Integrating Partial Model Knowledge in Model Free RL Algorithms

Aviv Tamar AVIVT@TX.TECHNION.AC.IL
Dotan Di Castro DOT@TX.TECHNION.AC.IL
Ron Meir RMEIR@EE.TECHNION.AC.IL
Department of Electrical Engineering, The Technion - Israel Institute of Technology, Haifa, Israel 32000

Abstract

In reinforcement learning an agent uses online feedback from the environment and prior knowledge in order to adaptively select an effective policy. Model free approaches address this task by directly mapping external and internal states to actions, while model based methods attempt to construct a model of the environment, followed by a selection of optimal actions based on that model. Given the complementary advantages of both approaches, we suggest a novel algorithm which combines them into a single algorithm, which switches between a model based and a model free mode, depending on the current environmental state and on the status of the agent's knowledge. We prove that such an approach leads to improved performance whenever environmental knowledge is available, without compromising performance when such knowledge is absent. Numerical simulations demonstrate the effectiveness of the approach and suggest its efficacy in boosting policy gradient learning.

1. Introduction

In Reinforcement Learning (RL) an agent attempts to improve its performance over time at a given task, based on continual interaction with the (usually unknown) environment, (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). This improvement takes place by modifying the action selection policy, based on feedback from the environment and prior knowledge available to the agent. Formally, RL is often formulated as the problem of finding a mapping, the so called *policy*, from the environment's states to the agent's actions that maximizes a given functional of a reward function.

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

Most RL algorithms can be classified into either *model based* or *model free* approaches (Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996). In the former setting, taking its inspiration from the field of Adaptive Control (Kumar, 1985), the agent builds a model of the environment, typically in the form of a *Markov Decision Process* (MDP), while interacting with it. Based on this model, a *planning* problem is solved where techniques from *Dynamic Programming* (Bertsekas and Tsitsiklis, 1996) are applied in order to find the optimal policy function. On the other hand, within the model free setting, the agent does not try to build a model of the MDP, but rather attempts to find the optimal policy by directly mapping environmental states to actions. In this sense, no model of the environment is required in order to act optimally. While both approaches possess significant merits, the view taken in this paper is that their advantages are in fact complementary, with each approach possessing distinct advantages in particular situations. Since the nature of the problem is often not known in advance, it would seem advantageous to combine both perspectives allowing the agent to benefit from both approaches. In this paper we pursue such a hybrid approach applicable to the case where partial model information is available. As a concrete example, consider a scenario where the environment consists of a known stationary component and an unknown time varying element. In this case the agent should continuously adapt to the environment, but may take advantage of knowledge of the known part to improve performance. We provide a method for integrating partial model information into an existing model free algorithm, prove through theoretical analysis that our method improves performance, and demonstrate its effectiveness via computer simulation.

A high level sketch of the method The model free algorithms we are concerned with are of the Stochastic Approximation (SA) type (Kushner and Yin, 2003). These online algorithms attempt to optimize some parameter of the system, using "noise corrupted" system measurements as a data stream for an iterative optimization process. These algorithms deal with noise by

making only small changes to the parameters at each step, so that over many iterations the noise averages out, and the parameters asymptotically follow a *mean trajectory.* Intuitively, any prior knowledge about the system should reduce our uncertainty about its behavior and thus enable some noise reduction. In this paper we propose a method that *reduces the noise at each step.* We do this by observing that the update at each step can be viewed as a simple estimate of the mean update. Prior knowledge enables us to propose an estimator which has smaller estimation error, thereby reducing the noise variance. A key property of our estimator is that it is unbiased, thus it *preserves the algorithm's mean trajectory.* This assures us that the overall *function* of the algorithm will remain intact, while the reduction in noise variance gives reason to expect an increase in *performance.*

2. Estimation of a Random Variable Mean with Partial Knowledge

Our method of using partial knowledge in an SA algorithm is based on constructing a better estimator for the mean update at each step. In this section we describe our estimator in the context of estimating the mean of a random variable. This allows us to derive all its important properties without the notational burden of the SA setting. The results we derive will then easily transfer to the more complex SA setting.

Let X be a random variable over a discrete set Ω and let $P(\omega) \triangleq \Pr(X=\omega)$ denote the probability distribution of X. Assume that, prior to the estimation process, we are given the probability distribution values of X over a subset of Ω. Denote by K this set for which P values are known, $K \triangleq \{\omega : \omega \in \Omega \text{ s.t. } P(\omega) \text{ is known}\}$, which we will refer to as the *partial knowledge* set. Suppose we are given a sample of X, denoted by x, and we wish to estimate (without bias) the expectation $\mu = \mathrm{E}[X] \triangleq \sum_{\omega\in\Omega} \omega P(\omega)$. Denote by $\bar{K}$ the complement of the set K , and by $\mathbf{1}_x^K$ the indicator function that equals 1 if $x \in K$ and 0 otherwise. Consider the estimator

$$\hat{\mu}_K(x) = \mathbf{1}_x^K \cdot \frac{\mathrm{E}[X \cdot \mathbf{1}_X^K]}{\mathrm{E}[\mathbf{1}_X^K]} + \mathbf{1}_x^{\bar{K}} \cdot x. \tag{1}$$

The intuition behind (1) is simple. If the sample is part of the known set, then it is of no use to us, and our estimate is a normalized mean over the known part of Ω. Otherwise, our estimate is the sample itself, maintaining the estimator unbiased. It is easy to see that $\hat{\mu}_K$ is unbiased, as expressed in Lemma 2.1.

Lemma 2.1. *The estimator* $\hat{\mu}_K$ *satisfies* $\mathrm{E}[\hat{\mu}_K] = \mu$.

In the next Lemma the Mean Squared Error (MSE) of $\hat{\mu}_K$ is computed. Let $P(K) = \sum_{\omega\in K} P(\omega)$, and let $P_K(\omega)$ denote the probability measure over the known set K, namely $P_K(\omega) = \mathbf{1}_\omega^K P(\omega)/P(K)$. Denote by $\mathrm{E}_K[\cdot]$ and $\mathrm{Var}_K[\cdot]$ the expectation and variance under the probability measure P_K.

Lemma 2.2. *The MSE of* $\hat{\mu}_K$ *is* $\mathrm{E}[\hat{\mu}_K - \mu]^2 = \mathrm{Var}[X] - P(K)\cdot\mathrm{Var}_K[X]$.

Proof. First, observe that for any function $f(\cdot)$

$$\begin{aligned}\mathrm{E}_K[f(X)-\mu]^2 &= \mathrm{E}_K[f(X)-\mathrm{E}_K f(X)+\mathrm{E}_K f(X)-\mu]^2 \\ &= \mathrm{Var}_K f(X) + (\mathrm{E}_K[f(X)]-\mu)^2,\end{aligned}$$

where the cross terms in the second equality vanish.

Since $\mathrm{Var}_K[\hat{\mu}_K(X)] = 0$, and $\mathrm{E}_K[\hat{\mu}_K(X)] = \mathrm{E}_K[X]$ we have

$$\begin{aligned}\mathrm{E}[\hat{\mu}_K(X)-\mu]^2 &= \mathrm{E}\left[\left(\mathbf{1}_X^K + \mathbf{1}_X^{\bar{K}}\right)(\hat{\mu}_K(X)-\mu)^2\right] \\ &= P(K)\,\mathrm{E}_K\left[(\hat{\mu}_K(X)-\mu)^2\right] + \mathrm{E}\left[\mathbf{1}_X^{\bar{K}}(X-\mu)^2\right] \\ &= P(K)(\mathrm{E}_K[X]-\mu)^2 + \mathrm{E}\left[\mathbf{1}_X^{\bar{K}}(X-\mu)^2\right] \\ &= E[X-\mu]^2 - P(K)\cdot\mathrm{Var}_K[X].\end{aligned}$$

□

One could disregard the partial knowledge altogether, and choose to use the sample x itself as an unbiased estimate for μ. Denote this estimator by

$$\hat{\mu}(x) = x. \tag{2}$$

It is easy to see that the MSE of $\hat{\mu}$ is $\mathrm{Var}[X]$. From Lemma 2.2 we deduce that when the cardinality of the known set satisfies $|K| > 1$, and $P(K) > 0$, the MSE of $\hat{\mu}_K$ is smaller than that of $\hat{\mu}$.

An appropriate question is whether a better estimator than $\hat{\mu}_K(x)$ exists. We refer the interested reader to the supplementary material[1], where we show that under certain assumptions $\hat{\mu}_K(x)$ is admissible, but note that the results of Lemmas 2.1 and 2.2 suffice for the following discussion.

3. A Stochastic Approximation Algorithm with Partial Model Knowledge

In this section we describe our method of endowing a model free RL algorithm with partial model knowledge, based on the estimator developed in the previous section.

[1] http://webee.technion.ac.il/~rmeir/tamar2011supp.zip

3.1. Preliminaries

Notation Throughout the rest of the paper the following notation is used. All vectors are column vectors, and $(\cdot)^T$ denotes the transpose operator. The product $A \circ B$ denotes the element-wise product (Hadamard product) of A and B. $\mathrm{Tr}[\cdot]$ is the trace of a matrix. The cardinality of a set K is denoted by $|K|$, and itsby $\bar{K}$. Unless noted otherwise, a subscript of a value denotes time. An individual element i of a vector A is denoted by $[A]_i$, and similarly, for a matrix B we denote the element (i,j) by $[B]_{ij}$.

RL Environment We consider an agent interacting with an unknown environment, modeled by a MDP in discrete time with a finite state set $\mathcal{X}$ and action set $\mathcal{U}$. Each selected action $u \in \mathcal{U}$ of the agent determines a stochastic transition matrix $P_u \triangleq [P_u(y|x)]_{x,y\in\mathcal{X}}$, where y is the state following the state x.

For each state $x \in \mathcal{X}$ the agent receives a corresponding deterministic reward $r(x)$, which is bounded, and depends only on the current state[2]. The agent maintains a *policy function,* $\mu_\theta(u|x)$*,* parametrized by a vector $\theta \in \mathbb{R}^L$, mapping an observation $x \in \mathcal{X}$ into a probability distribution over the controls $\mathcal{U}$. Under policy μ_θ, the environment and the agent induce a Markovian transition matrix, denoted by P_{μ_θ}, which we assume to be ergodic[3]. This Markovian transition matrix has a stationary distribution over the state space $\mathcal{X}$, denoted by π_{μ_θ}, and we construct a diagonal matrix $\Pi_{\mu_\theta} \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|} = \mathrm{diag}(\pi_{\mu_\theta})$. Our goal is to optimize θ with respect to some performance criteria. The tuning of θ is performed online in the following fashion. At time n, the current parameter value equals θ_n and the agent is in state x_n. It then chooses a control u_n according to $\mu_{\theta_n}(u|x_n)$, observes x_{n+1}, and updates θ_{n+1} according to some protocol.

Stochastic Approximation Stochastic approximation methods (Kushner and Yin, 2003) are a class of iterative stochastic algorithms, to which many model free RL algorithms belong (Bertsekas and Tsitsiklis, 1996). Analysis of SA methods has received considerable attention over the past decade, and many analysis techniques are available. In particular, the ODE approach is a widely used method for investigating the asymptotic behavior of SA iterates. The algorithms with which we deal in this paper will all be cast in the following SA form,

$$\theta_{n+1} = \theta_n + \epsilon_n F(\theta_n, x_n, u_n, x_{n+1}), \qquad (3)$$

where $\{\epsilon_n\}$ are positive step sizes. The key idea of the technique is the following. Suppose that iterate (3) can be decomposed into a mean function of the current state, action and parameter, denoted by $g(\theta_n, x_n, u_n) = \mathrm{E}[F(\theta, x_n, u_n, x_{n+1})|\,\theta_n, x_n, u_n]$, and a martingale difference noise term denoted by δM_n, which is a result of the stochastic transition to the next state. Formally,

$$\theta_{n+1} = \theta_n + \epsilon_n (g(\theta_n, x_n, u_n) + \delta M_n). \qquad (4)$$

Suppose that the effect of the martingale difference noise weakens due to repeated averaging, and further assume that there exists a continuous function $\bar{g}(\theta) = \mathrm{E}[g(\theta, x, u)|\,\theta]$ where the expectation is over the states and actions[4]. Consider the following ordinary differential equation (ODE)

$$d\theta/dt = \bar{g}(\theta). \qquad (5)$$

Then, a typical result of the ODE method in the SA setup suggests that the asymptotic limits of (3) and (5) are identical. Another aspect of SA relates to the rate of convergence of such iterates (Kushner and Yin, 2003), an issue we will return to.

A note on types of convergence The type of convergence to the asymptotic limit depends primarily on the step size used. Let θ^* denote an asymptotically stable point of (5). Then, for a suitably decreasing step size, convergence w.p. 1 of θ_n to θ^* can be established. For a constant step size, θ_n can be shown to converge weakly to a random variable centered on θ^*. In the following we use the term convergence ambiguously, and the precise definition should be inferred from the context. For a detailed and rigorous discussion of the types of convergence in SA the reader is referred to Kushner and Yin (2003).

3.2. Partial Model Based Algorithm

A key observation obtained from examining equations (3-4), is that $F(\theta_n, x_n, u_n, x_{n+1})$ in the SA algorithm is just the sample estimator (2) of $g(\theta_n, x_n, u_n)$, where the estimation variance stems from the stochastic transitions in the MDP. In the following we assume that we have, prior to running the algorithm, some information about these transitions. Similarly to section 2, define the known set for state x and action u as

$$K_{x,u} \triangleq \{y : y \in \mathcal{X} \text{ s.t. } P_u(y|x) \text{ is known}\}.$$

Denote by $\mathbf{1}^K_{n+1}$ an indicator function that equals 1 if $\{x_{n+1}\}$ belongs to K_{x_n,u_n} and 0 otherwise. Based on the estimator introduced in section 2, we propose

[2] Generalizing the results presented here to state-action rewards is straightforward. Generalization to stochastic rewards is also possible by considering mean rewards.

[3] i.e. every state is visited infinitely often.

[4] Explicitly $\bar{g}(\theta) = \sum_{x\in\mathcal{X}} \pi_{\mu_\theta}(x) \sum_{u\in\mathcal{U}} \mu_\theta(u|x) g(\theta, x, u)$

the following update rule for the tunable parameter, denoted by θ^K, which we refer to as the *Integrated Partial Model* (IPM) iteration

$$\theta_{n+1}^K = \theta_n^K + \epsilon_n \left(\mathbf{1}_{n+1}^K F_n^K + \mathbf{1}_{n+1}^{\bar{K}} F\left(\theta_n^K, x_n, u_n, x_{n+1}\right) \right), \tag{6}$$

where, abusing notation, $F_n^K = F_n^K\left(\theta_n^K, x_n, u_n\right)$, and

$$F_n^K \triangleq \frac{\sum\limits_{y \in K_{x_n,u_n}} P_{u_n}(y|x_n) F\left(\theta_n^K, x_n, u_n, y\right)}{\sum\limits_{y \in K_{x_n,u_n}} P_{u_n}(y|x_n)}. \tag{7}$$

Similarly to (4), iterate (6) can also be decomposed into a mean function $g^K\left(\theta_n^K, x_n, u_n\right)$ and a martingale difference noise δM_n^K

$$\theta_{n+1}^K = \theta_n^K + \epsilon_n \left(g^K\left(\theta_n^K, x_n, u_n\right) + \delta M_n^K \right),$$

and by Lemma 2.1 we have $g^K(\theta, x, u) = g(\theta, x, u)$. Similarly, defining $\bar{g}^K(\theta) = \mathrm{E}\left[g^K(\theta, x, u) | \theta \right]$ we get that $\bar{g}^K(\theta) = \bar{g}(\theta)$, therefore (6) converges to the same asymptotic trajectory as (3). Furthermore, Lemma 2.2 shows that if our partial knowledge set is not null, then on each iteration the variance of the noise term is reduced. This gives us reason to believe that an improvement in the overall performance of the algorithm can be expected.

3.3. Step size considerations

As it turns out, the improvement in performance attained by the IPM iteration is heavily influenced by the step size used. This can be intuitively explained using the following example. Let $\{z_i\}$ be a sequence of i.i.d. bounded random variables, with mean μ_z and variance σ_z^2. Consider the following SA iteration

$$\theta_{n+1} = \theta_n + \epsilon_n \left(z_{n+1} - \theta_n \right).$$

For a decreasing step size of the form $\epsilon_n = 1/(n+1)$, the value of θ_n is simply the empirical average, which converges w.p. 1 to μ_z. As a performance measure, consider the MSE defined by $\mathrm{E}\left\|\theta_n - \mu_z\right\|^2$, which equals σ_z^2/n. Integration of partial knowledge based on (6) in this case is equivalent to averaging variables with the same mean but with a reduced variance, and the MSE still approaches zero at a rate $O(1/n)$. On the other hand, when the step size is constant, θ_n converges in mean to μ_z, but the MSE converges to a non-zero value which, intuitively, is proportional[5] to the variance σ_z^2. Any variance reduction in this case would thus prove valuable.

[5] A precise value is given in the next section.

The use of a constant step size, though clearly undesirable in the preceding example, is quite common in RL applications, as it allows the iterates to quickly reach a neighborhood of the desired solution, and can cope with time varying environments. In the following discussion, we shall thus focus our analysis on algorithms with a constant step size.

4. TD(0) with Partial Knowledge

In this section we apply our method to the well known model free algorithm Temporal Difference Prediction (TD(0)) (Sutton and Barto, 1998). The simplicity of TD(0) allows us to mathematically characterize its performance in terms of *convergence rate*, and to quantify the impact of partial knowledge integration on it. The mathematical results we derive specifically for TD(0) are also characteristic of more complex algorithms, as will be shown in simulations.

A similar analysis of TD(0) without the partial knowledge was given in Dayan and Sejnowski (1994), though for a decreasing step size, and without explicit convergence rate results. Singh and Dayan (1998) provided update equations for the MSE of TD(0), which we also use as a measure of convergence rate, though their equations were only solvable by simulation.

Throughout this section, we assume that the agent's policy μ is deterministic and constant, mapping a specific action to each state, denoted by $\mu(x)$. Letting $0 < \gamma < 1$ denote a discount factor, define the value function for state x under policy μ as the expected discounted return when starting from state x and executing policy μ

$$V(x) \triangleq \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t) \middle| x_0 = x \right].$$

Here, for simplicity of notation, we have omitted μ in $V^\mu(x)$. The value function is a vector of size $|\mathcal{X}|$. When the state space is large, Function Approximation (FA) is often used to find an approximation to the value function in a subspace of size $L < |\mathcal{X}|$. Linear FA is implemented as follows. Given a set of $|\mathcal{X}|$ linearly independent basis vectors $\phi(x) \in \mathbb{R}^L$, our goal is to find an approximation to $V(x)$, denoted by $\hat{V}(x, \theta)$ and defined as $\hat{V}(x, \theta) = \phi(x)^T \theta$, i.e. our tunable parameter θ in this case is a vector of L linear weights. In vector form we write $\hat{V}(\theta) = \Phi\theta$, where $\Phi \in \mathbb{R}^{|\mathcal{X}| \times L}$ is a matrix with the basis vectors in its rows.

The fixed step TD(0) algorithm updates θ online in the following manner (Sutton and Barto, 1998)

$$\begin{aligned} \theta_{n+1} &= \theta_n + \epsilon d_n \phi\left(x_n\right), \\ d_n &\triangleq r\left(x_n\right) + \gamma \phi(x_{n+1})^T \theta_n - \phi(x_n)^T \theta_n, \end{aligned} \tag{8}$$

where ϵ is a small and constant step size. Using (6-7) we define IPM-TD(0)

$$\theta_{n+1}^K = \theta_n^K + \epsilon d_n^K \phi(x_n), \quad (9)$$
$$d_n^K \triangleq r(x_n) + \gamma\left(\mathbf{1}_{n+1}^K F_n^K + \mathbf{1}_{n+1}^{\bar{K}} \phi(x_{n+1})^T \theta_n^K\right) - \phi(x_n)^T \theta_n^K,$$
$$F_n^K \triangleq \frac{\sum_{y \in K_{x_n,u_n}} P_{u_n}(y|x_n)\,\phi(y)^T \theta_n^K}{\sum_{y \in K_{x_n,u_n}} P_{u_n}(y|x_n)}.$$

The mean function $\bar{g}(\theta) \triangleq \mathrm{E}[d_n \phi(x_n) | \theta_n = \theta]$ is calculated to be (Bertsekas and Tsitsiklis, 1996; Lemma 6.5)

$$\bar{g}(\theta) = \Phi^T \Pi_\mu r + \Phi^T \Pi_\mu (\gamma P_\mu - I)\Phi\theta.$$

From Lemma 2.1 we conclude that $\bar{g}^K(\theta) \triangleq \mathrm{E}[d_n^K \phi(x_n) | \theta_n^K = \theta] = \bar{g}(\theta)$. Denote by θ^* the asymptotic limit of the iterate mean, $\mathrm{E}[\theta_n]$, which is the unique fixed point of the ODE $d\theta/dt = \bar{g}(\theta)$[6]. Since both iterates (8) and (9) have the same asymptotic mean, the MSE, $\mathrm{E}\|\theta_n - \theta^*\|^2$, can be defined as a *measure of performance* of the iterates. The linearity of $\bar{g}(\theta)$ allows us to analytically quantify this measure.

The remainder of this section is concerned with proving that the performance, in MSE terms, of iterate (9) is better than that of iterate (8). In order to accomplish this we first present a general expression for the asymptotic MSE of both iterates. Then, we solve it for the simplified case of table based TD(0). The solution for linear FA can also be obtained, but it is technically tedious, and is deferred to the full paper. Nevertheless, simulations with linear FA are presented in section 5, and display similar behavior to the tabular case.

In their treatment of rate of convergence, Kushner and Yin (2003; p. 315) discuss the properties of the process $\rho_n \triangleq (\theta_n - \theta^*)/\sqrt{\epsilon}$. Their Theorem 10.1.3 states that, given a lengthy set of assumptions[7], ρ_n converges in distribution (as $\epsilon \to 0$ and $n \to \infty$ such that $n = \omega(1/\epsilon)$[8] to a normally distributed random variable, which is the stationary distribution of the stochastic differential equation

$$dU = AUdt + dW. \quad (10)$$

A is the Jacobian of $\bar{g}$, namely $A = \Phi^T \Pi_\mu (\gamma P_\mu - I)\Phi$, and W is a Wiener process with covariance matrix $\Sigma = \Sigma_0 + \Sigma_1 + \Sigma_1^T$ where

$$\Sigma_0 = \lim_{n\to\infty} \mathrm{E}\left[(d_n \phi(x_n))(d_n \phi(x_n))^T \middle| \theta_n = \theta^*\right], \quad (11)$$

$$\Sigma_1 = \sum_{j=1}^{\infty} \lim_{n\to\infty} \mathrm{E}\left[(d_n \phi(x_n))(d_{n+j} \phi(x_{n+j}))^T \middle| \theta_n, \theta_{n+j} = \theta^*\right].$$

For iteration (9) we have Σ_0^K, Σ_1^K where d_n^K replaces d_n in (11). The stationary solution to (10) is normally distributed with zero mean and covariance R, which can be easily computed (Papoulis and Pillai, 2002) by observing that (10) describes white noise filtered through a linear system, leading to

$$R = \lim_{t\to\infty} e^{At} \left\{ \int_0^t e^{-As} \Sigma \left(e^{-As}\right)^T ds \right\} \left(e^{At}\right)^T. \quad (12)$$

Let $\{\lambda\}_{i=1}^L$ denote the eigenvalues of A, which all have a negative real part (Bertsekas and Tsitsiklis, 1996, Lemma 6.6b), and let Γ be its diagonalizing matrix, i.e., $A = \Gamma\Lambda\Gamma^{-1}$ where Λ is diagonal. Also, define a matrix $\chi \in \mathbb{R}^{N\times N}$ such that $[\chi]_{ij} = -1/(\lambda_i + \lambda_j)$. Then the limit in (12) can be written as

$$R = \lim_{t\to\infty} \Gamma \left\{ \int_0^t e^{\Lambda(t-s)} \Gamma^{-1} \Sigma \left(\Gamma^{-1}\right)^T e^{\Lambda(t-s)} ds \right\} \Gamma^T$$
$$= \Gamma\left(\chi \circ \left(\Gamma^{-1} \Sigma \left(\Gamma^{-1}\right)^T\right)\right)\Gamma^T,$$

and the limit of the MSE is therefore

$$\mathrm{E}\|\theta_n - \theta^*\|^2 \to \epsilon \mathrm{Tr}\left[\Gamma\left(\chi \circ \left(\Gamma^{-1}\Sigma\left(\Gamma^{-1}\right)^T\right)\right)\Gamma^T\right]. \quad (13)$$

The difference in MSE between iterates (8) and (9) lies in the difference between Σ_0, Σ_0^K and Σ_1, Σ_1^K. In order to simplify calculations, in the remainder of the analysis we deal with a table based algorithm where $\Phi = I$. The results can easily be generalized to linear FA, though we defer the detailed analysis to the full paper. In the tabular case, $\Sigma_1 = \Sigma_1^K = 0$. This is the case since when we are in state x only the x'th value of θ is updated. The next update of that value will take place when we are again in state x. The state sequence is governed by an ergodic Markov chain, and from the Regenerative Cycle Theorem (see Brémaud, 1999, pp. 87) we have that trajectories between visits to x are independent. The updates, which are a function of these trajectories, are therefore also independent, and each element in the sum in (11) is zero. The calculation of Σ_0 is straightforward. It is a diagonal matrix, and its elements are

$$[\Sigma_0]_{xx} = [\pi_\mu]_x \sum_y P_\mu(y|x)\left(r(x) + \gamma[\theta^*]_y - [\theta^*]_x\right)^2.$$

[6] Convergence in mean of TD(0) was shown by Sutton (1988). It is also a consequence of weak convergence, which, due to lack of space we do not prove.

[7] Due to lack of space, we do not prove here that algorithms (8) and (9) fulfill all the required assumptions. Satisfaction of these assumptions relies on the following properties. The iterates are continuous in θ, and the value function is bounded. The stationary Markov chain controlling state transitions is ergodic, $\bar{g}(\theta)$ is linear, and the matrix $\Phi^T \Pi_\mu (\gamma P_\mu - I)\Phi$ is Hurwitz.

[8] We retain this assumption on ϵ and n in the sequel.

Let Δ_Σ denote the diagonal matrix defined by $\Delta_\Sigma \triangleq \Sigma_0 - \Sigma_0^K$. Using Lemma 2.2, Δ_Σ is a diagonal matrix with elements

$$[\Delta_\Sigma]_{xx} = [\pi_\mu]_x P\big(K_{x,\mu(x)}\big) \mathrm{Var}_{K_{x,\mu(x)}}\Big[r(x)+\gamma [\theta^*]_y - [\theta^*]_x \Big| x\Big]$$
$$= \gamma^2 [\pi_\mu]_x P\left(K_{x,\mu(x)}\right) \mathrm{Var}_{K_{x,\mu(x)}}\left[[\theta^*]_y \Big| x\right].$$

Note that Δ_Σ has no negative elements. We are interested in the difference in the asymptotic MSE

$$\begin{aligned} \delta_{MSE} &= \mathrm{E}\left\|\theta_n - \theta^*\right\|^2 - \mathrm{E}\left\|\theta_n^K - \theta^*\right\|^2 \qquad (14) \\ &\to \epsilon \cdot \mathrm{Tr}\left[\Gamma\left(\chi \circ \left(\Gamma^{-1}\Delta_\Sigma\left(\Gamma^{-1}\right)^T\right)\right)\Gamma^T\right]. \end{aligned}$$

If the known set is not null, then δ_{MSE} is positive (it can be seen as the asymptotic MSE of an iterate with the same matrix A, but with Δ_Σ instead of Σ_0, which by definition is positive), and thus the algorithm's performance improves. We summarize this result in the following theorem.

Theorem 4.1. *Consider the table based online TD(0) iterate for θ_n described by (8) with $\Phi = I$, and the* IPM − TD(0) *iterate for θ_n^K described by (9) with the same requirement on Φ. Then both iterations converge in mean to the same value θ^*. Furthermore, assuming that there is at least one state $x \in \mathcal{X}$ such that $P\left(K_{x,\mu(x)}\right) \cdot \mathrm{Var}_{K_{x,\mu(x)}}\left[[\theta^*]_y \Big| x\right] > 0$, then the asymptotic MSE of the iterates satisfy $\lim_{n\to\infty} E\|\theta_n^K - \theta^*\|^2 = \lim_{n\to\infty} E\|\theta_n - \theta^*\|^2 - \delta_{MSE}$, where δ_{MSE} is given in (14), and $\delta_{MSE} > 0$.*

Theorem 4.1 therefore assures us that the reduction in noise variance at each step, guaranteed by Lemma 2.2, translates into a reduction in the overall error of the algorithm.

Note that the simple dependence of the MSE on ϵ allows for a different interpretation of the performance in terms of *convergence rate* - for some desired MSE, the partial knowledge allows us to use a larger step size ϵ, and thus *converge faster*. This issue will also be demonstrated in simulation.

In the case of function approximation the results will be similar, but one has to also take into account the correlation between different time steps - the Σ_1 and Σ_1^K matrices.

We comment on a decreasing step size. For a step size of the form $\epsilon_n = 1/n^\alpha$, $0.5 < \alpha \leq 1$, a similar analysis can be performed with ρ_n defined as $\rho_n = n^{\alpha/2}(\theta_n - \theta^*)$. In this case, θ_n converges to θ^* w.p. 1, and the MSE decreases to zero at a rate $O\left(n^{-\alpha/2}\right)$. Integrating partial knowledge in this case will only reduce fluctuations in the converging path of the system. The performance gain of incorporating partial knowledge is therefore much stronger when the step size is constant.

5. Simulation results

In this section we demonstrate the performance of our method in simulation[9]. We first describe results on the TD(0) algorithm, and compare with the theory established in the previous section. Next, we demonstrate a policy gradient type algorithm which also changes the policy online, and show how partial knowledge allows for better performance.

General Setup Our simulations are on a set of abstract randomly constructed MDP's termed Generalized Average Reward Non-stationary Environment Test-bench or in short GARNET (Bhatnagar et al., 2007). GARNET MDP's comprise a class of randomly constructed finite MDP's serving as a test-bench for RL algorithms. A GARNET MDP is characterized in our case by four parameters and is denoted by GARNET$(|\mathcal{X}|, |\mathcal{U}|, B, \sigma)$. The parameter $|\mathcal{X}|$ is the number of states in the MDP, $|\mathcal{U}|$ is the number of actions, B is the branching factor of the MDP, i.e., the number of uniformly distributed non-zero entries in each line of the MDP's transition matrices, and the reward in each state is normally distributed with variance σ. For each GARNET MDP we also construct a 'Known' MDP characterized by a parameter p_K, $0 \leq p_K \leq 1$ such that each transition in the original MDP is known w.p. p_K. The value of p_K therefore indicates our level of knowledge about the MDP, ranging from no knowledge at all ($p_K = 0$) up to knowing the complete MDP ($p_K = 1$).

A linear FA is used as follows. For approximating a function of the states, we use a linear approximation $f(x,\theta) = \phi(x)'\theta$, where $\phi(x) \in \{0,1\}^L$, and define l to be the number nonzero values in $\phi(x)$. The nonzero values are chosen uniformly at random, where any two states have different feature vectors. When approximating a function of states and actions the approximation is performed only over the states, and we define feature vectors $\xi(x,u) \in \{0,1\}^{L\cdot|\mathcal{U}|}$ which are constructed as $\xi(x,u) \triangleq (0,...(L \times (u-1) \text{ zeros}), \phi(x), 0, ...(L \times (|\mathcal{U}| - u) \text{ zeros}))^T$.

5.1. Policy Evaluation with TD(0)

For a GARNET$(10,5,10,1)$ MDP, a random deterministic policy was chosen and its value function was evaluated using algorithm (9). The error $\|\theta_n - \theta^*\|^2$, aver-

[9] The code for generating the results presented here can be found in the supplementary material

aged over 500 different runs with the same initial conditions, is plotted in figure 1 (left) for different values of p_K. The asymptotic MSE was calculated using (13) and is shown for comparison. In figure 1 (middle), the step size for an iteration with partial knowledge was set such that the asymptotic MSE would match that of the iteration without partial knowledge. As can be seen, this caused the IPM iteration to converge *faster*.

Figure 1 (right) shows results on a GARNET$(30, 5, 10, 1)$ MDP, where we used linear FA as described above with $L = 10$ and $l = 2$. As can be seen, the behavior observed in the tabular case is characteristic of the FA case as well.

5.2. A Policy Gradient Algorithm

In this simulation the agent maintains a stochastic policy function parametrized by $\theta \in \mathbb{R}^{L\cdot|\mathcal{U}|}$, given by $\mu_\theta(u|x) = e^{\theta^T \xi(x,u)} / \sum_{u'} e^{\theta^T \xi(x,u')}$. The agent's goal is to find the parameter θ which maximizes the *average reward* $\eta = \mathrm{E}[r(x)]$. Policy Gradient algorithms estimate the gradient w.r.t. θ of the average reward, $\nabla_\theta \eta$, and perform a stochastic gradient ascent on the parameters to reach a local maximum. One such algorithm was proposed by Marbach and Tsitsiklis (1998). At time n we update the parameter vector θ and a scalar λ which is an estimate of η,

$$\theta_{n+1} = \theta_n + \epsilon \left(r\left(x_n\right) - \lambda_n\right) z_n, \tag{15}$$

$$\lambda_{n+1} = \lambda_n + \epsilon' \left(r\left(x_n\right) - \lambda_n\right), \tag{16}$$

where ϵ and ϵ' are step sizes, and $\epsilon' < \epsilon$. We then simulate a transition to the next state, and update the vector z by $z_{n+1} = z_n + L_{x_n,u_n}(\theta_n)$, where $L_{x_n,u_n}(\theta_n)$ is the likelihood ratio $L_{x,u}(\theta) = \nabla_\theta \log \mu_\theta(u|x)$. Every time a predefined recurrent state of the MDP is visited, z_{n+1} is reset to zero. Denote by $\mathbf{1}_n^K$ an indicator function that equals 1 if x_n belongs to $K_{x_{n-1},u_{n-1}}$ and 0 otherwise. Incorporating partial knowledge into the algorithm using (6) simply amounts to replacing $r(x_n)$ in (15-16) with

$$\mathbf{1}_n^K \cdot \frac{\sum_{y \in K_{x_{n-1},u_{n-1}}} P_{u_{n-1}}(y|x_{n-1}) r(y)}{\sum_{y \in K_{x_{n-1},u_{n-1}}} P_{u_{n-1}}(y|x_{n-1})} + \mathbf{1}_n^{\bar{K}} \cdot r(x_n).$$

We simulated the policy gradient algorithm on a GARNET$(30, 5, 10, 1)$ MDP. Linear FA was used with $L = 10, l = 2$. Figure 2 shows the average reward η as a function of iteration. These results indicate that the variance reduction in each iteration (guaranteed by Lemma 2.2) resulted, on average, in a better estimation of the gradient $\nabla_\theta \eta$, and therefore a better policy at each step.

6. Discussion and Future Work

Generally, when devising a solution to a difficult problem, one should incorporate into it all reliably available information. Model Free RL algorithms typically operate without explicit knowledge of the underlying environment, and therefore, when such knowledge is available, using these algorithms 'out of the box' is clearly suboptimal. In this work we have presented a general method of integrating partial environmental knowledge into a large class of model free algorithms. Our method improves the asymptotic behavior of the algorithm, and at each iteration reduces the estimation variance due to uncertainty in the environment. We have shown analytically (for TD(0)) and in simulation (for Policy Gradient) an improvement in the algorithm's overall performance.

A few issues are in need of further investigation. The first concerns the accuracy of the known set. In this work we have assumed that the partial environmental knowledge is precise. Clearly, such an assumption is not realizable in real world situations, and in a more realistic setup it is assumed that our partial knowledge contains some degree of error. This implies that our estimator (1) is no longer unbiased, though, if the bias is small then the asymptotic limits of algorithms (3) and (6) should be close. Therefore, one way to tackle this issue is to refine our known set such that some bound on the bias is guaranteed.

The second issue concerns the online acquisition of the known set. Using the SA algorithm (3), at the time of the n'th update of θ, we have already encountered a state-action trajectory of size n. Can we use this trajectory to construct an *estimated* partial MDP model, use it as in algorithm (6), and guarantee an improvement in the algorithm's performance? While one may easily construct such an algorithm, this should be done with caution, since using the same trajectory for updating the parameter and the estimated model may cause overfitting. Though this issue deserves careful analysis, we note that when a constant step size is used, the major influences on the current value of the parameter are the most recent measurements, which should render this effect negligible.

Relation to Real Time Dynamic Programming We conclude with an interesting note. When the model is fully known, implementing our method on Q-Learning and choosing a step size $\epsilon = 1$ results in an algorithm similar to Real Time Dynamic Programming (Barto et al., 1995) (RTDP). Thus in this case the IPM algorithm bridges between the model free Q-Learning and RTDP which uses a complete model.

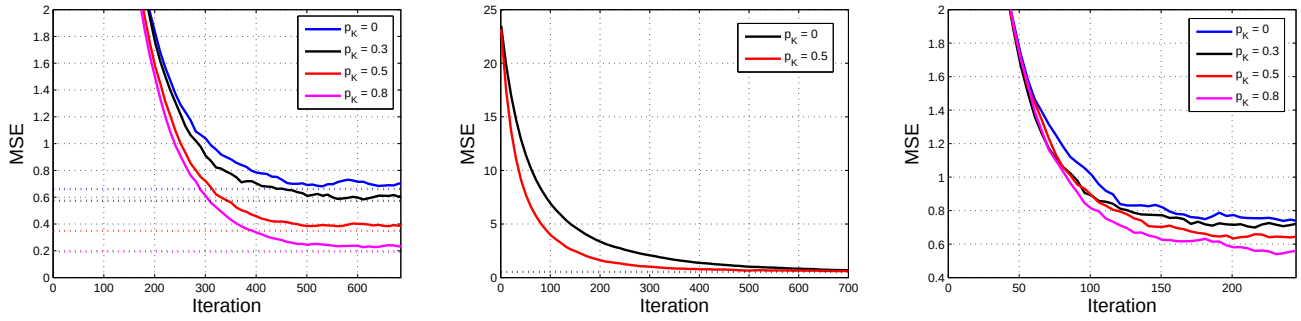

Figure 1. TD(0) with a Partial Model. *Left* : MSE of Table Based IPM-TD(0) on a GARNET(10,5,10,1) MDP with a deterministic random policy, for different values of p_K. Step size is $\epsilon = 0.2$. Dashed lines show the asymptotic MSE calculated by (13). *Middle* : MSE of Table Based IPM-TD(0) on a GARNET(10,5,10,1) MDP with a deterministic random policy. For $p_K = 0$ (black-solid) a step size $\epsilon = 0.15$ was used, and the asymptotic MSE was calculated using (13) (black-dashed). For $p_K = 0.5$ (red-solid) a step size was calculated (using (13)) such that its asymptotic MSE would equal that of $p_K = 0$. *Right* : MSE of linear FA IPM-TD(0) on a GARNET(30,5,10,1) MDP with a deterministic random policy, for different values of p_K. Step size is $\epsilon = 0.15$. The linear FA parameters are $L = 10$ and $l = 2$. A discount factor of $\gamma = 0.7$ was used in all simulations. All results are averaged over 500 different runs with the same initial conditions.

Acknowledgements

The research leading to these results has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under PASCAL2 (PUMP PRIMING) grant agreement no. 216886 and under a Marie Curie Reintegration Fellowship (IRG) grant agreement no. 249254.

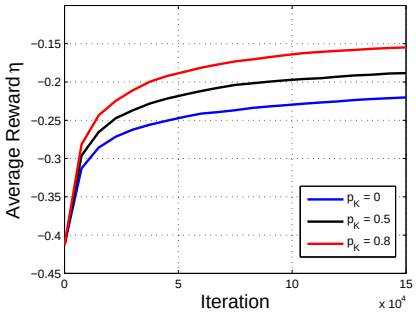

Figure 2. Policy Gradient with a Partial Model. Implementation of the algorithm described in section 5.2 on a GARNET(30,5,10,1) MDP, with step size parameters $\epsilon = 0.03$ and $\epsilon' = 0.003$. The linear FA parameters are $L = 10$ and $l = 2$. Average reward is plotted vs. iteration number for different values of p_K. Results are averaged over 500 different runs with the same initial conditions.

References

A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

D.P. Bertsekas and J. Tsitsiklis. *Neuro-dynamic Programming*. Athena Scinetific, 1996.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor–critic algorithms. Technical Report TR09-10, Univ. of Alberta, 2007.

P. Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999.

P. Dayan and T.J. Sejnowski. TD(λ) Converges with Probability 1. *Machine Learning*, 14:295–301, 1994.

P.R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal on Control and Optimization*, 23:329–380, 1985.

H.J. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer Verlag, 2003.

P. Marbach and J. Tsitsiklis. Simulation-based optimization of markov reward processes. *IEEE. Trans. Auto. Cont.*, 46(2):191–209, 1998.

A. Papoulis and S.U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, fourth edition, 2002.

S. Singh and P. Dayan. Analytical mean squared error curves for temporal difference learning. *Machine Learning*, 32:5–40, 1998.

R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.