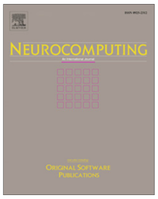
Neurocomputing 467 (2022) 476–490

Contents lists available at ScienceDirect

Neurocomputing

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



AKF-SR: Adaptive Kalman filtering-based successor representation ☆

Parvin Malekzadeh [a],*, Mohammad Salimibeni [b], Ming Hou [c], Arash Mohammadi [b], Konstantinos N. Plataniotis [a]

[a] *The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada*
[b] *Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC H3G 2W1, Canada*
[c] *Defence Research and Development Canada (DRDC) Toronto Research Centre, Toronto, ON M3K 2C9, Canada*

ARTICLE INFO

Article history:
Received 16 April 2021
Revised 6 July 2021
Accepted 2 October 2021
Available online 7 October 2021
Communicated by Zidong Wang

Keywords:
Reinforcement Learning
Successor representation
Kalman filter
Kalman temporal difference
Multiple model adaptive estimation
Radial basis function

ABSTRACT

To understand animals' behavior in finding relations between similar tasks and adapting themselves to changes in the tasks, it is necessary to know how the brain generalizes the learned knowledge from a previous task to unseen tasks. Recent studies in neuroscience suggest that Successor Representation (SR)-based models provide adaptation to changes in the goal locations or reward function faster than model-free algorithms, together with lower computational cost compared to that of model-based algorithms. However, it is not known how such representation might help animals to manage uncertainty in their decision making. Existing methods for the SR learning based on standard temporal difference methods (e.g., deep neural network-based algorithms) do not capture uncertainty about the estimated SR. In order to address this issue, the paper presents a Kalman filter-based SR framework, referred to as Adaptive Kalman Filtering-based Successor Representation (AKF–SR). First, Kalman temporal difference approach, which is a combination of Kalman filter and the temporal difference method, is used within the AKF–SR framework to cast the SR learning procedure into a filtering problem to benefit from uncertainty estimation of the SR, and also decreases in memory requirement and sensitivity to model's parameters in comparison to deep neural network-based algorithms. An adaptive Kalman filtering approach is then applied within the proposed AKF–SR framework in order to tune the measurement noise covariance and measurement mapping function of Kalman filter as the most important parameters affecting the filter's performance. Moreover, an active learning method that exploits the estimated uncertainty of the SR to form the behaviour policy leading to more visits to less certain values is proposed to improve the overall performance of an agent in terms of received rewards while interacting with its environment. Experimental results based on three reinforcement learning environments illustrate the efficacy of the proposed AKF–SR framework over state-of-the-art frameworks in terms of cumulative reward, reliability, time and computational cost, and speed of convergence to changes in the reward function.

1. Introduction

Human and animals have the learning capabilities for evaluating consequences of their actions; therefore, can adapt their behavior based on the received reward after each action [1–3]. Such an adaptive behavior can be achieved in the context of Artificial Intelligence (AI) via an optimal control policy, which can be obtained by Reinforcement Learning (RL) algorithms. Generally speaking, RL is a class of Machine Learning (ML) algorithms enabling an autonomous agent to learn a task to maximize expected future discounted rewards while interacting with its environment [4–6]. Unlike supervised learning, RL approaches do not need labeled data, which is generally difficult to acquire, for performing their learning processes. Traditionally, RL algorithms are categorized into two main classes: (i) Model-Free (MF) methods [4,7–9], which learn the value function using sample trajectories, and; (ii) Model-Based (MB) methods [10–12] that estimate transition and reward functions through search trees or dynamic programming. Algorithms belonging to the former category (MF algorithms) are particularly slow to adjust an agent to changes in its task (e.g., changes in the goal locations or reward function). The MB algorithms, on the other hand, can quickly adjust the agent to the changes. Such a rapid adaptation, however, comes with a high computational cost [13,14]. Reusing previous knowledge to facilitate learning of new tasks has been proposed as an appealing solu-

☆ This Project was partially supported by the Department of National Defence's Innovation for Defence Excellence and Security (IDEaS) program, Canada.

* Corresponding author.
E-mail address: p.malekzadeh@mail.utoronto.ca (P. Malekzadeh).

https://doi.org/10.1016/j.neucom.2021.10.008
0925-2312/Crown Copyright © 2021 Published by Elsevier B.V. All rights reserved.

tion to the mentioned adaptation problems of MB and MF algorithms [2]. To improve speed of an agent for learning a new task, which results from changes in its learned tasks, without considerable increase in computational cost, the agent should be able to identify the changes and reuse its knowledge from previous tasks. Reusing the transferred information learned from the previous tasks would not be feasible if the tasks are completely irrelevant. Therefore, in this paper, we focus on the changes in the reward function while the environment remains same.

Successor Representation (SR) methods [15,16] are proposed as a potential solution to aforementioned adaptation problems of both MF and MB categories of RL algorithms in the case of changes in the reward function. The SR-based algorithms are faster to adapt to changes than MF algorithm, and provide more efficient computation than MB algorithms. The SR-based algorithms learn the expected immediate reward received after each action, together with the expected discounted future state occupancy (referred to as the SR) [2,14,15]. The value function is then factorized into the SR and the immediate reward in each of successor states. This factorization enables rapid policy evaluation under changes in the reward conditions as only the reward function needs to be relearned in new tasks.

When the number of states are limited, the SR and the reward function (consequently, the value function) can be computed for each state. However, computation of the value function is not possible in RL problems with large number of states or when states are continuous; therefore, value function needs to be approximated. The function approximators in RL problems can be generally categorized into linear and non-linear function approximators [17,18]. In both cases, value of the approximate function is defined by a set of tunable parameters. Non-linear function approximators, such as artificial neural network [17,19–21] algorithms suffer from different issues including sensitivity of the model's performance to a large number of parameters, lack of theoretical convergence guarantees, and also the need for a large number of episodes to achieve acceptable results. The linear function approximators transform the approximation problem into the problem of computing weights to fuse several local estimators. Since linear function approximators are simpler and better understood than the non-linear ones; therefore, several convergence guarantees have been provided [15,22,23]. In this regard, Cerebellar Model Articulation Controllers (CMACs) [24] and Radial Basis Functions (RBFs) [25] are the commonly used linear estimators. It has been discovered that the RBFs can better represent gradual-continuous transitions in the functions to be approximated [26–29]. Due to this advantage, in this paper, we use RBFs for the SR and reward function linear approximation task.

Gershman et al. [30] have established that the temporal context model, a model of episodic memory, is actually direct estimation of the SR via Temporal Difference (TD) learning algorithms. Kulkarni et al. [17] proposed a deep learning method based on the combination of TD learning with Deep Neural Networks (DNNs) for estimation of the SR. Some other works [2,14,31,32] focused on the development of TD learning method by using different DNNs algorithms to represent the SR. CTDL framework proposed in [14] combines a DNN with a Self-Organising Map (SOM) to calculate action values to improve the performance and robustness of a DNN-based RL agent. Ma et al. [2] proposed a DNN framework, referred to as Universal Successor Representation (USR), to approximate the SR and incorporate it with actor-critic method to learn the SR. The methods proposed in [2,14,31,32], which are based on the combination of the standard TD learning and DNNs, however, do not consider uncertainty of the SR (consequently, uncertainty in the value function), which exists at the heart of RL problems in the real world. When there is uncertainty about the environment, the agent should not be overconfident of its knowledge and exploit it all the time, but instead explore other available actions, which might be better and reduce uncertainty. The optimal solution for the exploration/exploitation trade-off is computationally intractable, but it has been shown that uncertainty can cause exploration through two different ways: by adding randomness to the value function or directing actions toward uncertain ones [33]. The estimated uncertainty of the value function, therefore, is known as a useful information for quandary between exploration and exploitation dilemma [33,34]. Lehnert et al., [35] proposed a different method from the TD-based SR learning algorithms, where the transition probability of the RL problem is first learned; the SR is then approximated based on the learned transition model. Although the proposed framework is more sample efficient than the TD-based frameworks, the trajectory for finding the transition model is far more complex due to the difficulty in learning the accurate transition model, which makes the TD-based methods more favorable in the literature. Machado et al. [36] has recently proposed a framework, known as the Substochastic Successor Representation (SSR), which uses the norm of the SR as an exploration bonus. In the proposed scheme, first, the SSR is learned by minimizing the TD error through a DNN. The value function is then learned through a DNN structured similar to DQN framework [37], which adopts the estimated SSR for exploration/exploitation trade-off. Geist et al. [34] proposed Kalman Temporal Difference (KTD) framework for uncertainty estimation of the value function, but in the context of MF methods. The proposed KTD framework also benefits from less sensitivity of the framework's performance to model's parameters and reduced time and memory requirement for finding and learning of the best model in comparison to DNN-based frameworks [2,14,31,32,36]. Less sensitivity to parameter settings improves the reproducibility aspect of a reliable algorithm to regenerate more consistent performances across multiple learning runs; therefore, reduces the risk of generating unpredictable performances in different practical applications [38]. Geerts et al. [39] and Salimibeni et al. [40] applied KTD framework for the SR estimation in RL problems, respectively, with discrete state spaces and multiple agents. The proposed algorithms, however, do not use the uncertainty information of the estimated SR, which can be achieved from KTD algorithm, for the action selection process. Since value function is computed as a dot product between the SR and the reward function, there is a need for approximation of the reward function. However, Geerts et al. [39] do not discuss the reward function learning process. In this paper, as an initial step, we adopt the KTD framework for the SR learning process and propose a reward learning algorithm. Then, an innovative action selection scheme benefiting from the achieved uncertainty/belief of the estimated SR in order to deal with exploration/exploitation dilemma.

As stated previously, within the SR domain, for computation of the value function, both the SR and reward function need to be learned. So far, we proposed application of KTD within this context for the SR learning. To learn the reward function, on the other hand, we develop a Kalman Filter (KF) to estimate the RBFs' weights. The challenge here is selection of KF's parameters. Performance of KF-based algorithms are highly dependent on the filter's parameters. It has been shown that measurement noise covariances of a KF is one of the most important parameters of the filter and its improper selection can significantly degrade the filter's act and even cause divergence of the filter [41,42]. In the earlier studies, the underlying parameter assumed to be constant during the estimation process with its value being adapted manually by trial and error. Such a bruteforce approach, however, is significantly challenging due to variation of the parameter value. Lately, multiple studies have proposed to adjust the parameter value at each step via Adaptive Kalman Filter (AKF) approaches to enhance the overall accuracy of the filter [43]. Generally speaking, AKF approaches can be categorized into two main groups: (i) Innovation-based Adaptive Estimation (IAE) methods [44], and; (ii) Multiple Model Adaptive Estimation (MMAE) methods [45]. The former category uses a single Extended Kalman Filter (EKF) to adapt measurement noise covariances based on the innovation or the residual sequence. In the IAE methods, perfect knowledge of the system's model and an appropriate window size is required for the parameter computation. The MMAE methodologies, on the other hand, use a weighted sum of multiple KFs running in parallel for the parameter adaptation addressing the aforementioned issue. Consequently, the MMAE methods have attracted more attention owing to their capabilities to manage parametric uncertainty and their independence from imposing specific requirements on the system model. Capitalizing on the success of MMAE frameworks, in this paper, we develop an innovative MMAE-based modeling framework for learning of the reward function within the SR context. Different algorithms were proposed for fusion of the KFs in MMAE systems [46], among which the classical scheme is used in this paper due to its exponentially fast convergence speed to the best candidate model (filter). In the classical MMAE scheme, weight of each filter is achieved from a Bayesian approach.

Duration of the learning process and quality of the learned policies, which are obtained based on the proposed combination of KTD and MMAE techniques, highly depend on managing a trade-off between exploration and exploitation. Too much exploration prevents from maximizing the immediate rewards since the selected actions may result in a negative reward from the environment. On the other hand, exploiting uncertain knowledge offered by the environment reduces the expected future rewards because actions are selected given current information; therefore, may not be the optimal ones [47]. As the value function is modeled as a function of stochastic variables; therefore, there is a stochastic variable for each state-action pair. The dilemma between exploration and exploitation should profit from such uncertain information [34]. In this paper, we propose an active learning scheme, which uses the KTD framework to tackle the uncertainty computation, an important issue which has been overlooked in the literature.

In summary, this paper proposes a SR-based framework, referred to as Adaptive Kalman Filtering-based Successor Representations (AKF – SR), which can adapt quickly to changes in the reward function or goal locations faster than MF methods and with the lower computational cost compared to MB algorithms. The following 4 key contributions are made in this paper:

- RBFs estimators are incorporated within the AKF – SR framework to project continuous states into feature vectors such that the SR and the reward function can be modeled as linear functions of the feature vectors.
- Within the proposed AKF – SR framework, we adopt MMAE and gradient descent-based schemes for the reward function estimation (learning) via KF, which respectively compensate for the insufficient information about the measurement noise covariance and measurement mapping function of the KF as the most important parameters of a KF.
- The KTD framework is used within the AKF – SR framework, which casts the SR learning procedure into a filtering problem to estimate uncertainty of the learned SR. Moreover, by applying KTD, we benefit from decreases in memory and time spent for the SR learning and also sensitivity of the framework's performance to its parameters (i.e., more reliable) when compared with DNN-based algorithms.
- An active learning scheme is exploited to balance the amount of exploration and exploitation based on uncertainty information of the value function obtained from KTD algorithm of the SR learning. The proposed active learning mechanism effectively improves performance of the proposed AKF – SR framework in terms of cumulative reward as shown via three RL platforms.

The remainder of the paper is organized as follows: In Section 2, an overview of RL and SR is presented. The proposed AKF – SR framework is developed in Section 3. Section 4 presents experimental results obtained based on three benchmark RL platforms illustrating effectiveness of the proposed AKF – SR framework. Finally, Section 5 provides conclusion of the paper.

2. Problem Formulation

In this section, first, the required background to follow developments presented in the remainder of the paper will be presented. In what follows, we use following notation: Scalar variables are represented by Non-bold letter (e.g., X or x)); Vectors are represented by lowercase bold letter (e.g., $\boldsymbol{x}$); Matrices are denoted by capital bold letter (e.g., $\boldsymbol{X}$), and transpose of matrix $\boldsymbol{X}$ is shown by $\boldsymbol{X}^T$.

2.1. Reinforcement Learning (RL)

Generally speaking, the main objective of an autonomous agent in RL models is to learn an optimum control policy through interaction with the dynamic system (also considered as the agent's environment) [33]. More specifically, within a RL context, an autonomous agent selects actions from the action set $\mathscr{A}$ by following an optimal policy π^* in such a way that its cumulative reward will be maximized over time of the agent's interaction with its environment. At time step k, the autonomous agent takes an action $a_k \in \mathscr{A}$ given its current state $\boldsymbol{s}_k \in \mathscr{S}$ based on a given policy $\pi(.|\boldsymbol{s}_k)$, which maps the state $\boldsymbol{s}_k$ to a probability distribution over the action space $\mathscr{A}$. If the policy is deterministic, the agent will always map to one specified action in a given state. The environment then answers to the selected action by taking the agent to state $\boldsymbol{s}_{k+1} \in \mathscr{S}$ with the transition probability of $\Pr(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k, a_k)$ and returning a reward $R(\boldsymbol{s}_k, a_k)$ to the agent, where $R(.)$ is the reward function. Given initial state $\boldsymbol{s}_0 = \boldsymbol{s}$ and action $a_0 = a, R(\boldsymbol{s}_0, a_0) = R(\boldsymbol{s}, a)$ has no randomness; however, $R(\boldsymbol{s}_k, a_k)$ for $k \geqslant 1$ is a function of random variable $\boldsymbol{s}_k \sim \Pr(.|\boldsymbol{s}_{k-1}, a_{k-1})$ and possibly random variable $a_k \sim \pi(.|\boldsymbol{s}_k)$ (i.e., possibly stochastic policy). The expected value of the reward function $R(\boldsymbol{s}_k, a_k)$ for $(k \geqslant 1)$ is, therefore, calculated by taking the expectation with respect to the probabilities $\Pr(.|\boldsymbol{s}_{k-1}, a_{k-1})$ and $\pi(.|\boldsymbol{s}_k)$. The rewards are discounted using discount factor $\gamma \in (0, 1)$ to control the importance of the future rewards versus the immediate ones. The 5-tuple $\{\mathscr{S}, \mathscr{A}, \mathscr{P}_a, R, \gamma\}$ defines a Markov Decision Process (MDP), which provides a rigorous mathematical framework for modeling the RL tasks. For a fixed policy π, the return $\sum_{k=0}^{\infty}\gamma^k R(\boldsymbol{s}_k, a_k)$, represents the sum of discounted rewards observed along one trajectory of states while following π. The state-action value function $Q^{\pi}(\boldsymbol{s}, a)$ for a policy π estimates expected value of the return $\sum_{k=0}^{\infty}\gamma^k R(\boldsymbol{s}_k, a_k)$ over all infinite length trajectories that start at state $\boldsymbol{s}_0 = \boldsymbol{s}$ with action $a_0 = a$, then acting according to π:

$$Q^{\pi}(\boldsymbol{s}, a) = \mathbb{E}\left\{\sum_{k=0}^{\infty}\gamma^k R(\boldsymbol{s}_k, a_k)\right\} \tag{1}$$
$$\boldsymbol{s}_k \sim \Pr(.|\boldsymbol{s}_{k-1}, a_{k-1}), a_k \sim \pi(.|\boldsymbol{s}_k), a_0 = a, \boldsymbol{s}_0 = \boldsymbol{s}.$$

Using the linearity of the expectation, the expectation function $\mathbb{E}$ in Eq. (1) is, therefore, calculated over $\Pr(.|\boldsymbol{s}_{k-1}, a_{k-1})$ and possibly stochastic policy $\pi(.|\boldsymbol{s}_k)$ for all $k \geqslant 1$, i.e.,

$$\begin{aligned} Q^{\pi}(\boldsymbol{s},a) = & \mathbb{E}\left\{\sum_{k=0}^{\infty}\gamma^{k}R(\boldsymbol{s}_k,a_k)\right\} = \sum_{k=0}^{\infty}\gamma^{k}\mathbb{E}\{R(\boldsymbol{s}_k,a_k)\} \\ = & R(\boldsymbol{s},a) + \sum_{k=1}^{\infty}\gamma^{k}\mathbb{E}\{R(\boldsymbol{s}_k,a_k)\}, \end{aligned} \tag{2}$$

where $\boldsymbol{s}_k \sim \Pr(.|\boldsymbol{s}_{k-1},a_{k-1}), a_k \sim \pi(.|\boldsymbol{s}_k), a_0 = a$, and $\boldsymbol{s}_0 = \boldsymbol{s}$. The state-action value function ($Q^{\pi}(\boldsymbol{s},a)$) can be gradually update based on the Bellman update idea [48]. The iterative update of the state-action value function forms a learning algorithm known as Temporal Difference (TD) learning [33,49], i.e.,

$$Q^{\pi}_{\text{new}}(\boldsymbol{s}_k,a_k)=Q^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k)+\alpha\big(R(\boldsymbol{s}_k,a_k)+\gamma Q^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1},a_{k+1})-Q^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k)\big),$$

where $Q^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k)$ and $Q^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1},a_{k+1})$ represent the latest estimates of the state-action value function at state-action pairs $(\boldsymbol{s}_k,a_k)$ and $(\boldsymbol{s}_{k+1},a_{k+1})$, respectively. $Q^{\pi}_{\text{new}}(\boldsymbol{s}_k,a_k)$ is an updated representation given an observed transition $(\boldsymbol{s}_k,a_k,\boldsymbol{s}_{k+1},R(\boldsymbol{s}_k,a_k))$ and $(R(\boldsymbol{s}_k,a_k)+\gamma Q^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1},a_{k+1})-Q^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k))$ is known as the TD error that intuitively represents the difference between the predicted reward according to the current estimate of the state-action value function and the actual observed reward at time step k. The parameter $(0 < \alpha \leqslant 1)$ is the learning rate determining how much of the error should we accept to adjust our estimates towards. During the learning process of state-action value function, actions are selected using the current policy π. After convergence of the learning process (i.e., the TD error converges to zero) where optimal policy π^{*} is achieved, actions are obtained based on the optimal policy π^{*} as follows

$$a_k = \arg\max_{a\in\mathscr{A}} Q^{\pi^{*}}(\boldsymbol{s}_k,a). \tag{4}$$

This completes a brief introduction to basics of RL approaches. Next, we present the SR modeling framework.

2.2. The Successor Representation (SR)

Dayan [15] claimed that distances between states can express similarities of future paths of an agent given the policy π. Successor Representation (also referred to as SR) [15] estimates the cumulative time expected to be spent in future state $\boldsymbol{s}'$ given the initial state $\boldsymbol{s}$ and initial action a by following policy π:

$$\boldsymbol{M}^{\pi}(\boldsymbol{s},\boldsymbol{s}',a) = \mathbb{E}\left\{\sum_{k=0}^{\infty}\gamma^{k}\mathbb{1}[\boldsymbol{s}_{\mathbb{k}}=\boldsymbol{s}']|\boldsymbol{s}_{\mathbb{0}}=\boldsymbol{s},\mathbb{a}_{\mathbb{0}}=\mathbb{a}\right\}, \tag{5}$$

In a discrete state-space scenario, the SR $(\boldsymbol{M}^{\pi}(:,:,a))$ is a $(|\mathscr{S}|\times|\mathscr{S}|)$ matrix, where $|\mathscr{S}|$ is cardinality of $\mathscr{S}$. Gershman et al. [30] showed that similar to the TD learning of state-action value function in Eq. (3), the SR can be also updated in an on-policy recursive form [30], known as SARSA, as follows

$$\begin{aligned} \boldsymbol{M}^{\pi}_{\text{new}}(\boldsymbol{s}_k,\boldsymbol{s}',a_k)=&\boldsymbol{M}^{\pi}_{\text{old}}(\boldsymbol{s}_k,\boldsymbol{s}',a_k) \\ &+\alpha\big(\mathbb{1}[\boldsymbol{s}_{\mathbb{k}}=\boldsymbol{s}']+\gamma\boldsymbol{M}^{\pi}_{\text{old}}(\boldsymbol{s}_{\mathbb{k}+\mathbb{1}},\boldsymbol{s}',\mathbb{a}_{\mathbb{k}+\mathbb{1}})-\boldsymbol{M}^{\pi}_{\text{old}}(\boldsymbol{s}_{\mathbb{k}},\boldsymbol{s}',\mathbb{a}_{\mathbb{k}})\big). \end{aligned} \tag{6}$$

where $\boldsymbol{M}^{\pi}_{\text{old}}(\boldsymbol{s}_k,\boldsymbol{s}',a_k)$ and $\boldsymbol{M}^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1},\boldsymbol{s}',a_{k+1})$ represent the latest estimates of the expected number of times that the agent visit state $\boldsymbol{s}'$ given initial state-action pair $(\boldsymbol{s}_k,a_k)$ and $(\boldsymbol{s}_{k+1},a_{k+1})$, respectively. The TD error here represents the error in state visitation count of state $\boldsymbol{s}'$ starting at initial state-action pair $(\boldsymbol{s}_k,a_k)$. Given the SR, the state-action value function, Eq. (1), can be expressed as the inner product of the SR and the estimated reward function [15], i.e.,

$$Q^{\pi}(\boldsymbol{s}_k,a_k) = \sum_{\boldsymbol{s}'\in\mathscr{S}}\sum_{a'\in\mathscr{A}}\boldsymbol{M}^{\pi}(\boldsymbol{s}_k,\boldsymbol{s}',a_k)R(\boldsymbol{s}',a'). \tag{7}$$

Eq. (7) demonstrates the most important feature of SR-based frameworks, as it represents a linear mapping that allows the state-action value function $Q^{\pi}(\boldsymbol{s}_k,a_k)$ to be reconstructed straightforwardly based on reevaluation of the reward function $(R(\boldsymbol{s}',a'))$ in case of changes in the reward function. Next, we focus on extending SR model (Eqs. (5)–(7)))) to scenarios with continuous state spaces.

2.3. Feature-based successor representations

Estimation of the SR and the reward function are impractical for each state in RL problems with large number of states or continuous state spaces; hence, as such one needs to resort to their approximated forms. In such cases, each pair $(\boldsymbol{s},a)$, is mapped into a L-dimensional state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s},a)$ $(\boldsymbol{\psi}:\mathscr{A}\times\mathscr{S}\to\mathbb{R}^{L})$. In this setting, the successor representation is generalized to a feature-based SR vector [50], which encodes the expected feature values as follows

$$\boldsymbol{m}^{\pi}(\boldsymbol{s},a) = \mathbb{E}\left\{\sum_{k=0}^{\infty}\gamma^{k}\boldsymbol{\psi}(\boldsymbol{s}_k,a_k)|\boldsymbol{s}_0=\boldsymbol{s},a_0=a\right\}, \tag{8}$$

For approximation of the SR and reward function, linear function approximators are reasonable choices due to their simpler implementations and faster computation speed compared to the non-linear ones. The convergence of linear function approximators have been already guaranteed in RL problems [15,22,23]. A linear function of the feature vectors is, therefore, used to approximately factorize the immediate reward function for the pair $(\boldsymbol{s},a)$ as follows

$$R(\boldsymbol{s}_k,a_k) \approx \boldsymbol{\psi}(\boldsymbol{s}_k,a_k)^{T}\boldsymbol{\theta}_k, \tag{9}$$

where $\boldsymbol{\theta}_k$ is the reward's weight vector. The state-action value function (Eq. (7)), therefore, can be computed as

$$Q^{\pi}(\boldsymbol{s}_k,a_k) \approx \boldsymbol{\theta}_k^{T}\boldsymbol{m}^{\pi}(\boldsymbol{s}_k,a_k). \tag{10}$$

We assume that the SR vector $\boldsymbol{m}^{\pi}(\boldsymbol{s}_k,a_k)$ in Eq. (8) can be also approximated as a linear function of the same feature vector:

$$\boldsymbol{m}^{\pi}(\boldsymbol{s}_k,a_k) \approx \boldsymbol{W}_k\boldsymbol{\psi}(\boldsymbol{s}_k,a_k), \tag{11}$$

where $\boldsymbol{W}_k$ is a $(L\times L)$ matrix embodied by the weights $W_{i,j}$. The TD learning of the SR in Eq. (6) is update as

$$\begin{aligned} \boldsymbol{m}^{\pi}_{\text{new}}(\boldsymbol{s}_k,a_k) = & \boldsymbol{m}^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k) \\ & + \alpha\big(\boldsymbol{\psi}(\boldsymbol{s}_k,a_k)+\gamma\boldsymbol{m}^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1},a_{k+1})-\boldsymbol{m}^{\pi}_{\text{old}}(\boldsymbol{s}_k,a_k)\big). \end{aligned} \tag{12}$$

Our discussion on background of the SR is complete. Next, the proposed AKF – SR framework is presented. In the remainder of this paper, for simplicity, we assume that the policy and transition function of the system are deterministic.

3. AK-SR: Adaptive Kalman filtering-based successor representations

Once the approximation structure of the SR and the reward function have been defined, a suitable algorithm needs to be designed to learn (estimate) the reward function $R(\boldsymbol{s},a)$ and the SR vector $\boldsymbol{m}^{\pi}(\boldsymbol{s},a)$. For these estimations, sample transition of the system, the feature vectors, and the received reward from the environment are used as the measurements. DNN-based RL methods [2,17,36–38], require to store all these measurements together with the network's parameters and activations to be learned in batches. In the learning process, one needs to store the activations from a forward propagation to be utilized later for computation of the error gradients in a back propagation process. Therefore, considerably high memory is required for implementation of deep learning-based techniques. For instance, there are 26 million parameters in a 50-layer ResNet network resulting in the need to compute 16 million activations during the forward pass. Approximately, the memory required for training of a ResNet-50 network with a mini-batch of 32, is over 7.5 GB of local DRAM. Furthermore, similar to other standard TD learning-based algorithm, most of the DNN-based frameworks [14,17,37] do not consider uncertainty within the value approximation context. The difficulty between exploration and exploitation should use from such uncertainty information. Finally, reliability of a learning algorithm is another important factor which needs to be verified for applications to real word scenarios. For a learning process to be reliable in different applications, it should be capable of regenerating consistent performances over multiple runs with the least frangibility to the model's parameters (reproducibility aspect of reliability) [38]. However, performance of a DNN model, is highly affected by its large number of parameters required to be tuned. Tuning of such a large number of parameters, therefore, leads to high sensitivity and considerable time and effort needed to tune the parameters, which make DNN-based RL algorithms unreliable for practical applications.

By contrast, filtering algorithms [51,52], which are efficient techniques to process sequential data, can be implemented based only on the last measurement. Such approaches eliminate the necessity of the learning process to record the complete measurement history, which, in turn, translates into significant reduction in time and memory requirements in comparison to DNN-based techniques. It also has been shown that applied filtering-based algorithms (such as KTD [34]) estimate uncertainty of the value function and consider that for action selection during the learning process in order to make a balance between exploring unknowns and exploiting the agent's knowledge. Furthermore, despite DNNs, a small number of parameters is required to be tuned in filtering-based methods resulting in less time and effort required for the parameters tuning and also less vulnerability of its performance to the model's parameters in comparison to its DNN-based counterpart. The performance of filtering-based algorithms, however, are highly related to the filter's parameters and complete information about theses parameters is not accessible. In order to tackle this problem and tune the filter's parameters, earlier studies proposed adaptive multiple model filters [53,54] to adapt the filter's parameters.

The proposed AKF – SR framework is, therefore, based on development of KTD framework to the SR learning and adaptable KF in order to provide a powerful efficient means for estimation the reward function. The AKF – SR consists of the following four main modules as follows:

1. *RBF-based Feature Vector Construction*, which projects a pair $(\boldsymbol{s}, a)$ into a feature vector consisting of RBFs in order to generalize the SR and the reward function (consequently, value function) to continuous state spaces such that the SR and the reward function can be modeled as linear functions of the feature vectors.
2. *Reward Learning*, which estimates the reward weight vector $\boldsymbol{\theta}$ via a KF. To tune the measurement noise covariance of the KF a multiple model adaptive estimation method is utilized within KF formulation for reward learning. Furthermore, restricted gradient descent is adopted which regularizes the measurement mapping function of the KF by updating the means and covariances of the underlying RBFs.
3. *KTD-based SR Learning*, which estimates the SR weight matrix $\boldsymbol{W}$ using KTD algorithm in order to estimate uncertainty of the SR (consequently, the value function).
4. *An Active Learning Process*, which uses the value function's uncertainty achieved from utilized KTD in the SR learning process to select the action that reduces the system's uncertainty more than any other potential actions.

Fig. 1 provides a block diagram of the proposed AKF – SR framework. The aforementioned four components of the proposed AKF – SR framework are detailed in the following sub-sections.

3.1. RBF-based Feature Vector Construction

As mentioned earlier, in RL problems with large/continuous state spaces, we need to represent each pair $(\boldsymbol{s}, a)$ with a state-action feature vector in order to approximate value function for all of the states in $\mathscr{S}$. In this study, state $\boldsymbol{s}$ is mapped into a N_{RBF}-dimensional state feature vector $\boldsymbol{\phi}(\boldsymbol{s})$ consisting of basis functions $\phi_n(\boldsymbol{s})$ as follows

$$\boldsymbol{\phi}(\boldsymbol{s}_k) = \left[\phi_1(\boldsymbol{s}_k), \phi_2(\boldsymbol{s}_k), \ldots, \phi_{N-1}(\boldsymbol{s}_k), \phi_{N_{\text{RBF}}}(\boldsymbol{s}_k)\right]^T, \tag{13}$$

where each element of vector $\boldsymbol{\phi}(\boldsymbol{s}_k)$ in Eq. (13) is selected as a radial basis function given by

$$\phi_n(\boldsymbol{s}_k) = e^{\frac{-1}{2}\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)^T \left(\boldsymbol{\Sigma}_k^n\right)^{-1} \left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)}, \tag{14}$$

where the mean and covariance of the RBFs at step k are represented by $\boldsymbol{\mu}_k^n$ and $\boldsymbol{\Sigma}_k^n$. The state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k = a_d)$ for $(1 \leqslant d \leqslant N_{\text{actions}})$, where N_{actions} is the total number of actions, is then generated from $\boldsymbol{\phi}(\boldsymbol{s}_k)$ by assigning this state feature vector to the corresponding spot for action $(a_k = a_d)$ while the feature vector values for the remainder of the actions are set to zero, i.e.,

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k = a_d) = \left[0, \ldots 0, \phi_1(\boldsymbol{s}_k), \ldots, \phi_{N_{\text{RBF}}}(\boldsymbol{s}_k), 0, \ldots 0\right]^T. \tag{15}$$

For each pair $(\boldsymbol{s}, a)$, the generated state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}, a)$ is, therefore, a vector of size $L = N_{\text{RBF}} \times N_{\text{actions}}$.

After construction of the state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ through Eq. (15), the reward weight vector $\boldsymbol{\theta}_k$ in Eq. (9) and the SR weight matrix $\boldsymbol{W}_k$ in Eq. (11) need to be learned. Sub-Sections 3.2 and 3.3 detail the learning process of the reward function and the SR, respectively.

3.2. Reward Learning

As mentioned in Sub-Section 2.3, the reward function can be approximated as a linear function of the state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ with the weigh vector of $\boldsymbol{\theta}_k$. In the proposed AKF – SR framework, the reward's weight vector $\boldsymbol{\theta}_k$ considered as a random variable, which can be estimated via KF with the following measurement model

$$R(\boldsymbol{s}_k, a_k) = \underbrace{\boldsymbol{\psi}^T(\boldsymbol{s}_k, a_k)}_{\boldsymbol{h}_k} \boldsymbol{\theta}_k + X_k, \tag{16}$$

where $\boldsymbol{h}_k$ is the measurement mapping function, and X_k is a zero-mean Gaussian noise with unknown variance of Ω_k. The evolution of the reward function is supposed to be determined by a so-called evolution equation, which connects the current value $\boldsymbol{\theta}_k$ with the previous one $\boldsymbol{\theta}_{k-1}$. However, true dynamic of the reward function cannot anyway be obtained in a general case. Following earlier works[4,15,39], where the dynamic is assumed to be a random diffusion process, such that the passage of time increases uncertainty without changing the mean belief/uncertainty about the estimated $\boldsymbol{\theta}$, in this paper, a heuristic evolution model following Occam razor principle is adopted as follows

$$\boldsymbol{\theta}_k = \boldsymbol{F}_k \boldsymbol{\theta}_{k-1} + \boldsymbol{b}_k, \tag{17}$$

where $\boldsymbol{F}_k = 0.9\boldsymbol{I}_L$ in order to make the filter stable, and $\boldsymbol{b}_k$ is supposed to be a white Gaussian noise with zero mean and covariance of $\boldsymbol{B}_k$. Given the assumptions of the linear evolution model and additional noise with Gaussian distribution, a KF is guaranteed to provide the optimal solution. For solving this KF problem, after

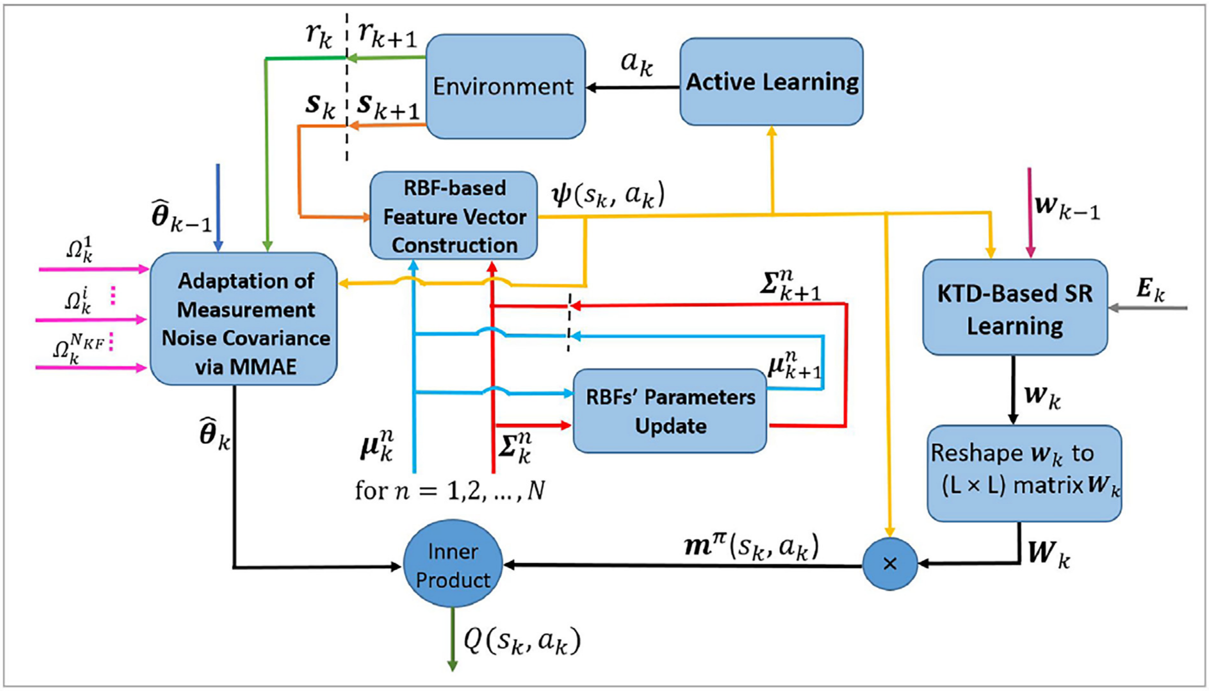


Fig. 1. Block diagram of the proposed AKF – SR framework.

the initialization step, the weights and their associated covariance matrices are computed as follows

$$\hat{\theta}_{k|k-1} = \boldsymbol{F}_k \hat{\theta}_{k-1}, \tag{18}$$

$$\text{and } \boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1} \boldsymbol{F}_k^T + \boldsymbol{B}_k. \tag{19}$$

Then, the received reward $r_k = R(\boldsymbol{s}_k, a_k)$ from the environment is utilized to update the estimates as follows

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{h}_k^T \left(\boldsymbol{h}_k \boldsymbol{P}_{k|k-1} \boldsymbol{h}_k^T + \Omega_k \right)^{-1}, \tag{20}$$

$$\hat{\theta}_k = \hat{\theta}_{k|k-1} + \boldsymbol{K}_k \left(r_k - \boldsymbol{h}_k \hat{\theta}_{k|k-1} \right), \tag{21}$$

$$\text{and } \boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{h}_k) \boldsymbol{P}_{k|k-1}. \tag{22}$$

Note that the performance of KF is highly affected by the filter's parameters and improper selection of the parameters can cause the instability of the process for applications in practical RL problems. Complete information about these parameters is, typically, not available in practical scenarios. In most of previous studies, it has been supposed that these parameters were fixed during the estimation and were manually adapted by trial and error. Due to changes of noise levels in different contexts, however, it can be difficult to utilize such a method to set up the correct value of the parameters. Within a KF framework, the measurement noise covariance matrix and the measurement mapping function are the most critical parameters as they control flow of new information. Improper choices of theses parameters could degrade performance of the KF, even cause divergence of the system, and are, therefore, considered for adaptation in this work. In the following, we represent adaptation process of measurement noise covariance Ω. Adaptation of the mapping function $\boldsymbol{h}$ is discussed in Sub-Section 3.2.2.

3.2.1. Adaptation of Measurement Noise Covariance via MMAE

For adaptation of the measurement noise covariance, MMAE scheme is used via implementation of a set of KFs, where different values for Ω is considered in each mode-matched filter. Eqs. (20)–(22) are then modified as follows

$$\boldsymbol{K}_k^i = \boldsymbol{P}_{k|k-1} \boldsymbol{h}_k^T \left(\boldsymbol{h}_k \boldsymbol{P}_{k|k-1} \boldsymbol{h}_k^T + \Omega_k^i \right)^{-1} \tag{23}$$

$$\hat{\theta}_k^i = \hat{\theta}_{k|k-1} + \boldsymbol{K}_k^i \left(r_k - \boldsymbol{h}_k \hat{\theta}_{k|k-1} \right) \tag{24}$$

$$\boldsymbol{P}_k^i = \left(\boldsymbol{I} - \boldsymbol{K}_k^i \boldsymbol{h}_k \right) \boldsymbol{P}_{k|k-1}, \tag{25}$$

where superscript i for $(1 \leqslant i \leqslant N_{\text{KF}})$, shows the i^{th} filter within the bank of N_{KF} Kalman filters. The i^{th} mode-matched filter, denoted by m^i, exploits Ω_k^i as its measurement noise covariance. The output posteriori of localized matched filters are then averaged based on their associated normalized weights ω_k^i as follows

$$\Pr(\boldsymbol{\theta}_k | \boldsymbol{Y}_k) = \sum_{i=1}^{N_{\text{KF}}} \omega_k^i \Pr\left(\boldsymbol{\theta}_k | \boldsymbol{Y}_k, \Omega_k^i \right), \tag{26}$$

where $\boldsymbol{Y}_k$ shows the reward sequence $\{r_1, r_2, \ldots, r_k\}$. The weight of mode m^i is calculated recursively using the Bayesian rule as

$$\omega_k^i \triangleq \Pr(m_k^i | \boldsymbol{Y}_k) = \frac{P(r_k | \boldsymbol{Y}_{k-1}, m_k^i) P(m_k^i | \boldsymbol{Y}_{k-1})}{\sum_{j=1}^{N_{\text{KF}}} P\left(r_k | \boldsymbol{Y}_{k-1}, m_k^j \right) P\left(m_k^j | \boldsymbol{Y}_{k-1} \right)}, \tag{27}$$

where the denominator is just a normalizing factor to make sure that $P(m_k^i | \boldsymbol{Y}_k)$ is an appropriate probability density function (PDF). Term $\mathscr{L}_k^i \triangleq \Pr(r_k | \boldsymbol{Y}_{k-1}, m_k^i)$ in the nominator is the likelihood function of mode i, which is calculated as a PDF of Kalman filter measurement residual ($\epsilon_k = r_k - \boldsymbol{h}_k \hat{\theta}_{k|k-1}$) as follows

$$\begin{aligned} \mathscr{L}_k^i &= \Pr\left(r_k | \hat{\boldsymbol{\theta}}_{k|k-1}, \Omega_k^i \right) \\ &= \frac{1}{\sqrt{\det\left[2\pi Z_k^i \right]}} . e^{\frac{-1}{2} \epsilon_k^T \left(Z_k^i \right)^{-1} \epsilon_k}, \quad \text{where} \quad Z_k^i \\ &= \boldsymbol{h}_k \boldsymbol{P}_{\theta,k|k-1} \boldsymbol{h}_k^T + \Omega_k^i. \end{aligned} \tag{28}$$

Eq. (27) for computing ω_k^i is, therefore, reduced to

$$\omega_k^i = \frac{\omega_{k-1}^i \mathscr{L}_k^i}{\sum_{j=1}^{M} \omega_{k-1}^j \mathscr{L}_k^j}. \tag{29}$$

The initial value of the weights are set to $\omega_0^i = 1/N_{\mathrm{KF}}$ for $i = 1, 2, \ldots, M$. The posteriori estimate $\hat{\boldsymbol{\theta}}_k$ and its error covariance are then obtained as

$$\hat{\boldsymbol{\theta}}_k = \sum_{i=1}^{N_{\mathrm{KF}}} \omega_k^i \hat{\boldsymbol{\theta}}_k^i, \tag{30}$$

$$\boldsymbol{P}_k = \sum_{i=1}^{N_{\mathrm{KF}}} \omega_k^i \left(\boldsymbol{P}_{\theta,k}^i + \left(\hat{\boldsymbol{\theta}}_k^i - \hat{\boldsymbol{\theta}}_k \right) \left(\hat{\boldsymbol{\theta}}_k^i - \hat{\boldsymbol{\theta}}_k \right)^T \right). \tag{31}$$

The MMAE process is shown in Fig. 2. This completes presentation of the proposed MMAE-based algorithm for reward function learning. Next, we focus on adaptation of the measurement mapping function of the KF used for reward function learning.

3.2.2. Adaptation of Measurement Mapping Function via RBFs' Parameters Update

As already mentioned, the measurement mapping function needs to be properly set up in KF-based estimations as one of the most important parameters in a Kalman filter. Such a priori knowledge, however, is usually not available, and; consequently it has to be adapted to its correct value. The measurement mapping function employed for the reward weight learning (i.e., $\boldsymbol{h}$) is modeled by basis functions. Its adaptations, therefore, necessitates recursive update of the basis functions, which is developed below. As the measurement mapping function depends on a large number of parameters (i.e., $2 \times N$), a gradient descent adaptation approach is used instead of the MMAE model to adapt these parameters. In this regard, we use the Restricted Gradient Descent (RGD) method [27], where partial derivations are exploited to compute the gradient of a defined loss function (L_k) in terms of the underlying parameters of the basis functions. We considered the proposed generalized robust lost function in [55] due to its robustness to outliers. Based on Eq. (16), probability distribution of $(r_k - \boldsymbol{\psi}^T(\boldsymbol{s}_k, a_k)\boldsymbol{\theta}_k = X_k)$ is a Gaussian distribution. The value of shape parameter of the proposed loss [55] is, therefore, considered as 2, and the loss for RBFs' parameters update will approach to L2 loss function in the limit:

$$L_k = \left(r_k - \boldsymbol{\psi}^T(\boldsymbol{s}_k, a_k)\boldsymbol{\theta}_k \right)^2 \tag{32}$$

Here, the focus is on minimization of the loss function (L_k) defined in Eq. (32). To achieve this goal, the required gradients with respect to the mean ($\boldsymbol{\mu}$) and covariance ($\boldsymbol{\Sigma}$) of RBFs are computed using the chain rule as follows

$$\Delta\boldsymbol{\mu} = \frac{\partial L_k}{\partial \boldsymbol{\mu}} = \frac{\partial L_k}{\partial \boldsymbol{\psi}} \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\mu}}, \tag{33}$$

$$\text{and } \Delta\boldsymbol{\Sigma} = \frac{\partial L_k}{\partial \boldsymbol{\Sigma}} = \frac{\partial L_k}{\partial \boldsymbol{\psi}} \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\Sigma}}, \tag{34}$$

where the partial derivations are calculated as

$$\frac{\partial L_k}{\partial \boldsymbol{\psi}} = -2\,\boldsymbol{\theta}_k^T (L_k)^{\frac{1}{2}}, \tag{35}$$

$$\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\mu}} = \boldsymbol{\psi}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu}), \tag{36}$$

$$\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\psi}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}_k - \boldsymbol{\mu})(\boldsymbol{s}_k - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}. \tag{37}$$

The partial derivatives are then used to update the means and covariances of RBFs as

$$\boldsymbol{\mu}_{k+1}^n = \boldsymbol{\mu}_k^n - \lambda_{\boldsymbol{\mu}}\Delta\boldsymbol{\mu} = \boldsymbol{\mu}_k^n + 2\lambda_{\boldsymbol{\mu}}(L_k)^{\frac{1}{2}}\boldsymbol{\theta}_k^T\boldsymbol{\psi}\left(\boldsymbol{\Sigma}_k^n\right)^{-1}\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right), \tag{38}$$

$$\boldsymbol{\Sigma}_{k+1}^n = \boldsymbol{\Sigma}_k^n - \lambda_{\Sigma}\Delta\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_k^n + 2\lambda_{\Sigma}(L_k)^{\frac{1}{2}}\boldsymbol{\theta}_k^T\boldsymbol{\psi}\left(\boldsymbol{\Sigma}_k^n\right)^{-1}\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)^T\left(\boldsymbol{\Sigma}_k^n\right)^{-1}, \tag{39}$$

where $\lambda_{\boldsymbol{\mu}}$ and λ_{Σ} shown adaptation rate of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively. For stability of the underlying system, according to [27], at each step,

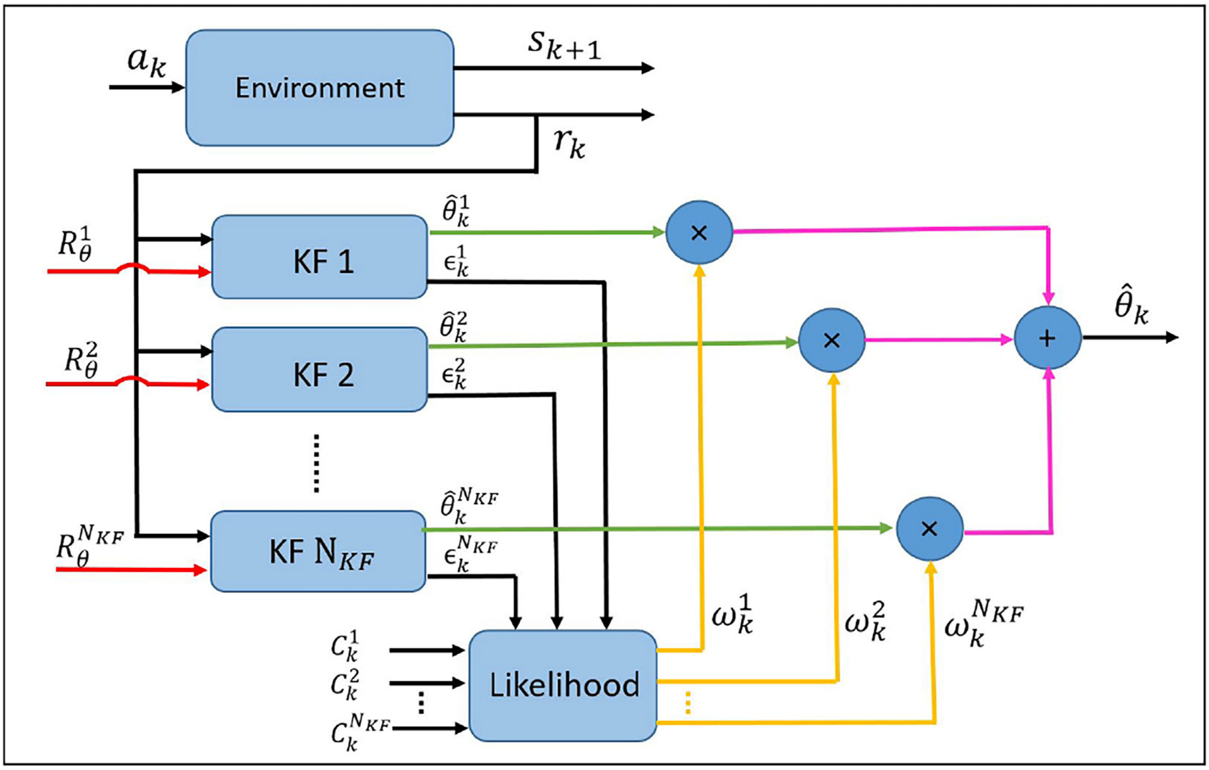


Fig. 2. MMAE structure.

we only execute one of the updates in Eqs. (38) and (39). More specifically, we update error covariance matrices of the RBFs via Eq. (39) in scenarios where the covariance is decreasing in size (i.e., $L_k^{\frac{1}{2}}(\boldsymbol{\theta}_k^T\boldsymbol{\psi}) < 0$). Otherwise, the means are updated based on Eq. (38). It is worth mentioning that the above approach avoids unlimited spread of the RBFs error covariance matrices.

3.3. KTD-based SR Learning

As explained in Section 3, we apply KTD algorithm, which is the combination of TD and Kalamn filter, for the SR learning due to its advantages over DNN-based frameworks. The TD approach of Eq. (12) is, therefore, used to approximate the SR as follows

$$\boldsymbol{m}^{\pi}_{\text{new}}(\boldsymbol{s}_k, a_k) \approx \boldsymbol{\psi}(\boldsymbol{s}_k, a_k) + \gamma \boldsymbol{m}^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1}, a_{k+1}). \tag{40}$$

By reordering terms in Eq. (40), state-action feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)$ can be considered as the measurement model within the KTD framework as follows

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k) = \boldsymbol{m}^{\pi}_{\text{new}}(\boldsymbol{s}_k, a_k) - \gamma \boldsymbol{m}^{\pi}_{\text{old}}(\boldsymbol{s}_{k+1}, a_{k+1}) + \boldsymbol{e}_k, \tag{41}$$

where $\boldsymbol{e}_k$ is considered to be a zero-mean Gaussian noise with the covariance of $\boldsymbol{E}_k$. By jointly considering Eqs. (11) and (41), the feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)$ can be approximated as

$$\begin{aligned}\boldsymbol{\psi}(\boldsymbol{s}_k, a_k) = \;& \boldsymbol{W}_k \boldsymbol{\psi}(\boldsymbol{s}_k, a_k) - \gamma \boldsymbol{W}_k \boldsymbol{\psi}(\boldsymbol{s}_{k+1}, a_{k+1}) + \boldsymbol{e}_k \\ = \;& \boldsymbol{W}_k \underbrace{\{\boldsymbol{\psi}(\boldsymbol{s}_k, a_k) - \gamma \boldsymbol{\psi}(\boldsymbol{s}_{k+1}, a_{k+1})\}}_{\boldsymbol{g}_k} + \boldsymbol{e}_k. \quad (42)\end{aligned}$$

To form a KF-based estimate of matrix $\boldsymbol{W}_k$, we construct a column vector $\boldsymbol{w}_k$ by stacking the columns of matrix $\boldsymbol{W}_k$. Based on the vector-trick property of Kronecker product, Eq. (42) can be rewritten as follows

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k) = (\boldsymbol{g}_k^T \otimes \boldsymbol{I})\boldsymbol{w}_k + \boldsymbol{e}_k,$$

where $\otimes$ denotes the Kronecker product and $\boldsymbol{I}$ is an identity matrix of a suitable dimension. Eq. (43) defines the observation of the system ($\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$) as a linear function of vector $\boldsymbol{w}_k$, which is to be estimated. Similar to the evolution model of the reward function's parameter $\boldsymbol{\theta}$ in Eq. (17), the following linear dynamic model is defined to represent evolution of $\boldsymbol{w}_k$ over time

$$\boldsymbol{w}_k = \boldsymbol{A}_k \boldsymbol{w}_{k-1} + \boldsymbol{u}_k, \tag{44}$$

where $\boldsymbol{A}_k = 0.9\boldsymbol{I}$ and the forcing term, $\boldsymbol{u}_k$, is considered to be a white Gaussian noise with zero mean and design covariance of $\boldsymbol{U}_k$. In a similar fashion to that of the reward weight estimation approach of Eqs. (18)–(22), a priori estimates of $\boldsymbol{w}_k$ and its covariance matrix $\boldsymbol{C}_k$ at time step k are obtained and then updated using measurement vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$. Matrix $\boldsymbol{W}_k$ can be then reconstructed via reshaping the estimated vector $\boldsymbol{w}_k$ into a $L \times L$ matrix. Now, the state-action value function for each pair $(\boldsymbol{s}_k, a_k)$ can easily be computed using Eq. (10).

3.4. Active Learning Scheme

A critical existing challenge in RL problems is the exploration/-exploitation trade-off, i.e., selection between exploiting a familiar action for a known reward or exploring unfamiliar actions for unknown rewards. Computation of uncertainty associated with the state-action value function is a key advantage of the proposed AKF – SR learning framework against its DNN-based counterparts. Actions are selected based on computed uncertainty at each filtering step. In other words, at each step, the action leading to the largest decrease in uncertainty of the state-action value function is selected. Since, the state-action value function ($Q(\boldsymbol{s}_k, a_k)$), has been modeled as a function of estimates $\boldsymbol{\theta}_k$ (reward's weight vector) and $\boldsymbol{W}_k$ (SR's weight matrix), their uncertainty can be used to approximate uncertainty of the state-action value function. Using the information form of KF (also referred to as information filter [52]), the information associate with $\boldsymbol{\theta}_k$ and $\boldsymbol{W}_k$, which are denoted by the inverse of their posteriori covariance matrices (i.e., $\boldsymbol{P}$ and $\boldsymbol{C}$) are updated as follows

$$\boldsymbol{P}_k^{-1} = \boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{h}_k^T \Omega_k^{-1} \boldsymbol{h}_k, \tag{45}$$

$$\boldsymbol{C}_k^{-1} = \boldsymbol{C}_{k|k-1}^{-1} + \boldsymbol{g}_k^T \boldsymbol{E}_k^{-1} \boldsymbol{g}_k \tag{46}$$

The choice of actions only affects terms $(\boldsymbol{h}_k^T\boldsymbol{h}_k)$ and $(\boldsymbol{g}_k^T\boldsymbol{g}_k)$ in Eqs. (45) and (46) as it changes $\boldsymbol{h}_k$ and $\boldsymbol{g}_k$. Therefore, $\boldsymbol{h}_k$ and $\boldsymbol{g}_k$ can change uncertainty of the state-action value function by choosing different actions. Since the information matrix $\boldsymbol{h}_k^T\boldsymbol{h}_k$ cannot be maximized; therefore, its trace ($\text{tr}(\boldsymbol{h}_k^T\boldsymbol{h}_k) = \boldsymbol{h}_k\boldsymbol{h}_k^T$) will be maximized. As stated previously, $\boldsymbol{h}_k = \boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$, where vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ has been constructed by placing the action-independent vector $\boldsymbol{\phi}(\boldsymbol{s}_k)$ in the corresponding spot for action a_k and setting the feature values for the other actions to zero. The value of $\boldsymbol{h}_k\boldsymbol{h}_k^T$ at the specific state $\boldsymbol{s}_k$, therefore, would be the square norm of the state feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k)$ for different choices of actions (i.e., action-independent). The action a_k is then selected by maximizing the SR weight's information as follows

$$a_k = \arg\max_a (\boldsymbol{g}_k^T(\boldsymbol{s}_k, a)\boldsymbol{g}_k(\boldsymbol{s}_k, a)). \tag{47}$$

This contribution is shown in the next section to effectively improve the performance of learning process in terms of achieved rewards from the environment. The proposed AKF – SR framework is briefed in Algorithm 1, and the model's parameters are listed in Table 1.

Algorithm 1: The AKF – SR Framework

```
Learning Phase:
Input: $\gamma, \boldsymbol{B}_k, \boldsymbol{U}_k, \lambda_{\boldsymbol{\mu}}, \lambda_{\Sigma}, \boldsymbol{E}_k$, and $\{\Omega_k^i\}$ for $i = 1, 2, \ldots, N_{\text{KF}}$
Initialize: $\theta_0, \boldsymbol{P}_0, \boldsymbol{w}_0, \boldsymbol{C}_0, \boldsymbol{\mu}_0^n$, and $\Sigma_0^n$ for $n = 1, 2, \ldots, N_{\text{RBF}}$
Repeat (for each episode):
  Initialize $\boldsymbol{s}_k$.
  for $k = 1, 2, \ldots$ do:
    RBF-based Feature Vector Construction: Construct $\boldsymbol{\psi}(\boldsymbol{s}_k, a)$ through Eqs. (14) and (15).
    Active Learning: $a_k = \arg\max_a(\boldsymbol{g}_k^T(\boldsymbol{s}_k, a)\boldsymbol{g}_k(\boldsymbol{s}_k, a))$.
    Take action $a_k$ , observe $\boldsymbol{s}_{k+1}$ and $r_k = R(\boldsymbol{s}_k, a_k)$.
    MMAE-based Reward Learning: Perform Eqs. (16)–(29), (31) to estimate $\boldsymbol{\theta}_k$.
    KTD-based SR learning: Perform KF on Eqs. (43) and (44) to update $\boldsymbol{w}_k$.
    Reshape $\boldsymbol{w}_k$ to construct matrix $\boldsymbol{W}_k$.
    Calculate feature-based SR vector as $\boldsymbol{m}^{\pi}(\boldsymbol{s}_k, a_k) = \boldsymbol{W}_k\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$.
    Calculate the state-action value function $Q(\boldsymbol{s}_k, a_k) = \boldsymbol{\theta}_k^T\boldsymbol{m}^{\pi}(\boldsymbol{s}_k, a_k)$.
    Update RBFs Parameters: Perform RGD, Eqs. (32), (34), (37)–(39), to achieve $\Sigma_{k+1}^n$ and $\boldsymbol{\mu}_{k+1}^n$ for $1 \leqslant n \leqslant N_{\text{RBF}}$.
  end for
```

4. Experimental results

The proposed AKF – SR framework is evaluated in this section and compared to the state of the art RL algorithms: Deep Q-Network [37], Substochastic Successor Representation (SSR)

Table 1
Parameters of the proposed AKF – SR framework.

Name	Symbol
Discount factor	γ
Number of possible actions	N_{actions}
Number of states' variables	D
Order of RBFs	O_{RBF}
Number of RBFs	N_{RBF}
RBF's mean	$\boldsymbol{\mu}^n$
RBF's covariance	$\boldsymbol{\Sigma}^n$
Length of the state-action feature vector	L
Number of KFs in MMAE scheme	N_{KF}
Process noise covariance of reward learning	$\boldsymbol{B}$
Measurement noise variance of reward learning	Ω
Posteriori estimate covariance of reward learning	$\boldsymbol{P}$
Process noise covariance of SR learning	$\boldsymbol{U}$
Measurement noise covariance of SR learning	$\boldsymbol{E}$
Posteriori estimate covariance of SR learning	$\boldsymbol{C}$
Adaptation rate for RBF's mean update	$\lambda_{\boldsymbol{\mu}}$
Adaptation rate for RBF's covariance update	λ_{Σ}

framework [36], and Universal Successor Representations (USR) [2]. In order to demonstrate efficacy of the proposed AKF – SR framework in the learning process as well as its rapid adaptation to the reward changes, the following popular RL benchmarks are considered: (1) *Mountain Car*, (2) *Inverted Pendulum*, and; (3) Lunar Lander.

The classical DQN described by [37], uses a convolutional neural network for calculation of state-action value function. The TD error is then used to perform a gradient descent step with respect to the parameters of the network. The SSR [36] computes the norm of the SR while it is being learned with combination of the TD learning and a DNN. It has been shown that SSR implicitly counts state visitation, explaining some light for the uncertainty about that state for exploration/exploitation trade-off. The state-action value function is then learned through a DNN model while it considers the agent's next state uncertainty as an exploration bonus obtained from the SSR. Similar to the proposed AKF – SR, SSR estimates the uncertainty of the state (associated value function). But, there are two main differences between AKF – SR and SSR: (i) SSR learns the SR through a DNN model while as previously mentioned, AKF – SR learns the SR via KTD (i.e., incorporation of the TD and KF), and: (ii) SSR considers the norm of the estimated SR as an exploration factor while AKF – SR uses the covariance of the estimated SR for exploration/exploitation trade-off. The USR [2] uses a deep learning framework to approximate the SR and incorporate it with actor-critic method to learn the SR. There are two key differences to highlight between the proposed AKF – SR and USR approaches. Firstly, USR adopts actor-critic method, which is a TD method that has a separate memory structure to explicitly represent the policy independent of the value function while our proposed AKF – SR framework is a value-based TD method that estimates the policy by estimating the associated value function. The second key difference is that similar to DQN and SSR schemes, USR uses a DNN for the SR learning while our proposed AKF – SR adopts KTD for the SR learning.

4.1. Choosing AKF – SR *Parameters*

For adaptation of RBFs' parameters, generally, finding a proper shape parameter of RBFs is not easy. The centers of RBFs are typically distributed evenly along each dimension of state vector, leading to O_{RBF}^D centers for D state variables and an order O_{RBF} given by user; variance of each state variable (σ^2) is often initialized to $\frac{2}{O_{\text{RBF}}-1}$. It has been shown that RBFs only generalize locally-changes in one area of the state space and do not affect the entire state space [29]. The covariance between two different state variables is thus set as zero. Therefore, $\boldsymbol{\Sigma}_0 = \sigma^2$ is a scalar if there is one state variable (i.e., $D = 1$); otherwise, it is a $D \times D$ diagonal positive definite matrix with the entries of $\frac{2}{O_{\text{RBF}}-1}$. The number of state variables for both the Inverted Pendulum ($\boldsymbol{s}_k = \left[\theta, \dot{\theta}\right]$) and Mountain Car ($\boldsymbol{s}_k = [x, \dot{x}]$) environments used in the manuscript are $D = 2$. Since we have selected ($O_{\text{RBF}} = 3$), $\boldsymbol{\Sigma}_k^n$ for all $3^2 = 9$ RBFs (i.e, $N_{\text{RBF}} = 9$) are initialized as $\boldsymbol{\Sigma}_0^n = \boldsymbol{I}_2$, which is a positive definite (invertible) matrix. For the Lunar Lander, $D = 6$ and the value of O_{RBF} is selected to be 2. Therefore, $\boldsymbol{\Sigma}_0^n = 2\boldsymbol{I}_6$ for all 64 RBFs.

As explained in Sub-Section 3.2.2, the measurement mapping function in Eq. (16) (i.e., $\boldsymbol{h}$) is constructed as a function of state-action feature vector $\boldsymbol{h} = \boldsymbol{\psi}^T(\boldsymbol{s}_k, a_k)$, its adaptation, therefore, requires adaptation of the basis functions parameters $\boldsymbol{\mu}_k^n, \boldsymbol{\Sigma}_k^n$ for $(1 \leqslant n \leqslant N_{\text{RBF}})$. In this regard, a gradient descent adaptation approach is adopted to update either of the means and covariances of the RBFs at each time step k based on Eqs. (38) and (39), respectively. The four terms in the right-hand side of Eq. (39) (i.e., $\left(\boldsymbol{\Sigma}_k^n\right)^{-1}\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)\left(\boldsymbol{s}_k - \boldsymbol{\mu}_k^n\right)^T\left(\boldsymbol{\Sigma}_k^n\right)^{-1}$) form a positive semi-definite matrix and the multiplication of four terms in the left-hand side of Eq. (39) (i.e., $2\lambda_{\Sigma}(L_k)^{\frac{1}{2}}\boldsymbol{\theta}_k^T\boldsymbol{\psi}$) results in a scalar positive number; therefore, $(-\lambda_{\Sigma}\Delta\boldsymbol{\Sigma})$ is a positive semi-definite matrix. Since, the covariance matrix has been initialized to a positive definite matrix $(\frac{2}{O_{\text{RBF}}-1}\boldsymbol{I}_D)$, the updated $\boldsymbol{\Sigma}_k^n$ for $k \geqslant 1$ achieved from Eq. (39) is always a positive definite matrix (consequently, invertible).

The values for $\lambda_{\boldsymbol{\mu}}$ and λ_{Σ} are selected in such a way to keep the system stable. The discount factor denoted by γ affects how much weight is given to the future rewards in the value function. A discount factor $\gamma = 0$ will result in state/action values representing the immediate reward, while a higher discount factor will result in the values representing the cumulative discounted future reward that an agent is expected to receive (behaving under a given policy). Commonly (as is the case in the implemented environments), a large portion of the reward is earned upon reaching the goal. To prioritize this final success, we expect an acceptable γ to be close 1. In our manuscript, γ is set to the high values of 0.95 and 0.99. In order to use the KFs implemented for reward learning and SR learning components of the proposed AKF – SR framework, the parameters of two filters have to be chosen: the variances/covariances of the observation noises, the priors and the variances/covariances of the process noise. The priors ($\boldsymbol{\theta}_0$ and $\boldsymbol{w}_0$) should be initialized to the values close to the ones looks optimal, or to a default value (i.e., the zero vector). The priors $\boldsymbol{P}_0$ and $\boldsymbol{C}_0$ respectively show the certainty in the prior guess of $\boldsymbol{\theta}_0$ and $\boldsymbol{w}_0$, the lower the less certain. The process noise covariance of a KF is a design parameter. If some knowledge about non-stationarity is available, it can be used to choose this matrix. However, such a knowledge is generally difficult to obtain in advance. The proposed AKF – SR algorithm systematically allows to use a set of different values of process noise covariance ($\boldsymbol{B}$ and $\boldsymbol{U}$). The priors $\boldsymbol{\theta}_0, \boldsymbol{w}_0, \boldsymbol{P}_0$, and $\boldsymbol{C}_0$ of the proposed AKF – SR framework are chosen by trial and error. They may not be the best ones, but orders of magnitude are correct. Finally, as stated previously, the measurement noise covariance of a KF, is one of the most important parameters to be identified. To select this parameter for the KF used in reward function learning process (i.e., Ω), our intuition in the proposed AKF – SR framework is to use MMAE scheme, which covers the potential range of the measurement noise variance Ω using N_{KF} mode-matched filters. The parameter N_{KF} is set a priori denoting the number of candidates Ω^i, for $(1 \leqslant i \leqslant N_{\text{KF}})$. In the experiments, we have selected N_{KF} to be equal to 11. It is worth mentioning that measurement noise covariance of the KF used in KTD for the SR learning process ($\boldsymbol{E}$), can be also selected via MMAE in a similar way to Ω. But, since $\boldsymbol{E}$ is a square high dimensional matrix ($L \times L$), applying MMAE comes with high computational cost. The measurement noise covariance $\boldsymbol{E}$ is, therefore, selected by trial and error.

4.2. Inverted Pendulum

In the first experiment, Inverted Pendulum environment, shown in Fig. 3, is considered. A pole is attached to a cart moving along a friction-less track. At each time step, an agent can move the cart to the left or to the right. The goal is to prevent it from falling over. The pendulum's angle from upright position and its angular velocity (i.e., $\boldsymbol{s}_k = \left[\theta, \dot{\theta}\right]^T$) constitute the state of the system. A reward of $+1$ is given at each step that the pendulum is above the horizontal line. Reward 0 will be provided to the agent when $|\theta| > \pi/2$, i.e., when the pendulum in the next state is beyond the horizontal. Each episode terminates upon falling of the pendulum or when its length approaches 200 runs.

For implementation of the proposed AKF – SR, we use 9 RBFs together with a bias parameter. The size of the state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ is then equal 30 as $\mathscr{A} = \{-50, 0, +50\}$, i.e., three actions are possible. Initialization is performed as follows

$$\boldsymbol{\mu}_0^n \in \{-\pi/4, 0, +\pi/4\} \times \{-0.5, 0, +0.5\} \tag{48}$$

$$\boldsymbol{\Sigma}_0^n = \boldsymbol{I}_2, \tag{49}$$

where $\boldsymbol{I}_2$ denotes an identity matrix of dimension of 2×2. Based on the Eq. (15), the state-action feature vector is given by

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k = +50) = [1, \phi_1, \ldots, \phi_9, 0, \ldots 0, 0, \ldots, 0]^T \tag{50}$$

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k = -50) = [0, \ldots, 0, 1, \phi_1, \ldots \phi_9, 0, \ldots, 0]^T \tag{51}$$

$$\boldsymbol{\psi}(\boldsymbol{s}_k, a_k = 0) = [0, \ldots, 0, 0, \ldots, 0, 1, \phi_1, \ldots \phi_9]^T, \tag{52}$$

where the value of ϕ_n, for ($1 \leqslant n \leqslant 9$), is computed based on Eq. (14). We use the initial values of $\lambda_{\boldsymbol{u}} = 200$ and $\lambda_\Sigma = 100$ for initialization, which are obtained via trial and error to have a stable system. We use $\gamma = 0.95$ for the discount factor. The process noise covariances for learning of the reward weight and the SR weight are selected as time-invariant values $\boldsymbol{B}_k = 10^{-3}\boldsymbol{I}_{30}$ and $\boldsymbol{U}_k = 10^{-2}\boldsymbol{I}_{900}$. The measurement noise covariance of SR learning ($\boldsymbol{E}_k$) is set to $\boldsymbol{I}_{30}$ and the measurement noise variance candidates for estimation of the reward's weight comes from

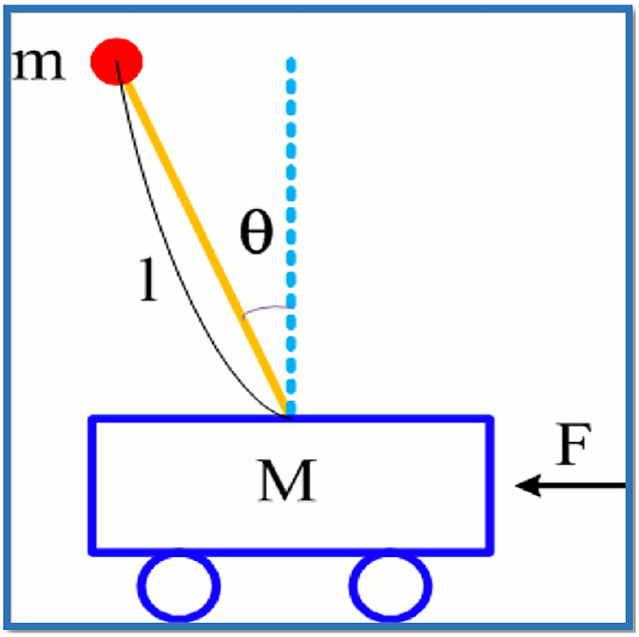


Fig. 3. The Inverted Pendulum environment.

$$\Omega_k^i \in \{0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}. \tag{53}$$

Finally, $\boldsymbol{\theta}_0 = \boldsymbol{0}$ and $\boldsymbol{w}_0 = \boldsymbol{0}$ are used to initialize the weights, and $\boldsymbol{P}_0 = 10\boldsymbol{I}_{30}$ and $\boldsymbol{C}_0 = 10\boldsymbol{I}_{900}$ are utilized to initialize the posteriori covariance matrices.

A zero-mean Gaussian distribution with standard deviation of 0.1 is used to randomly select starting angle for each episode. Learning process is performed over 1,000 episodes. In order to evaluate effect of the proposed uncertainty-based action selection process within the AKF – SR framework, we measure the averaged achieved rewards at each episode over 50 runs to investigate how the proposed method helps the agent to choose the optimum action, which leads to the least future penalties (maximum future rewards). By selecting such an action at each step, the agent will achieve its goal by doing as few interactions as possible. The achieved reward at each episode, therefore, demonstrates how far the selected actions at each step are from the optimum ones. To illustrate effectiveness of the proposed AKF – SR framework, we compare the results obtained from that with those obtained from the DQN [37], SSR [36], and USR [2]. As Fig. 4 shows, AKF – SR algorithm performs better than others in terms of achieved rewards toward keeping the inverted pendulum above the horizontal line. These results demonstrate the positive effect of uncertainty usage in action selection process of AKF – SR. For a RL algorithm to be usable in different applications, it should be able to reproduce consistent performances across multiple training runs (i.e., reproducibility). As it can be observed in Fig. 4, at each episode, DNN-based frameworks have a greater variance across 50 runs than AKF – SR, which can show sensitivity of DNNs to different factors such as random initialization of the optimization and hyper-parameter setting. Such a high sensitivity can lead to unpredictable performance in practical scenarios and also a large search (consequently, more time and more cost) to find the best model. Another aspect for measuring the reliability of an algorithm is the ability of the algorithm in generating stable performance across episodes (i.e., stability). As can be seen from the results, AKF – SR has less sudden changes in its performance over episodes in comparison to others, which makes AKF – SR framework more reliable to apply to different scenarios. For evaluating the reliability of an RL algorithm, the time and effort spent for tuning of the

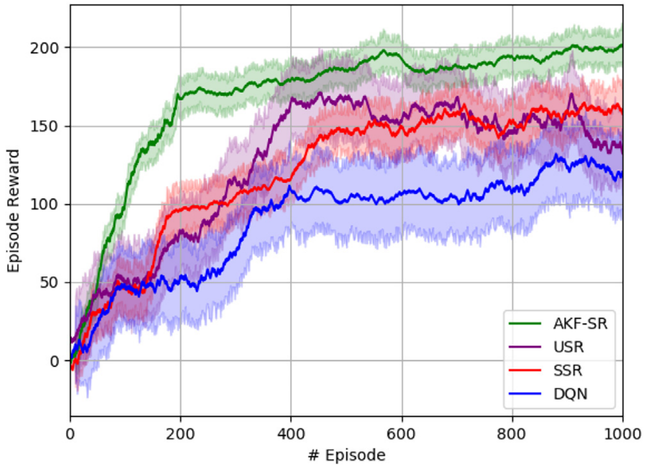


Fig. 4. The mean (solid lines) and standard deviation (shaded regions) of cumulative episode's reward on the Inverted Pendulum environment over the learning process.

algorithm's parameters should also be considered. While a DNN has a large number of parameters to be tuned (usually million number of parameters), the proposed filtering-based AKF – SR framework only requires to tune the process and measurement noise covariances ($\boldsymbol{B}, \boldsymbol{U}, \Omega, \boldsymbol{E}$), means and covariances of RBFs ($\boldsymbol{\mu}^n, \boldsymbol{\Sigma}^n$), and adaptation rates used for RBFs parameters tuning ($\lambda_{\boldsymbol{\mu}}, \lambda_{\Sigma}$). Much less time and effort is, therefore, required for parameters tuning of AKF – SR framework as compared to its DNN counterparts.

In addition, the key advantage of the proposed SR-based method, which is the decomposition of state-action value function into the SR and reward function, allows the proposed AKF – SR framework to rapidly adapt to changes in the reward function. In order to investigate this strength, after convergence of both models, value of the reward upon keeping the pendulum above the horizontal line is changed from $+1$ to $+3$ and the state-action value is relearned through different frameworks. The estimated state-action value function in the SR context of AKF – SR and USR [2], will converge to this change by just updating (relearning) the reward given the new external rewards (the SR remains same and the agent does not need to learn the SR another time). However, adjustment of DQN and SSR algorithm to changes comes with updating cached values at all states in the environment. Results shown in Fig. 5 confirm that the computing the value function via a dot product between the SR and reward enables the agent to adapt itself rapidly to the new value function by just updating the reward function.

Furthermore, to illustrate the stability of the RBFs in the manuscript, in the Inverted Pendulum task, we have fixed the number of RBFs to be 10. We have performed proposed AKF – SR scheme for 200 episodes. The entire process have been repeated 50 times for three different values of widths ($\boldsymbol{\Sigma}$) of RBFs and the number of steps to goal was averaged over 50 times. As Fig. 6 shows, by using the RBFs for the reward function and the SR approximations, we can achieve a steady state performance.

4.3. Mountain Car

The Mountain Car benchmark RL platform shown in Fig. 7 is used to conduct the second experiment. In this classic RL problem, one car is positioned on a 1-D track between two mountains. The objective of the RL model is to bring the car to the top of the right mountain. It is assumed that the car cannot climb the right mountain in one single try. For success in this scenario, therefore, the car needs to go back and forth building up momentum to reach its goal. Position and velocity of the car (i.e., $\boldsymbol{s} = [x, \dot{x}]^T$) constitute the state of the system. Available actions are "push left", "push right", and "no push", represented by set $\mathscr{A} = \{0, 1, 2\}$. The objective of the RL model is to reach the top position on the right mountain, i.e., car's position becomes greater than or equal to $0.5m$. In scenarios that the objective is not achieved (i.e., car can not mount the right hill), a -1 reward will be given to the agent for each step. No penalty will be assigned in case that the car reaches the left hill (this state is treated as hitting a wall). The starting state of each episode is a random position within $-0.6m$ to $-0.4m$ range with zero velocity.

In a similar fashion to the Inverted Pendulum experiment in Sub-Section 4.2, the state-action feature vector for Mountain Car is also constructed based on 9 RBFs and a bias parameter (resulting in a feature vector of size 30). Initialization is performed as follows

$$\boldsymbol{\mu}_0^{(n)} \in \{-0.775, -0.35, +0.775\} \times \{-0.035, 0, +0.035\} \tag{54}$$

$$\boldsymbol{\Sigma}_0^{(n)} = \boldsymbol{I}_2 \tag{55}$$

The process noise covariances for learning the reward's weight and the SR weight are selected as $\boldsymbol{B}_k = 10^{-2}\boldsymbol{I}_{30}$ and $\boldsymbol{U}_k = 10^{-2}\boldsymbol{I}_{900}$. The discount factor γ is set to 0.95. Values $\boldsymbol{\theta}_0 = \boldsymbol{0}$ and $\boldsymbol{w}_0 = \boldsymbol{0}$ are used to initialize the weights. Values $\boldsymbol{P}_0 = 10\boldsymbol{I}_{30}$ and $\boldsymbol{C}_0 = 10\boldsymbol{I}_{900}$ are exploited to initialize the posteriori covariance matrices. The measurement noise covariance of SR learning ($\boldsymbol{E}_k$) and the measurement noise variance candidates for estimation of the reward's weight are selected from the same set as the Inverted Pendulum environment. The model learns from $1,000$ episodes, with a maximum steps of 200 at each episode. Each episode is repeated 50 times and averaged. Fig. 8 demonstrates that AKF – SR outperforms DNN-based frameworks in the terms of performance criteria defined for Inverted Pendulum environment, i.e., cumulative reward received at each episode, time and computational cost, and reproducibility and stability aspects of the algorithms.

4.4. Lunar Lander

In the third experiment, we focus on the Lunar Lander environment, which is a more complicated environment compared to the two previous ones. In the Lunar Lander environment, the goal for the RL agent is to learn to land successfully on a landing pad located at coordinate $(0, 0)$ in a randomly generated surface on the moon as shown in Fig. 9. The state space of the system consists of the agent's position (x, y) in space, horizontal and vertical veloc-

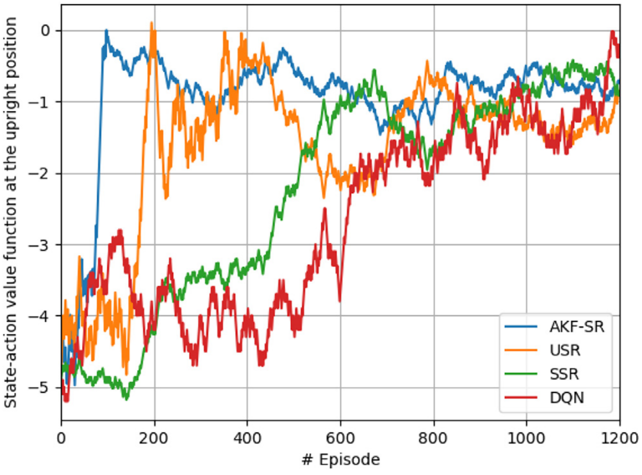


Fig. 5. State-action value function after change of the reward's value at the upright position of the Inverted Pendulum.

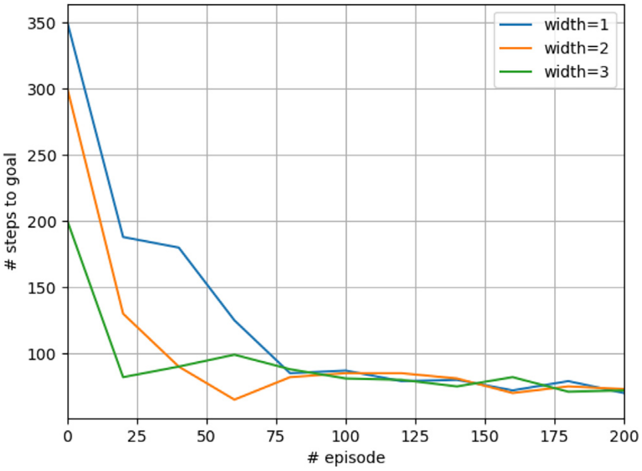


Fig. 6. Stability analysis of the RBFs.

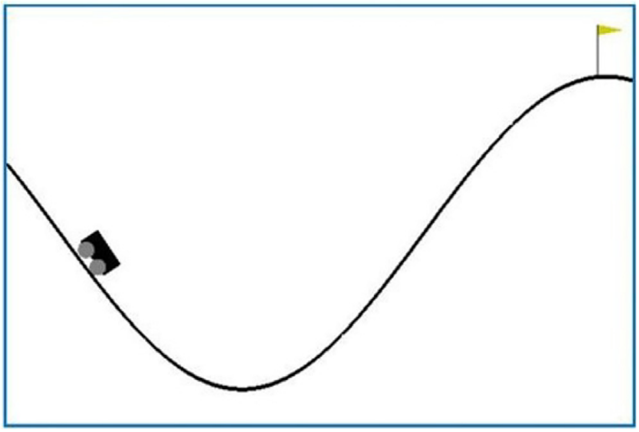


Fig. 7. The Mountain Car environment.

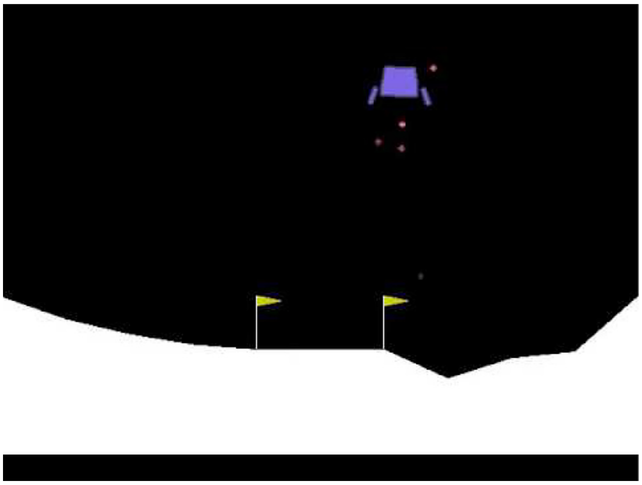


Fig. 9. The Lunar Lander environment..

ity (v_x, v_y), orientation in space θ, and angular velocity $\dot{\theta}$. The agent has four possible actions, i.e., do nothing; firing the left engine, firing the main engine, and; firing the right engine ($\mathscr{A} = \{0, 1, 2, 3\}$). Reward for landing on the pad is about 100 to 140 points, varying on the lander placement on the pad. If the lander moves away from the landing pad it loses reward. Each episode terminates if the lander lands or crashes, receiving additional $+100$ or -100 points, respectively. Each leg ground contact worth $+10$ points. Firing the main engine results in a -0.3 point penalty for each frame. The problem is considered solved if the agent receives $+200$ points over 100 iterations. The RBFs of order two are considered for each state variable resulting in 64 RBFs for each action. Consequently, the size of the feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k, a_k)$ will be 256. Based on the useful range of each variable of state vector, the initial mean and covariance of the RBFs are chosen as follows

$$\boldsymbol{u}_0^n \in \{-0.333, +0.333\}^6, \tag{56}$$

$$\boldsymbol{\Sigma}_0^n = 2\boldsymbol{I}_6. \tag{57}$$

The initial values of $\lambda_{\boldsymbol{mu}}$ and λ_{Σ} are both selected as 200 to keep the system stable. The discount factor is selected as 0.99. The noise covariances are selected as $\boldsymbol{B}_k = 10^{-2}\boldsymbol{I}_{256}$ and $\boldsymbol{E}_k = \boldsymbol{I}_{256}$. Candidate values for Ω are selected from the same set as was used for the Inverted Pendulum environment. Like the two previous environments, the agent is trained through different 1000 episodes and training process is repeated 50 runs. Fig. 10 depicts the cumulative reward averaged over 50 runs reward per each episode for different approaches. Due to the complexity of the lunar lander environment, the cumulative rewards achieved from $AKF-SR$ has more fluctuations in comparison to inverted pendulum and mountain car. But as it can be observed, $AKF-SR$ still outperforms its counterparts in terms of cumulative reward received at each episode, and reproducibility and stability of the algorithms.

In order to evaluate the linear assumptions of the reward function and the SR, the Mean Squared Error (MSE) loss of the estimated reward and the SR for 1000 episodes over 50 training runs are calculated and are shown in Fig. 11. As expected, both the reward and the SR losses decrease over training process. Despite the fluctuations in the loss values, the linear model estimates seem to be good enough to achieve a better performance over model-free DQN method and SR-based SSR and USR frameworks. The loss of state-action value function, which is summation of the SR loss and the reward loss, are averaged over 1000 episodes and shown in Table 2 for different frameworks. As it can be seen in Table 2, the average loss in the proposed $AKF-SR$ is less than that of the DNN-based approaches.

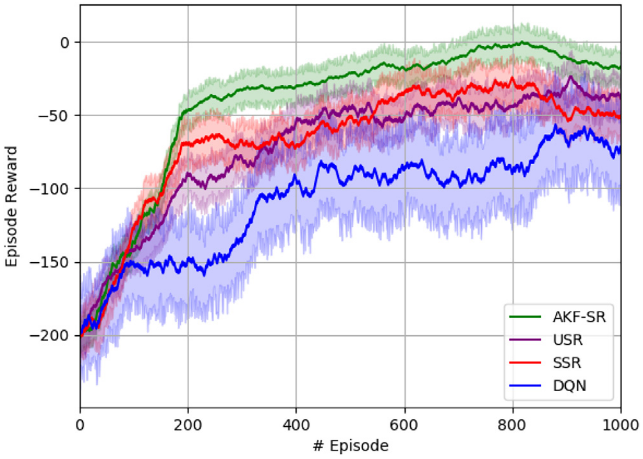


Fig. 8. The mean (solid lines) and standard deviation (shaded regions) of cumulative episode's reward on the Mountain Car environment over the learning process.

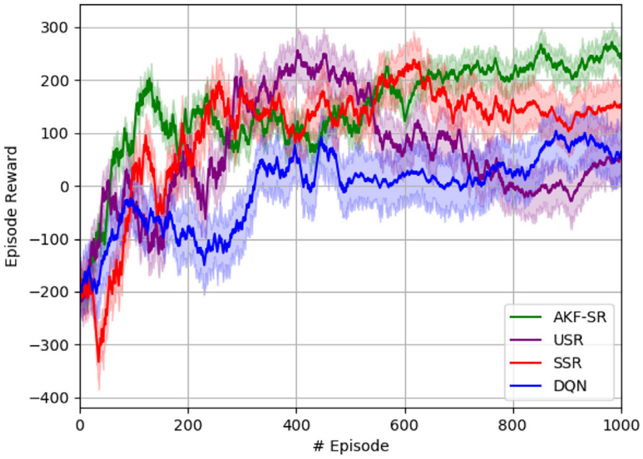


Fig. 10. The mean (solid lines) and standard deviation (shaded regions) of cumulative episode's reward on the Lunar Lander environment over the learning process.

4.5. Algorithmic complexity

As explained in the previous sections, the proposed AKF – SR framework consists of four main modules: (i) RBF-based Feature Vector Construction, which maps each state $\boldsymbol{s}_k$ to N_{RBF}-dimensional state feature vector $\boldsymbol{\phi}(\boldsymbol{s}_k)$ consisting of basis functions achieved from Eq. (14). The state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ is then generated by assigning $\boldsymbol{\phi}(\boldsymbol{s}_k)$ to the corresponding spot for action a_k while the feature vector values for the remainder of the actions are set to zero. The whole construction process of state-action feature vector $\boldsymbol{\psi}(\boldsymbol{s}_k, a_k)$ has a computational complexity of $O\left(N_{\text{RBF}} \times D^3\right)$ per iteration, (ii) Reward Learning, which first, estimates the reward weight vector θ_k via implementation of N_{RBF} parallel KFs. The global computational complexity (per iteration) of MMAE is $O\left(N_{\text{KF}} \times L^3\right)$. The measurement mapping function of the reward weight vector is then adapted by updating the means and covariances of basis functions through Eqs. (38) and (39). Since at each time step, only the mean or covariance is updated, this process results in memory complexity of $O\left(\frac{L+D^3}{2}\right)$, (iii) KTD-based SR Learning, which estimates the SR weight matrix $\boldsymbol{W}_k$ via KTD algorithm. This estimation procedure can be computed in $O\left(L^4\right)$ [34] per iteration, and; (iv) Active Learning Scheme, which selects the action leading to the largest decrease in uncertainty of the state-action value function in $O(L \times N_{\text{actions}})$. The global computational complexity (per iteration) of the proposed AKF – SR scheme is, therefore, in $O\left(N_{\text{RBF}} \times D^3 + N_{\text{KF}} \times L^3 + L^4\right)$. It should be noted that the main focus of this paper is on the improvement of the performance of RL agents by considering their uncertainties/beliefs for choosing different actions. As it has been shown in the manuscript, this uncertainty can be achieved by using KF for the reward function learning and KTD for the SR learning. The proposed MMAE and RGD improve the performances of the used KF by adaptation of its two most important parameters: measurement noise covariance and measurement mapping function. The computational complexities of DQN [37], USR [2] and SSR [36] are not specified; however, based on our experiments and since the proposed AKF – SR has much less trainable parameters than the DQN [37], USR [2], and SSR [36] frameworks, which all use DNNs with large number of parameters for the learning process, AKF – SR has less computational cost compared to its DNN-based counterparts. As an example, DQN with only one hidden layer of size 64 has 11140 parameters resulting in high computational complexity and memory requirement. Training with a mini-batch of 32 on a typical performance GPU shows that it needs over 3.5 GB of local DRAM.

Table 2
The MSE loss of state-action value function averaged over 1000 episodes.

Environment	AKF – SR	DQN	SSR	USR
Inverted Pendulum	6.43	11.98	7.36	7.69
Mountain Car	6.78	14.78	9.94	8.46
Lunar Lander	8.93	14.59	16.76	11.98

5. Conclusion

This paper proposed a novel KTD-based SR framework, referred to as the AKF – SR, to learn goal reaching behavior in RL problems. The AKF – SR learns the value function by computing the inner product of the SR and reward function's weight vector. This factorization of the value function, enables the learned SR to be reused for other tasks with the same domains but different reward values. Moreover, we extended KTD framework of MF algorithms [34] to the SR learning, which estimates uncertainty of the value function, and also deals with over-fitting, parameter sensitivity (consequently, unreliability in reproducing consistent performances across multiple runs), and high memory requirement problems of DNN-based algorithms. Another key advantage of the proposed AKF – SR algorithm is using RGD and an innovative MMAE schemes for the reward function learning, which tackles the effect of improper selection of the filter's parameters on its performance and stability to make the framework more usable in practical problems. Furthermore, an innovative active learning scheme, which considers uncertainty of the value function achieved from KTD algorithm for actions selection, was adopted within the AKF – SR framework to deal with the exploration/exploitation dilemma. The proposed AKF – SR framework was evaluated based on three continuous state space RL benchmarks: Inverted Pendulum, Mountain Car, and Lunar Lander. The averaged accumulative reward and MSE loss for 1000 episode over 50 runs were calculated. The results showed that the proposed AKF – SR framework outperforms its counterparts in terms of cumulative reward, reproducibility and stability aspects of reliability, and time and effort spent for finding and learning the best model. To compare adaptation speed of AKF – SR with DNN-based frameworks to changes in the reward

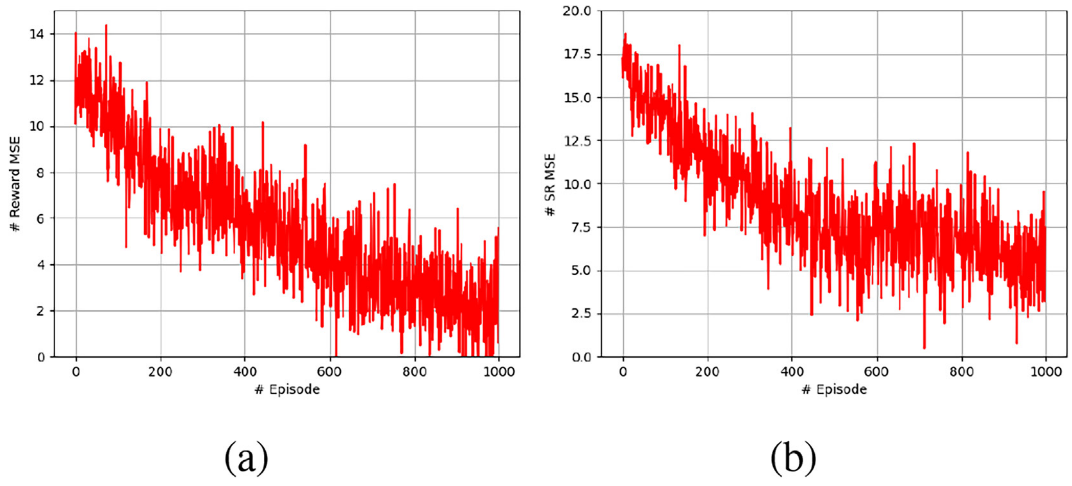


Fig. 11. The calculated MSE loss of the estimated reward function and the SR in the AKF – SR framework over 1000 episodes for the Lunar Lander: (a) MSE of the estimated reward, and (b) MSE of the estimated SR.

function, the reward value of Inverted Pendulum environment was changed and all algorithms were relearned. As it was expected, $AKF-SR$ could adapt itself to the changes faster than other frameworks.

For estimating of the SR weight matrix $\boldsymbol{W}_k$ through implemented Kalman filter within the KTD framework, we mapped the matrix into a column vector by stacking its columns one underneath the other and proceed with standard Kalman filter for vectors. However, it is often difficult to ensure that this vectorized estimation method does not cause any consequential losses in the original structure of the problem. Furthermore, for high-dimensional estimation models (e.g., a RL environment with a large number of states variables), such a method comes with an excessive computational cost. A number of studies have investigated to develop estimation algorithms in terms of the original system matrices [56,57]. As future work, we plan to apply matrix-based filtering algorithms for estimation of the SR weight matrix in order to extend the proposed $AKF-SR$ framework to more complex RL tasks with high-dimensional states in an efficient manner.

CRediT authorship contribution statement

Parvin Malekzadeh: Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Investigation, Writing - original draft, Writing - review & editing. **Mohammad Salimibeni:** Conceptualization, Software, Validation, Investigation, Writing - original draft, Writing - review & editing, Data curation. **Ming Hou:** Supervision, Writing - review & editing, Conceptualization, Funding acquisition. **Arash Mohammadi:** Supervision, Writing - review & editing, Conceptualization, Project administration. **Konstantinos N. Plataniotis:** Supervision, Writing - review & editing, Conceptualization, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] S. Spanó, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Matta, A. Nannarelli, M. Re, An Efficient Hardware Implementation of Reinforcement Learning: The Q-Learning Algorithm, IEEE Access 7 (2019) 186340–186351.

[2] C. Ma, J. Wen, and Y. Bengio, "Universal successor representations for transfer reinforcement learning, arXiv preprint arXiv:1804.03758, 2018..

[3] M. Seo, L.F. Vecchietti, S. Lee, D. Har, Rewards Prediction-Based Credit Assignment for Reinforcement Learning With Sparse Binary Rewards, IEEE Access 7 (2019) 118776–118791.

[4] P. Malekzadeh, M. Salimibeni, A. Mohammadi, A. Assa, K.N. Plataniotis, MM-KTD: Multiple Model Kalman Temporal Differences for Reinforcement Learning, IEEE Access 8 (2020) 128716–128729.

[5] A. Toubman et al., "Modeling behavior of Computer Generated Forces with Machine Learning Techniques, the NATO Task Group approach, IEEE Int. Con. Systems, Man, and Cyb. (SMC), Budapest, 2016, pp. 001906–001911..

[6] J.J. Roessingh et al., "Machine Learning Techniques for Autonomous Agents in Military Simulations - Multum in Parvo, IEEE Int. Con. Systems, Man, and Cyb. (SMC), Banff, AB, 2017, pp. 3445–3450..

[7] H.K. Venkataraman, and P.J. Seiler, "Recovering Robustness in Model-Free Reinforcement Learning, American Control Conference (ACC), Philadelphia, PA, USA, 2019, pp. 4210-4216..

[8] H. Hu, S. Song, C.L.P. Chen, Plume Tracing via Model-Free Reinforcement Learning Method, IEEE Transactions on Neural Networks and Learning Systems 30 (8) (Aug. 2019) 2515–2527.

[9] S.D.C. Shashua, and S. Mannor, "Kalman meets bellman: Improving policy evaluation through value tracking, arXiv preprint arXiv:2002.07171, 2020..

[10] G. Williams, N. Wagener, B. Goldfain, P. Drews, J.M. Rehg, B. Boots, E.A. Theodorou, Information Theoretic MPC for Model-based Reinforcement Learning, in: International Conference on Robotics and Automation (ICRA), 2017.

[11] S. Ross and J.A. Bagnell, "Agnostic System Identification for Model-based Reinforcement Learning, arXiv:1203.1007, 2012..

[12] A. Ayoub, Z. Jia, C. Szepesvari, M. Wang, M., and L. Yang, "Model-based reinforcement learning with value-targeted regression, International Conference on Machine Learning, pp. 463–474, PMLR, 2020..

[13] E. Vértes, M. Sahani, A neurally plausible model learns successor representations in partially observable environments, Advances in Neural Information Processing Systems 32 (2019) 13714–13724.

[14] S. Blakeman, D. Mareschal, A complementary learning systems approach to temporal difference learning, Neural Networks 122 (2020) 218–230.

[15] P. Dayan, Improving generalization for temporal difference learning: The successor representation, Neural Computation 5 (4) (1993) 613–624.

[16] A. Ducarouge, O. Sigaud, "The Successor Representation as a model of behavioural flexibility, Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA), 2017..

[17] T.D. Kulkarni, A. Saeedi, S. Gautam, and S.J. Gershman, "Deep successor reinforcement learning, arXiv preprint, arXiv:1606.02396, 2016..

[18] M. Riedmiller, Neural Fitted Q Iteration-first Experiences with a Data Efficient Neural Reinforcement Learning Method, in: European Conference on Machine Learning, Springer, 2005, pp. 317–328.

[19] Y. Tang, H. Guo, T. Yuan, X. Gao, X. Hong, Y. Li, J. Qiu, Y. Zuo, J. Wu, Flow Splitter: A Deep Reinforcement Learning-Based Flow Scheduler for Hybrid Optical-Electrical Data Center Network, IEEE Access 7 (2019) 129955–129965.

[20] M. Kim, S. Lee, J. Lim, J. Choi, S.G. Kang, Unexpected Collision Avoidance Driving Strategy Using Deep Reinforcement Learning, IEEE Access 8 (2020) 17243–17252.

[21] J. Xie, Z. Shao, Y. Li, Y. Guan, J. Tan, Deep reinforcement learning with optimized reward functions for robotic trajectory planning, IEEE Access 7 (2019) 105669–105679.

[22] J.N. Tsitsiklis, B. Van Roy, An analysis of temporal-difference learning with function approximation, IEEE Transactions on Automatic Control 42 (5) (May 1997) 674–690.

[23] D.P. Bertsekas, V.S. Borkar, A. Nedic, Improved temporal difference methods with linear function approximation, Learning and Approximate Dynamic Programming, 2004, pp. 231–255.

[24] W.T. Miller, F.H. Glanz, L.G. Kraft, Cmas: An Associative Neural Network Alternative to Backpropagation, Proceedings of the IEEE 78 (10) (1990) 1561–1567.

[25] S. Haykin, "Neural Networks: A Comprehensive Foundation," Prentice Hall PTR, 1994..

[26] I. Menache, S. Mannor, N. Shimkin, Basis Function Adaptation in Temporal Difference Reinforcement Learning, Annals of Operations Research 134 (1) (2005) 215–238.

[27] A. d. M.S. Barreto and C.W. Anderson, "Restricted Gradient-descent Algorithm for Value-function Approximation in Reinforcement Learning, Artificial Intelligence, vol. 172, no. 4-5, pp. 454-482, 2008..

[28] G.S. Babu, S. and Suresh, " Meta-cognitive neural network for classification problems in a sequential learning framework, Neurocomputing, 81, pp.86-96, 2012..

[29] R.M. Kretchmar and C.W. Anderson, "Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning, Proceedings of International Conference on Neural Networks (ICNN), 1997, pp. 834–837 vol 2..

[30] S.J. Gershman, C.D. Moore, M.T. Todd, K.A. Norman, P.B. Sederberg, The successor representation and temporal context, Neural Computation 24 (6) (2012) 1553–1568.

[31] I. Momennejad, E.M. Russek, J.H. Cheong, M.M. Botvinick, N.D. Daw, S.J. Gershman, The successor representation in human reinforcement learning, Nature Human Behaviour 1 (9) (2017) 680–692.

[32] E.M. Russek, I. Momennejad, M.M. Botvinick, S.J. Gershman, N.D. Daw, Predictive representations can link model-based reinforcement learning to model-free mechanisms, PLoS Computational Biology 13 (9) (2017) e1005768.

[33] R.S. Sutton, A.G. Barto, F. Bach, et al., Reinforcement Learning: An Introduction, MIT Press, 1998.

[34] M. Geist, O. Pietquin, Kalman Temporal Differences, Journal of Artificial Intelligence Research 39 (2010) 483–532.

[35] L. Lehnert, M.L. and Littman, " Successor features combine elements of model-free and model-based reinforcement learning, Journal of Machine Learning Research, 21(196), pp.1-53, 2020..

[36] M.C. Machado, M.G. Bellemare, and M. Bowling, "Count-based exploration with the successor representation, Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 04, 2020, pp. 5125–5133..

[37] V. Mnih, K. Kavukcuoglu, D. Silver, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[38] S.C. Chan, S. Fishman, J. Canny, et al., "Measuring the reliability of reinforcement learning algorithms," International Conference on Learning Representations, 2020..

[39] J.P., Geerts, K.L. Stachenfeld, and N. Burgess, " Probabilistic successor representations with Kalman temporal differences, arXiv preprint arXiv:1910.02532, 2019..

[40] M. Salimibeni, P. Malekzadeh, A. Mohammadi, P. Spachos, K.N. Plataniotis, Makf-Sr: Multi-Agent Adaptive Kalman Filtering-Based Successor Representations, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 8037–8041.

[41] K. Doya, K. Samejima, K.-I. Katagiri, M. Kawato, Multiple Model-based Reinforcement Learning, Neural Computation 14 (6) (2002) 1347–1369.
[42] T. Kitao, M. Shirai, T. Miura, Model Selection based on Kalman Temporal Differences Learning, in: IEEE International Conference on Collaboration and Internet Computing (CIC), 2017, pp. 41–47.
[43] S. Akhlaghi, N. Zhou, Z. Huang, Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation, in: IEEE Power & Energy Society General Meeting, Chicago, IL, 2017, pp. 1–5.
[44] R. Mehra, On the Identification of Variances and Adaptive Kalman Filtering, IEEE Transactions on Automatic Control 15 (2) (1970) 175–184.
[45] D.G. Lainiotis, Partitioning: A Unifying Framework for Adaptive Systems, i: Estimation, Proceedings of the IEEE 64 (8) (1976) 1126–1143.
[46] A. Assa and K.N. Plataniotis, "Similarity-based Multiple Model Adaptive Estimation, IEEE Access, vol. 6, pp. 36 632–36 644, 2018..
[47] T. Michel, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences, in: Annual Conference on Artificial Intelligence, Springer, Berlin, Heidelberg, 2010, pp. 203–210.
[48] M. Hutter, S. Legg, Temporal Difference Updating without a Learning Rate, Advances in Neural Information Processing Systems (2008) 705–712.
[49] W. Xia, C. Di, H. Guo, S. Li, Reinforcement Learning Based Stochastic Shortest Path Finding in Wireless Sensor Networks, IEEE Access 7 (2019) 157807–157817.
[50] A. Barreto, R. Dabney, R. Munos, J.J. Hunt, T. Schaul, H.V. Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning, Advances in Neural Information Processing Systems, pp. 4055–4065, 2017..
[51] P. Malekzadeh, A. Mohammadi, M. Barbulescu, K.N. Plataniotis, STUPEFY: Set-Valued Box Particle Filtering for Bluetooth Low Energy-Based Indoor Localization, IEEE Signal Processing Letters 26 (12) (Dec. 2019) 1773–1777.
[52] A. Mohammadi and K.N. Plataniotis, "Event-Based Estimation With Information-Based Triggering and Adaptive Update, IEEE Transactions on Signal Processing, vol. 65, no. 18, pp. 4924–4939, 15 Sept. 2017..
[53] A. Mohammadi, K.N. Plataniotis, Improper Complex-Valued Bhattacharyya Distance, IEEE Transactions on Neural Networks and Learning Systems 27 (5) (May 2016) 1049–1064.
[54] A. Mohammadi, K.N. Plataniotis, Distributed Widely Linear Multiple-Model Adaptive Estimation, IEEE Trans Signal & Information Processing over Networks 1 (3) (Sept. 2015) 164–179.
[55] J.T. Barron, A General and Adaptive Robust Loss Function, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4326–4334.
[56] D. Choukroun, H. Weiss, I.Y. Bar-Itzhack, Y. Oshman, Kalman filtering for matrix estimation, IEEE Transactions on Aerospace and Electronic Systems 42 (1) (2006) 147–159.
[57] D.H. Nissen, A note on the variance of a matrix, Econometrica 36 (1968) 603–604.

Parvin Malekzadeh, received the B.Sc. degree in Electrical Engineering from Sharif University of Technology, Iran, in 2018, and MASc. in Electrical Engineering from Concordia University, Canada, in 2020. She is currently pursuing Ph.D. in the Department of Electrical and Computer Engineering at University of Toronto, Toronto, ON, Canada. Her research interests lie in the areas of signal processing, Machine Learning, Reinforcement Learning and Internet of Things.

Mohammad Salimibeni, received the B.Sc. degree in Electrical Engineering from the Isfahan University of Technology (IUT), in 2011 and the M.Sc. degree in Electrical Engineering from the same university (IUT) in 2014. He is currently pursuing the Ph.D. degree at Concordia University. His research interests include signal processing, Reinforcement Learning and distributed IoT-based networks with a particular interest in distributed ledger technology and Blockchain.

Ming Hou (M'05-SM'07) is currently a Senior Defence Scientist with Defence Research and Development Canada (DRDC) and the Principal Authority of Human-Technology Interactions with the Department of National Defence (DND), Canada. He is responsible for providing science-based advice at national and international levels to the Canadian Armed Forces (CAF) and coalition partners about the investment in and application of advanced technologies for human–machine systems requirements. He is an Integrator for the Canadian government 16 billion IDEaS program and one of the three Scientific Advisors to the Canadian National Centre of Expertise in Human Systems Performance with responsibilities for guiding national research and development activities in automation, robotics, and telepresence. He also gives advice for the development of National Defence AI Science and Technology Strategy and Roadmap to the CAF and DND. He is the Co-Chair of Human Factors Specialist Team within NATO Joint Capability Group on Unmanned Aircraft Systems (UAS). His book Intelligent Adaptive Systems: An Interaction-Centered Design Perspective became a guiding document to the development of NATO Standard Recommendations on UAS Human Systems Integration Guide book, UAS Human Factors Experimentation Guidelines and UAS Sense and Avoid Guidance. As one of the four invited lecturers, he delivers NATO Lecture Series on UAVs: Technological Challenges, Concepts of Operations, and Regulatory Issues. He also serves for multiple international associations/programs as a chair and a board members.

Arash Mohammadi (S'08-M'14-SM'17) is currently an Associate Professor with Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada. Prior to joining Concordia University and for 2 years, he was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Dr. Mohammadi is a registered professional engineer in Ontario. He is Director-Membership Developments of IEEE Signal Processing Society (SPS); General Co-Chair of "2021 IEEE International Conference on Autonomous Systems (ICAS)," and; Guest Editor for IEEE Signal Processing Magazine (SPM) Special Issue on "Signal Processing for Neurorehabilitation and Assistive Technologies". He was Co-Chair of "Symposium on Advanced Bio-Signal Processing and Machine Learning for Assistive and Neuro-Rehabilitation Systems" as part of 2019 IEEE GlobalSIP, and "Symposium on Advanced Bio-Signal Processing and Machine Learning for Medical Cyber-Physical Systems," as a part of IEEE GlobalSIP'18; The Organizing Chair of 2018 IEEE Signal Processing Society Video and Image Processing (VIP) Cup, and the Lead Guest Editor for IEEE Transactions on Signal & Information Processing over Networks Special Issue on "Distributed Signal Processing for Security and Privacy in Networked Cyber-Physical Systems". He is recipient of several distinguishing awards including the Eshrat Arjomandi Award for outstanding Ph.D. dissertation from Electrical Engineering and Computer Science Department, York University, in 2013; Concordia President's Excellence in Teaching Award in 2018, and; 2019 Gina Cody School of Engineering and Computer Science's Research and Teaching awards in the new scholar category.

Konstantinos N. Plataniotis (S'93-M'95-SM'03-F'12) is currently a Professor and the Bell Canada Chair in multimedia with the ECE Department, University of Toronto. He is also the Founder and the Inaugural Director-Research for the Identity, Privacy and Security Institute (IPSI), University of Toronto, where he was the Director for the Knowledge Media Design Institute (KMDI), from January 2010 to July 2012. He is a registered Professional Engineer in Ontario. Among his publications in these fields are the recent books WLAN Positioning Systems (2012) and Multi-Linear Subspace Learning: Reduction of Multidimensional Data (2013). His research interests are knowledge and digital media design,multimedia systems, biometrics, image and signal processing, communications systems, and pattern recognition. Dr. Plataniotis is a fellow of the Engineering Institute of Canada. He has served as the Editor-in-Chief for the IEEE SIGNAL PROCESSING LETTERS, and as the Technical Co-Chair for the IEEE 2013 International Conference in Acoustics, Speech and Signal Processing. He was the Vice President of the IEEE Signal Processing Society for Membership from 2014 to 2016. He was the General Co-Chair of the 2017 IEEE GlobalSIP, the 2018 IEEE International Conference on Image Processing (ICIP 2018), and is the General Co-Chair of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021).