filtering can be seen as a statistically linearized recursive least-squares approach [26]. Therefore, KTD can be seen as a second-order algorithm minimizing cost function (4) using a derivative-free statistical linearization. The full derivation of KTD using this recursive least-squares point of view follows ideas of [26] and is provided in [6]. As other residual approaches, KTD suffers from the bias problem when system transitions are stochastic. In order to handle this issue, a new colored noise model, based on the idea of eligibility traces and generalizing the GPTD colored noise [18], has been introduced [22]. This leads to the KTD(λ) family of algorithms; as far as we know, it is the only residual approach which makes use of eligibility traces (there is a GPTD(λ) algorithm [16], however it should be linked to projected fixed-point approaches, not to residual ones). Nevertheless, as mentioned before, this induces some memory effects which prevent from learning in an off-policy manner. Consequently, the sampled Bellman optimality operator can no longer be considered in this setting, because of its off-policy aspect. Using a colored noise also leads to minimize a slightly different cost function [22]. Notice also that the available uncertainty information (the P_i matrix) can be useful for the dilemma between exploration and exploitation [27].

V. PROJECTED FIXED-POINT APPROACHES

Projected fixed-point approaches seek at minimizing the distance between the estimated state-action value function \hat{Q}_θ and the projection $\Pi T\hat{Q}_\theta$ (the projection being noted Π) of the image $T\hat{Q}_\theta$ of this function under a Bellman operator onto the hypothesis space \mathcal{H} : $J(\theta) = \|\hat{Q}_\theta - \Pi T\hat{Q}_\theta\|^2$ with $\Pi f = \operatorname{argmin}_{f \in \mathcal{H}} \|f - \hat{f}\|^2$. This is illustrated in Figure 1. The state-action value function estimate \hat{Q}_θ lies in the hypothesis space \mathcal{H} . Its image under a Bellman operator $T\hat{Q}_\theta$ does not necessarily lies on this hypothesis space. Residual approaches of Section IV try to minimize the distance between these two functions, that is line ① in Figure 1, with the drawback that using a sampled Bellman operator leads to biased estimates, as discussed before. The function $T\hat{Q}_\theta$ can be projected onto the hypothesis space, this projection minimizing the distance between $T\hat{Q}_\theta$ and the hypothesis space (line ② in Figure 1). Projected fixed-point methods aim at minimizing the distance between this projection and \hat{Q}_θ , represented by line ③ in Figure 1.

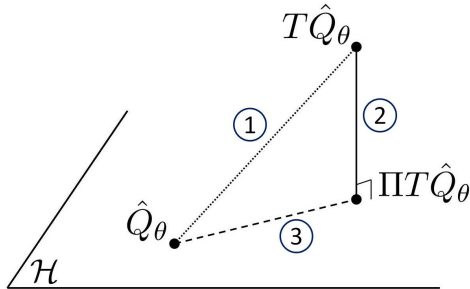


Fig. 1. Projected fixed-point principle.

The related empirical cost function is $\theta_i = \operatorname{argmin}_{\theta \in \mathbb{R}^p} \sum_{j=1}^i (\hat{Q}_\theta(s_j, a_j) - \hat{Q}_{\omega_\theta}(s_j, a_j))^2$, with $\omega_\theta = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i (\hat{Q}_\omega(s_j, a_j) - \hat{T}\hat{Q}_\theta(s_j, a_j))^2$. Obviously, this cost is minimized for $\theta_i = \omega_{\theta_i}$ (admitting that this equation has a solution), which summarizes the two nested optimization problems:

$$\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i \left(\hat{Q}_\omega(s_j, a_j) - \hat{T}\hat{Q}_{\theta_i}(s_j, a_j) \right)^2 \quad (5)$$

Notice that as θ_i appears in both sides of this equation, this is not a pure quadratic cost function. Least-squares methods are presented here before stochastic-gradient-based approaches for historical reasons.

A. Least-Squares-based Approaches

The least-squares temporal differences (LSTD) algorithm [28] assumes a linear parameterization and the (sampled) Bellman evaluation operator in order to solve the above optimization problem. It has been originally introduced as the minimization of a residual cost function. Using a sampled Bellman operator can be interpreted as a correlation between the noise and inputs in the corresponding observation model (therefore the noise is not white, which is a mandatory assumption for least-squares). This correlation can be shown to cause a bias (in this case, the bias presented in Section IV). A classic method to cope with this problem are instrumental variables [29], which were used to provide the first derivation of the LSTD algorithm. This point of view is historical. Later, it has been interpreted as a projected fixed-point minimization [30], and it is the point of view adopted here.

LSTD assumes a linear parameterization as well as the sampled Bellman evaluation operator. Using the same notations as before, optimization problem (5) can be rewritten as: $\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i (r_j + \gamma \phi_{j+1}^T \theta_i - \phi_j^T \omega)^2$. Thanks to linearity in ω (linear parameterization assumption), this can be analytically solved. Moreover, the update can be done recursively thanks to the Sherman-Morrison formula. This provides the LSTD algorithm (assuming initial θ_0 and $p \times p$ M_0 matrix):

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

Once again, K_i is a gain and $r_i - (\Delta \phi_i)^T \theta_{i-1}$ a temporal difference error, to be linked to a Widrow-Hoff update. This algorithm has been extended to eligibility traces [31] and can be shown to be convergent under general assumptions [32].

The statistically linearized LSTD (sLSTD) algorithm [33] generalizes LSTD: it does not assume a linear parameterization nor the Bellman evaluation operator. Notably, this allows using nonlinear parameterization such as multilayered perceptrons and it provides a ‘‘Q-learning extension’’ of the LSTD algorithm. How sLSTD generalizes LSTD is very close to how KTD generalizes GPTD: a statistical linearization is performed, which allows solving the related optimization problem analytically. See [33][6] for details.

B. Stochastic Gradient Descent-based Approaches

GTD2 and TDC algorithms [34] aim at minimizing cost (5) while considering the Bellman evaluation operator, and they differs on the route taken to express the gradient followed

to perform the stochastic gradient descent. Both methods rely on a linear parameterization, and are based on a re-worked expression of the cost function. Let $\delta_j(\theta) = r_j + \gamma\phi_{j+1}^T\theta - \phi_j^T\theta$ be the temporal difference error, Equation (5) can be rewritten as $\theta_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} J_i(\omega)$ with $J_i(\omega) = (\sum_{j=1}^i \phi_j \delta_j(\omega))^T (\sum_{j=1}^i \phi_j \phi_j^T)^{-1} (\sum_{j=1}^i \phi_j \delta_j(\omega))$. This result is far from being direct, see [34][6] for details. The negative gradient of this cost is given by: $-\frac{1}{2}\nabla_{\theta} J_i(\theta) = (\sum_{j=1}^i \Delta\phi_j \phi_j^T) w_i$ with $w_i = (\sum_{j=1}^i \phi_j \phi_j^T)^{-1} (\sum_{j=1}^i \delta_j(\theta) \phi_j)$. In order to avoid a bias problem, a second modifiable parameter vector $\omega \in \mathbb{R}^p$ is used to form a quasi-stationary estimate of the term $w_i = (\sum_{j=1}^i \phi_j \phi_j^T)^{-1} (\sum_{j=1}^i \delta_j(\theta) \phi_j)$, this being called the weight-doubling trick. Parameter vector θ is updated according to a stochastic gradient descent: $\theta_i = \theta_{i-1} + \alpha_i \Delta\phi_i \phi_i^T w_{i-1}$. There remains to find an update rule for w_i . In order to obtain a $O(p)$ algorithm, it is estimated using a stochastic gradient descent too [34]. One can remark that w_i is actually the solution of a linear least-squares optimization problem: $w_i = \operatorname{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^i (\phi_j^T \omega - \delta_j(\theta))^2$. This suggests the following update rule: $w_i = w_{i-1} + \beta_i \phi_i (\delta_i(\theta_{i-1}) - \phi_i^T w_{i-1})$. Learning rates satisfy the classical stochastic approximation criterion. Moreover, they are chosen such that $\beta_i = \eta \alpha_i$ with $\eta > 0$. The GTD2 algorithm is thus (assuming an initial θ_0, ω_0 being set to zero):

$$\begin{cases} \theta_i &= \theta_{i-1} + \alpha_i \Delta\phi_i \phi_i^T w_{i-1} \\ w_i &= w_{i-1} + \beta_i \phi_i (\delta_i(\theta_{i-1}) - \phi_i^T w_{i-1}) \end{cases}$$

Contrary to previous algorithms, the GTD update is not in the form of Equation (2). However, one can recognize from this update rule that $w^T \phi(s, a)$ is actually a parameterization of the advantage function $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ (the temporal difference error being a bootstrapped estimate of the advantage [35]). Under some assumptions, this algorithm can be shown to be convergent [34].

By expressing the gradient in a slightly different way, another algorithm called TDC can be derived, the difference being how the θ parameter vector is updated. The gradient can be rewritten as: $-\frac{1}{2}\nabla_{\theta} J_i(\theta) = (\sum_{j=1}^i \delta_j(\theta) \phi_j) - \gamma (\sum_{j=1}^i \phi_{j+1} \phi_j^T) w_i$. This gives rise to the following update for θ, ω being updated as before: $\theta_i = \theta_{i-1} + \alpha_i \phi_i \delta_i(\theta_{i-1}) - \alpha_i \gamma \phi_{i+1} \phi_i^T w_{i-1}$. This algorithm is called TD with gradient correction because the first term, $\alpha_i \phi_i \delta_i(\theta_{i-1})$, is the same as for TD with function approximation (see Section III-A), and the second term, $-\alpha_i \gamma \phi_{i+1} \phi_i^T w_{i-1}$, acts as a correction. For TDC, learning rates α_i and β_i are chosen such as satisfying the classic stochastic approximation criterion, and such that $\lim_{i \rightarrow \infty} \frac{\alpha_i}{\beta_i} = 0$. This means that θ_i is updated on a slower time-scale. The idea behind this is that w_i should look stationary from the θ_i point of view. The TDC algorithm can be summarized as follows (assuming an initial θ_0, ω_0 being set to zero):

$$\begin{cases} \theta_i &= \theta_{i-1} + \alpha_i \phi_i \delta_i(\theta_{i-1}) - \alpha_i \gamma \phi_{i+1} \phi_i^T w_{i-1} \\ w_i &= w_{i-1} + \beta_i \phi_i (\delta_i(\theta_{i-1}) - \phi_i^T w_{i-1}) \end{cases}$$

As GTD, TDC is not in the form of Equation (2), but ω is still a parameter vector to be linked to the advantage function. This algorithm can also be shown to be convergent under some assumptions, see [34] again.

GTD2 and TDC algorithms have been extended to the case of a general nonlinear parameterization \hat{Q}_θ , as long

as it is differentiable respectively to θ [36]. The underlying idea is to perform a projection onto the tangent plane of the hypothesis space. The TDC algorithm has also been extended to eligibility traces [37]. Moreover, this algorithm, called GQ(λ), allows off-policy learning, that is learning the value of one target policy while following another behavioral policy. This new algorithm (for which some convergence guarantees can be provided) still minimizes the empirical cost function linked to (5). However, instead of the T^π Bellman operator considered so far, an eligibility-based T_λ^π operator is used (λ being the eligibility factor), this operator being defined as the expected λ -return. See [37] for more details. Using eligibility traces induces a memory effect which prevents from learning in an off-policy manner without caution, see [19] for example. To cope with this problem, [37] use ideas from the importance sampling field (as [19] actually). They present GQ(λ) as an extension of Q -learning, which can be misleading. Actually, they consider off-policy learning (with a known, fixed, target policy), but not the Bellman optimality operator. Nevertheless, the TDC algorithm has also been extended to this operator [38] (this new algorithm being called Greedy-GQ). To do so, the Bellman evaluation operator T^{π_θ} for a policy π_θ , which depends on the currently estimated state-action value function (through parameters θ), is considered. Therefore, the learnt policy is non-stationary (it evolves with parameters' estimation). If π_θ is greedy respectively to the learnt value function, then it is equivalent to considering the Bellman optimality operator. However, in this case, there are some non-differentiability problems (due to the max operator), and Fréchet subgradients are required [38]. They also consider the case where π_θ is a stochastic Gibbs policy built upon the currently estimated value function. In this case, there are no differentiability problems. There is a convergence analysis for these algorithms [38].

C. Iterated-Solving-based Approaches

Methods presented so far in Section V aim at minimizing the distance between the state-action value function and the projection of the image of this function under a Bellman operator onto the hypothesis space: $J(\theta) = \|\hat{Q}_\theta - \Pi T \hat{Q}_\theta\|^2$. This can be interpreted as trying to find a fixed-point of the operator ΠT , which is the composition of the projection operator Π and of one of the Bellman operator T . Assuming that this operator is a contraction (which is not always the case, it depends on the projection operator), there exists a unique fixed-point which can be found by iterating the application of the ΠT operator: $\hat{Q}_{\theta_i} = \Pi T \hat{Q}_{\theta_{i-1}}$. Methods presented in this section adopt this point of view to provide algorithms.

Fitted-Q is a batch algorithm (therefore, it cannot be linked to the Widrow-Hoff update). It assumes that a set of N transitions is available beforehand: $\{(s_j, a_j, r_j, s_{j+1}, a_{j+1})\}_{1 \leq j \leq N}$. An initial Q -function \hat{Q}_{θ_0} is (more or less arbitrary) initialized, and estimates are refined by iterating the ΠT operator: $\hat{Q}_{\theta_i} = \Pi \hat{T} \hat{Q}_{\theta_{i-1}}, \forall i > 0$. A sampled Bellman operator is used, because as usual transition probabilities are not known. Most of time, fitted-Q suggests the sampled Bellman optimality operator, but this approach is of course still valid for the sampled Bellman evaluation operator. The Π operator indeed represents any supervised learning algorithm matching inputs (s_j, a_j) to pseudo-outputs $\hat{T} \hat{Q}_{\theta_{i-1}}(s_j, a_j)$. Notice that the representation of

the estimated state-action value function is not necessarily parametric (e.g., if a support vector machine is used as the supervised learning algorithm). The fitted-Q idea probably dates back to [39]. Its convergence properties have been analyzed in [40] through the contraction property of the ΠT operator. Its performance bounds in L_p norm has been analyzed in [41]. This is particularly judicious: if performance bounds of supervised learning algorithms are very often analyzed in L_p norm, this is not the case for (approximate) dynamic programming which is most of the time analyzed in L_∞ norm. Fitted-Q has been considered with many different function approximators: kernel-based regression [42], neural networks [43], tree-based methods [44], etc.

The least-squares policy evaluation (LSPE) algorithm has been directly introduced using the concept of eligibility traces [45], but this aspect is left apart in this article. It can be roughly seen as a fitted-Q algorithm using a linear parameterization, the (sampled) Bellman evaluation operator and for which a new training sample is added to the training set at each iteration. Thanks to linearity (linear parameterization and evaluation operator), an efficient online algorithm can be obtained.

The LSPE algorithm solves the following optimization problem:

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

Thanks to the linearity (linear parameterization and Bellman evaluation operator), and using the Sherman-Morrison formula, a recursive estimate can be provided (assuming initial θ_0 and B_0 , A_0 and b_0 being set to 0):

$$\begin{cases} B_i^{-1} &= B_{i-1}^{-1} - \frac{B_{i-1}^{-1} \phi_i \phi_i^T B_{i-1}^{-1}}{1 + \phi_i^T B_{i-1}^{-1} \phi_i} \\ A_i &= A_{i-1} + \phi_i (\Delta \phi_i)^T \\ b_i &= b_{i-1} + \phi_i r_i \\ \theta_i &= \theta_{i-1} + B_i^{-1} (b_i - A_i \theta_{i-1}) \end{cases}$$

Actually, the way LSPE is presented here differs from [45] in the sense that the B_i^{-1} matrix is originally scaled with a learning rate. The algorithm presented here is the case where the learning rate is chosen constant and equal to one, which can be shown to be convergent [46]. This Widrow-Hoff update is not of the form of Eq. 2, but one can remark that the term $b_i - A_i \theta_{i-1} = \sum_{j=1}^i \phi_j \delta_j(\theta_{i-1})$ implies the temporal differences errors of all transitions encountered so far. Notice that ideas behind LSPE have other applications [47] and can also be linked to variational inequalities [48].

The LSPE algorithm has been extended to the Bellman optimality operator in the case of optimal stopping problems [49], a restrictive class of MDPs. Convergence of the resulting Q-OSP algorithm can still be ensure, however obtaining a computationally efficient recursive estimate requires some approximations.

VI. CONCLUSION

This article has reviewed a large part of the state of the art in (state-action) value function approximation. Basically, it has been shown that all these approaches can be categorized in three main classes, given the considered cost function (related to bootstrapping, residual or projected fixed-point). In each of these groups, they can be categorized given that the cost function is minimized using a stochastic gradient descent

or a recursive least-squares approach (except fitted-Q, which can be considered with any supervised learning algorithm). Projected fixed-point approaches can be divided into two approaches, given that the cost function is directly minimized or that the underlying possible fixed-point is searched for using an iterative scheme. All of this is summarized in Table I. A link between the Widrow-Hoff update and most of the reviewed methods has also be drawn through this article.

A point not discussed so far is the computational (and memory) complexity of the reviewed algorithms. There are basically two cases. For stochastic gradient descent-based algorithms, complexity is in $O(p)$, and for recursive least-squares-based approach, complexity is in $O(p^2)$. Notice that for algorithms handling the Bellman optimality operator, these complexities have to be scaled by the number of actions (because of the max computation). Exceptions (to the two above cases) are fitted-Q (which complexity depends on the considered supervised learning scheme) and Q-OSP. Considering the sample efficiency of these algorithms, second order approaches are usually more efficient.

All algorithms for value function approximation have not been discussed here. Notably, most of the presented ones have been extended to eligibility traces. Basically, methods presented here are based on a one-step prediction (the reward plus the estimate of the value function in the transition state), whereas eligibility traces are based on a weighted average of multiple step predictions [2]. Such an approach as a clear advantage for methods based on stochastic gradient descent, as it speeds up the learning. However, for least-squares-based approaches, this advantage is less clear, notably because these methods are generally much more sample efficient [31]. Moreover, eligibility traces present a (possibly severe) drawback: they induces a memory effect which prevents off-policy learning without caution. Off-policy learning is actually still possible, but it implies to add some importance sampling scheme [19]. By the way, using eligibility traces can be expressed as using a modified T^λ Bellman operator (see [37] for example), and the work presented here can globally be extended to this case.

Other approaches extend algorithms presented here. For example, LSTD has been kernelized [50] as well as LSPE [51]. A variation of LSTD has been proposed by [52] who use the sparsity of used feature vectors to reduce the computational complexity. The LSTD algorithm has also been extended to L_2 regularization in reproducing kernel Hilbert spaces [53] as well as to L_1 regularization [54]. These are examples, among other.

REFERENCES

- [1] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 3rd ed. The MIT Press, March 1998.
- [3] O. Sigaud and O. Buffet, Eds., *Markov Decision Processes and Artificial Intelligence*. Wiley - ISTE, 2010.
- [4] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.
- [5] R. Bellman, *Dynamic Programming*, 6th ed. Dover Publications, 1957.
- [6] M. Geist and O. Pietquin, "A Brief Survey of Parametric Value Function Approximation," Supélec, Tech. Rep., September 2010.
- [7] 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.
- [8] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, 1997.

- [9] G. Tesauro, "Temporal Difference Learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, March 1995.
- [10] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proceedings of the 25th International Conference on Machine Learning*, 2009, pp. 664–671.
- [11] 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.
- [12] L. C. Baird, "Residual Algorithms: Reinforcement Learning with Function Approximation," in *Proc. of the International Conference on Machine Learning (ICML 95)*, 1995, pp. 30–37.
- [13] 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.
- [14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
- [15] Y. Engel, S. Mannor, and R. Meir, "The Kernel Recursive Least Squares Algorithm," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2275–2285, 2004. [Online]. Available: <http://www.cs.ualberta.ca/yaki/>
- [16] Y. Engel, "Algorithms and Representations for Reinforcement Learning," Ph.D. dissertation, Hebrew University, April 2005.
- [17] Z. Chen, "Bayesian Filtering : From Kalman Filters to Particle Filters, and Beyond," Adaptive Systems Lab, McMaster University, Tech. Rep., 2003.
- [18] Y. Engel, S. Mannor, and R. Meir, "Reinforcement Learning with Gaussian Processes," in *Proceedings of the International Conference on Machine Learning (ICML 05)*, 2005.
- [19] D. Precup, R. S. Sutton, and S. P. Singh, "Eligibility Traces for Off-Policy Policy Evaluation," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 00)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 759–766.
- [20] M. Geist, O. Pietquin, and G. Fricout, "Kalman Temporal Differences: the deterministic case," in *Proceedings of the IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, April 2009.
- [21] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research (JAIR)*, 2010.
- [22] —, "Eligibility Traces through Colored Noises," in *Proceedings of the IEEE International Conference on Ultra Modern Control Systems (ICUMT 2010)*. Moscow (Russia): IEEE, October 2010.
- [23] C. W. Phua and R. Fitch, "Tracking value function dynamics to improve reinforcement learning with piecewise linear function approximation," in *Proceedings of the International Conference on Machine Learning (ICML 07)*, 2007.
- [24] 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 (Thailand): Springer, 2009.
- [25] 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.
- [26] 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): IEEE, 2010.
- [27] —, "Managing Uncertainty within Value Function Approximation in Reinforcement Learning," in *Active Learning and Experimental Design Workshop (collocated with AISTATS 2010)*, ser. Journal of Machine Learning Research - Workshop and Conference Proceedings, Sardinia, Italy, 2010.
- [28] 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.
- [29] T. Söderström and P. Stoica, "Instrumental variable methods for system identification," *Circuits, Systems, and Signal Processing*, vol. 21, pp. 1–9, 2002.
- [30] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.
- [31] J. A. Boyan, "Technical Update: Least-Squares Temporal Difference Learning," *Machine Learning*, vol. 49, no. 2-3, pp. 233–246, 1999.
- [32] H. Yu, "Convergence of Least Squares Temporal Difference Methods Under General Conditions," in *International Conference on Machine Learning (ICML 2010)*, 2010, pp. 1207–1214.
- [33] M. Geist and O. Pietquin, "Statistically Linearized Least-Squares Temporal Differences," in *Proceedings of the IEEE International Conference on Ultra Modern Control Systems (ICUMT 2010)*. Moscow (Russia): IEEE, October 2010.
- [34] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 993–1000.
- [35] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Incremental natural actor-critic algorithms," in *Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2007.
- [36] H. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton, "Convergent temporal-difference learning with arbitrary smooth function approximation," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., 2009, pp. 1204–1212.
- [37] H. R. Maei and R. S. Sutton, "GQ(λ): a general gradient algorithm for temporal-differences prediction learning with eligibility traces," in *Third Conference on Artificial General Intelligence*, 2010.
- [38] H. R. Maei, C. Szepesvari, S. Bhatnagar, and R. S. Sutton, "Toward Off-Policy Learning Control with Function Approximation," in *27th conference on Machine Learning (ICML 2010)*, 2010.
- [39] A. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal on Research and Development*, pp. 210–229, 1959.
- [40] G. Gordon, "Stable Function Approximation in Dynamic Programming," in *Proceedings of the International Conference on Machine Learning (ICML 95)*, 1995.
- [41] R. Munos, "Performance Bounds in Lp norm for Approximate Value Iteration," *SIAM Journal on Control and Optimization*, 2007.
- [42] D. Ormoneit and S. Sen, "Kernel-Based Reinforcement Learning," *Machine Learning*, vol. 49, pp. 161–178, 2002.
- [43] M. Riedmiller, "Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method," in *European Conference on Machine Learning (ECML)*, 2005.
- [44] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-Based Batch Mode Reinforcement Learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.
- [45] 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.
- [46] D. P. Bertsekas, V. Borkar, and A. Nedic, *Learning and Approximate Dynamic Programming*. IEEE Press, 2004, ch. Improved Temporal Difference Methods with Linear Function Approximation, pp. 231–235.
- [47] D. P. Bertsekas and H. Yu, "Projected Equation Methods for Approximate Solution of Large Linear Systems," *Journal of Computational and Applied Mathematics*, vol. 227, pp. 27–50, 2007.
- [48] D. P. Bertsekas, "Projected Equations, Variational Inequalities, and Temporal Difference Methods," in *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.
- [49] H. Yu and D. P. Bertsekas, "Q-Learning Algorithms for Optimal Stopping Based on Least Squares," in *Proceedings of European Control Conference*, Kos, Greece, 2007.
- [50] X. Xu, D. Hu, and X. Lu, "Kernel-Based Least Squares Policy Iteration for Reinforcement Learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, July 2007.
- [51] T. Jung and D. Polani, "Kernelizing LSPE(λ)," in *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007, pp. 338–345.
- [52] A. Geramifard, M. Bowling, and R. S. Sutton, "Incremental Least-Squares Temporal Difference Learning," in *21st Conference of American Association for Artificial Intelligence (AAAI 06)*, 2006, pp. 356–361.
- [53] A. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor, "Regularized policy iteration," in *22nd Annual Conference on Neural Information Processing Systems (NIPS 21)*, Vancouver, Canada, 2008.
- [54] J. Z. Kolter and A. Y. Ng, "Regularization and Feature Selection in Least-Squares Temporal Difference Learning," in *proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, Montreal Canada, 2009.