Automatica 162 (2024) 111517

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

A kernel-based approximate dynamic programming approach: Theory and application ☆

Ali Forootani [a,*], Raffaele Iervolino [b], Massimo Tipaldi [c], Silvio Baccari [d]

[a] Hamilton Institute, Maynooth University, Maynooth, Co. Kildare W23F2K8, Ireland
[b] Department of Electrical Engineering and Information Technology, University of Naples, 80125 Napoli, Italy
[c] Department of Electrical and Information Engineering, Polytechnic University of Bari, 70126 Bari, Italy
[d] Department of Mathematics and Physics, University of Campania Luigi Vanvitelli, 81100 Caserta, Italy

ARTICLE INFO

Article history:
Received 23 January 2023
Received in revised form 14 October 2023
Accepted 11 December 2023
Available online xxxx

Keywords:
Dynamic Programming
Markov Decision Process
Approximate Dynamic Programming
Kernel function
Support Vector Machine
Sensor scheduling

ABSTRACT

This article proposes a novel kernel-based Dynamic Programming (DP) approximation method to tackle the typical curse of dimensionality of stochastic DP problems over the finite time horizon. Such a method utilizes kernel functions in combination with Support Vector Machine (SVM) regression to determine an approximate cost function for the entire state space of the underlying Markov Decision Process (MDP), by leveraging cost function computed for selected representative states. Kernel functions are used to define the so-called kernel matrix, while the parameter vector of the given kernel-based cost function approximation is computed by moving backwards in time from the terminal condition and by applying SVM regression. This way, the difficulty of selecting a proper set of features is also tackled. The proposed method is then extended to the infinite time horizon case. To show the effectiveness of the proposed approach, the resulting Recursive Residual Approximate Dynamic Programming (RR-ADP) algorithm is applied to the sensor scheduling design in multi-process remote state estimation problems.

1. Introduction

There are many engineering applications that employ Markov Decision Processes (MDPs) to address sequential decision making problems under uncertainties, e.g., sensor scheduling (Forootani, Iervolino, Tipaldi, & Dey, 2022), resource allocation in distributed energy systems (Kandil, Farag, Shaaban, & El-Sharafy, 2018), and autonomous vehicles (Pouya & Madni, 2021). However, in large MDPs, the curse of dimensionality in both the state and action space is unavoidable, and this makes their resolution via exact Dynamic Programming (DP) algorithms (e.g., Value Iteration (VI)) infeasible (Bertsekas, 2017; Powell, 2007). Therefore, efforts have been devoted to investigating methods able to solve the underlying stochastic DP problem approximately (Bertsekas, 2011, 2019a; Forootani et al., 2022). Such methods have been named differently in the literature and, among them, we can mention Approximate Dynamic Programming (ADP, used in this paper), Reinforcement Learning (RL), and Neural Dynamic Programming (Bertsekas, 2017, 2019b; Sutton & Barto, 2018).

All the effective implementations of ADP techniques have approximation approaches at their core, which can be grouped into two main categories. The former (addressed in this paper) is the value space approximation where the original cost function is approximated with another linear or non-linear function (Bertsekas, 2019b). The policy space approximation is a common alternative, where a policy is determined by optimizing across a sufficiently constrained class of policies, generally a parametric family of some sort (Bertsekas, 2019a; Powell, 2007). The value space approximation is more directly related to the basic DP notions of value and policy iteration, while the policy space approximation exploits more widely applicable optimization techniques, e.g., gradient-like descent methods (Bertsekas, 2017; Powell, 2007).

In the domain of cost function approximation, the choice of a suitable parametric approximation architecture for cost functions is significant for the success of the applied approximation approach (Bertsekas, 2017, 2019a). Several parametric approximation architectures have been considered in the literature, e.g., neural networks and linear-based architectures (Bertsekas, 2019a, 2019b; Forootani, Iervolino, Tipaldi, & Neilson, 2020). As for linear approximation architectures, we can mention projected equation approaches and aggregation methods.

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Vijay Gupta under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: aliforootani@ieee.org (A. Forootani), rafierv@unina.it (R. Iervolino), massimo.tipaldi@poliba.it (M. Tipaldi), silvio.baccari@unicampania.it (S. Baccari).

https://doi.org/10.1016/j.automatica.2024.111517
0005-1098/

The projected equation techniques are linked to Least Squares Temporal Difference (LSTD), which has its origins in RL (Bertsekas, 2011; Forootani, Iervolino, & Tipaldi, 2019; Sutton & Barto, 2018). Aggregate methods address the curse of dimensionality by defining some aggregation rules of the original system states: for instance, one can compute the costs of nonrepresentative states by interpolating the costs of specific representative states via aggregation probabilities (Bertsekas, 2019a). One of the key aspects of ADP methods is to ensure the convergence of the derived Bellman's operator (implicitly defined in the approximate Bellman's equation), used to produce better estimations of the optimal cost function (Bertsekas, 2019a). Furthermore, ADP techniques make use of sampled data (generated by computer simulators or real systems, and processed via Monte Carlo or bootstrapping) to solve the underlying optimality condition related to their approximate Bellman's equation and tune the parameter vector of the chosen approximation architecture (Bertsekas, 2019a; Powell, 2007). ADP cost function approximations usually employ feature vectors in order to replace the original cost function of a given state with a function that depends on such state through its feature vector (Bertsekas, 2011, 2019b). However, the definition of an appropriate set of features can be a difficult task since it requires a deep knowledge of the problem at hand. And, the feature selection process affects both the computational complexity of ADP methods and the quality of the resulting approximate cost function (Bertsekas, 2019a, 2019b).

This paper proposes a novel kernel-based function approximation approach in conjunction with Support Vector Machine (SVM) regression to address the curse of dimensionality of stochastic DP problems and also alleviate the difficulty of defining a proper set of features for the problem at hand. More specifically, by defining a set of representative states from the original MDP state space and employing a kernel function on this set, a kernel matrix can be built. Such kernel matrix replaces the feature matrix in the proposed ADP kernel-based architecture. Like aggregate methods, it is possible to compute the cost function of an arbitrary state from such representative states by using the kernel function and the parameter vectors computed by our proposed algorithm, named Recursive Residual ADP (RR-ADP). More specifically, by moving backwards over a finite time horizon from the terminal conditions and employing the resulting Bellman's operator, the cost-to-go function for the set of representative states is determined. This way, for each time slot, the parameter vector is calculated by minimizing the error between the cost-to-go functions of two consecutive iterations of the RR-ADP algorithm via SVM. It is also worth mentioning the two following aspects: (i) defining a kernel function over the state space of an MDP can also provide a continuous framework able to handle its possibly infinite dimensional state space; (ii) representative states can have particular properties, e.g., they can be the ones with more transitions for specific control actions (such states in network science are often called hubs, i.e., nodes with a number of links that greatly exceeds the average (Albert & Barabási, 2002)).

In the literature, there exist a few works applying kernel-based methods to DP. For instance, in Dietterich and Wang (2001) a kernel version of the linear programming approach for solving stochastic DP problems was considered. In Ormoneit and Sen (2002), the authors proposed a kernel-based approach to overcome the stability problems of RL temporal-difference learning in continuous state-spaces. In Jung and Polani (2006), the LSTD method was formulated in the framework of least squares SVM to cope with the large amount of training data, while the QR decomposition was employed to solve the underlying kernel-based optimization. In Bhat, Farias, and Moallemi (2023), a kernel-based variation of the smoothed approximate linear programming approach was proposed as a non parametric ADP method to alleviate sample complexity in the original parametric linear programming version reported in Desai, Farias, and Moallemi (2012) with theoretical guarantees. In Xu, Lian, Zuo, and He (2013), a kernel-based dual heuristic programming method, integrated into an actor-critic framework, was applied and tested on real-time control systems, with gradient-based learning techniques for approximating near-optimal control policies.

Unlike the previous works, in this paper, we integrate kernel methods and SVM regression into the native mechanism of the DP backward algorithm (Bertsekas, 2017) to solve approximately stochastic DP problems with large state space and mitigate the difficulty of defining a proper set of features for the chosen approximation architecture. This also allows the application of the resulting solution to both the finite and infinite time horizon. The contributions of this article can be summarized as follows: (i) proposing a novel ADP-based approach to mitigate the hardship of defining feature matrices by using a kernel function approximation architecture for the finite time horizon case; (ii) analyzing the properties of the consequent kernel-based Bellman's operator when applying the SVM regression; (iii) extending the proposed approach to the infinite time horizon case; (iv) evaluating the resulting RR-ADP algorithm on the transmission scheduling problem (Forootani et al., 2022) to show its effectiveness and applicability. The remainder of this paper is organized as follows. After providing some background on MDP, ADP, and SVM in Section 2, Section 3 presents the kernel-based ADP method with SVM for the finite time horizon case. In Section 4, the proposed method is extended to the infinite time horizon case. The RR-ADP algorithm is applied to the transmission scheduling problem in Section 5. Finally, Section 6 concludes the paper.

2. Preliminaries

In this paper, stochastic DP problems are formulated via discrete-time finite space MDPs, defined as a 4-tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{T}, \mathcal{R} \rangle$ where:

- $\mathcal{X}$ is the finite set of states with cardinality Ω. We denote with $x(l) = x \in \mathcal{X}$ and $x(l+1) = x' \in \mathcal{X}$ two generic components of this set at the time slots l and $l+1$, respectively.
- $\mathcal{U}$ is the finite set of actions. We denote with $u(l) = u \in \mathcal{U}$ a generic element of this set at the time slot l.
- $\mathcal{T} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is the state transition probability function. In particular, it is defined as $\mathcal{T}_{xx'}(u) := \left[P(x'|x, u)\right]$ and provides the probability of going from a given state x to the state x', when the control action u is applied.
- $\mathcal{R} : \mathcal{X} \to [0, +\infty)$ is the (non-negative) cost-per-stage function. We denote with $\mathcal{R}(x)$ the cost for a given state $x \in \mathcal{X}$.

We denote with $\mu(x, l)$ a decision function, i.e., a mapping between the entire state space $\mathcal{X}$ and the action set $\mathcal{U}$ at a given time step l. With a slight abuse of notation, in this paper we omit the explicit dependency on the state and define $\mu_l := \mu(x, l)$. If we define the policy $\pi = \{\mu_0, \ldots, \mu_{\mathcal{N}-1}\}$ as the collection of subsequent decision functions over a finite time horizon of length $\mathcal{N}$, then the expected cost function associated with such policy π and for a given initial state $x(0)$ can be expressed as follows

$$J_\pi\big(x(0)\big) = E\left[\sum_{l=0}^{\mathcal{N}-1} \alpha^l \mathcal{R}\left(x(l)\right) + J_{\mathcal{N}}\big(x(\mathcal{N})\big)\right], \tag{1}$$

where $E\{\cdot\}$ represents the expectation operator evaluated along the MDP trajectory under the policy π, $\alpha \in]0, 1[$ is the discount factor, and $J_{\mathcal{N}}(\cdot)$ is a known terminal cost for the final time slot. As

for the finite time horizon, the optimal discounted cost function can be defined as

$$J^*\big(x(0)\big) = \min_{\pi} \; E\left[\sum_{l=0}^{\mathcal{N}-1} \alpha^l \mathcal{R}\,\big(x(l)\big) + J_{\mathcal{N}}\big(x(\mathcal{N})\big)\right], \tag{2}$$

with $\pi^* = \{\mu_0^*, \ldots, \mu_{\mathcal{N}-1}^*\}$ being the associate optimal policy. With a slight abuse of notation, we can define (1) and (2) for a generic state $x \in \mathcal{X}$ as $J_\pi(x)$ and $J^*\big(x\big)$, respectively. If we refer to the entire set $\mathcal{X}$, the global (optimal) cost function vector is denoted with $J_\pi \in \mathbb{R}^{\Omega}$, with components $J_\pi(x)$ ($J^* \in \mathbb{R}^{\Omega}$, with components $J^*(x)$). For any cost function J and decision function μ, we introduce the corresponding Bellman's operator $\mathcal{F} : \mathbb{R}^{\Omega} \to \mathbb{R}^{\Omega}$, which, in compact matrix form, can be expressed as $\mathcal{F}J = \mathcal{R} + \alpha\mathcal{P}J$, with $\mathcal{P} \in \mathbb{R}^{\Omega\times\Omega}$ being the state transition probability matrix associated to μ, whose elements are $\mathcal{P}_{xx'} := \mathcal{T}_{xx'}(\mu)$ (Bertsekas, 2011).[1] The optimal Bellman's operator $\mathcal{F}^* : \mathbb{R}^{\Omega} \to \mathbb{R}^{\Omega}$ is defined as

$$(\mathcal{F}^*J)(x) = \min_{u\in\mathcal{U}}\big[\mathcal{R}(x) + \alpha \sum_{x'\in\mathcal{X}} \mathcal{P}_{xx'}J(x')\big], \tag{3}$$

which, in compact matrix form, becomes $\mathcal{F}^*J = \mathcal{R} + \alpha\mathcal{P}^*J$, with $\mathcal{P}^*$ being the state transition probability matrix associated to the resolution of the minimization operator in (3). The DP backward algorithm can be used to solve stochastic DP problems over the finite time horizon (Bertsekas, 2017). This means applying iteratively the optimal Bellman operator $J_l^* = \mathcal{F}^*J_{l+1}^*$ by moving backwards in time (i.e., $l = \mathcal{N}-1, \ldots, 0$) and starting from the known terminal cost $J_{\mathcal{N}}$. It is worth noting that both the resulting optimal cost function J_l^* and its related optimal decision function μ_l^* depend on l in general.[2]

It is worth noting that, by letting $\mathcal{N} \to \infty$ and setting $J_{\mathcal{N}} = 0$ in the formulas (1), (2), the infinite time horizon case can be addressed (Bertsekas, 2017). In such a case, the dependency on l is completely dropped since only stationary policies $\pi = \{\mu, \mu, \ldots\}$ are considered (Bertsekas, 2019a; Forootani et al., 2022). In case of discounted problems and bounded costs, the optimal stationary cost function J^* is the fixed point of the Bellman's equation, i.e., $J^* = \mathcal{F}^*J^*$ and $J^* = \big(I - \alpha\mathcal{P}^*\big)^{-1}\mathcal{R}$ (Bertsekas, 2011, 2019a; Forootani et al., 2019).

2.1. Brief on ADP and Support Vector Regression (SVR)

When dealing with stochastic DP problems of real applications, large state space issues can easily occur (Bertsekas, 2011; Forootani et al., 2022). In this paper, the following linear approximation architecture for cost functions is adopted

$$\tilde{J}(x) = \sum_{i=1}^{m} w_i\phi_i(x) = \Big\langle w, \phi(x)\Big\rangle, \tag{4}$$

where $\phi(x) = [\phi_1(x), \ldots, \phi_m(x)]^T$ is known as the feature (or basis function) vector with the corresponding elements $\phi_i(x)$ being scalar functions, $i = 1, \ldots, m$. We denote by Φ the feature matrix, i.e., the matrix whose rows are the features evaluated at each state. Moreover, we call $w = [w_1, \ldots, w_m]^T$ as the parameter (or weight) vector. Here the notation $\big\langle\cdot,\cdot\big\rangle$ represents the standard inner product. The aim of the SVM regression (or Support Vector Regression (SVR)) is to learn the cost function $\tilde{J}(x)$ by using the above linear approximation architecture (Smola & Schölkopf, 2004). The representation (4) provides a suitable approximation to the training data set $\mathcal{D} = \big\{\big(x, J(x)\big)\big|x \in \mathcal{X}\big\}$, where $x \in \mathcal{X}$ and $J(x)$ are the input and the observed output data, respectively. During the SVR training procedure, the following quadratic problem has to be solved (Smola & Schölkopf, 2004)

$$\min_{w,\xi,\xi^*} \frac{1}{2}\|w\|^2 + c\sum_{x\in\mathcal{X}}\big(\xi(x) + \xi^*(x)\big) \tag{5}$$

subject to

$$J(x) - \Big\langle w, \phi(x)\Big\rangle \le \epsilon + \xi(x) \tag{6a}$$

$$-J(x) + \Big\langle w, \phi(x)\Big\rangle \le \epsilon + \xi^*(x) \tag{6b}$$

$$\xi(x), \xi^*(x) \ge 0, \quad \forall x \in \mathcal{X}. \tag{6c}$$

where the term $\frac{1}{2}\|w\|^2$ is used to enforce the flatness of the function to optimize, for regularization purposes, and to achieve a proper bias–variance trade-off in the resulting approximate cost function (Smola & Schölkopf, 2004). Moreover, $\xi(x)$ and $\xi^*(x)$ are called slack variables. They become active whenever the distance between the training point $J(x)$ and the approximating function $\tilde{J}(x)$ is greater than ϵ (this is called ϵ-insensitive loss function (Smola & Schölkopf, 2004)). The resolution of the quadratic optimization problem (5), (6) is

computationally expensive. Therefore, such optimization problem can be solved in its dual form as follows (Keerthi, Shevade, Bhattacharyya, & Murthy, 2001; Smola & Schölkopf, 2004)

$$\max_{\lambda,\lambda^*} -\frac{1}{2}\sum_{x,x'\in\mathcal{X}}\big(\lambda^*(x) - \lambda(x)\big)\big(\lambda^*(x') - \lambda(x')\big)\Big\langle\phi(x), \phi(x')\Big\rangle - \epsilon\sum_{x\in\mathcal{X}}\big(\lambda^*(x) + \lambda(x)\big) + \sum_{x\in\mathcal{X}}J(x)\big(\lambda^*(x) - \lambda(x)\big) \tag{7}$$

subject to $0 < \lambda(x), \lambda^*(x) < c, \ \forall x \in \mathcal{X}$, with $\lambda(\cdot)$ and $\lambda^*(\cdot)$ being the dual variables. In this setting, the feature vector $\phi(x)$ enters into the optimization as an inner product. This allows defining the following kernel function

$$\mathcal{K}(x, x') := \Big\langle\phi(x), \phi(x')\Big\rangle. \tag{8}$$

Using kernel functions allows reducing the computational overhead for the parameter vector tuning since it becomes possible to avoid the direct computation of the feature vectors $\phi(x)$ and $\phi(x')$. Then, we can compute both the parameter vector w and the approximate cost function $\tilde{J}(x)$ by the two following relations (Keerthi et al., 2001; Smola & Schölkopf, 2004)

$$w = \sum_{x\in\mathcal{X}}\big(\lambda(x) - \lambda^*(x)\big)\phi(x), \tag{9}$$

$$\tilde{J}(x) = \Big\langle w, \phi(x)\Big\rangle = \sum_{x'\in\mathcal{X}}\big(\lambda(x') - \lambda^*(x')\big)\Big\langle\phi(x'), \phi(x)\Big\rangle = \sum_{x'\in\mathcal{X}}\big(\lambda(x') - \lambda^*(x')\big)\mathcal{K}(x, x'). \tag{10}$$

A kernel function $\mathcal{K}(\cdot,\cdot) : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ measures the similarity between two states x and x'. Note that the selection of a suitable kernel function has a great impact on the success of the learning method. The kernel matrix $\mathcal{Y}$ is defined as the matrix with components $\mathcal{Y}_{xx'} = \mathcal{K}(x, x')$, or equivalently, as $\mathcal{Y} = \Phi\Phi^T$. In case the kernel matrix $\mathcal{Y}$ is symmetric and positive semi-definite, it is said to be *admissible*. If $\mathcal{Y}$ is positive definite, it is *non-degenerate*. Note that the last property holds whenever the feature matrix Φ is full column rank.

[1] Note that, for the sake of simplicity, the dependency of $\mathcal{P}$ on the policy has been omitted.

[2] Since μ_l^* can vary over the finite time horizon, the matrix $\mathcal{P}^*$ implicitly depends on l. For the sake of simplicity in the adopted notation, such dependency has been omitted in this paper.

3. Kernel-based ADP with SVR for the finite time horizon

This section presents the details of our ADP approach which combines the standard DP backward algorithm with kernel function approximation architecture and SVR. Starting from the known terminal condition $J_{\mathcal{N}}$ and by applying Bellman's principle of optimality (Bertsekas, 2017), we move backwards in time and, at each time slot, we use SVR to approximate the cost function. More specifically, let us consider one iteration of Bellman's operator at an arbitrary time slot l which is $J_l^* = \mathcal{F}^* J_{l+1}^*$, with $l = \mathcal{N}-1, \ldots, 0$. This relation can be written in the form of $J_l^* - \mathcal{F}^* J_{l+1}^* = 0$. As mentioned earlier, solving it for large-scale MDPs is impractical since it is a linear system of Ω equations. Let us call $\tilde{J}_l^*$ the approximate cost-to-go function at the iteration l. We introduce the following error norm between two iterations

$$\left\|\mathcal{RE}_l\right\| = \left\|\tilde{J}_l^* - \mathcal{F}^*\tilde{J}_{l+1}^*\right\| = \left(\sum_{x\in\mathcal{X}} \left|\tilde{J}_l^* - \mathcal{R} - \alpha\mathcal{P}^*\tilde{J}_{l+1}^*\right|^2\right)^{\frac{1}{2}}, \tag{11}$$

where the term $\tilde{J}_l^* - \mathcal{F}^*\tilde{J}_{l+1}$ is called Residual Error (RE) at the time slot l. Finding an ADP approach to minimize the $\mathcal{RE}_l$ for a collection of candidate cost-to-go functions is a kind of art and complicated since achieving zero error implies solving it exactly. It is quite obvious that computing such error is impractical since it requires to be computed for every state of the MDP. Therefore, it is natural to take representative states from the state space $\mathcal{X}$. The resulting set is denoted by $\tilde{\mathcal{X}}$. Hence, we will work with the norm of the Residual Error approximation at time slot l, say $\tilde{\mathcal{RE}}_l$, based on these representative states $x \in \tilde{\mathcal{X}}$, i.e.

$$\left\|\tilde{\mathcal{RE}}_l\right\| = \left(\sum_{x\in\tilde{\mathcal{X}}} \left|\tilde{J}_l^* - \mathcal{R} - \alpha\mathcal{P}^*\tilde{J}_{l+1}^*\right|^2\right)^{\frac{1}{2}}. \tag{12}$$

Several approximation architectures have been considered in the literature, in particular, linear and neural network-based architectures (Bertsekas, 2019b; Bishop, 1995). Such approximation architectures are finite-dimensional, hence it is not always possible to satisfy the condition $\tilde{\mathcal{RE}}_l = 0$. This work proposes a novel kernel-based ADP method and shows the potential of kernel-based approximation architectures (Schölkopf, Smola, Bach, et al., 2002) to guarantee $\tilde{\mathcal{RE}}_l = 0$. Moreover, SVR is used solve the regression problem resulting from (12). In other words, for a given MDP and a set of representative states $\tilde{\mathcal{X}}$, the parameter vector of the kernel-based approximate cost function $\tilde{J}_l^*$ is computed under the condition that, for each l, $\tilde{\mathcal{RE}}_l$ is equal to zero. Then, by employing the chosen kernel function, the linear mapping (10) is applied to calculate the associate cost-to-go function $\tilde{J}_l^*(x)$ for each $x \in \mathcal{X}$.

To start with, by using (4), we can write the approximate cost function $\tilde{J}_l^*$ at the generic state $x \in \mathcal{X}$ as an inner product between the feature vector $\phi(x)$ and the weight vector w_l as follows

$$\tilde{J}_l^*(x) = \left\langle w_l, \phi(x)\right\rangle, \quad x \in \mathcal{X}. \tag{13}$$

Then, the recursive residual error can be expressed as follows

$$\begin{aligned}\mathcal{RE}_l(x) &= \tilde{J}_l^*(x) - \mathcal{F}^*\tilde{J}_{l+1}(x)\\ &= \tilde{J}_l^*(x) - \Big(\mathcal{R}(x) + \alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\tilde{J}_{l+1}^*(x'')\Big),\end{aligned} \tag{14}$$

where $x'' \in \mathcal{X}$ is another generic state at the time slot $l+1$. By replacing the first term with its approximate value (13), we have

$$\mathcal{RE}_l(x) = \left\langle w_l, \phi(x)\right\rangle - \Big(\mathcal{R}(x) + \alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\tilde{J}_{l+1}^*(x'')\Big). \tag{15}$$

Since we are moving backwards, the parameter vector w_{l+1} is known from the previous iteration, therefore we have

$$\mathcal{RE}_l(x) = \left\langle w_l, \phi(x)\right\rangle - \Big(\mathcal{R}(x) + \alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\left\langle w_{l+1}, \phi(x'')\right\rangle\Big). \tag{16}$$

By replacing the $\mathcal{RE}_l$ in the SVR framework and by employing only the representative states $x \in \tilde{\mathcal{X}}$, we have

$$\min_{w_l,\xi+\xi^*} \frac{1}{2}\|w_l\|^2 + c\sum_{x\in\tilde{\mathcal{X}}}\big(\xi(x) + \xi^*(x)\big) \tag{17}$$

subject to

$$-\mathcal{R}(x) + \left\langle w_l, \phi(x)\right\rangle - \Big(\alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\left\langle w_{l+1}, \phi(x'')\right\rangle\Big) \le \epsilon + \xi^*(x), \tag{18a}$$

$$\mathcal{R}(x) - \left\langle w_l, \phi(x)\right\rangle + \Big(\alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\left\langle w_{l+1}, \phi(x'')\right\rangle\Big) \le \epsilon + \xi(x), \tag{18b}$$

$$\xi(x), \xi^*(x) \ge 0, \quad \forall x \in \tilde{\mathcal{X}}. \tag{18c}$$

Then, the dual problem is

$$\begin{aligned}\max_{\lambda,\lambda^*} &-\frac{1}{2}\sum_{x,x'\in\tilde{\mathcal{X}}}\big(\lambda^*(x) - \lambda(x)\big)\big(\lambda^*(x') - \lambda(x')\big)\left\langle\phi(x), \phi(x')\right\rangle\\ &-\epsilon\sum_{x\in\tilde{\mathcal{X}}}\big(\lambda^*(x) + \lambda(x)\big)\\ &+\sum_{x\in\tilde{\mathcal{X}}}\Big(\big(\mathcal{R}(x) + \alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\left\langle w_{l+1}, \phi(x'')\right\rangle\big)\big(\lambda^*(x) - \lambda(x)\big)\Big),\end{aligned} \tag{19}$$

subject to $0 \le \lambda(x), \lambda^*(x) \le c, \forall x \in \tilde{\mathcal{X}}$. Note that, in (18) and (19), the state x'' may not necessarily belong to the set of the representative states $\tilde{\mathcal{X}}$, since it comes from the transition probability matrix $\mathcal{P}^*_{xx''}$ which takes into account all the possible one-step transitions starting from $x \in \tilde{\mathcal{X}}$.

In the SVR framework, the primal parameter vector w_l can be computed as follows

$$w_l = \sum_{x\in\tilde{\mathcal{X}}}(\lambda_l(x) - \lambda_l^*(x))\phi(x), \tag{20}$$

where $\lambda_l(x)$ and $\lambda_l^*(x)$ represent the dual variables computed at the iteration l and for any $x \in \tilde{\mathcal{X}}$ Note that, as for their vector form, we have $\lambda_l, \lambda_l^* \in \mathbb{R}^{\tilde{\Omega}}$, with $\tilde{\Omega}$ being the cardinality of $\tilde{\mathcal{X}}$. As a result, for any state $x \in \mathcal{X}$, the cost function $\tilde{J}_l^*(x)$ can be computed as follows

$$\begin{aligned}\tilde{J}_l^*(x) &= \left\langle w_l, \phi(x)\right\rangle = \sum_{x'\in\tilde{\mathcal{X}}}\big(\lambda_l(x') - \lambda_l^*(x')\big)\left\langle\phi(x'), \phi(x)\right\rangle\\ &= \sum_{x'\in\tilde{\mathcal{X}}}\big(\lambda_l(x') - \lambda_l^*(x')\big)\mathcal{K}(x, x').\end{aligned} \tag{21}$$

The SVR approach in its traditional form was introduced to solve the problem of noisy data. Therefore, the selection of a regression function that exactly matches the noisy data is not suitable. Employing ϵ-insensitive loss function of the forms (18) allows the data points to be within a distance ϵ of the regression function without increasing the objective function. However, in our setup, the instantaneous cost function $\mathcal{R}(x)$ is considered exact and we have a perfect observation. Therefore, to make the recursive residual $\tilde{\mathcal{RE}}_l$ zero, the regression function $\left\langle w_l, \phi(x)\right\rangle$ has to reproduce the instantaneous cost function precisely, so that

$$\tilde{\mathcal{RE}}_l(x) = \left\langle w_l, \phi(x)\right\rangle - \mathcal{R}(x) - \alpha\sum_{x''\in\mathcal{X}} \mathcal{P}^*_{xx''}\left\langle w_{l+1}, \phi(x'')\right\rangle = 0. \tag{22}$$

If we fix the values c and ϵ, then the constrained minimization problem (17), (18) can be expressed in a simpler form. If $c = \infty$, then a feasible solution must satisfy $\xi(x) = \xi^*(x) = 0, \forall x \in \tilde{\mathcal{X}}$, otherwise the objective function will be unbounded. Moreover, if $\epsilon = 0$, then inequalities in (18) turn to equalities. It is worth highlighting that if we set $\epsilon = 0$ and $c = \infty$, we can reach the standard SVR problem to satisfy (22). With these explanations, for $\epsilon = 0$ and $c = \infty$, the constrained minimization problem (17), (18) can be written as

$$\min_{w_l} \frac{1}{2}\|w_l\|^2, \tag{23}$$

subject to

$$\Big\langle w_l, \phi(x)\Big\rangle - \mathcal{R}(x) - \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \Big\langle w_{l+1}, \phi(x'')\Big\rangle = 0, \ \forall x \in \tilde{\mathcal{X}}. \tag{24}$$

By using the Lagrange multiplier approach, we can write

$$\mathcal{L}(w_l, \lambda) = \frac{1}{2}\|w_l\|^2 + \sum_{x \in \tilde{\mathcal{X}}} \lambda(x)\Big(\Big\langle w_l, \phi(x)\Big\rangle - \mathcal{R}(x) - \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \Big\langle w_{l+1}, \phi(x'')\Big\rangle\Big). \tag{25}$$

To minimize the above relation, we take the gradient of $\mathcal{L}(w_l, \lambda)$ with respect to w_l and set it to 0

$$\frac{\partial \mathcal{L}}{\partial w_l} = w_l - \sum_{x \in \tilde{\mathcal{X}}} \lambda(x)\phi(x) = 0, \tag{26}$$

hence

$$w_l = \sum_{x \in \tilde{\mathcal{X}}} \lambda(x)\phi(x). \tag{27}$$

Consequently, we can express the cost function $\tilde{J}^*_l(x)$ as

$$\tilde{J}^*_l(x) = \Big\langle w_l, \phi(x)\Big\rangle = \sum_{x' \in \tilde{\mathcal{X}}} \lambda(x')\Big\langle \phi(x'), \phi(x)\Big\rangle = \sum_{x' \in \tilde{\mathcal{X}}} \lambda(x')\Big(\mathcal{K}(x', x)\Big), \quad \forall x \in \mathcal{X}. \tag{28}$$

To reach the dual problem, at a generic step, we need to substitute (27) into (25) and maximize it with respect to λ, then we have

$$\max_{\lambda} \ -\frac{1}{2} \sum_{x, x' \in \tilde{\mathcal{X}}} \lambda(x)\lambda(x')\Big\langle \phi(x), \phi(x')\Big\rangle + \sum_{x \in \tilde{\mathcal{X}}} \lambda(x)\Big(\mathcal{R}(x) + \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \Big\langle w_{l+1}, \phi(x'')\Big\rangle\Big). \tag{29}$$

Eq. (29) can be shown in the vector form as follows

$$\max_{\lambda} -\frac{1}{2}\lambda^T \mathcal{Y}\lambda + \lambda^T \mathcal{J}_l, \tag{30}$$

where $\mathcal{Y}_{xx'} = \Big\langle \phi(x), \phi(x')\Big\rangle = \mathcal{K}(x, x'), \ \forall x, x' \in \tilde{\mathcal{X}}$, and $\mathcal{J}_l \in \mathbb{R}^{\tilde{\Omega}}$ with its components given by

$$\mathcal{J}_l(x) = \mathcal{R}(x) + \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \Big\langle w_{l+1}, \phi(x'')\Big\rangle, \forall x \in \tilde{\mathcal{X}}. \tag{31}$$

As we can see, (30) is an unconstrained maximization quadratic problem. Note that since $\mathcal{K}$ is non-degenerative, the associated residual recursive error is also non-degenerative. Moreover, matrix $\mathcal{Y}$ is full-rank and positive definite, hence the quadratic optimization problem (30) has a unique optimum. Taking the gradient of (30) and set to 0 we have

$$\mathcal{Y}\lambda_l = \mathcal{J}_l, \quad \lambda_l \in \mathbb{R}^{\tilde{\Omega}}. \tag{32}$$

It is worth to highlight that the linear system of equations (32) has the dimension $\tilde{\Omega}$. The dimensionality of this equation can be made much lower compared to the original one if $\tilde{\Omega} \ll \Omega$.

In the following theorem, we claim that recursive residual errors are precisely 0 at the representative states $\tilde{\mathcal{X}}$.

Theorem 3.1. *If the kernel function $\mathcal{K}(x, x') = \Big\langle \phi(x), \phi(x')\Big\rangle$ is non-degenerate, then the approximate cost function $\tilde{J}^*_l(x)$, computed by solving the constrained minimization problem (23), (24), satisfies the following*

$$\tilde{J}^*_l(x) = \mathcal{R}(x) + \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \tilde{J}^*_{l+1}(x''), \quad \forall x \in \tilde{\mathcal{X}}, \tag{33}$$

which implies that recursive errors $\mathcal{RE}_l(x)$ are equal to 0 for all state $x \in \tilde{\mathcal{X}}$.

Proof. We know that $\mathcal{K}(x, x')$ is non-degenerate, then the matrix $\mathcal{Y}$ is positive definite and invertible. Hence, the constrained minimization problem (23), (24) has the unique solution given by (32). This implies that (33) is satisfied and $\tilde{\mathcal{RE}}_l(x) = 0$, see (13), (22), (24), and (28). □

Lemma 3.2. *If the kernel function $\mathcal{K}(x, x') = \Big\langle \phi(x), \phi(x')\Big\rangle$ is non-degenerate, and we have $\tilde{\mathcal{X}} = \mathcal{X}$, then cost function $\tilde{J}^*_l(x)$ generated by our proposed method satisfies the expression $\tilde{J}^*_l(x) = J^*_l(x), \ \forall x \in \mathcal{X}$, which says that the approximation is identical to the exact solution.*

Proof. Using the results of Theorem 3.1, if $\tilde{\mathcal{X}} = \mathcal{X}$, then $\mathcal{RE}_l(x) = 0, \ \forall x \in \mathcal{X}$. Considering the definition of recursive residual, we have $\mathcal{RE}_l(x) = \tilde{J}^*_l(x) - \mathcal{F}^*\tilde{J}^*_{l+1}(x)$, which means that $\tilde{J}^*_l(x) = J^*_l(x)$. □

Remark 1. Since Theorem 3.1 holds, we have $\tilde{J}^*_l = \mathcal{J}_l$. This can be easily verified by comparing the right-hand side of (31) and (33).

Theorem 3.3. *If the kernel function $\mathcal{K}(x, x') = \Big\langle \phi(x), \phi(x')\Big\rangle$ is non-degenerate, and we consider stationary policies, then the sequence of computed vectors λ_l is convergent to its steady-state value $\bar{\lambda} = \Big(I - \alpha \mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}\Big)^{-1}\mathcal{Y}^{-1}\mathcal{R}$.*

Proof. Since $\mathcal{K}(x, x')$ is non-degenerate, then the matrix $\mathcal{Y}$ is positive definite and invertible. Based on the results of Theorem 3.1 and from Remark 1, we can replace $\tilde{J}^*_l$ with $\mathcal{Y}\lambda_l$ in Bellman's recursive iteration. Therefore, for the case of a stationary policy with associated transition probability matrix $\mathcal{P}^*$, we have $\mathcal{Y}\lambda_l = \mathcal{R} + \alpha\mathcal{P}^*\mathcal{Y}\lambda_{l+1}$ and

$$\lambda_l = \mathcal{Y}^{-1}\mathcal{R} + \alpha\mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}\lambda_{l+1}. \tag{34}$$

Such relation is a difference equation whose behavior is determined by the term $\alpha\mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}$. Being the matrices $\mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}$ and $\mathcal{P}^*$ similar, they have the same eigenvalues. Since the eigenvalues of $\mathcal{P}^*$ are ≤ 1 and being $\alpha < 1$, the eigenvalues of $\alpha\mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}$ are then strictly inside the unit circle, which implies the sequence of λ_l is convergent[3] and its unique (constant) stationary solution can be found from (34) by $\bar{\lambda} = \Big(I - \alpha\mathcal{Y}^{-1}\mathcal{P}^*\mathcal{Y}\Big)^{-1}\mathcal{Y}^{-1}\mathcal{R}$, where $\bar{\lambda}$ is the steady state value of λ, which completes the proof. □

[3] The eigenvalues of $\alpha\mathcal{P}^*$ also determine its convergence rate.

Since we make use of Bellman's backward operator and apply the residual-based approximation architecture recursively, we call our approach Recursive Residual Approximate Dynamic Programming (RR-ADP). Algorithm 1 summarizes the steps of the RR-ADP for the finite time horizon. As we can notice, the proposed algorithm requires the choice of kernel functions without the need of specifying the feature vectors. Moreover, by exploiting the native mechanism of the DP backward algorithm (Bertsekas, 2017), one can simultaneously compute, at any stage l, the approximate optimal decision function, say $\tilde{\mu}_l^*$.

Algorithm 1 RR-ADP Algorithm

- Select the inputs:
  - Appropriate representative set of states $\tilde{\mathcal{X}}$
  - Suitable kernel function $\mathcal{K}$ defined on $\mathcal{X} \times \mathcal{X}$
- $l \leftarrow \mathcal{N} - 1$
- Compute the associated kernel function $\mathcal{K}$ and the kernel matrix $\mathcal{Y}$ for all $x, x' \in \tilde{\mathcal{X}}$, i.e. $\mathcal{Y}_{x,x'} = \mathcal{K}(x, x')$
- $\forall x \in \mathcal{X}$ put $J_{\mathcal{N}}(x) = \mathcal{R}(x)$, hence we set $\tilde{J}^*_{\mathcal{N}} = J_{\mathcal{N}}$

while $l \geq 0$ **do**

  - $\forall x \in \tilde{\mathcal{X}}$, compute $\mathcal{R}(x)$
  - $\forall x \in \tilde{\mathcal{X}}$, construct $\mathcal{J}_l(x)$ based on the Eq. (31)
  - Solve linear system of equations (32) to compute λ_l by using $\mathcal{Y}$ and $\mathcal{J}_l$
  - $\forall x \in \mathcal{X}$, compute the cost to go function $\tilde{J}^*_l(x)$ based on (28)
  - $l \leftarrow l - 1$

end

- Output: λ_l, $l = 0, 1, \ldots, \mathcal{N} - 1$.

4. Extension to the infinite time horizon

In this section, we consider the case when time goes to infinity and we analyze the proposed recursive residual error scheme and its properties. For the infinite time horizon case, it is known that $J^* = \mathcal{F}^* J^*$ and only stationary cost functions and policies are considered (Bertsekas, 2011, 2019a). Consequently, the dependency on l can be dropped. The residual error at the state x becomes

$$\mathcal{RE}(x) = \tilde{J}^*(x) - \mathcal{F}^*\tilde{J}^*(x) = \tilde{J}^*(x) - \Big(\mathcal{R}(x) + \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}^*_{xx'} \tilde{J}^*(x')\Big). \quad (35)$$

By replacing the first term of the above relation with the right-hand side of (13), we have

$$\mathcal{RE}(x) = \Big\langle w, \phi(x)\Big\rangle - \Big(\mathcal{R}(x) + \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}^*_{xx'} \Big\langle w, \phi(x')\Big\rangle\Big). \quad (36)$$

Since the inner product $\langle \cdot, \cdot \rangle$ is linear, we can write $\mathcal{RE}(x)$ as

$$\begin{aligned}\mathcal{RE}(x) &= -\mathcal{R}(x) + \Big\langle w, \Big(\phi(x) - \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}^*_{xx'} \phi(x')\Big)\Big\rangle \\ &= -\mathcal{R}(x) + \Big\langle w, \psi(x)\Big\rangle, \end{aligned} \quad (37)$$

with

$$\psi(x) = \phi(x) - \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}^*_{xx'} \phi(x'). \quad (38)$$

Note that $\psi(x)$ constructs a new feature mapping that represents a linear combination of the original features ϕ for the state x and all the states x' that are reached by one-step transition from x. The following Lemma proves that the columns of the matrix whose rows are made of the feature vectors $\psi(x)$ for all $x \in \mathcal{X}$ are linearly independent.

Lemma 4.1. *If the feature matrix Φ has full column rank, then the feature matrix Ψ, whose rows are $\psi(x) = \phi(x) - \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}_{xx'} \phi(x')$, has full column rank as well.*

Proof. Say Υ the real vector space spanned by the vectors $\{\phi(x) | x \in \mathcal{X}\}$. From (38), we know that $\psi(x)$ is a linear combination of vectors in Υ. Hence, there exists a linear operator B that maps Φ to Ψ, i.e., $\Psi = B\Phi$, with $B = (I - \alpha\mathcal{P})$. From the properties of stochastic matrices, the largest eigenvalue of $\mathcal{P}$ is 1, while the others lie within the unit circle. Hence, all the corresponding eigenvalues of $\alpha\mathcal{P}$ have modulus not greater than $\alpha < 1$. Moreover, all the eigenvalues of I are equal to 1, therefore $(I - \alpha\mathcal{P})$ is full rank and $dim\big(ker(B)\big) = 0$. Hence the matrix Ψ is full column rank, being $\Psi = B\Phi$ and Φ of full column rank. □

By replacing the $\mathcal{RE}$ in the SVR framework and by employing only the representative states $x \in \tilde{\mathcal{X}}$, we have

$$\min_{w, \xi + \xi^*} \frac{1}{2}\|w\|^2 + c \sum_{x \in \tilde{\mathcal{X}}} \big(\xi(x) + \xi^*(x)\big) \quad (39)$$

subject to

$$-\mathcal{R}(x) + \Big\langle w, \psi(x)\Big\rangle \leq \epsilon + \xi^*(x), \quad (40a)$$

$$\mathcal{R}(x) - \Big\langle w, \psi(x)\Big\rangle \leq \epsilon + \xi(x), \quad (40b)$$

$$\xi(x), \xi^*(x) \geq 0, \quad \forall x \in \tilde{\mathcal{X}}. \quad (40c)$$

Note that the original feature vector $\phi(x)$ is substituted by the feature vector $\psi(x)$. Then, the dual problem is

$$\begin{aligned}\max_{\lambda, \lambda^*} &-\frac{1}{2} \sum_{x, x' \in \tilde{\mathcal{X}}} \big(\lambda^*(x) - \lambda(x)\big)\big(\lambda^*(x') - \lambda(x')\big)\Big\langle \psi(x), \psi(x')\Big\rangle \\ &-\epsilon \sum_{x \in \tilde{\mathcal{X}}} \big(\lambda^*(x) + \lambda(x)\big) + \sum_{x \in \tilde{\mathcal{X}}} \mathcal{R}(x)\big(\lambda^*(x) - \lambda(x)\big) \end{aligned} \quad (41)$$

subject to $0 \leq \lambda(x), \lambda^*(x) \leq c, \forall x \in \tilde{\mathcal{X}}$. Note that the dual minimization problem has the kernel function $\Gamma(x, x') = \Big\langle \psi(x), \psi(x')\Big\rangle$. The following relation holds between the two kernel functions $\mathcal{K}$ and Γ

$$\begin{aligned}\Gamma(x, x') &= \Big\langle \psi(x), \psi(x')\Big\rangle \\ &= \Big\langle \phi(x) - \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{xx''} \phi(x''), \phi(x') - \alpha \sum_{x'' \in \mathcal{X}} \mathcal{P}^*_{x'x''} \phi(x'')\Big\rangle \\ &= \Big\langle \phi(x), \phi(x')\Big\rangle \\ &-\alpha \sum_{x'' \in \mathcal{X}} \Big(\mathcal{P}^*_{x'x''} \Big\langle \phi(x), \phi(x'')\Big\rangle + \mathcal{P}^*_{xx''} \Big\langle \phi(x'), \phi(x'')\Big\rangle\Big) \\ &+\alpha^2 \sum_{x'', x''' \in \mathcal{X}} \mathcal{P}^*_{xx''} \mathcal{P}^*_{x'x'''} \Big\langle \phi(x''), \phi(x''')\Big\rangle \\ &= \mathcal{K}(x, x') - \alpha \sum_{x'' \in \mathcal{X}} \Big(\mathcal{P}^*_{x'x''} \mathcal{K}(x, x'') + \mathcal{P}^*_{xx''} \mathcal{K}(x', x'')\Big) \\ &+\alpha^2 \sum_{x'', x''' \in \mathcal{X}} \mathcal{P}^*_{xx''} \mathcal{P}^*_{x'x'''} \mathcal{K}(x'', x'''). \end{aligned} \quad (42)$$

The following theorem proves the non-degenerate property of $\Gamma(x, x')$.

Theorem 4.2. *If the kernel function $\mathcal{K}(x, x') = \Big\langle \phi(x), \phi(x')\Big\rangle$ is non-degenerate, then the new kernel mapping $\Gamma(x, x') = \Big\langle \psi(x), \psi(x')\Big\rangle$ is also non-degenerative, with $\psi(x) = \phi(x) - \alpha \sum_{x' \in \mathcal{X}} \mathcal{P}^*_{xx'} \phi(x')$.*

Proof. If $\mathcal{K}(x,x')$ is non-degenerate, the matrix Φ is full column rank, and then the kernel matrix $\mathcal{Y} = \Phi\Phi^T$ is positive definite. In this regard, as proved in Lemma 4.1, the matrix Ψ is also full column rank which directly implies that $\Gamma(x,x')$ is non-degenerate, being its kernel matrix given by $\Psi\Psi^T = B\Phi\Phi^T B^T$ positive definite. □

After computing the kernel function $\Gamma(x,x')$, the dual problem can be solved. By applying (9) with $\psi(x)$ and by using the representative states $x \in \tilde{\mathcal{X}}$, the cost function $\tilde{J}^*(x)$ can be defined as below

$$\begin{aligned}\tilde{J}^*(x) &= \Big\langle w, \phi(x)\Big\rangle \\ &= \sum_{x'\in\tilde{\mathcal{X}}}(\lambda(x)-\lambda^*(x'))\Big\langle\psi(x'),\phi(x)\Big\rangle \\ &= \sum_{x'\in\tilde{\mathcal{X}}}(\lambda(x')-\lambda^*(x'))\Big\langle\Big(\phi(x')-\alpha\sum_{x''\in\mathcal{X}}\mathcal{P}^*_{x'x''}\phi(x'')\Big),\phi(x)\Big\rangle \\ &= \sum_{x'\in\tilde{\mathcal{X}}}(\lambda(x')-\lambda^*(x'))\Big(\Big\langle\phi(x'),\phi(x)\Big\rangle-\alpha\mathcal{P}^*_{x'x''}\Big\langle\phi(x''),\phi(x)\Big\rangle\Big) \\ &= \sum_{x'\in\tilde{\mathcal{X}}}(\lambda(x')-\lambda^*(x'))\Big(\mathcal{K}(x,x')-\alpha\sum_{x''\in\mathcal{X}}\mathcal{P}^*_{x'x''}\mathcal{K}(x'',x)\Big). \end{aligned} \tag{43}$$

To make the recursive residual $\mathcal{RE}_l$ equal to zero, the regression function $\Big\langle w,\psi(x)\Big\rangle$ has to satisfy the following relation

$$\tilde{\mathcal{RE}}(x) = -\mathcal{R}(x) + \Big\langle w,\psi(x)\Big\rangle = 0. \tag{44}$$

If we fix the values c and ϵ, the constrained minimization problem (39), (40) can be expressed in a simpler form. As done before, if we set $\epsilon = 0$ and $c = \infty$, the derived SVR problem satisfies (44). In other words, we can write the constrained minimization problem (39), (40) as follows

$$\min_w \frac{1}{2}\|w\|^2, \tag{45}$$

subject to

$$\mathcal{R}(x) - \Big\langle w,\psi(x)\Big\rangle = 0, \forall x\in\tilde{\mathcal{X}}. \tag{46}$$

Using the Lagrange multiplier approach, we can write

$$\mathcal{L}(w,\lambda) = \frac{1}{2}\|w\|^2 + \sum_{x\in\tilde{\mathcal{X}}}\lambda(x)\Big(\mathcal{R}(x)-\Big\langle w,\psi(x)\Big\rangle\Big). \tag{47}$$

To minimize the above relation, we take the gradient of $\mathcal{L}(w,\lambda)$ with respect to w and set it to 0

$$\frac{\partial\mathcal{L}}{\partial w} = w - \sum_{x\in\tilde{\mathcal{X}}}\lambda(x)\psi(x) = 0, \tag{48}$$

hence

$$w = \sum_{x\in\tilde{\mathcal{X}}}\lambda(x)\psi(x). \tag{49}$$

Consequently, we can write the cost function $\tilde{J}^*(x)$ as

$$\begin{aligned}\tilde{J}^*(x) &= \Big\langle w,\phi(x)\Big\rangle = \sum_{x'\in\tilde{\mathcal{X}}}\lambda(x')\Big\langle\psi(x'),\phi(x)\Big\rangle \\ &= \sum_{x'\in\tilde{\mathcal{X}}}\lambda(x')\Big(\mathcal{K}(x',x)-\alpha\sum_{x''\in\mathcal{X}}\mathcal{P}^*_{x'x''}\mathcal{K}(x'',x)\Big), \quad \forall x\in\mathcal{X}. \end{aligned} \tag{50}$$

To reach the dual problem, we need to substitute (49) into (47) and maximize it with respect to λ, then we have

$$\max_\lambda \; -\frac{1}{2}\sum_{x,x'\in\tilde{\mathcal{X}}}\lambda(x)\lambda(x')\Big\langle\psi(x),\psi(x')\Big\rangle + \sum_{x\in\tilde{\mathcal{X}}}\lambda(x)\mathcal{R}(x). \tag{51}$$

In the vector form, we can write

$$\max_\lambda -\frac{1}{2}\lambda^T\mathcal{G}\lambda + \lambda^T\mathcal{R}, \tag{52}$$

where $\lambda \in \mathbb{R}^{\tilde{\Omega}}$ and $\mathcal{G} = \Psi\Psi^T$, with $\mathcal{G}_{x,x'} = \Big\langle\psi(x),\psi(x')\Big\rangle = \Gamma(x,x')$, is the kernel matrix corresponding to the kernel function Γ (here, it is only computed for the representative states). The analytical solution to the optimization problem (52) is given by

$$\mathcal{G}\lambda = \mathcal{R}. \tag{53}$$

Note that the solution of (53) exists and is unique since the kernel matrix $\mathcal{G}$ with components $\Gamma(\cdot,\cdot)$ is non-degenerate and positive definite, see Theorem 4.2. Moreover, if $\tilde{\Omega} \ll \Omega$, the system of equations in (53) has considerably lower size than the original problem.

Finally, the following interesting result is presented for the case of symmetric matrix $\mathcal{P}^*$, $\tilde{\mathcal{X}} = \mathcal{X}$, and the kernel matrix $\mathcal{Y}$ set to the identity matrix.

Theorem 4.3. *Suppose $\mathcal{P}^*$ be symmetric and $\tilde{\mathcal{X}} = \mathcal{X}$. Choose the kernel function to satisfy the following conditions: (i) $\mathcal{K}(x,x') = 0$ for $x \neq x'$; (ii) $\mathcal{K}(x,x') = 1$ for $x = x'$. Then, the constrained minimization problem (45), (46) provides the same solution of the Bellman's equation applied over the infinite time horizon.*

Proof. By using assumptions (i) and (ii), we can formulate the kernel function Γ from (42) as follows

$$\begin{aligned}\Gamma(x,x') &= \delta_{xx'} - \alpha\sum_{x''\in\mathcal{X}}\big(\mathcal{P}^*_{x'x''}\delta_{xx''} + \mathcal{P}^*_{xx''}\delta_{x'x''}\big) \\ &\quad +\alpha^2\sum_{x'',x'''\in\mathcal{X}}\mathcal{P}^*_{xx''}\mathcal{P}^*_{x'x'''}\delta_{x''x'''} \\ &= \delta_{xx'} - \alpha\big(\mathcal{P}^*_{x'x} + \mathcal{P}^*_{xx'}\big) + \alpha^2\sum_{x''\in\mathcal{X}}\mathcal{P}^*_{xx''}\mathcal{P}^*_{x'x''}, \end{aligned} \tag{54}$$

where $\delta_{xx'}$ is the Kronecker delta impulse. Thus, the associate kernel matrix can be expressed as $\mathcal{G} = \big(I-\alpha\mathcal{P}^*\big)^2$. By denoting with $\hat{\lambda}$ the solution of the unconstrained maximization problem (52), we have $\hat{\lambda} = \mathcal{G}^{-1}\mathcal{R} = \big(I-\alpha\mathcal{P}^*\big)^{-2}\mathcal{R}$. Hence, by using (50), we can calculate the cost function $\tilde{J}^*(x)$ as follows

$$\begin{aligned}\tilde{J}^*(x) &= \sum_{x'\in\mathcal{X}}\hat{\lambda}(x')\Big(\delta_{x'x} - \alpha\sum_{x''\in\mathcal{X}}\mathcal{P}^*_{x'x''}\delta_{x''x}\Big) \\ &= \hat{\lambda}(x) - \alpha\sum_{x'\in\mathcal{X}}\hat{\lambda}(x')\mathcal{P}^*_{x'x}. \end{aligned} \tag{55}$$

In the vector form, being $\mathcal{P}^*$ symmetric, we have

$$\begin{aligned}\tilde{J}^* &= \hat{\lambda} - \alpha\mathcal{P}^*\hat{\lambda} = \big(I-\alpha\mathcal{P}^*\big)\hat{\lambda} \\ &= \big(I-\alpha\mathcal{P}^*\big)\big(I-\alpha\mathcal{P}^*\big)^{-2}\mathcal{R} = \big(I-\alpha\mathcal{P}^*\big)^{-1}\mathcal{R} = J^*, \end{aligned} \tag{56}$$

which concludes the proof. □

5. Application of the RR-ADP to the sensor scheduling problem

In this section, the RR-ADP algorithm is applied to the transmission scheduling problem for the remote state estimation of multiple processes. A dedicated sensor is allocated to each process and sends its local measurements to a remote estimator through a wireless communication channel subject to packet loss. Since multiple sensors can transmit their measurements simultaneously, interference may occur among them. The remote estimator has multi-packet reception capability and can receive

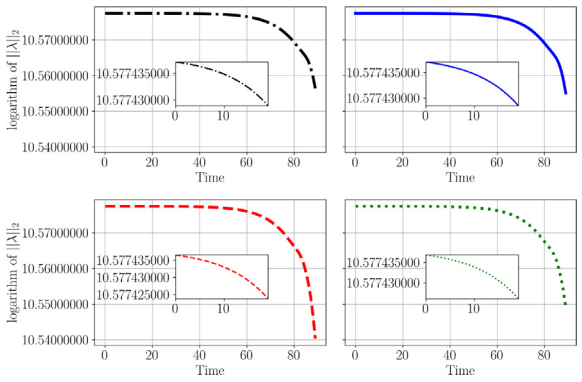


Fig. 1. Base-10 logarithm of the norm-2 for the vector λ_l in case of 4 different values of $\lambda_{\mathcal{N}}$ ($N = 4$ dynamical systems and $d = 5$ maximum packet drops).

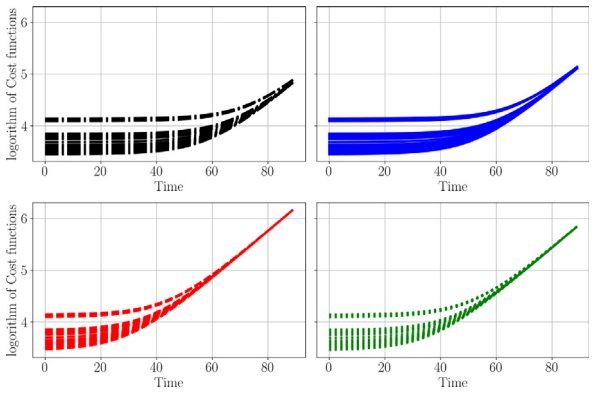


Fig. 2. Cost function of the representative states in case of 4 different values of $\lambda_{\mathcal{N}}$ ($N = 4$ dynamical systems and $d = 5$ maximum packet drops).

sensor estimates if their signal-to-interference-and-noise ratio (SINR) exceeds a prescribed lower bound. As a consequence, having a proper sensor scheduling policy becomes necessary. As shown in Forootani et al. (2022), such policy can be computed by minimizing the discounted error covariance cost function over a finite time horizon, which depends on the state estimation error covariance of each sensor. The whole transmission scheduling problem for Multi Sensors-Multi Processes (MSMP) can be modeled as an MDP (Forootani et al., 2022), which can require ADP-based techniques in case of large state space.

Let us consider a set of N processes, modeled as a set of discrete time Linear Time Invariant (LTI) dynamical systems $z_i(l+1) = A_i z_i(l) + \omega_i(l)$, where, for each $i = 1, \dots N$, $z_i(l) \in \mathbb{R}^{n_i}$ is the state vector of process i, $n_i \in \mathbb{N}$, and $\omega_i(l)$ is i.i.d. Gaussian process noise with zero mean and covariances $Q_i \in \mathbb{R}^{n_i \times n_i}$. We consider a scenario with N sensors, as the number of systems to monitor, in which the ith sensor has measurement $y_i(l) = C_i z_i(l) + v_i(l)$, where $y_i(l) \in \mathbb{R}^{m_i}$, $m_i \in \mathbb{N}$, and the measurement noise $v_i(l)$ is an i.i.d. Gaussian noise with zero mean and covariance R_i. It is supposed that $\omega_i(l)$ and $v_i(l)$ are mutually independent and do not depend on the initial state $z_i(0)$. Moreover, all the local sensors are supposed to be smart and able to run Kalman filters (Forootani et al., 2022). By denoting with $x_i \in \mathcal{X}_i$ the error covariance matrix associated with the process i at the remote estimator, the MDP state for the MSMP transmission scheduling problem can be defined as $x = (x_1, x_2, \dots, x_N)$. Moreover, it is $\mathcal{R}(x) = \sum_{i=1}^{N} Tr(x_i)$ (Forootani et al., 2022).[4] In this setting, the state space dimension is proportional to the number of successive packet drops. Thus, by denoting with d the maximum number of successive packet drops for each process, we have $\Omega = d^N$ (see Forootani et al. (2022) for more details).

For this application, the radial basis function of the form $\mathcal{K}(x, x') = exp^{-(x-x')^T \Sigma (x-x')}$ was chosen as kernel function, where Σ is a positive definite matrix (in particular, the matrix Σ was set equal to the identity matrix multiplied by a small positive coefficient). This function measures the distance between two states, and has its maximum when $x = x'$. It is worth highlighting that the kernel matrix $\mathcal{Y}$ with elements $\mathcal{K}(x, x')$ is non-degenerate and symmetric positive definite. In addition, Σ can be regarded as a weight matrix to trade off the value of the kernel function for two given states. The RR-ADP algorithm was implemented as a Python package for computing the parameter vector λ_l, and then determining the approximate optimal policy over the finite time horizon (see Forootani et al. (2019, 2022, 2020)). The MSMP described in Forootani et al. (2022) was considered. In particular, we considered the case of $N = 4$ and $d = 5$, therefore the cardinality of the spate space was $\Omega = 5^N = 625$. Moreover, we set the discount factor α to 0.9 and the time horizon $\mathcal{N}$ to 100. We chose 80 representative states, and thus the cardinality $\tilde{\Omega}$ of $\tilde{\mathcal{X}}$ was equal to 80 and the parameter vector $\lambda_l \in \mathbb{R}^{80}$. In Fig. 1, the logarithm of the norm-2 for the dual parameter vector λ_l over the finite time horizon is shown in case of 4 different values of the parameter $\lambda_{\mathcal{N}}$ (with each subplot of Fig. 1 corresponding to a specific $\lambda_{\mathcal{N}}$). Fig. 2 shows the backward evolution of the cost function of the representative states in logarithmic scale. For given any state $x \in \mathcal{X}$ and at any arbitrary time slot l, the cost function $\tilde{J}_l^*(x)$ can be computed by means of (28), being the vector λ_l computed for each time slot and being the kernel function $\mathcal{K}(x, x')$ chosen at the beginning of the algorithm. Finally, Fig. 3 displays the evolution of the norm-2 of the error vector $\tilde{J}_l^* - \tilde{J}_{l+1}^*$, which converges to zero as the backward procedure approaches the initial time step.

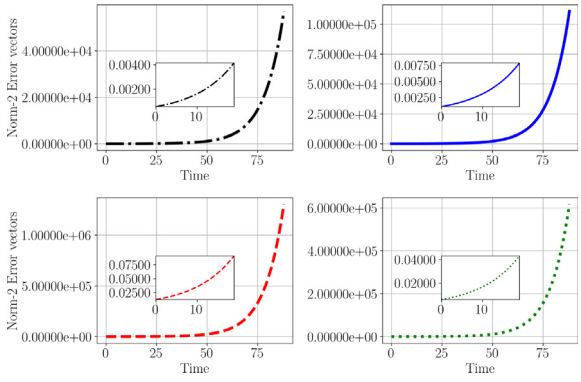


Fig. 3. Norm-2 of error vectors $\tilde{J}_l^* - \tilde{J}_{l+1}^*$ in case of 4 different values of $\lambda_{\mathcal{N}}$ ($N = 4$ dynamical systems and $d = 5$ maximum packet drops).

6. Conclusion

This paper proposed a kernel-based cost function approximation approach in conjunction with Support Vector Machine (SVM) regression in order to: (i) tackle the curse of dimensionality of stochastic Dynamic Programming (DP) problems (over both

[4] Note that in Forootani et al. (2022) the instantaneous cost function depends on both control action and state.

the finite and infinite time horizon); (ii) mitigate the difficulty of defining a proper set of features for the problem at hand. It was shown that kernel functions can provide a framework able to handle large-space Markov Decision Processes (MDPs) and compute approximate cost functions of the whole MDP state space. The properties of the resulting kernel-based Bellman's operator were also studied. Finally, the proposed Recursive Residual Approximate DP (RR-ADP) algorithm was applied to the transmission scheduling problem for a multi-process multi-sensor remote state estimator. As future work, it is planned to investigate the connection between the finite time stability and the convergence over the finite time horizon of kernel-based cost function approximations in case of stochastic continuous-time systems.

References

Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*(1), 47.

Bertsekas, D. P. (2011). Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control*, *56*(9), 2128–2139.

Bertsekas, D. P. (2017). *Dynamic programming and optimal control, volume I* (4th ed.). Belmont, Massachusetts: Athena Scientific.

Bertsekas, D. P. (2019a). Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. *IEEE/CAA Journal of Automatica Sinica*, *6*(1), 1–31.

Bertsekas, D. P. (2019b). *Reinforcement learning and optimal control*. Athena Scientific.

Bhat, N., Farias, V., & Moallemi, C. C. (2023). Non-parametric approximate dynamic programming via the kernel method. *Stochastic Systems*, *25*, 1–22.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Desai, V. V., Farias, V. F., & Moallemi, C. C. (2012). Approximate dynamic programming via a smoothed linear program. *Operations Research*, *60*(3), 655–674.

Dietterich, T., & Wang, X. (2001). Batch value function approximation via support vectors. In *Advances in neural information processing systems, vol. 14*.

Forootani, A., Iervolino, R., & Tipaldi, M. (2019). Applying unweighted least-squares based techniques to stochastic dynamic programming: Theory and application. *IET Control Theory & Applications*, *13*(15), 2387–2398.

Forootani, A., Iervolino, R., Tipaldi, M., & Dey, S. (2022). Transmission scheduling for multi-process multi-sensor remote estimation via approximate dynamic programming. *Automatica*, *136*, Article 110061.

Forootani, A., Iervolino, R., Tipaldi, M., & Neilson, J. (2020). Approximate dynamic programming for stochastic resource allocation problems. *IEEE/CAA Journal of Automatica Sinica*, *7*(4), 975–990.

Jung, T., & Polani, D. (2006). Least squares SVM for least squares TD learning. In *Proceedings of the 2006 conference on ECAI 2006: 17th European conference on artificial intelligence* (pp. 499–503).

Kandil, S. M., Farag, H. E. Z., Shaaban, M. F., & El-Sharafy, M. Z. (2018). A combined resource allocation framework for PEVs charging stations, renewable energy resources and distributed energy storage systems. *Energy*, *143*, 961–972.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, *13*(3), 637–649.

Ormoneit, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, *49*(2), 161–178.

Pouya, P., & Madni, A. M. (2021). Expandable-partially observable Markov decision-process framework for modeling and analysis of autonomous vehicle behavior. *IEEE Systems Journal*, *15*(3), 3714–3725.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality, vol. 703*. John Wiley & Sons.

Schölkopf, B., Smola, A. J., Bach, F., et al. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

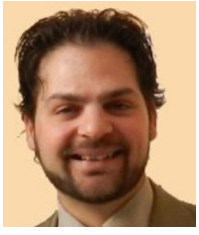
Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Xu, X., Lian, C., Zuo, L., & He, H. (2013). Kernel-based approximate dynamic programming for real-time online learning control: An experimental study. *IEEE Transactions on Control Systems Technology*, *22*(1), 146–156.

Ali Forootani received the M.Sc. degree in electrical engineering and automatic control system from Power and Water University of Technology, Iran, in 2011, and the Ph.D. degree in automatic control system and information technology from Department of Engineering, University of Sannio, Italy, in 2019. From 2011 to 2015 he worked both on research and industry at Niroo Research Institute (Tehran, Iran) and at the Ministry of Energy and Power (Water and Sewage Khuzestan Engineering Company). From 2019 to 2020 he served as Postdoctoral researcher at the Measurement and Instrumentation Laboratory University of Sannio, as well as University of Salerno on the topics of drone image signal processing and AI based network disease analysis. From 2020 to 2022 he was at the Hamilton Institute, Maynooth University, Ireland, as a postdoctoral researcher. Since 2022 he is with Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany, as a postdoctoral researcher. His current research interests include Markov decision processes, approximate dynamic programming, reinforcement learning in optimal control, learning in network control systems, Physics Informed Neural Network for Nonlinear System Identification and Partial Differential Equation parameter estimation. He is a Senior Member of IEEE Control System Society and his papers were considered as the selected publications on International Journal of Control and IEEE Transaction of Automatica Sinica.

Raffaele Iervolino received the laurea degree cum laude in aerospace engineering from the University of Naples, Italy, in 1996, where he also obtained the Ph.D. degree in electronic and computer science engineering in 2002. Since 2003 he is an Assistant Professor of automatic control at the University of Naples. From 2005 he is also a Adjoint Professor of automatic control with the Department of Electrical Engineering and Information Technology at the same University. He is Senior Member of IEEE Control System Society. His research interests include piecewise affine systems, opinion dynamics and consensus in social networks, and human telemetry systems. He is a Senior Member of IEEE Control System Society.

Massimo Tipaldi received the master's degree in computer science engineering and the Ph.D. degree in information technology from the University of Sannio, Benevento, Italy, in 1998 and 2017, respectively. In 2023, he achieved the National Scientific qualification as associate professor in the Italian higher education system for the disciplinary field of 09/G1 - Systems and control engineering. He possesses about 25 years of industrial experience in the managerial/technical coordination of European Space Agency (ESA), Agenzia Spaziale Italiana (ASI), and Centre National D'Etudes Spatiales (CNES) space projects (satellite systems, experimental equipment for the International Space Station, and ground segments) and in the writing of proposals, mainly related to space and research projects. He holds two patents and has coauthored more than 50 papers published in proceedings of international conferences or international archival journals. His research interests include reinforcement learning, approximate dynamic programming, multiagent systems, advanced system control techniques, safety-critical systems, and space systems engineering.



Silvio Baccari Born in 1975, he is an engineer with a Ph.D. in automatic control and currently works as a researcher at the University of Vanvitelli affiliated with the Department of Mathematics and Physics. He is specializing primarily in modeling and control of electronic power systems and numerical optimization. Originally from Benevento, he was a visiting student at Stanford University during his doctoral studies. A father of five, he is the founder of two innovative start-ups at the University of Sannio. He is a senior member of the IEEE and holds several patents. Apart from his academic passion, he has a keen interest in quantitative finance. Researcher with a contract co-funded by the European Union - PON Ricerca e Innovazione 2014–2020 ai sensi dell'art. 24, comma 3, lett. (a), della Legge 30 dicembre 2010, n. 240 e s.m.i. e del D.M. 10 agosto 2021 n. 1062. He is a Senior Member of IEEE Control System Society.