A Brief Survey of Parametric Value Function Approximation

Matthieu Geist matthieu.geist@supelec.fr
Olivier Pietquin olivier.pietquin@supelec.fr
IMS Research Group
Supélec
Metz, France

Supélec Technical Report (september 2010)

Abstract

Reinforcement learning is a machine learning answer to the optimal control problem. It consists in 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 subtopic of reinforcement learning is to compute an approximation of this value function when the system is too large for an exact representation. This survey reviews 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.

Keywords: Reinforcement learning, value function approximation, survey

Contents

1. 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, in which an artificial agent learns an optimal control policy through interactions with the dynamic system (also considered as its environment). After each interaction, the agent receives an immediate scalar reward information and the optimal policy it searches for is the one that maximizes the cumulative reward over the long term.

The system to be controlled is usually modeled as a Markovian Decision Process (MDP). An MDP is made up of a set of states (the different configurations of the system), a set of actions (which cause a change of the system's state), a set of Markovian transition probabilities (the probability to transit from one state to another under a given action; the Markovian property states that the probability depends on the current state-action pair and not on the path followed to reach it), a reward function associating a scalar to each transition and a discounting factor which decreases long-term rewards' influence. How the agent acts with the system is modeled by a so-called policy which associates to each state a probability distribution over actions. The quality of such a policy is quantified by a so-called value function which associates to each state the expected cumulative discounted reward from starting in the considered state and then following the given policy (expectation being done over all possible trajectories). An optimal policy is one of those which maximize the associated value function for each state.

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. These Bellman equations are very important for dynamic programming and reinforcement learning, as they allow computing the value function. If the Markovian hypothesis is not satisfied (something known as partial observability or perceptual aliasing), there are roughly two solutions: the first one is to transform the problem such as to work with something for which the Markovian property holds and the second one is to stop using Bellman equations. In the rest of this article, it is assumed that this property holds.

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. 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.

Reinforcement learning aims at estimating the optimal policy without knowing the model and from interactions with the system. Value functions can no longer be computed, they have to be estimated, which is the main scope of this paper. Reinforcement learning 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 (or Q-) function alleviate this problem by providing an additional degree of freedom on the first action to be chosen. It is defined, for a given policy and for a state-action couple, as the expected discounted cumulative reward starting in the given state, applying the given action and then following the fixed policy. A greedy policy can thus be obtained by maximizing the Q-function over actions.

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 (with the drawback that errors in the Q-function estimation can lead to a bad derived 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 (which is known as optimistic policy iteration). 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). There is generally an underlying dilemma between exploration and exploitation. At each time step, the agent should decide between acting greedily respectively to its uncertain and imperfect knowledge of the world (exploitation) and taking another action which improves this knowledge and possibly leads to a better policy (exploration). The policy can also have its own representation, which leads to actor-critic architectures. The actor is the policy, and the critic is an estimated value or Q-function which is used to correct the policy representation.

All these approaches share a common subproblem: estimating the (state-action) value function (of a given policy or the optimal one directly). This issue is even more difficult when state or action spaces are too large for a tabular representation, which implies to use some approximate representation. Generally speaking, estimating a function from samples is addressed by the supervised learning paradigm. However, in reinforcement learning, the (state-action) values are not directly observed, which renders the underlying estimation problem more difficult. Despite this, a number of (state-action) value function approximators have been proposed in the past decades. The aim of this paper is to review the more classic ones by adopting an unifying view which classifies them into three main categories: bootstrapping approaches, residual approaches and projected fixed-point approaches. Each of these approaches is related to a specific cost function, and algorithms are derived considering one of these costs and a specific way to minimize it (almost always a stochastic

gradient descent or a recursive least-squares approach). Before this, the underlying formalism is presented. More details can be found in reference textbooks (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Sigaud and Buffet, 2010).

2. Preliminaries

A Markovian decision process (MDP) is a tuple $\{S, A, P, R, \gamma\}$ where S is the (finite) state space, A the (finite) action space, $P : s, a \in S \times A \rightarrow p(.|s, a) \in \mathcal{P}(S)$ the family of Markovian transition probabilities, $R : s, a, s' \in S \times A \times S \rightarrow r = R(s, a, s') \in \mathbb{R}$ the bounded reward function and γ the discount factor weighting long term rewards. According to these definitions, the system stochastically steps from state to state conditionally to the actions the agent performed. Let i be the discrete time step. To each transition (s_i, a_i, s_{i+1}) is associated an immediate reward r_i. The action selection process is driven by a policy $\pi : s \in S \rightarrow \pi(.|s) \in \mathcal{P}(A)$.

The quality of a policy is quantified by the value function V^π, defined as the expected discounted cumulative reward starting in a state s and then following the policy π:

$$V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi] \tag{1}$$

Thanks to the Markovian property, the value function of a policy π satisfies the linear Bellman evaluation equation:

$$V^\pi(s) = E_{s',a|s,\pi}[R(s, a, s') + \gamma V^\pi(s')] \tag{2}$$

$$= \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V^\pi(s')) \tag{3}$$

Let define the Bellman evaluation operator T^π:

$$T^\pi : V \in \mathbb{R}^S \rightarrow T^\pi V \in \mathbb{R}^S : T^\pi V(s) = E_{s',a|s,\pi}[R(s, a, s') + \gamma V(s')] \tag{4}$$

The operator T^π is a contraction and V^π is its unique fixed-point:

$$V^\pi = T^\pi V^\pi \tag{5}$$

For a practical purpose, the *sampled* Bellman operator $\hat{T}^\pi$ is defined as the Bellman operator for a sampled transition. Assume that a transition (s_i, s_{i+1}) and associated reward r_i are observed, then:

$$\hat{T}^\pi V(s_i) = r_i + \gamma V(s_{i+1}) \tag{6}$$

An optimal policy π^* maximizes the associated value function for each state: $\pi^* \in \text{argmax}_{\pi \in \mathcal{P}(A)^S} V^\pi$. The associated optimal value function, noted V^*, satisfies the nonlinear Bellman optimality equation:

$$V^*(s) = \max_{a \in A} E_{s'|s,\pi}[R(s, a, s') + \gamma V^*(s')] \tag{7}$$

$$= \max_{a \in A} \sum_{s' \in S} p(s'|s, a)(R(s, a, s') + \gamma V^*(s')) \tag{8}$$

Notice that if the optimal value function is unique, it is not the case for the optimal policy. For this, consider an MDP with one state, two actions, each one providing the same reward: any policy is optimal. Let define the Bellman optimality operator T^*:

$$T^* : V \in \mathbb{R}^S \to T^*V \in \mathbb{R}^S : T^*V(s) = \max_{a \in A} E_{s'|s}[R(s,a,s') + \gamma V(s')] \tag{9}$$

The operator T^* is a contraction and V^* is its unique fixed-point:

$$V^* = T^*V^* \tag{10}$$

Remark that a sampled Bellman optimality operator cannot be defined for the value function, as the maximum depends on the expectation.

The state-action value (or Q-) function provides an additional degree of freedom on the choice of the first action to be applied. This provides useful in a model-free context. It is defined as the expected cumulative reward starting in a state s, taking an action a and then following the policy π:

$$Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi] \tag{11}$$

The state-action value function Q^π also satisfies the linear Bellman evaluation equation:

$$\begin{aligned} Q^\pi(s,a) &= E_{s',a'|s,a,\pi}[R(s,a,s') + \gamma Q^\pi(s',a')] & (12)\\ &= \sum_{s' \in S} p(s'|s,a)(R(s,a,s') + \gamma \sum_{a' \in A} \pi(a'|s') Q^\pi(s',a')) & (13) \end{aligned}$$

It is clear that value and state-action value functions are directly linked:

$$V^\pi(s) = E_{a|s,\pi}[Q^\pi(s,a)] \tag{14}$$

A Bellman evaluation operator related to the Q-function can also be defined. By a slight abuse of notation, it is also noted T^π, the distinction being clear from the context.

$$T^\pi : Q \in \mathbb{R}^{S \times A} \to T^\pi Q \in \mathbb{R}^{S \times A} : T^\pi Q(s,a) = E_{s',a'|s,\pi}[R(s,a,s') + \gamma Q(s',a')] \tag{15}$$

This operator is also a contraction and Q^π is its unique fixed-point:

$$Q^\pi = T^\pi Q^\pi \tag{16}$$

Similarly to what has been done for the value function, a sampled Bellman evaluation is also introduced. For a transition $(s_i, a_i, s_{i+1}, a_{i+1})$ and the associated reward r_i, it is defined as:

$$\hat{T}^\pi Q(s_i, a_i) = r_i + \gamma Q(s_{i+1}, a_{i+1}) \tag{17}$$

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

$$\begin{aligned} Q^*(s,a) &= E_{s'|s,a}[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a')] & (18)\\ &= \sum_{s' \in S} p(s'|s,a)(R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a')) & (19) \end{aligned}$$

The associated Bellman optimality operator is defined as (with the same slight abuse of notation):

$$T^* : Q \in \mathbb{R}^{S \times A} \rightarrow T^*Q \in \mathbb{R}^{S \times A} : T^*Q(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{a' \in A} Q(s',a')] \quad (20)$$

This is still a contraction and Q^* is its unique fixed-point:

$$Q^* = T^*Q^* \quad (21)$$

Contrary to the optimality operator related to the value function, here the maximum does not depend on the expectation, but the expectation depends on the maximum. Consequently, a sampled Bellman optimality operator can be defined. Assume that a transition (s_i, a_i, s_{i+1}) and associated reward r_i are observed, it is given by:

$$\hat{T}^*Q(s_i,a_i) = r_i + \gamma \max_{a \in A} Q(s_{i+1},a) \quad (22)$$

As mentioned in Section 1, an important subtopic of reinforcement learning 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 estimate value (resp. state-action value) function is of the form $\hat{V}_\theta$ (resp. $\hat{Q}_\theta$), where θ is the parameter vector; this estimate belongs to an hypothesis space $\mathcal{H} = \{\hat{V}_\theta \text{ (resp. } \hat{Q}_\theta) | \theta \in \mathbb{R}^p\}$ 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 $\hat{V}_\theta(s) = e_s^T\theta$ with e_s being an unitary vector which is equal to one in the component corresponding to state s and zero elsewhere. More complex hypothesis spaces can be envisioned, such as neural networks. However, notice that some of the approaches reviewed in this paper do not allow handling nonlinear representations.

Estimating a function from samples is a common topic of supervised learning. However, estimating a (state-action) value function is a more difficult problem. Indeed, values are never directly observed, just rewards which define them. Therefore supervised learning techniques cannot be directly applied to learn such a function. This article reviews state of the art value function (parametric) approximators by grouping them into three categories. First, bootstrapping approaches 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 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 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.

Notice that if this paper focuses on how learning the (state-action) value function from samples, it is not concerned with how this samples are generated. Otherwise speaking, the control problem is not addressed, as announced before. Extension of the proposed algorithms to eligibility traces are not considered too (but this is briefly discussed in the conclusion).

3. Bootstrapping Approaches

Bootstrapping approaches deal with (state-action) value function approximation as 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 optimal value function estimation is not considered because it does not allow defining an associated sampled Bellman optimality operator):

$$J_{V^\pi}(\theta) = \|V^\pi - \hat{V}_\theta\|^2 \tag{23}$$

$$J_{Q^\pi}(\theta) = \|Q^\pi - \hat{Q}_\theta\|^2 \tag{24}$$

$$J_{Q^*}(\theta) = \|Q^* - \hat{Q}_\theta\|^2 \tag{25}$$

Related algorithms depend on what cost function is minimized (actually on what associated empirical cost function is minimized) and how it is minimized (gradient descent or recursive least-squares approaches in the subsequent reviewed methods). However, resulting algorithms make use of a (state-action) value which is actually not observed. Bootstrapping consists in replacing this missing observation by a pseudo-observation computed by applying a sampled Bellman operator to the current estimate of the (state-action) value function.

3.1 Bootstrapped Stochastic Gradient Descent

Algorithms presented in this section aim at estimating respectively the value function of a given policy (TD), the Q-function of a given policy (SARSA) or directly the optimal state-action value function (Q-learning) by combining the bootstrapping principle with a stochastic gradient descent over the associated empirical cost function (Sutton and Barto, 1998).

3.1.1 TD with Function Approximation

TD with function approximation (TD-VFA) aims at estimating the value function V^π of a fixed policy π. Its objective function is the empirical cost function linked to (23). Let the notation v_j^π depict a (possibly noisy, as long as the noise is additive and white) observation of $V^\pi(s_j)$. The empirical cost is:

$$\hat{J}_{V^\pi}(\theta) = \sum_j \left(v_j^\pi - \hat{V}_\theta(s_j)\right)^2 \tag{26}$$

More precisely, TD with function approximation minimizes this empirical cost function using a stochastic gradient descent: parameters are adjusted by an amount proportional to an approximation of the gradient of cost function (23), only evaluated on a single training example. Let α_i be a learning rate satisfying the classical stochastic approximation criterion:

$$\sum_{i=1}^{\infty} \alpha_i = \infty, \quad \sum_{i=1}^{\infty} \alpha_i^2 < \infty \tag{27}$$

Parameters are updated according to the following Widrow-Hoff equation, given the i^{th} observed state s_i:

$$\theta_i = \theta_{i-1} - \frac{\alpha_i}{2}\nabla_{\theta_{i-1}}\left(v_i^\pi - \hat{V}_\theta(s_i)\right)^2 \tag{28}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{V}_\theta(s_i)\right)\left(v_i^\pi - \hat{V}_{\theta_{i-1}}(s_i)\right) \tag{29}$$

However, as mentioned above, the value of the state s_i is not observed. It is where the bootstrapping principle applies. The unobserved value v_i^π is replaced by an estimate computed by applying the sampled Bellman evaluation operator (6) to the current estimate $\hat{V}_{\theta_{i-1}}(s_i)$. Assume that not only the current state is observed, but the whole transition (s_i, s_{i+1}) (sampled according to policy π) as well as associated reward r_i. The corresponding update rule is therefore:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{V}_\theta(s_i)\right)\left(\hat{T}^\pi\hat{V}_{\theta_{i-1}}(s_i) - \hat{V}_{\theta_{i-1}}(s_i)\right) \tag{30}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{V}_\theta(s_i)\right)\left(r_i + \gamma\hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)\right) \tag{31}$$

The idea behind using this sampled operator (and more generally behind bootstrapping) is twofold: if parameters are perfectly estimated and if the value function belongs to the hypothesis space, this provides an unbiased estimate of the actual value (the value function being the fixed point of the unsampled operator) and this estimate provides more information as it is computed using the observed reward. Under some assumptions, notably a linear parameterization hypothesis, TD with function approximation can be shown to be convergent, see Tsitsiklis and Van Roy (1997). However, this is no longer the case when it is combined with a nonlinear function approximator, see Tsitsiklis and Van Roy (1997) again for a counterexample. Despite this, one of the important success of reinforcement learning is based on TD with neural network-based function approximation (Tesauro, 1995).

3.1.2 SARSA with Function Approximation

SARSA with function approximation (SARSA-VFA) aims at estimating the Q-function of a fixed policy π. Notice that usually SARSA is presented combined with an ϵ-greedy policy, which is a control component. This paper focuses on the pure estimation problem, therefore SARSA should be here really understood as Q-function evaluation, independently from the control scheme. Notice also that an MDP defines a valued Markov chain over the state-action space, therefore value and Q-function evaluation are two very close problems. Let q_j^π be a (possibly noisy, as long as the noise is additive and white) observation of $Q^\pi(s_j)$. The considered algorithm aims at minimizing the following empirical cost function, linked to (24):

$$\hat{J}_{Q^\pi}(\theta) = \sum_j \left(q_j^\pi - \hat{Q}_\theta(s_j, a_j)\right)^2 \tag{32}$$

SARSA with function approximation also minimizes the related empirical cost function using a stochastic gradient descent. Parameters are thus updated as follows, given the i^{th}

state-action pair (s_i, a_i):

$$\theta_i = \theta_{i-1} - \frac{\alpha_i}{2}\nabla_{\theta_{i-1}}\left(q_i^\pi - \hat{Q}_\theta(s_i, a_i)\right)^2 \tag{33}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(q_i^\pi - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{34}$$

As before, q_i^π is not observed and it is replaced by an estimate computed by applying the sampled Bellman evaluation operator to the current estimate $\hat{Q}_{\theta_{i-1}}(s_i, a_i)$. Assume that the whole transition $(s_i, a_i, s_{i+1}, a_{i+1})$, a_{i+1} being sampled according to policy π, as well as associated reward r_i are observed. Parameters are therefore updated according to:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(\hat{T}^\pi\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{35}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(r_i + \gamma\hat{Q}_{\theta_{i-1}}(s_{i+1}, a_{i+1}) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{36}$$

From a practical point of view, using the Q-function instead of the value function is of interest because it does not require the model (transition probabilities and reward function) to be known in order to derive a greedy policy. Convergence results holding for TD with function approximation apply rather directly to SARSA with function approximation.

3.1.3 Q-learning with Function Approximation

Q-learning with function approximation (QL-VFA) aims at estimating directly the optimal state-action value function Q^*. Let q_j^* be a (possibly noisy, as long as the noise is additive and white) observation of $Q^*(s_j)$. This algorithm aims at minimizing the empirical cost function linked to (25):

$$\hat{J}_{Q^*}(\theta) = \sum_j \left(q_j^* - \hat{Q}_\theta(s_j, a_j)\right)^2 \tag{37}$$

The same approach is used, and parameters are recursively estimated using a stochastic gradient descent. Given the i^{th} state-action pair (s_i, a_i), parameters should be updated according to:

$$\theta_i = \theta_{i-1} - \frac{\alpha_i}{2}\nabla_{\theta_{i-1}}\left(q_i^* - \hat{Q}_\theta(s_i, a_i)\right)^2 \tag{38}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(q_i^* - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{39}$$

As for preceding algorithms, the bootstrapping principle is applied to estimate the unobserved q_i^* value, using the sampled Bellman optimality operator now. Assume that the transition (s_i, a_i, s_{i+1}) as well as associated reward r_i are observed. Notice that Q-learning with function approximation is an off-policy algorithm, which means that it can evaluate a policy (the optimal one in this case) from samples generated according to a different policy. Practically, transitions can be sampled according to any sufficiently explorative policy. Parameters are updated according to:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(\hat{T}^*\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{40}$$

$$= \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\hat{Q}_\theta(s_i, a_i)\right)\left(r_i + \gamma\max_{a\in A}\hat{Q}_{\theta_{i-1}}(s_{i+1}, a) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{41}$$

Under some assumptions, notably a linear parameterization hypothesis, Q-learning with function approximation can be shown to be convergent (Melo et al., 2009).

3.1.4 An Unified View

These algorithms can be formalized using the same unified notation. First, value and Q-function evaluation (TD and SARSA with function approximation) are somehow redundant. As $V^\pi(s) = E_{a|\pi,s}[Q^\pi(s,a)]$, any algorithm aiming at estimating a Q-function can easily be specialized to the related value function, as long as the policy is known (this therefore notably does not apply to Q-learning with function approximation). Practically, it consists in replacing the Q-function by the value function and state-action pairs by states. Consequently, value function estimation is not considered anymore in this paper. Let $\hat{T}$ denote either the evaluation or the optimality operator, depending on the context. Let also q_j be either q_j^π or q_j^*, also depending on the context. SARSA and Q-learning with function approximation aim at minimizing the following empirical cost function, which is instantiated by specifying if evaluation or direct optimization is considered:

$$\hat{J}(\theta) = \sum_j \left(q_j - \hat{Q}_\theta(s_j, a_j)\right)^2 \tag{42}$$

As before, parameters are estimated using a stochastic gradient descent and by applying the bootstrapping principle, which leads to the following update:

$$\theta_i = \theta_{i-1} + \alpha_i \left(\nabla_{\theta_{i-1}} \hat{Q}_\theta(s_i, a_i)\right) \left(\hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)\right) \tag{43}$$

A practical algorithm is instantiated by specifying which of the sampled Bellman operator is used for $\hat{T}$, that is $\hat{T}^\pi$ or $\hat{T}^*$. Algorithms have been detailed so far, for the sake of clarity. However, in the rest of this article this summarizing point of view is adopted. Notice also that this update is actually a Widrow-Hoff equation of the following form:

$$\theta_i = \theta_{i-1} + K_i \delta_i \tag{44}$$

In this expression, $\delta_i = \hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)$ is the so-called temporal difference (TD) error, which is the reward prediction error given the current estimate of the state action value function, and which depends on what Bellman operator is considered, and $K_i = \alpha_i \nabla_{\theta_{i-1}} \hat{Q}_\theta(s_i, a_i)$ is a gain indicating in what direction the parameter vector should be corrected in order to improve the estimate. Most of (online) algorithms presented in this paper satisfy a Widrow-Hoff update.

3.2 Fixed-point Kalman Filter (a Bootstrapped Least-Squares Approach)

The fixed-point Kalman Filter (FPKF) of Choi and Van Roy (2006) also seeks at minimizing the empirical cost function linking (actually unobserved) state-action values to the estimated Q-function (still with a bootstrapping approach):

$$\hat{J}_i(\theta) = \sum_{j=1}^{i} \left(q_j - \hat{Q}_\theta(s_j, a_j)\right)^2 \tag{45}$$

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

$$\mathcal{H} = \{\hat{Q}_\theta : (s,a) \in S \times A \rightarrow \phi(s,a)^T\theta \in \mathbb{R} | \theta \in \mathbb{R}^p\} \tag{46}$$

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. The corresponding empirical objective function can thus be rewritten as:

$$\hat{J}_i(\theta) = \sum_{j=1}^{i} \left(q_j - \phi_j^T\theta\right)^2 \tag{47}$$

Thanks to linearity in parameters, this cost function is convex and has a unique minimum:

$$\theta_i = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}}\, \hat{J}_i(\theta) \tag{48}$$

This optimization problem can be solved analytically by zeroing the gradient of $\hat{J}_i(\theta)$; this is the principle of the least-squares method. Parameters are thus estimated as:

$$\theta_i = \left(\sum_{j=1}^{i} \phi_j\phi_j^T\right)^{-1} \sum_{j=1}^{i} \phi_j q_j \tag{49}$$

Let write $P_i^{-1} = \sum_{j=1}^{i} \phi_j\phi_j^T$. The Sherman-Morrison formula allows updating directly the inverse of a rank-one perturbed matrix:

$$P_i = P_{i-1} - \frac{P_{i-1}\phi_i\phi_i^T P_{i-1}}{1 + \phi_i^T P_{i-1}\phi_i} \tag{50}$$

This allows estimating parameters recursively:

$$\theta_i = \theta_{i-1} + \frac{P_{i-1}\phi_i}{1 + \phi_i^T P_{i-1}\phi_i}\left(q_i - \hat{Q}_{\theta_{i-1}}(s_i,a_i)\right) \tag{51}$$

As for algorithms presented in Section 3.1, the bootstrapping principle is applied and the unobserved q_i state-action value is replaced by the estimate $\hat{T}\hat{Q}_{\theta_{i-1}}(s_i,a_i)$:

$$\theta_i = \theta_{i-1} + \frac{P_{i-1}\phi_i}{1 + \phi_i^T P_{i-1}\phi_i}\left(\hat{T}\hat{Q}_{\theta_{i-1}}(s_i,a_i) - \hat{Q}_{\theta_{i-1}}(s_i,a_i)\right) \tag{52}$$

As for algorithms of Section 3.1, this equation is actually a Widrow-Hoff update (44). The temporal difference error is still $\delta_i = \hat{T}\hat{Q}_{\theta_{i-1}}(s_i,a_i) - \hat{Q}_{\theta_{i-1}}(s_i,a_i)$ (this prediction error term is actually common to all algorithms aiming at estimating the state-action value function which can be expresse as a Widrow-Hoff update). The gain depends on the fact that a least-squares minimization has been considered:

$$K_i = \frac{P_{i-1}\phi_i}{1 + \phi_i^T P_{i-1}\phi_i} \tag{53}$$

If the sampled Bellman evaluation operator is considered, this update rule specializes as:

$$\theta_i = \theta_{i-1} + \frac{P_{i-1}\phi_i}{1 + \phi_i^T P_{i-1}\phi_i}\left(r_i + \gamma\phi_{i+1}^T\theta_{i-1} - \phi_i^T\theta_{i-1}\right) \tag{54}$$

If the sampled Bellman optimality operator is considered, this update rule specializes as:

$$\theta_i = \theta_{i-1} + \frac{P_{i-1}\phi_i}{1 + \phi_i^T P_{i-1}\phi_i}\left(r_i + \gamma\max_{a\in A}\left(\phi(s_{i+1},a)^T\theta_{i-1}\right) - \phi_i^T\theta_{i-1}\right) \tag{55}$$

This algorithm can be shown to be convergent under some assumptions, for both sampled operators (evaluation and optimality). See Choi and Van Roy (2006) for details.

4. 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 associated cost function is:

$$J(\theta) = \|\hat{Q}_\theta - T\hat{Q}_\theta\|^2 \tag{56}$$

Practically, learning is done using samples and the Bellman operator is replaced by a sampled Bellman operator, the model (particularly transition probabilities) being not known. The associated empirical cost function is therefore:

$$\hat{J}(\theta) = \sum_j \left(\hat{Q}_\theta(s_j,a_j) - \hat{T}\hat{Q}_\theta(s_j,a_j)\right)^2 \tag{57}$$

A common drawback of all approaches aiming at minimizing this cost function is that they produce biased estimates of the (state-action) value function. Basically, this is due to the fact that the expectation of a square is not the square of the expectation:

$$E[(\hat{Q}_\theta(s,a) - \hat{T}\hat{Q}_\theta(s,a))^2] = (\hat{Q}_\theta(s,a) - T\hat{Q}_\theta(s,a))^2 + \mathrm{Var}(\hat{T}\hat{Q}_\theta(s,a)) \tag{58}$$

There is an unwanted variance term acting as a penalty factor which favorises smooth functions. If such penalties are commonly used for regularization, this one is harmful here as it cannot be controlled. See Antos et al. (2008) for a discussion of this aspect. All methods presented below can be modified so as to handle this problem, this will be shortly discussed. However, it is important to note that any algorithm aiming at minimizing this cost presents this bias problem.

4.1 Residual Stochastic Gradient Descent

So-called residual algorithms (R-SGD for residual stochastic gradient descent) have been introduced by Baird (1995). Their principle is to minimize the empirical cost function (57) using a stochastic gradient descent. The corresponding update rule is therefore:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\nabla_{\theta_{i-1}}\left(\hat{Q}_\theta(s_i,a_i) - \hat{T}\hat{Q}_\theta(s_i,a_i)\right)\right)\left(\hat{T}\hat{Q}_{\theta_{i-1}}(s_i,a_i) - \hat{Q}_{\theta_{i-1}}(s_i,a_i)\right) \tag{59}$$

Here again, this update is actually a Widrow-Hoff equation (44) with $\delta_i = \hat{T}\hat{Q}_{\theta_{i-1}}(s_i, a_i) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)$ and $K_i = \alpha_i \nabla_{\theta_{i-1}} (\hat{Q}_\theta(s_i, a_i) - \hat{T}\hat{Q}_\theta(s_i, a_i))$. If the sampled Bellman evaluation operator is considered, gain and temporal difference error are given by:

$$K_i = \alpha_i \nabla_{\theta_{i-1}} \left(\hat{Q}_\theta(s_i, a_i) - \gamma \hat{Q}_\theta(s_{i+1}, a_{i+1}) \right) \tag{60}$$

$$\delta_i = r_i + \gamma \hat{Q}_{\theta_{i-1}}(s_{i+1}, a_{i+1}) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \tag{61}$$

A first problem arises when the sampled Bellman optimality operator is considered. In this case, gain and TD error are:

$$K_i = \alpha_i \nabla_{\theta_{i-1}} \left(\hat{Q}_\theta(s_i, a_i) - \gamma \max_{a \in A} \hat{Q}_\theta(s_{i+1}, a) \right) \tag{62}$$

$$\delta_i = r_i + \gamma \max_{a \in A} \hat{Q}_{\theta_{i-1}}(s_{i+1}, a) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \tag{63}$$

In this case, the gradient of the max operator must be computed: $\nabla_{\theta_{i-1}}(\max_{a \in A} \hat{Q}_\theta(s_{i+1}, a))$. This is far from being straightforward, and Baird (1995) does not propose a solution to this issue, even if he introduces this update rule[1]. Another problem is that these algorithms compute biased estimates of the (state-action) value function, as explained above. This is inherent to all approaches minimizing a residual cost function using a sampled Bellman operator. In order to handle this problem, Baird (1995) proposes to use a double sampling scheme. Let consider the Bellman evaluation operator. Two transitions are independently generated from the state-action couple (s_i, a_i): $(s_i, a_i, r'_i, s'_{i+1}, a'_{i+1})$ and $(s_i, a_i, r''_i, s''_{i+1}, a''_{i+1})$. One of these transitions is used to compute the gain, and the other one to compute the TD error:

$$K_i = \alpha_i \nabla_{\theta_{i-1}} \left(r_i + \gamma \hat{Q}_\theta(s'_{i+1}, a'_{i+1}) - \hat{Q}_\theta(s_i, a_i) \right) \tag{64}$$

$$\delta_i = r_i + \gamma \hat{Q}_{\theta_{i-1}}(s''_{i+1}, a''_{i+1}) - \hat{Q}_{\theta_{i-1}}(s_i, a_i) \tag{65}$$

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:

$$E[K_i \delta_i] = \alpha_i \left(\nabla_{\theta_{i-1}} \left(\hat{Q}_\theta(s_i, a_i) - T^\pi \hat{Q}_\theta(s_i, a_i) \right) \right) \left(\hat{Q}_{\theta_{i-1}}(s_i, a_i) - T^\pi \hat{Q}_{\theta_{i-1}}(s_i, a_i) \right) \tag{66}$$

However, this suggests that transitions can be sampled on demand (for example using a simulator), which can be a strong assumption.

4.2 Residual Least-Squares

In this section, methods based on a least-squares minimization of cost function (57) are reviewed. The Gaussian Process Temporal Differences (Engel et al., 2003) algorithm minimizes it by assuming a linear parameterization as well as the Bellman evaluation operator, and the Kalman Temporal Differences framework (Geist et al., 2009; Geist and Pietquin, 2010b) generalizes it to nonlinear parameterizations as well as to the Bellman optimality operator thanks to a statistical linearization (Anderson, 1984).

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

4.2.1 Gaussian Process Temporal Differences

Engel et al. (2003) introduced the so-called Gaussian Process Temporal Differences (GPTD) framework. The underlying principle is to model the value function as a Gaussian process (that is a set of jointly Gaussian random variables, random variables being here values of each state). 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 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). 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 problem, Engel et al. (2003) propose an online sparsification scheme. It relies on the fact that the covariance matrix of any Gaussian process actually defines a Mercer kernel which can be viewed as an inner product in a high (possibly infinite) dimensional Hilbert space (Scholkopf and Smola, 2001). Using an approximate linear independence argument, the sparsification procedure only retains states (observed during learning) which help defining an approximate basis in this Hilbert space.

This sparsification scheme actually constructs online a kernel-based linear parametric representation. As announced in Section 2, this paper focuses on (pure) parametric representations. However, 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, Engel (2005) proposes a parametric version of the Gaussian Process Temporal Differences framework (not necessarily assuming that the linear parameterization is based on Mercer kernels). 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 is now derived more formally, however from a different (least-squares-based) point of view. Validity of the oncoming derivation relies strongly on the link between Bayesian inference, Kalman (1960) filtering and recursive least-squares under Gaussian and linear assumptions (Chen, 2003).

The (parametric) GPTD algorithm actually minimizes a cost function close to (57) using a classical linear least-squares approach. To apply it, linearity is mandatory. The state-action value function is assumed linear, $\hat{Q}_\theta(s_j, a_j) = \phi(s_j, a_j)^T\theta = \phi_j^T\theta$, and only the (sampled) Bellman evaluation operator is considered. An observation model (actually the sampled Bellman evaluation equation) using a not necessarily unitary noise n_i is considered:

$$r_j = \phi_j^T\theta - \gamma\phi_{j+1}^T\theta + n_j \tag{67}$$

Notice that this equation actually models the temporal difference error as a white noise:

$$r_j = \phi_j^T\theta - \gamma\phi_{j+1}^T\theta + n_j \Leftrightarrow n_j = \hat{T}\hat{Q}_\theta(s_j, a_j) - \hat{Q}_\theta(s_j, a_j) \tag{68}$$

Let write P_{n_j} the variance of n_j, its effect is to weight square terms in the minimized cost function, this is the slight difference with cost (57):

$$J_i(\theta) = \sum_{j=1}^{i} \frac{1}{P_{n_j}} \left(r_j + \gamma \phi_{j+1}^T \theta - \phi_j^T \theta\right)^2 \tag{69}$$

Let note $\Delta\phi_j = \phi_j - \gamma\phi_{j+1}$. The unique parameter vector minimizing the above convex cost-function can be computed analytically:

$$\theta_i = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} J_i(\theta) \tag{70}$$

$$= \left(\sum_{j=1}^{i} \frac{1}{P_{n_j}} \Delta\phi_j \Delta\phi_j^T\right)^{-1} \sum_{j=1}^{i} \frac{1}{P_{n_j}} \Delta\phi_j^T r_j \tag{71}$$

Let write $P_i = (\sum_{j=1}^{i} \Delta\phi_j \Delta\phi_j^T)^{-1}$. Thanks to the Sherman-Morrison formula, P_i can be computed iteratively and parameters can be estimated recursively. Let θ_0 and P_0 be some priors, the (parametric) GPTD algorithm is given by these two equations:

$$\theta_i = \theta_{i-1} + \frac{P_{i-1}\Delta\phi_i}{P_{n_i} + \Delta\phi_i^T P_{i-1} \Delta\phi_i} \left(r_i - \Delta\phi_i^T \theta_{i-1}\right) \tag{72}$$

$$P_i = P_{i-1} - \frac{P_{i-1}\Delta\phi_i \Delta\phi_i^T P_{i-1}}{P_{n_i} + \Delta\phi_i^T P_{i-1} \Delta\phi_i} \tag{73}$$

One can recognize the temporal difference error $\delta_i = r_i - \Delta\phi_i^T \theta_{i-1}$ and a gain $K_i = \frac{P_{i-1}\Delta\phi_i}{P_{n_i} + \Delta\phi_i^T P_{i-1}\Delta\phi_i}$, to be linked again with the generic Widrow-Hoff update (44). 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. Remark that this interpretation of P_i as being a variance matrix can provide useful for handling the dilemma between exploration and exploitation, as noted by Engel (2005) (even if no specific scheme is proposed).

As all other residual methods, GPTD produces biased estimates of the value function when transitions are stochastic. To alleviate this problem, Engel et al. (2005) introduced a colored noise instead of the classical white noise assumption (a noise being white if $\forall i \neq j$, n_i and n_j are independent, and a colored noise is any non-white noise). This noise allows removing the bias, but it also induces a memory effect which prevents from learning in an off-policy manner, much like eligibility traces does (e.g., see Precup et al. (2000)). Note also that this leads to minimize another cost function, linking states' estimates to Monte Carlo samples of the discounted return (Engel et al., 2005). These developments are interesting, but not pursued here.

4.2.2 Kalman Temporal Differences

Geist et al. (2009) (see also Geist and Pietquin (2010b) for an extended version) introduced the so-called Kalman Temporal Differences (KTD) framework, which can be seen as a generalization of the GPTD framework. They start by noting that the (state-action) value

function to be estimated is generally not stationary. This is mainly due to two reasons. First, the system to be controlled can be non-stationary. More important, (state-action) value function estimation generally occurs as a part of a generalized policy iteration process. Each time the policy is modified, the associated (state-action) value function to be estimated changes too, hence non-stationarity. This has also been discussed by Phua and Fitch (2007) before. The idea behind KTD is to cast (state-action) value function approximation in the Kalman filtering paradigm. Parameters are modeled as random variables following a random walk (this allows handling non-stationarities). These hidden parameters have to be tracked from observed rewards and transitions, the link between them being a sampled Bellman equation. KTD algorithms are derived by finding the best linear estimator minimizing the expectation of parameters conditioned on past observed rewards. Moreover, they make use of an efficient derivative-free approximation scheme, the unscented transform (UT) of Julier and Uhlmann (1997). This allows considering both nonlinear parameterizations and the Bellman optimality operator. Compared to GPTD, KTD handles nonlinearities and non-stationarities, however it is a pure parametric approach (even if the online kernel-based linear parameterization construction of Engel et al. (2003) can be adapted to this framework). See Geist and Pietquin (2010b) for the original derivation of the KTD framework. Here it is derived using a statistically linearized recursive least-squares point of view. It is slightly restrictive, as the non-stationarities handling aspect is somehow lost (as introducing GPTD from a pure parametric least-squares perspective is also limiting). However, it allows linking more easily KTD to GPTD and to other residual approaches, and thus this provides a more unified view. Moreover, statistically linearized recursive least-squares and unscented Kalman filtering are strongly linked, the second being a generalization of the first. See Geist and Pietquin (2010e) for details.

As the GPTD framework, KTD also seeks at minimizing cost function (57), not necessarily considering a unitary noise variance too. Let write $\hat{T}Q(s_i,a_i) = r_i + \gamma \hat{P}Q(s_i,a_i)$ with:

$$\hat{P}Q(s_i,a_i) = \begin{cases} Q(s_{i+1},a_{i+1}) \text{ (if sampled Bellman evaluation operator)} \\ \max_{a\in A} Q(s_{i+1},a) \text{ (if sampled Bellman optimality operator)} \end{cases} \tag{74}$$

Estimated parameters should satisfy:

$$\theta_i = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}}\, \hat{J}_i(\theta) \text{ with } \hat{J}_i(\theta) = \sum_{j=1}^{i} \frac{1}{P_{n_j}} \left(r_j - \left(\hat{Q}_\theta(s_j,a_j) - \gamma \hat{P}\hat{Q}_\theta(s_j,a_j) \right) \right)^2 \tag{75}$$

Contrary to GPTD, KTD does not assume a linear parameterization nor the sampled evaluation operator. Instead, it makes use of a derivative-free approximation scheme (the derivative-free aspect allows considering the sampled optimality operator), the so-called statistical linearization (Anderson, 1984). The quantity to be linearized is the following observation model (to be considered as a function of θ, the state-action couple being fixed);

$$r_j = \hat{Q}_\theta(s_j,a_j) - \gamma \hat{P}\hat{Q}_\theta(s_j,a_j) + n_j \tag{76}$$

Assume that it is evaluated in n sampled parameter vectors $\theta^{(k)}$ of associated weights w_k (how to sample them practically and efficiently being addressed later):

$$\left(\theta^{(k)}, r_j^{(k)} = \hat{Q}_{\theta^{(k)}}(s_j,a_j) - \gamma \hat{P}\hat{Q}_{\theta^{(k)}}(s_j,a_j) \right)_{1\leq k\leq n} \tag{77}$$

The following statistics of interest are defined:

$$\bar{\theta} = \sum_{k=1}^{n} w_k \theta^{(k)}, \quad \bar{r}_j = \sum_{k=1}^{n} w_k r_j^{(k)} \tag{78}$$

$$P_\theta = \sum_{k=1}^{n} w_k \left(\theta^{(k)} - \bar{\theta}\right)\left(\theta^{(k)} - \bar{\theta}\right)^T \tag{79}$$

$$P_{\theta r_j} = \sum_{k=1}^{n} w_k \left(\theta^{(k)} - \bar{\theta}\right)\left(r_j^{(k)} - \bar{r}_j\right)^T = P_{r_j\theta}^T \tag{80}$$

$$P_{r_j} = \sum_{k=1}^{n} w_k \left(r_j^{(k)} - \bar{r}_j\right)^2 \tag{81}$$

Statistical linearization consists in linearizing the nonlinear observation model (76) around $\bar{\theta}$ (with P_θ being actually the spread of sampling) by adopting a statistical point of view. It finds a linear model $r_j = A_j\theta + b_j + u_j$, u_j being a noise, by minimizing the sum of squared errors between values of nonlinear and linearized functions in the regression points:

$$(A_j, b_j) = \underset{A,b}{\operatorname{argmin}} \sum_{k=1}^{n} \left(e_j^{(k)}\right)^2 \text{ with } e_j^{(k)} = r_j^{(k)} - \left(A\theta^{(k)} + b\right) \tag{82}$$

The solution of this optimization problem is given by (Anderson, 1984):

$$A_j = P_{r_j\theta} P_\theta^{-1} \text{ and } b_j = \bar{r}_j - A_j\bar{\theta} \tag{83}$$

Moreover, it is easy to check that the covariance matrix of the error is given by:

$$P_{e_j} = \sum_{k=1}^{n} \left(e_j^{(k)}\right)^2 = P_{r_j} - A_j P_\theta A_j^T \tag{84}$$

The nonlinear observation model (76) can thus be replaced by the following equivalent linear observation model:

$$r_j = A_j\theta + b_j + u_j \text{ with } u_j = e_j + n_j \tag{85}$$

Notice that the linearization error is taken into account through the noise e_j. Noises e_j and n_j being independent, the variance of P_{u_j} is given by $P_{u_j} = P_{e_j} + P_{n_j}$. Given this statistically linearized observation model, the least-squares problem can be rewritten in a linear form:

$$\theta_i = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}} \left(\sum_{j=1}^{i} \frac{1}{P_{u_j}} \left(r_j - A_j\theta - b_j\right)^2\right) \tag{86}$$

$$= \left(\sum_{j=1}^{i} \frac{1}{P_{u_j}} A_j^T A_j\right)^{-1} \sum_{j=1}^{i} A_j \left(r_j - b_j\right) \tag{87}$$

With this cost function, the higher is the statistical linearization error of a given transition (quantified by P_{u_j} through P_{e_j}), the less the corresponding square term contributes to the

cost. Using the Sherman-Morrison formula, a recursive formulation of this estimation can be obtained (expressing directly the gain K_i and assuming some priors θ_0 and P_0):

$$K_i = \frac{P_{i-1}A_i^T}{P_{u_i} + A_i P_{i-1} A_i^T} \tag{88}$$

$$\theta_i = \theta_{i-1} + K_i \left(r_i - b_i - A_i \theta_{i-1}\right) \tag{89}$$

$$P_i = P_{i-1} - K_i \left(P_{u_i} + A_i P_{i-1} A_i^T\right) K_i^T \tag{90}$$

Here again, the gain's magnitude is directly linked to the linearization error, and large errors will results in small updates.

How to actually sample parameter vectors in order to perform statistical linearization is addressed now. With this recursive estimation, θ_{i-1} (the previous estimate) and P_{i-1} (the associated uncertainty matrix as explained in Section 4.2.1) are known, and the issue is to compute A_i and b_i. A first thing is to choose around what point to linearize and with which magnitude. It is legitimate to sample around the previous estimate θ_{i-1} and with a spread related to the uncertainty of these estimates. Otherwise speaking, n parameter vectors are sampled such that $\bar{\theta}_i = \theta_{i-1}$ and $P_{\theta_i} = P_{i-1}$. Notice that $\bar{\theta}_i$, the point around what linearization is performed in order to update parameters, is different from θ_i, the updated parameter vector. There remains the choice of how parameter vectors are sampled. A natural idea would be to assume a Gaussian distribution of mean θ_{i-1} and variance P_{i-1} and to compute statistics of interest (78-81) using a Monte Carlo approach. However, this would be particularly inefficient. Actually, the problem of sampling these points can be stated as follows: how to sample a random variable (here the parameter vector) of known mean and variance (here θ_{i-1} and P_{i-1}) in order to compute accurate estimates of first and second order moments of a nonlinear mapping of this random variable (here $\hat{Q}_\theta(s_i, a_i) - \gamma \hat{P} \hat{Q}_\theta(s_i, a_i)$). The unscented transform of Julier and Uhlmann (1997) provides a solution to this problem. It consists in sampling deterministically a set of $n = 2p + 1$ so-called sigma-points as follows (p being the number of parameters):

$$\theta_i^{(k)} = \theta_{i-1} \qquad k = 0 \tag{91}$$

$$\theta_i^{(k)} = \theta_{i-1} + \left(\sqrt{(p+\kappa)P_{i-1}}\right)_k \qquad 1 \leq k \leq p \tag{92}$$

$$\theta_i^{(k)} = \theta_{i-1} - \left(\sqrt{(p+\kappa)P_{i-1}}\right)_{k-p} \qquad p+1 \leq k \leq 2p \tag{93}$$

as well as associated weights:

$$w_0 = \frac{\kappa}{p+\kappa} \text{ and } w_k = \frac{1}{2(p+\kappa)} \forall k > 0 \tag{94}$$

where κ is a scaling factor controlling the accuracy of the unscented transform (Julier and Uhlmann, 2004) and $(\sqrt{(p+\kappa)P_{i-1}})_k$ is the k^{th} column of the Cholesky decomposition of the matrix $(p+\kappa)P_{i-1}$. Image of each of these sigma-points is computed:

$$r_i^{(k)} = \hat{Q}_{\theta_{i-1}^{(k)}}(s_i, a_i) - \gamma \hat{P} \hat{Q}_{\theta_{i-1}^{(k)}}(s_i, a_i), \quad 0 \leq k \leq 2p \tag{95}$$

Sigma-points, their images and associated weights can then be used to compute statistics of interest (78-81), which are in turn used to compute linearization terms A_i and b_i (83) as

well as noise variance P_{e_i} (84). Notice that other approximation schemes can be considered instead of the UT, such as the scaled unscented transform (Julier, 2002), approximation schemes based on Sterling interpolation (Nørgård et al., 2000) or more generally sigma-point-based transforms (van der Merwe, 2004). The KTD algorithm can also be expressed directly as a function of the statistics of interest (which only need a few algebraic manipulation):

$$K_i = \frac{P_{\theta_i r_i}}{P_{u_i} + P_{r_i}} \tag{96}$$

$$\theta_i = \theta_{i-1} + K_i \left(r_i - \bar{r}_i\right) \tag{97}$$

$$P_i = P_{i-1} - K_i \left(P_{v_i} + P_{r_i}\right) K_i^T \tag{98}$$

Once again, K_i is a gain and $r_i - \bar{r}_i$ a temporal difference error, to be linked to the Widrow-Hoff update (44). KTD generalizes GPTD in the sens that it allows handling nonlinearities: nonlinear parameterization thanks to the linearization, and the (sampled) Bellman optimality operator thanks to the fact that this linearization scheme does not rely on a gradient computation. Notice that KTD reduces to GPTD if a linear parameterization as well as the sampled Bellman evaluation operator are considered (this is not difficult to check, the unscented transform being no longer an approximation in the case of a linear mapping).

As other residual approaches, KTD suffers from the bias problem when system transitions are stochastic. In order to handle this issue, Geist and Pietquin (2010a) also use a colored noise model (based on the idea of eligibility traces), which is actually a generalization of the noise proposed by Engel et al. (2005). However, as mentioned in Section 4.2.1, 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 different cost function. These developments are interesting, however as for GPTD they are not pursued here, see corresponding papers. Notice that the available uncertainty information (matrix P_i) can provide useful for the dilemma between exploration and exploitation (Geist and Pietquin, 2010c).

5. Projected Fixed-point Approaches

Projected fixed-point approaches seek at minimizing the distance between the estimated state-action value function and the projection (the projection being noted Π) of the image of this function under a Bellman operator onto the hypothesis space $\mathcal{H}$:

$$J(\theta) = \|\hat{Q}_\theta - \Pi T \hat{Q}_\theta\|^2 \text{ with } \Pi f = \underset{\hat{f} \in \mathcal{H}}{\operatorname{argmin}} \|f - \hat{f}\|^2 \tag{99}$$

This is illustrated in Figure 1. The state-action value function estimate $\hat{Q}_\theta$ 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 4 try to minimize the distance between these two functions, that is the dotted line 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 (the solid line in Figure 1). Projected fixed-point methods aim at minimizing the distance between this projection and $\hat{Q}_\theta$, represented by a dashed line in Figure 1.

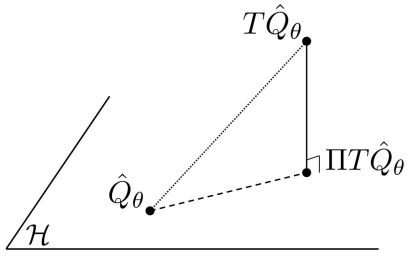

Figure 1: Projected fixed-point principle.

5.1 Least-Squares-based Approaches

In this section are reviewed approaches which use a least-squares approach to minimize the empirical cost linked to (99):

$$\theta_i = \underset{\theta\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\hat{Q}_\theta(s_j,a_j) - \hat{Q}_{\omega_\theta}(s_j,a_j)\right)^2 \tag{100}$$

$$\text{with } \omega_\theta = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\hat{Q}_\omega(s_j,a_j) - \hat{T}\hat{Q}_\theta(s_j,a_j)\right)^2 \tag{101}$$

Obviously, cost related to (100) is minimized for $\theta = \omega_\theta$ (admitting that this equation has a solution). Therefore, nested optimization problems (100) and (101) can be summarized as $\theta_i = \omega_{\theta_i}$:

$$\theta_i = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\hat{Q}_\omega(s_j,a_j) - \hat{T}\hat{Q}_{\theta_i}(s_j,a_j)\right)^2 \tag{102}$$

Notice that as θ_i appears in both sides of this equation, this is not a pure quadratic cost function. The least-squares temporal differences (LSTD) algorithm of Bradtke and Barto (1996) assumes a linear parameterization and the (sampled) Bellman evaluation operator in order to solve the above optimization problem. The statistically linearized LSTD (slLSTD) algorithm of Geist and Pietquin (2010d) generalizes it to nonlinear parameterization and to the (sampled) Bellman optimality operator thanks to a statistical linearization process (the generalization from LSTD to slLSTD being quite close to the generalization from GPTD to KTD).

5.1.1 Least-Squares Temporal Differences

The LSTD algorithm has been originally introduced by Bradtke and Barto (1996). The starting point of their derivation is the minimization of a residual cost function. Using a sampled Bellman operator (which leads to biased estimate, as explained in Section 4) 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 2). A classic method to cope with this problem are instrumental variables (Söderström and Stoica, 2002). Bradtke and Barto (1996) use instrumental variables to modify the least-squares problem, which leads to the LSTD algorithm. This point of view is historical. Later, it has been interpreted as a projected fixed-point minimization by Lagoudakis and Parr (2003), 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 (102) can be rewritten as:

$$\theta_i = \underset{\omega \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(r_j + \gamma \phi_{j+1}^T \theta_i - \phi_j^T \omega \right)^2 \tag{103}$$

Thanks to linearity in ω (linear parameterization assumption), this can be analytically solved:

$$\theta_i = \left(\sum_{j=1}^{i} \phi_j \phi_j^T \right)^{-1} \sum_{j=1}^{i} \phi_j \left(r_j + \gamma \phi_{j+1}^T \theta_i \right) \tag{104}$$

Thanks to linearity in θ_i (linear parameterization and evaluation operator assumptions), the parameter vector can be isolated:

$$\theta_i = \left(\sum_{j=1}^{i} \phi_j \phi_j^T \right)^{-1} \sum_{j=1}^{i} \phi_j \left(r_j + \gamma \phi_{j+1}^T \theta_i \right) \tag{105}$$

$$\Leftrightarrow \left(\sum_{j=1}^{i} \phi_j \phi_j^T \right) \theta_i = \sum_{j=1}^{i} \phi_j r_j + \gamma \left(\sum_{j=1}^{i} \phi_j \phi_{j+1}^T \right) \theta_i \tag{106}$$

$$\Leftrightarrow \theta_i = \left(\sum_{j=1}^{i} \phi_j \left(\phi_j - \gamma \phi_{j+1} \right)^T \right)^{-1} \sum_{j=1}^{i} \phi_j r_j \tag{107}$$

Equation (107) defines the (batch) LSTD estimate. Thanks to the Sherman-Morrison formula, a recursive form of this estimation process can be obtained (assuming that priors θ_0 and M_0 are defined beforehand):

$$K_i = \frac{M_{i-1} \phi_i}{1 + \left(\phi_i - \gamma \phi_{i+1} \right)^T M_{i-1} \phi_i} \tag{108}$$

$$\theta_i = \theta_{i-1} + K_i \left(r_i + \gamma \phi_{i+1}^T \theta_{i-1} - \phi_i^T \theta_{i-1} \right) \tag{109}$$

$$M_i = M_{i-1} - K_i \left(M_{i-1}^T \left(\phi_i - \gamma \phi_{i+1} \right) \right)^T \tag{110}$$

Once again, K_i is a gain and $r_i + \gamma \phi_{i+1}^T \theta_{i-1} - \phi_i^T \theta_{i-1}$ a temporal difference error, to be linked to the Widrow-Hoff update (44).

5.1.2 Statistically Linearized Least-Squares Temporal Differences

The slLSTD algorithm (Geist and Pietquin, 2010d) generalizes LSTD: it does not assume a linear parameterization nor the Bellman evaluation operator. The corresponding optimization problem is therefore:

$$\theta_i = \underset{\omega\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(r_j + \gamma \hat{P}\hat{Q}_{\theta_i}(s_j,a_j) - \hat{Q}_\omega(s_j,a_j)\right)^2 \tag{111}$$

How slLSTD generalizes LSTD is very close to how KTD generalizes GPTD: a statistical linearization is performed, which allows solving this optimization problem analytically. Equation (111) can be linked to the following observation model (n_j being here a unitary white and centered observation noise):

$$r_j + \gamma \hat{P}\hat{Q}_{\theta_i}(s_j,a_j) = \hat{Q}_\omega(s_j,a_j) + n_j \tag{112}$$

The noise is chosen unitary to strengthen parallel to LSTD, but extension to non-unitary noise is straightforward (it would lead to scale each square term of the cost function by the inverse of the associated variance P_{n_i}, equal to one here). As for KTD, a statistical linearization is performed. However, here two different quantities have to be linearized: $\hat{Q}_\omega(s_j,a_j)$ and $\hat{P}\hat{Q}_{\theta_i}(s_j,a_j)$.

Assume that n parameter vectors $\omega^{(k)}$ of associated weights w_k are sampled, and that their images are computed (how to sample them is addressed later, but note that the unscented transform will be used):

$$\left(\omega^{(k)}, q_j^{(k)} = \hat{Q}_{\omega^{(k)}}(s_j,a_j)\right)_{1\leq k\leq n} \tag{113}$$

Let define the following statistics:

$$\bar{\omega} = \sum_{k=1}^{n} w_k \omega^{(k)}, \quad \bar{q}_j = \sum_{k=1}^{n} w_k q_j^{(k)} \tag{114}$$

$$P_\omega = \sum_{k=1}^{n} w_k \left(\omega^{(k)} - \bar{\omega}\right)\left(\omega^{(k)} - \bar{\omega}\right)^T \tag{115}$$

$$P_{\omega q_j} = \sum_{k=1}^{n} w_k \left(\omega^{(k)} - \bar{\omega}\right)\left(q_j^{(k)} - \bar{q}_j\right)^T = P_{q_j\omega}^T \tag{116}$$

$$P_{q_j} = \sum_{k=1}^{n} w_k \left(q_j^{(k)} - \bar{q}_j\right)^2 \tag{117}$$

Using the statistical linearization process explained in Section 4.2.2, the following linear observation model is obtained:

$$\hat{Q}_\omega(s_j,a_j) = A_j\omega + b_j + e_j \tag{118}$$

$$\text{with } A_j = P_{q_j\omega}P_\omega^{-1},\ b_j = \bar{q}_j - A_j\bar{\omega} \text{ and } P_{e_j} = P_{q_j} - A_jP_\omega A_j^T \tag{119}$$

Recall that the noise e_j is centered and can be sampled as $e_j^{(k)} = q_j^{(k)} - (A_j\omega^{(k)} + b_j)$.

The term $\hat{P}\hat{Q}_{\theta_i}(s_j, a_j)$ also needs to be linearized. Assume that n parameter vectors $\theta_i^{(k)}$ of associated weights w_k are sampled, and that their images are computed (here again, how to sample them is addressed later).

$$\left(\theta_i^{(k)}, p_{q_j}^{(k)} = \hat{P}\hat{Q}_{\theta_i^{(k)}}(s_j, a_j)\right)_{1 \leq k \leq n} \tag{120}$$

Let define the following statistics:

$$\bar{\theta}_i = \sum_{k=1}^{n} w_k \theta_i^{(k)}, \quad \bar{p}_{q_j} = \sum_{k=1}^{n} w_k p_{q_j}^{(k)} \tag{121}$$

$$P_{\theta_i} = \sum_{k=1}^{n} w_k \left(\theta_i^{(k)} - \bar{\theta}_i\right)\left(\theta_i^{(k)} - \bar{\theta}_i\right)^T \tag{122}$$

$$P_{\theta_i p_{q_j}} = \sum_{k=1}^{n} w_k \left(\theta_i^{(k)} - \bar{\theta}_i\right)\left(p_{q_j}^{(k)} - \bar{p}_{q_j}\right)^T = P_{p_{q_j}\theta_i}^T \tag{123}$$

$$P_{p_{q_j}} = \sum_{k=1}^{n} w_k \left(p_{q_j}^{(k)} - \bar{p}_{q_j}\right)^2 \tag{124}$$

Notice that $\bar{\theta}_i$ is not equal to θ_i *a priori*. Using the statistical linearization process explained in Section 4.2.2, the following linear observation model is obtained:

$$\hat{P}\hat{Q}_{\theta_i}(s_j, a_j) = C_j \theta_i + d_j + \epsilon_j \tag{125}$$

$$\text{with } C_j = P_{p_{q_j}\theta_i} P_{\theta_i}^{-1},\ d_j = \bar{p}_{q_j} - C_j \bar{\theta}_i \text{ and } P_{\epsilon_j} = P_{p_{q_j}} - C_j P_{\theta_i} C_j^T \tag{126}$$

Recall that the noise ϵ_j is centered and can be sampled as $\epsilon_j^{(k)} = p_{q_j}^{(k)} - (C_j \theta_i^{(k)} + d_j)$.

Linearized models (118) and (125) can be injected into observation model (112):

$$r_j + \gamma\left(C_j\theta + d_j + \epsilon_j\right) = A_j\omega + b_j + e_j + n_j \tag{127}$$

$$\Leftrightarrow r_j + \gamma\left(C_j\theta_i + d_j\right) = A_j\omega + e_j - \gamma\epsilon_j + n_j \tag{128}$$

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

$$u_j = n_j + e_j - \gamma\epsilon_j \text{ and } P_{u_j} = E[u_j^2] \tag{129}$$

This equivalent observation model leads to the following optimization problem, which can be solved analytically:

$$\theta_i = \underset{\omega \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \frac{1}{P_{u_j}} \left(r_j + \gamma\left(C_j\theta_i + d_j\right) - \left(A_j\omega + b_j\right)\right)^2 \tag{130}$$

$$= \left(\sum_{j=1}^{i} \frac{1}{P_{u_j}} A_j^T A_j\right)^{-1} \sum_{j=1}^{i} \frac{1}{P_{u_j}} A_j^T \left(r_j + \gamma C_j\theta_i + \gamma d_j - b_j\right) \tag{131}$$

$$\Leftrightarrow \theta_i = \left(\sum_{j=1}^{i} \frac{1}{P_{u_j}} A_j^T \left(A_j - \gamma C_j\right)\right)^{-1} \sum_{j=1}^{i} \frac{1}{P_{u_j}} A_j \left(r_j + \gamma d_j - b_j\right) \tag{132}$$

Similarly to what happen with KTD, the statistical linearization error is taken into account through the noise variance P_{u_j}. The Sherman-Morrison formula allows again deriving a recursive estimation of θ_i. Assume that some priors θ_0 and M_0 are chosen, the slLSTD algorithm is defined as:

$$K_i = \frac{M_{i-1}A_i^T}{P_{u_i} + (A_i - \gamma C_i) M_{i-1} A_i^T} \tag{133}$$

$$\theta_i = \theta_{i-1} + K_i (r_i + \gamma d_i - b_i - (A_i - \gamma C_i)\theta_{i-1}) \tag{134}$$

$$M_i = M_{i-1} - K_i \left(M_{i-1}^T (A_i - \gamma C_i)^T \right)^T \tag{135}$$

Given this recursive formulation, there still remains to choose how to sample parameter vectors (related to ω and θ_i) in order to compute A_i, b_i, C_i, d_i and P_{u_i}.

The unscented transform, described in Section 4.2.2, is used to sample these parameter vectors. The parameter vector ω to be considered is the solution of Equation (130), that is the solution of the fixed-point problem $\theta_i = \omega_{\theta_i}$. In this recursive estimation context, 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}. Geist and Pietquin (2010d) use the same variance matrix as would have been provided by a statistically linearized recursive least-squares (Geist and Pietquin, 2010e) used to perform supervised learning of the approximate state-action value function given true observations of the Q-values. The fact that the unobserved state-action values are not used to update the variance matrix tends to legitimate this choice. The associated matrix update is:

$$P_i = P_{i-1} - \frac{P_{i-1}A_i^T A_i P_{i-1}}{1 + A_i P_{i-1} A_i^T} \tag{136}$$

These choices being made, A_i and b_i can be computed. A first step is to compute the set of sigma-points as well as associated weights w_k:

$$\left\{ \omega_i^{(k)}, 0 \leq k \leq 2p \right\} = \left[\theta_{i-1} \quad \theta_{i-1} \pm \left(\sqrt{(p+\kappa)P_{i-1}} \right)_j \right] \tag{137}$$

Images of these sigma-points are also computed:

$$\left\{ q_i^{(k)} = \hat{Q}_{\omega_i^{(k)}}(s_i, a_i), 0 \leq k \leq 2p \right\} \tag{138}$$

Statistics of interest are given by Equations (114-117), they are used to compute A_i and b_i (see Section 4.2.2):

$$A_i^T = P_{i-1}^{-1} P_{\omega q_i} \text{ and } b_i = \bar{q}_i - A_i \theta_{i-1} \tag{139}$$

The variance matrix update simplifies as:

$$P_i = P_{i-1} - P_{\omega q_i} (1 + P_{q_i})^{-1} P_{\omega q_i}^T \tag{140}$$

The inverse of the P_{i-1} matrix is necessary to compute A_i, it can be maintained recursively thanks to the Sherman-Morrison formula:

$$P_i^{-1} = P_{i-1}^{-1} + \frac{P_{i-1}^{-1} P_{\omega q_i} P_{\omega q_i}^T P_{i-1}^{-1}}{1 + P_{q_i} - P_{\omega q_i}^T P_{i-1}^{-1} P_{\omega q_i}} \tag{141}$$

The same approach is used to compute C_i and d_i, coming from the statistical linearization of $\hat{P}\hat{Q}_{\theta_i}(s_i, a_i)$. As before, the linearization is performed around the last estimate θ_{i-1} and considering the matrix variance Σ_{i-1} provided by a statistical linearized recursive least-squares that would perform a supervised regression of $\hat{P}\hat{Q}_{\theta_i}$:

$$\Sigma_i = \Sigma_{i-1} - \frac{\Sigma_{i-1} C_i^T C_i \Sigma_{i-1}}{1 + C_i \Sigma_{i-1} C_i^T} \tag{142}$$

A first step is to compute the set of sigma-points as well as associated weights w_k:

$$\left\{\theta_i^{(k)}, 0 \leq k \leq 2p\right\} = \left[\theta_{i-1} \quad \theta_{i-1} \pm \left(\sqrt{(p+\kappa)\Sigma_{i-1}}\right)_j\right] \tag{143}$$

Images of these sigma-points are also computed:

$$\left\{p_{q_i}^{(k)} = \hat{P}\hat{Q}_{\theta_i^{(k)}}(s_i, a_i), 0 \leq k \leq 2p\right\} \tag{144}$$

Statistics of interest are given by Equations (121-124), they are used to compute C_i and d_i (see Section 4.2.2 again):

$$C_i^T = \Sigma_{i-1}^{-1} \Sigma_{\theta_i p_{q_i}} \text{ and } d_i = \bar{p}_{q_i} - C_i \theta_{i-1} \tag{145}$$

The variance matrix update simplifies as:

$$\Sigma_i = \Sigma_{i-1} - P_{\theta_i p_{q_i}} \left(1 + P_{p_{q_i}}\right)^{-1} P_{\theta_i p_{q_i}}^T \tag{146}$$

The inverse of the Σ_{i-1} matrix is necessary to compute A_i, it can be maintained recursively thanks to the Sherman-Morrison formula:

$$\Sigma_i^{-1} = \Sigma_{i-1}^{-1} + \frac{\Sigma_{i-1}^{-1} P_{\theta_i p_{q_i}} P_{\theta_i p_{q_i}}^T \Sigma_{i-1}^{-1}}{1 + P_{p_{q_i}} - P_{\theta_i p_{q_i}}^T \Sigma_{i-1}^{-1} P_{\theta_i p_{q_i}}} \tag{147}$$

A last thing 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 of others, and the variance of $e_i - \gamma\epsilon_i$ can be computed using the UT:

$$P_{u_i} = E[(n_i + e_i - \gamma\epsilon_i)^2] \tag{148}$$

$$= 1 + \sum_{k=0}^{2p} w_k \left(e_i^{(k)} - \gamma\epsilon_i^{(k)}\right) \tag{149}$$

$$\text{with } e_i^{(k)} = q_i^{(k)} - A_i\omega_i^{(k)} - b_i = q_i^{(k)} - \bar{q}_i - A_i\left(\omega_i^{(k)} - \theta_{i-1}\right) \tag{150}$$

$$\text{and } \epsilon_j^{(k)} = p_{q_i}^{(k)} - C_i\theta_i^{(k)} - d_i = p_{q_i}^{(k)} - \bar{p}_{q_i} - C_i\left(\theta_i^{(k)} - \theta_{i-1}\right) \tag{151}$$

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 computed. 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. Parameters are then updated according to Equations (153-135), which simplifies as follows (given analytical expressions of b_i and d_i):

$$K_i = \frac{M_{i-1}A_i^T}{P_{u_i} + (A_i - \gamma C_i) M_{i-1} A_i^T} \tag{152}$$

$$\theta_i = \theta_{i-1} + K_i (r_i + \gamma \bar{p}_{q_i} - \bar{q}_i) \tag{153}$$

$$M_i = M_{i-1} - K_i \left(M_{i-1}^T (A_i - \gamma C_i)^T \right)^T \tag{154}$$

Therefore, it is not necessary to compute b_i and d_i. Notice that once again, this satisfies the Widrow-Hoff update, with a gain K_i and a temporal difference error $r_i + \gamma \bar{p}_{q_i} - \bar{q}_i$ (which depends on θ_{i-1}). Finally, matrices P_{i-1}, P_{i-1}^{-1}, Σ_{i-1} and Σ_{i-1}^{-1} are updated. Notice that it can be easily shown that with a linear parameterization and the (sampled) Bellman evaluation operator, slLSTD indeed reduces to LSTD (this relies on the fact that the unscented transform is no longer an approximation for a linear mapping).

5.2 Stochastic Gradient Descent-based Approaches

Algorithms presented in this section aim at minimizing the same cost function, that is :

$$J_i(\theta) = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\hat{Q}_\theta(s_j, a_j) - \hat{Q}_{\omega_\theta}(s_j, a_j) \right)^2 \text{ with } \hat{Q}_{\omega_\theta} = \Pi \hat{T} \hat{Q}_\theta \tag{155}$$

However, here a stochastic gradient descent approach is considered instead of the least-squares approach of the above section. Algorithms presented in Section 5.2.1, namely Gradient Temporal Difference 2 (GTD2) and Temporal Difference with Gradient Correction (TDC) of Sutton et al. (2009), assume a linear parameterization and the (sampled) Bellman evaluation operator. Algorithms presented in Section 5.2.2, namely nonlinear GTD2 (nlGTD2) and nonlinear TD (nlTDC) of Maei et al. (2009), extend them to the case of a nonlinear parameterization. The (linear) TDC algorithm has also been extended to eligibility traces (Maei and Sutton, 2010) and to the Bellman optimality operator (Maei et al., 2010), these extensions being briefly presented in Section 5.2.3.

5.2.1 Gradient Temporal Difference 2, Temporal Difference with Gradient Correction

GTD 2 and TDC algorithms of Sutton et al. (2009) aim at minimizing cost function (155) 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 reworked expression of the cost function. Let $\hat{\mathbf{Q}}_\theta$ and $\hat{\mathbf{Q}}_{\omega_\theta}$ be respectively:

$$\hat{\mathbf{Q}}_\theta = \begin{pmatrix} \hat{Q}_\theta(s_1, a_1) & \dots & \hat{Q}_\theta(s_i, a_i) \end{pmatrix}^T \tag{156}$$

$$\hat{\mathbf{Q}}_{\omega_\theta} = \begin{pmatrix} \hat{Q}_{\omega_\theta}(s_1, a_1) & \dots & \hat{Q}_{\omega_\theta}(s_i, a_i) \end{pmatrix}^T \tag{157}$$

Cost function (155) can be rewritten as:

$$J_i(\theta) = \left(\hat{\mathbf{Q}}_\theta - \hat{\mathbf{Q}}_{\omega_\theta} \right)^T \left(\hat{\mathbf{Q}}_\theta - \hat{\mathbf{Q}}_{\omega_\theta} \right) \tag{158}$$

Let also Φ_i (respectively Φ'_i) be the $p \times i$ matrix which columns are the features $\phi(s_j, a_j)$ (respectively $\phi(s_{j+1}, a_{j+1})$):

$$\Phi_i = \begin{bmatrix} \phi(s_1, a_1) & \dots & \phi(s_i, a_i) \end{bmatrix} \tag{159}$$

$$\Phi'_i = \begin{bmatrix} \phi(s_2, a_2) & \dots & \phi(s_{i+1}, a_{i+1}) \end{bmatrix} \tag{160}$$

Let R_i be the set of observed rewards:

$$R_i = \begin{pmatrix} r_1 & \dots & r_i \end{pmatrix}^T \tag{161}$$

As the parameterization is linear and as the Bellman evaluation is considered, the Q-values and their images through the sampled operator are given as:

$$\hat{\mathbf{Q}}_\theta = \Phi_i^T \theta \tag{162}$$

$$\hat{T}\hat{\mathbf{Q}}_\theta = R_i + \gamma \left(\Phi'_i\right)^T \theta \tag{163}$$

$\hat{\mathbf{Q}}_{\omega_\theta}$ is the projection of $\hat{T}\hat{\mathbf{Q}}_\theta$ onto the hypothesis space:

$$\omega_\theta = \underset{\omega \in \mathbb{R}^p}{\operatorname{argmin}} \left(\left(R_i + \gamma \left(\Phi'_i\right)^T \theta - \Phi_i^T \omega \right)^T \left(R_i + \gamma \left(\Phi'_i\right)^T \theta - \Phi_i^T \omega \right) \right) \tag{164}$$

$$= \left(\Phi_i \Phi_i^T\right)^{-1} \Phi_i \left(R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \tag{165}$$

Therefore, by writing $\Pi_i = \Phi_i^T \left(\Phi_i \Phi_i^T\right)^{-1} \Phi_i$ the projection operator, $\hat{\mathbf{Q}}_{\omega_\theta}$ satisfies:

$$\hat{\mathbf{Q}}_{\omega_\theta} = \Phi_i^T \omega_\theta = \Pi_i \hat{T} \hat{\mathbf{Q}}_\theta \tag{166}$$

Cost function (158) can thus be rewritten as:

$$J_i(\theta) = \left(\Phi_i^T \theta - \Pi_i \left(R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \right)^T \left(\Phi_i^T \theta - \Pi_i \left(R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \right) \tag{167}$$

Before developing this expression, two remarks of importance have to be made. First, $\Pi_i \Phi_i^T \theta = \Phi_i^T \theta$. As $\hat{\mathbf{Q}}_\theta$ belongs to the hypothesis space, it is invariant under the projection operator. This can also be easily checked algebraically in this case. Second, $\Pi_i \Pi_i^T = \Pi_i$. This is a basic property of a projection operator, which can also be easily checked algebraically here. Using these relationships, the cost can rewritten as:

$$J_i(\theta) = \left(\Pi_i \Phi_i^T \theta - \Pi_i \left(R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \right)^T \left(\Pi_i \Phi_i^T \theta - \Pi_i \left(R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \right) \tag{168}$$

$$= \left(\Phi_i^T \theta - R_i - \gamma \left(\Phi'_i\right)^T \theta \right)^T \Pi_i \Pi_i^T \left(\Phi_i^T \theta - R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \tag{169}$$

$$= \left(\Phi_i^T \theta - R_i - \gamma \left(\Phi'_i\right)^T \theta \right)^T \Pi_i \left(\Phi_i^T \theta - R_i + \gamma \left(\Phi'_i\right)^T \theta \right) \tag{170}$$

$$= \left(\Phi_i \left(\Phi_i^T \theta - R_i - \gamma \left(\Phi'_i\right)^T \theta \right) \right)^T \left(\Phi_i \Phi_i^T\right)^{-1} \left(\Phi_i \left(\Phi_i^T \theta - R_i - \gamma \left(\Phi'_i\right)^T \theta \right) \right) \tag{171}$$

Let $\delta_j(\theta) = r_j + \gamma \phi_{j+1}^T \theta - \phi_j^T \theta$ be the temporal difference error, $J_i(\theta)$ is finally given as:

$$J_i(\theta) = \left(\sum_{j=1}^{i} \phi_j \delta_j(\theta) \right)^T \left(\sum_{j=1}^{i} \phi_j \phi_j^T \right)^{-1} \left(\sum_{j=1}^{i} \phi_j \delta_j(\theta) \right) \tag{172}$$

Notice that a Gradient Temporal Difference (GTD) algorithm has been introduced by Sutton et al. (2008) by considering a slightly different cost function:

$$J'_i(\theta) = \left(\sum_{j=1}^{i} \phi_j \delta_j(\theta)\right)^T \left(\sum_{j=1}^{i} \phi_j \delta_j(\theta)\right) \tag{173}$$

This explains why the algorithm of Sutton et al. (2009) is called GTD2, and it is not further developed here.

The negative gradient of cost function (172) is given by:

$$-\frac{1}{2}\nabla_\theta J_i(\theta) = \left(\sum_{j=1}^{i} (\phi_j - \gamma\phi_{j+1})\,\phi_j^T\right) \left(\sum_{j=1}^{i} \phi_j \phi_j^T\right)^{-1} \left(\sum_{j=1}^{i} \delta_j(\theta)\phi_j\right) \tag{174}$$

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 $(\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1}(\sum_{j=1}^{i} \delta_j(\theta)\phi_j)$, this being called the weight-doubling trick. Parameter vector θ is updated according to a stochastic gradient descent:

$$\theta_i = \theta_{i-1} + \alpha_i\,(\phi_i - \gamma\phi_{i+1})\,\phi_i^T \omega_{i-1} \tag{175}$$

Their remains to find an update rule for ω_i. In order to obtain a $O(p)$ algorithm, Sutton et al. (2009) estimate it using a stochastic gradient descent too. One can remark that ω_i is actually the solution of a linear least-squares optimization problem:

$$\omega_i = \left(\sum_{j=1}^{i} \phi_j \phi_j^T\right)^{-1} \sum_{j=1}^{i} \delta_j(\theta)\phi_j \tag{176}$$

$$= \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} \left(\phi_j^T \omega - \delta_j(\theta)\right)^2 \tag{177}$$

This suggests the following update rule for ω_i (minimization of Equation (177) using a stochastic gradient descent):

$$\omega_i = \omega_{i-1} + \beta_i \phi_i \left(\delta_i(\theta_{i-1}) - \phi_i^T \omega_{i-1}\right) \tag{178}$$

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 given by:

$$\theta_i = \theta_{i-1} + \alpha_i\,(\phi_i - \gamma\phi_{i+1})\,\phi_i^T \omega_{i-1} \tag{179}$$

$$\omega_i = \omega_{i-1} + \beta_i \phi_i \left(\delta_i(\theta_{i-1}) - \phi_i^T \omega_{i-1}\right) \tag{180}$$

$$\text{with } \delta_i(\theta) = r_i + \gamma\phi_{i+1}^T\theta - \phi_i^T\theta \tag{181}$$

Under some assumptions, this algorithm can be shown to be convergent, see Sutton et al. (2009).

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. Starts from (174):

$$-\frac{1}{2}\nabla_\theta J_i(\theta) = \left(\sum_{j=1}^{i}\left(\phi_j - \gamma\phi_{j+1}\right)\phi_j^T\right)\left(\sum_{j=1}^{i}\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^{i}\delta_j(\theta)\phi_j\right) \tag{182}$$

$$= \left(\sum_{j=1}^{i}\phi_j\phi_j^T - \gamma\sum_{j=1}^{i}\phi_{j+1}\phi_j^T\right)\left(\sum_{j=1}^{i}\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^{i}\delta_j(\theta)\phi_j\right) \tag{183}$$

$$= \left(\sum_{j=1}^{i}\delta_j(\theta)\phi_j\right) - \gamma\left(\sum_{j=1}^{i}\phi_{j+1}\phi_j^T\right)\left(\sum_{j=1}^{i}\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^{i}\delta_j(\theta)\phi_j\right) \tag{184}$$

This gives rise to the following update for θ, ω being updated as before:

$$\theta_i = \theta_{i-1} + \alpha_i\phi_i\delta_i(\theta_{i-1}) - \alpha_i\gamma\phi_{i+1}\phi_i^T\omega_{i-1} \tag{185}$$

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 3.1), and the second term, $-\alpha_i\gamma\phi_{i+1}\phi_i^T\omega_{i-1}$, 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 $\lim_{i\to\infty}\frac{\alpha_i}{\beta_i} = 0$. This means that θ_i is updated on a slower time-scale. The idea behind this is that ω_i should look stationary from the θ_i point of view. The TDC algorithm can be summarized as follows:

$$\theta_i = \theta_{i-1} + \alpha_i\phi_i\delta_i(\theta_{i-1}) - \alpha_i\gamma\phi_{i+1}\phi_i^T\omega_{i-1} \tag{186}$$

$$\omega_i = \omega_{i-1} + \beta_i\phi_i\left(\delta_i(\theta_{i-1}) - \phi_i^T\omega_{i-1}\right) \tag{187}$$

$$\text{with } \delta_i(\theta) = r_i + \gamma\phi_{i+1}^T\theta - \phi_i^T\theta \tag{188}$$

This algorithm can also be shown to be convergent under some assumptions, see Sutton et al. (2009) again.

5.2.2 Nonlinear Gradient Temporal Difference 2, Nonlinear Temporal Difference with Gradient Correction

Maei et al. (2009) extend GTD2 and TDC algorithms to the case of a general nonlinear parameterization $\hat{Q}_\theta$, as long as it is differentiable respectively to θ. The corresponding hypothesis space $\mathcal{H} = \{\hat{Q}_\theta|\theta\in\mathbb{R}^p\}$ is a differentiable submanifold onto which projecting is not computationally feasible. They assume that the parameter vector θ is slightly updated in one step (given that learning rate are usually small), which causes the surface of the submanifold to be close to linear. Therefore, projection is done onto the tangent plane defined as $\mathcal{TH} = \{(s,a)\in S\times A\to\omega^T\nabla_\theta\hat{Q}(s,a)|\omega\in\mathbb{R}^p\}$. The corresponding projection operator Π_i^θ can be obtained as in Section 5.2.1, the tangent space being an hyperplane:

$$\Pi_i^\theta = \left(\Phi_i^\theta\right)^T\left(\Phi_i^\theta\left(\Phi_i^\theta\right)^T\right)^{-1}\Phi_i^\theta \tag{189}$$

$$\text{with } \Phi_i^\theta = \begin{bmatrix}\nabla_\theta\hat{Q}_\theta(s_1,a_1) & \dots & \nabla_\theta\hat{Q}_\theta(s_i,a_i)\end{bmatrix} \tag{190}$$

The corresponding cost function can therefore be derived as in Section 5.2.1, with basically feature vectors $\phi(s,a)$ being replaced by linearized features $\phi^\theta(s,a) = \nabla_\theta \hat{Q}_\theta(s,a)$:

$$J_i(\theta) = \left(\sum_{j=1}^{i} \phi_j^\theta \delta_j(\theta)\right)^T \left(\sum_{j=1}^{i} \phi_j^\theta \left(\phi_j^\theta\right)^T\right)^{-1} \left(\sum_{j=1}^{i} \phi_j^\theta \delta_j(\theta)\right) \tag{191}$$

$$\text{with } \phi_j^\theta = \nabla_\theta \hat{Q}_\theta(s_j, a_j) \tag{192}$$

$$\text{and } \delta_j(\theta) = r_j + \gamma \hat{Q}_\theta(s_{j+1}, a_{j+1}) - \hat{Q}_\theta(s_j, a_j) \tag{193}$$

Maei et al. (2009) show (see the paper for details) that the gradient of this cost function is given as:

$$-\frac{1}{2}\nabla_\theta J_i(\theta) = \left(\sum_{j=1}^{i} \left(\phi_j^\theta - \gamma \phi_{j+1}^\theta\right)\left(\phi_j^\theta\right)^T\right)\omega_i + h(\theta, \omega_i) \tag{194}$$

$$= \left(\sum_{j=1}^{i} \delta_j(\theta)\phi_j^\theta\right) - \gamma \left(\sum_{j=1}^{i} \phi_{j+1}^\theta \left(\phi_j^\theta\right)^T\right)\omega_i + h(\theta, \omega_i) \tag{195}$$

$$\text{with } \omega_i = \left(\sum_{j=1}^{i} \phi_j^\theta \left(\phi_j^\theta\right)^T\right)^{-1} \left(\sum_{j=1}^{i} \delta_j(\theta)\phi_j^\theta\right) \tag{196}$$

$$\text{and } h(\theta, \omega) = -\sum_{j=1}^{i} \left(\delta_j(\theta) - \left(\phi_j^\theta\right)^T \omega\right)\left(\nabla^2 \hat{Q}_\theta(s_j, a_j)\right)\omega \tag{197}$$

GTD2 and TDC are generalized to nlGTD2 and nlTDC using a stochastic gradient descent on the above cost function. Parameter vector ω_i is updated as in Section 5.2.1:

$$\omega_i = \omega_{i-1} + \beta_i \phi_i^{\theta_{i-1}} \left(\delta_i(\theta_{i-1}) - \left(\phi_i^{\theta_{i-1}}\right)^T \omega_{i-1}\right) \tag{198}$$

The nonlinear GTD2 algorithm performs a stochastic gradient descent according to (194):

$$\theta_i = \theta_{i-1} + \alpha_i \left(\left(\phi_i^{\theta_{i-1}} - \gamma \phi_{i+1}^{\theta_{i-1}}\right)\left(\phi_i^{\theta_{i-1}}\right)^T \omega_{i-1} - h_i\right) \tag{199}$$

$$\text{with } h_i = \left(\delta_i(\theta_{i-1}) - \left(\phi_i^{\theta_{i-1}}\right)^T \omega_{i-1}\right)\left(\nabla^2_{\theta_{i-1}} \hat{Q}_\theta(s_i, a_i)\right)\omega_{i-1} \tag{200}$$

Learning rates are chosen as for the NTD algorithm, that is satisfying the classic stochastic approximation criterion and such that $\lim_{i\to\infty} \frac{\alpha_i}{\beta_i} = 0$, which means that θ is updated on a slower timescale than ω. The nonlinear TDC algorithm performs a stochastic gradient descent according to (195):

$$\theta_i = \theta_{i-1} + \alpha_i \left(\phi_i^{\theta_{i-1}} \delta_i(\theta_{i-1}) - \gamma \phi_{i+1}^{\theta_{i-1}} \left(\phi_i^{\theta_{i-1}}\right)^T \omega_{i-1} - h_i\right) \tag{201}$$

$$\text{with } h_i = \left(\delta_i(\theta_{i-1}) - \left(\phi_i^{\theta_{i-1}}\right)^T \omega_{i-1}\right)\left(\nabla^2_{\theta_{i-1}} \hat{Q}_\theta(s_i, a_i)\right)\omega_{i-1} \tag{202}$$

Learning rate are chosen as above. Both nlGTD2 and nlTDC can be shown to be convergent under some assumptions, see Maei et al. (2009) for details.

5.2.3 Extension of TDC

The TDC algorithm (see Section 5.2.1) has been extended to eligibility traces by Maei and Sutton (2010). Moreover, this algorithm, called GQ(λ), allows off-policy learning, that is learning the value of one target policy while following another behaviorial policy. This new algorithm (for which some convergence guarantees can be provided) still minimizes the empirical cost function linked to (99). 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 Maei and Sutton (2010) for more details. Using eligibility traces induce a memory effect which prevents from learning in an off-policy manner without caution, see Precup et al. (2000) for example. To cope with this problem, Maei and Sutton (2010) use ideas from the importance sampling[2] field (as Precup et al. (2000) 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 by Maei et al. (2010) (this new algorithm being called Greedy-GQ). To do so, they consider 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). Therefore, the considered 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 Maei et al. (2010) relies on the Fréchet sub-gradient (roughly speaking a generalization of the gradient for non-differentiable functions) to provide their algorithm. 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. Maei et al. (2010) provide a convergence analysis for these algorithms.

5.3 Iterated solving-based approaches

Methods presented so far in Section 5 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:

$$J(\theta) = \|\hat{Q}_\theta - \Pi T \hat{Q}_\theta\|^2 \tag{203}$$

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:

$$\hat{Q}_{\theta_i} = \Pi T \hat{Q}_{\theta_{i-1}} \tag{204}$$

Methods presented in this section adopt this point of view to provide algorithms.

2. Generally speaking, importance sampling refers to technics allowing to compute some statistics linked to a random variable using samples drawn from another random variable. This is exactly the problem caused by learning in an off-policy manner while using eligibility traces.

5.3.1 Fitted-Q

Fitted-Q is a batch algorithm. It assumes that a set of N transitions is available beforehand:

$$\{(s_j, a_j, r_j, s_{j+1}, a_{j+1})\}_{1 \leq j \leq N} \tag{205}$$

A initial Q-function $\hat{Q}_{\theta_0}$ is (more or less arbitrary) initialized, and estimates are refined by iterating the ΠT operator:

$$\hat{Q}_{\theta_i} = \Pi \hat{T} \hat{Q}_{\theta_{i-1}}, \quad \forall i > 0 \tag{206}$$

Some remarks of importance have to be made here. 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 (which can be more or less directly seen as a projection). 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).

A practical example is given now. Assume that at iteration i the estimate $\hat{Q}_{\theta_{i-1}}$ is available. A first thing is to compute the image of this estimate trough the sampled Bellman (here optimality) operator. This consists in computing the following training base, composed of state-action couples and estimated associated optimal Q-values:

$$\left\{(s_j, a_j, r_j + \gamma \max_{a \in A} \hat{Q}_{\theta_{i-1}}(s_{j+1}, a))\right\}_{1 \leq j \leq N} \tag{207}$$

A supervised learning algorithm is then used on this set, associating inputs (s_j, a_j) to estimated outputs $r_j + \gamma \max_{a \in A} \hat{Q}_{\theta_{i-1}}(s_{j+1}, a)$. For example, if the parameterization is linear and if the supervised learning algorithm is the classic least-squares regression, the new parameter vector is given by:

$$\theta_i = \left(\sum_{j=1}^{N} \phi_j \phi_j^T\right)^{-1} \sum_{j=1}^{N} \phi_j \left(r_j + \gamma \max_{a \in A} \left(\phi(s_{j+1}, a)^T \theta_{i-1}\right)\right) \tag{208}$$

This iteration is repeated until a stopping criterion is met (*e.g.*, a maximum number of steps or little changes in the representation).

The fitted-Q idea probably dates back to Samuel (1959). Its convergence properties have been analyzed by Gordon (1995), who reasons on the contraction property of the ΠT operator. Munos (2007) analyses its performance bounds in L_p norm. 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. For example, see Ormoneit and Sen (2002) for fitted-Q with a kernel-based regression, Riedmiller (2005) for fitted-Q with neural networks trained by retro-propagation or Ernst et al. (2005) for fitted-Q with tree-based methods.

5.3.2 Least-Squares Policy Evaluation

The least-squares policy evaluation (LSPE) has been proposed[3] by Nedić and Bertsekas (2003). They introduce it directly using the concept of eligibility traces, but this aspect is left apart in this article. The LSPE algorithm 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 recursively $\hat{Q}_{\theta_i} = \Pi_i \hat{T} \hat{Q}_{\theta_{i-1}}$, the Π_i projection operator being defined in Section 5.2.1. Given linearity, this can be rewritten as:

$$\theta_i = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\phi_j^T \theta - r_j - \gamma \phi_{j+1}^T \theta_{i-1}\right)^2 \tag{209}$$

$$= \left(\sum_{j=1}^{i} \phi_j \phi_j^T\right)^{-1} \sum_{j=1}^{i} \phi_j \left(r_j + \gamma \phi_{j+1}^T \theta_{i-1}\right) \tag{210}$$

$$= \theta_{i-1} + \left(\sum_{j=1}^{i} \phi_j \phi_j^T\right)^{-1} \sum_{j=1}^{i} \phi_j \left(r_j - \left(\phi_j - \gamma \phi_{j+1}\right)^T \theta_{i-1}\right) \tag{211}$$

All terms involved can be computed recursively and efficiently, using the Sherman-Morrison formula for the inverse matrix:

$$B_i^{-1} = \left(\sum_{j=1}^{i} \phi_j \phi_j^T\right)^{-1} = B_{i-1}^{-1} - \frac{B_{i-1}^{-1} \phi_i \phi_i^T B_{i-1}^{-1}}{1 + \phi_i^T B_{i-1}^{-1} \phi_i} \tag{212}$$

$$A_i = \sum_{j=1}^{i} \phi_j \left(\phi_j - \gamma \phi_{j+1}\right)^T = A_{i-1} + \phi_i \left(\phi_i - \gamma \phi_{i+1}\right)^T \tag{213}$$

$$b_i = \sum_{j=1}^{i} \phi_j r_j = b_{i-1} + \phi_i r_i \tag{214}$$

The LSPE update can therefore be expressed in a form close to a Widrow-Hoff update:

$$\theta_i = \theta_{i-1} + B_i^{-1} \left(b_i - A_i \theta_{i-1}\right) \tag{215}$$

Actually, the way LSPE is presented here differs from Nedić and Bertsekas (2003) 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. This algorithm (with a constant learning rate equal to one) is shown to be convergent by Bertsekas et al. (2004). Notice that ideas behind LSPE have other applications (Bertsekas and Yu, 2007) and can also be linked to variational inequalities (Bertsekas, 2009).

3. Actually, if the name LSPE has been introduced by Nedić and Bertsekas (2003) (in a multistep value-iteration context), the related algorithm has first been introduced by Bertsekas and Ioffe (1996), where it is built upon λ-policy-iteration.

5.3.3 Q-learning for optimal stopping problems

The Q-learning for optimal stopping problems (Q-OSP) proposed by Yu and Bertsekas (2007) extends LSPE to the (sampled) Bellman optimality operator, the parameterization being still linear. They present this algorithm in the case of optimal stopping problems, which are a restrictive class of Markovian decision processes. However, it is presented here in the general case.

The derivation of this algorithm is the same as for the LSPE one, by considering the optimality operator instead of the evaluation one:

$$\theta_i = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{j=1}^{i} \left(\phi_j^T \theta - r_j - \gamma \max_{a \in A} \left(\phi(s_{j+1}, a)^T \theta_{i-1} \right) \right) \tag{216}$$

$$= \left(\sum_{j=1}^{i} \phi_j \phi_j^T \right)^{-1} \sum_{j=1}^{i} \left(r_j + \gamma \max_{a \in A} \left(\phi(s_{j+1}, a)^T \theta_{i-1} \right) \right) \tag{217}$$

The matrix $(\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1}$ can still be computed recursively and efficiently, however this is not the case for the term $\sum_{j=1}^{i}(r_j + \gamma \max_{a \in A}(\phi(s_{j+1}, a)^T \theta_{i-1}))$ which requires remembering the whole trajectory and needs to be computed at each iteration. Yu and Bertsekas (2007) show this algorithm to be convergent for optimal stopping problems and under some assumptions, and they also propose some more computationally efficient variations for the same restrictive class of MDP.

6. 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 1. A link between Widrow-Hoff update and most of 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 (which is moreover linear in the length of the trajectory). 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

	bootstrapping	residual	projected fixed-point direct / iterated	
stochastic gradient descent	TD-VFA SARSA-VFA QL-VFA	R-SGD	(nl)GTD2 (nl)TDC GQ(λ) Greedy-GQ	
(recursive) least-squares	FPKF	GPTD KTD	LSTD slLSTD	LSPE Q-OSP
other				fitted-Q

Table 1: Summary.

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, see Sutton and Barto (1998). Such an approach as a clear advantage for methods based on stochastic gradient descent, as it speed up learning. However, for least-squares-based approaches, this advantage is less clear, notably because these methods are generally much more sample efficient, see Boyan (1999) for example. 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, see Precup et al. (2000) for example. By the way, using eligibility traces can be expressed as using a modified T^λ Bellman operator (see Maei and Sutton (2010) 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 (Xu et al., 2007) as well as LSPE (Jung and Polani, 2007). A variation of LSTD has been proposed by Geramifard et al. (2006) 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 (Farahmand et al., 2008) as well as to L_1 regularization (Kolter and Ng, 2009). These are some examples among other.

References

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

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*, 71(1):89–129, 2008.

L. C. Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. In *Proceedings of the International Conference on Machine Learning (ICML 95)*, pages 30–37, 1995.

D. P. Bertsekas. Projected Equations, Variational Inequalities, and Temporal Difference Methods. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2009.

D. P. Bertsekas, V. Borkar, and A. Nedic. *Learning and Approximate Dynamic Programming*, chapter Improved Temporal Difference Methods with Linear Function Approximation, pages 231–235. IEEE Press, 2004.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3).* Athena Scientific, 1996.

D. P. Bertsekas and H. Yu. Projected Equation Methods for Approximate Solution of Large Linear Systems. *Journal of Computational and Applied Mathematics*, 227:27–50, 2007.

Dimitri P. Bertsekas and Sergey Ioffe. Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming. Technical Report LIDS-P-2349, Labs for Information and Decision Systems, MIT, 1996.

J. A. Boyan. Technical Update: Least-Squares Temporal Difference Learning. *Machine Learning*, 49(2-3):233–246, 1999.

S. J. Bradtke and A. G. Barto. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

Z. Chen. Bayesian Filtering : From Kalman Filters to Particle Filters, and Beyond. Technical report, Adaptive Systems Lab, McMaster University, 2003.

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

Y. Engel. *Algorithms and Representations for Reinforcement Learning.* PhD thesis, Hebrew University, April 2005.

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)*, pages 154–161, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement Learning with Gaussian Processes. In *Proceedings of the International Conference on Machine Learning (ICML 05)*, 2005.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6:503–556, 2005. ISSN 1533-7928.

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.

M. Geist and O. Pietquin. Eligibility Traces through Colored Noises. In *Proceedings of the IEEE International Conference on Ultra Modern Control Systems (ICUMT 2010)*, Moscow (Russia), October 2010a. IEEE.

M. Geist and O. Pietquin. Kalman Temporal Differences. *Journal of Artificial Intelligence Research (JAIR)*, 2010b.

M. Geist and O. Pietquin. Managing Uncertainty within Value Function Approximation in Reinforcement Learning. In *Active Learning and Experimental Design Workshop (collocated with AISTATS 2010)*, Journal of Machine Learning Research - Workshop and Conference Proceedings, Sardinia, Italy, 2010c.

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), October 2010d. IEEE.

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), 2010e. IEEE.

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.

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)*, pages 356–361, 2006.

G. Gordon. Stable Function Approximation in Dynamic Programming. In *Proceedings of the International Conference on Machine Learning (IMCL 95)*, 1995.

S. J. Julier. The scaled unscented transformation. In *American Control Conference*, volume 6, pages 4555–4559, 2002.

S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls 3*, 1997.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

T. Jung and D. Polani. Kernelizing LSPE(λ). In *IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 338–345, 2007.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

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.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003. ISSN 1533-7928.

H. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1204–1212, 2009.

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.

H. Reza 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.

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*, pages 664–671, 2009.

R. Munos. Performance Bounds in Lp norm for Approximate Value Iteration. *SIAM Journal on Control and Optimization*, 2007.

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

P. M. Nørgård, N.K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

D. Ormoneit and S. Sen. Kernel-Based Reinforcement Learning. *Machine Learning*, 49: 161–178, 2002.

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.

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)*, pages 759–766, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

M. Riedmiller. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method . In *Europeac Conference on Machine Learning (ECML)*, 2005.

A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, pages 210–229, 1959.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

O. Sigaud and O. Buffet, editors. *Markov Decision Processes and Artificial Intelligence*. Wiley - ISTE, 2010.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 3rd edition, March 1998.

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*, pages 993–1000, New York, NY, USA, 2009. ACM.

R. S. Sutton, C. Szepesvari, and H. R. Maei. A Convergent O(n) Algorithm for Off-policy Temporal-difference Learning with Linear Function Approximation. In *Advances in Neural Information Processing Systems 21*, Vancouver, BC, 2008.

T. Söderström and P. Stoica. Instrumental variable methods for system identification. *Circuits, Systems, and Signal Processing*, 21:1–9, 2002.

G. Tesauro. Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3), March 1995.

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42, 1997.

R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, Portland, OR, USA, April 2004.

X. Xu, D. Hu, and X. Lu. Kernel-Based Least Squares Policy Iteration for Reinforcement Learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, July 2007.

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.