MODEL-FREE MULTI-KERNEL LEARNING CONTROL FOR NONLINEAR DISCRETE-TIME SYSTEMS

Jiahang Liu,* Xin Xu,* Zhenhua Huang,* and Chuanqiang Lian**

Abstract

Reinforcement learning (RL) has become an important research topic to solve learning control problems of nonlinear dynamic systems. In RL, feature representation is a critical factor for improving the performance of online or offline learning controllers. Although multi-kernel learning has been studied in supervised learning problems, there is little work on multi-kernel-based feature representation in RL algorithms. In this paper, a model-free multi-kernel learning control (MMLC) approach is proposed for a class of nonlinear discrete-time systems. MMLC has advantages over other single-kernel-based RL algorithms in that the parameters in the kernel functions can be learned adaptively. Furthermore, MMLC uses a model-free actor–critic learning structure, where the critic is designed to approximate the derivatives of value functions. Different from the popularly studied dual heuristical programming algorithm, the proposed MMLC approach can learn the dynamics of the nonlinear system in a data-driven way. To evaluate the performance of MMLC, single-link and double-link inverted pendulums are employed as two benchmarks. The effectiveness of the MMLC algorithm has been demonstrated in simulation. It is shown that MMLC can achieve better performance than previous kernel-based dual heuristic programming with partial model information.

Key Words

Reinforcement learning, multiple kernel learning, approximate dynamic programming, model-free, nonlinear systems

1. Introduction

In general, machine learning algorithms can be categorized into three classes: supervised learning, unsupervised learning and reinforcement learning (RL). Different from supervised learning which uses labelled data, RL aims at learning an optimized action policy through the interaction with the environment. In the past decades, there have been many advances in the theory and practice of RL [1]–[6]. In RL, the formalism of Markov decision process (MDP) [7] has been a basic model to solve learning control problems where sequential decisions are made to maximize cumulative payoffs over uncertain outcomes. Classical RL algorithms including Q-learning [8] and Sarsa-learning always converge slowly with huge computational costs when the MDPs have large-scale or continuous state and action spaces in many practical applications. This phenomenon is considered as the "Curse of Dimensionality" which limits the application of RL algorithms.

To deal with this problem, approximate/adaptive dynamic programming (ADP) as a special family of RL techniques has received much attention in the machine learning and control engineering communities. One key idea of ADP is to use function approximation methods to realize generalization in large or continuous state and action spaces. RL and ADP methods with function approximation can be divided into three groups, including policy search [9], value function approximation (VFA) [10]–[13] and actor–critic methods [14]. In the scheme of actor–critic, the actor receives the signal from the critic for policy search and the critic receives the system states and the reward for VFA [15]. Typical online actor–critic methods are also called adaptive critic designs (ACDs) [16], which can be categorized as: heuristic dynamic programming, dual heuristic dynamic programming and globalized dual heuristic dynamic programming. In recent years, there have been various works [17]–[20] to approximate optimal control policies via ACDs. Meanwhile, the performance of ACDs heavily depends on the manual setup of the critic neural network. To deal with this problem, the study of kernel-based RL has been studied in the literature [21]–[23]. In [24], [25], a class of kernel-based ACDs were developed, which has the ability of automatically generating the feature representation for the critic. However, these single-kernel-based approaches still have some empirical parameters to be selected.

Multiple kernel learning (MKL) [26]–[28] has been an important research area for its two advantages compared with single kernel methods. Firstly, MKL can

* College of Mechatronics and Automation, National University of Defense Technology, Changsha, People's Republic of China; e-mail: liujiahang1992@foxmail.com, xuxin_mail@263.net, huangzhenhua_001@163.com

** Naval University of Engineering, Wuhan, People's Republic of China; e-mail: wzdslcq@163.com

Recommended by Dr. Francesco Pierri

(DOI: 10.2316/Journal.206.2017.5.206-5112)

automatically tune the weights of a set of chosen kernels rather than use the cross-validation procedure or do lots of experiments to determine the most suitable kernel. Secondly, the combination of multiple kernels can have better feature representation ability than a single kernel function. As different kernels have different measurement scales to the similarity of sources or modalities, MKL can be a method to combine many kinds of information sources [29]. The existing MKL algorithms can be divided into five major groups [26]: fixed rules, heuristic approaches, optimization approaches, Bayesian approaches and boosting approaches. In recent literature, MKL has been widely applied in many fields, such as visual object recognition [30], visual search [31], speaker verification [32], dimensionality reduction [33], structured prediction [34] and brain–computer interfacing [35].

In this paper, multiple kernel machines are integrated in RL and a model-free multi-kernel learning control (MMLC) algorithm is proposed for solving nonlinear control problems. There are two advantages of the proposed approach. One advantage is that MMLC is a data-driven RL approach so that an optimized control policy can be obtained without enough prior information. The second advantage is that multi-kernel methods are used to construct representative features for the VFA. MMLC can obtain the suitable parameter of kernel function automatically by combining a series of kernel functions which have different parameters with adaptive weights. Simulation results demonstrate that the proposed MMLC approach can avoid the process of complicated parameter setting and it can achieve better performance than the previous single kernel ACDs such as kernel-based dual heuristic programming with partial model information.

The rest of paper is organized as follows. In Section 2, brief details about MDPs and MKL are presented. In Section 3, the framework of multi-kernel actor–critic learning and the proposed MMLC algorithm are presented in detail. In Section 4, simulation results about the proposed algorithm are presented on the two benchmarks. Moreover, the final simulation result illustrates the superiority of the proposed algorithm compared with KDHP. The conclusion and future work are presented in Section 5.

2. Brief Introduction on MDPs and Kernel Machines

2.1 Markov Decision Processes

In RL, the whole system is usually modelled in the framework of MDP as a 4-tuple (S, A, P, R), where S is the state space, A is the action space, P is the state transition probability and R represents the reward or punishment from the environment. The policy π is a mapping from the state space to action space. The target is to approximate optimal policy π^* that satisfies

$$J_{\pi^*} = \max_{\pi} J_{\pi} = \max_{\pi} E^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{1}$$

where γ is a discount factor and r denotes the reward with respect to time step t, $E^{\pi}[\cdot]$ represents the expectation with respect to state transition probability and the policy π and J_{π} is the expected entire reward along the state trajectory with respect to the policy π.

The value function $V^{\pi}(s)$ under the policy π is defined as the expected total reward starting from the state s:

$$V^{\pi}(s) = E^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s\right] \tag{2}$$

The state value function $V^{\pi}(s)$ satisfies the following Bellman equation:

$$V^{\pi}(s) = E^{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right] \tag{3}$$

The optimal value function $V^*(s)$, maximizing the total γ-discount reward, is defined as follows:

$$V^*(s) = \max_{\pi} E^{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right] \tag{4}$$

Under the framework of MDP, the optimal policy can be described as follows:

$$\pi^*(s) = \operatorname*{argmax}_{a} E^{\pi}\left[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\right] \tag{5}$$

2.2 Kernel Machines

Based on the principle of Mercer kernel theory [36], the positive-definite kernel function known as the Mercer kernel is applied in the feature construction. The kernel matrix $K = [k(s_i, s_j)](1 \le i, j \le n)$ is a positive-definite matrix with any finite set of points $\{s_1, s_2, \ldots, s_n\}$. The kernel function can be represented as follows:

$$k(s_i, s_j) = < \phi(s_i), \phi(s_j) > \tag{6}$$

where $k(\cdot, \cdot)$ denotes the dot product in $\mathbb{H}$, and $\phi(\cdot)$ represents the feature vector that maps from state space S to a high-dimensional feature space F. Therefore, the similarity of two states can be easily measured through the dot product of $\phi(s_i)$ and $\phi(s_j)$. These are geometrically described in the dot product space also known as the feature space F.

It can be expected that multiple kernels achieve better performance than a single kernel for its better interpretability and generalization. So, the target multi-kernel function $K(s_i, s_j)$ is described as a convex combination of basis kernels:

$$\mathcal{K}(s_i, s_j) = \sum_{l=1}^{L} \eta_l k_l(s_i, s_j) s.t.\ \eta_l \geqslant 0,\ \sum_{l=1}^{L} \eta_l = 1 \tag{7}$$

where L is the number of kernels. The new parameter η is introduced as the weight of kernel which can be learned through online learning.

3. Model-Free Multi-Kernel Learning Control

As the MKL method is superior to single kernel methods in feature presentation, many kernel-based learning algorithms have been transformed into MKL in recent studies. The commonly used algorithms include multiple kernel support vector machine (SVM) [37] and multiple kernel support vector regression (SVR) [38]. In this part, we will make an introduction about the multi-kernel actor-critic learning method, especially for MMLC without precise model information. It is worth mentioning that the weights of the basis kernels can be updated automatically through the online learning process of MMLC.

3.1 The Overview of Model-Free Actor–Critic Learning Control

The structure of the proposed MMLC approach includes model identification, multi-kernel feature construction and actor–critic learning. The model is considered as a black box and estimated through model inputs and outputs. Different from traditional actor–critic learning methods, the multilayer perceptron neural network is replaced by a kernel-based linear approximator in the critic. The new linearized approximation structure is written as the formulation:

$$\mathbf{A}(\mathbf{s}) = \mathbf{B}^{\mathbf{T}}(\mathbf{s})\,\Omega \tag{8}$$

where Ω denotes the weights vector and $\mathbf{B}$ denotes the basis function. A convex combination of basis kernels (7) is chosen to be the basis function in MMLC.

In MDP, a reward function is defined as follows:

$$r(s,u) = s^T Q s + u^T R u \tag{9}$$

where s denotes state when the action u is performed. Q and R are both positive-definite matrices.

The process of model-free multi-kernel actor–critic learning can be summarized in three steps. (1) Model identification: the model information is obtained through hundreds of data collected randomly through the plant. In other words, these data are generated under the Gaussian distribution and then is taken as the input of the system, and the output needs to be observed as the samples in the state space simultaneously. The mapping from input to output can be approximated via the least-squares approach. (2) Kernel feature construction: the kernel dictionary is established from the collected samples by a sparsification method such as clustering and the approximately linear dependence (ALD) analysis method [22]. (3) Policy learning through actor–critic. Figure 1 shows the main structure of the model-free multi-kernel actor–critic learning.

3.2 The MMLC Algorithm

The whole algorithm procedure is described in Algorithm 1. In MMLC, $\mathbb{S}$ denotes the state space and kernel function denotes a mapping $\mathbb{S}\times\mathbb{S}\to\Re$. At first, the data need to be sampled randomly through the black box model, so that the state samples $\mathbb{D}=\{s_1,s_2,\ldots,s_n\}$ and control signal $U=(u_1,u_2,\ldots,u_n)$ are obtained. The kernel dictionary $\mathcal{D}=\{s_{d1},s_{d2},\ldots,s_{dm}\}$ is constructed from the samples $\mathbb{D}=\{s_1,s_2,\ldots,s_n\}$.

Consider the identified model being described by

$$y = W\Phi(x) \tag{10}$$

where

$$\Phi(x) = [\phi(x,x_{d1}),\phi(x,x_{d2}),\ldots,\phi(x,x_{dg})] \tag{11}$$

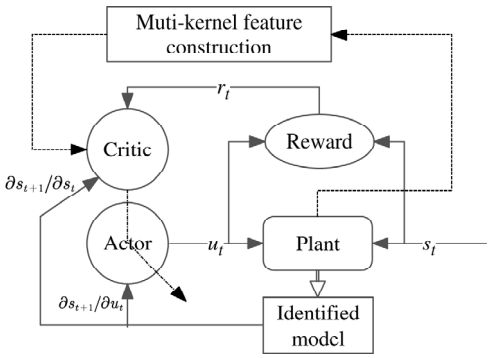

Figure 1. Main structure of the model-free multi-kernel actor–critic learning.

Algorithm 1 The MMLC Algorithm

Input: A series of basis kernels in multi-kernel function $\mathcal{K}(\cdot,\cdot)$, learning step size $\alpha(s_t)$ in actor, positive diagonal matrix Q and R, Gaussian kernel function $\phi(\cdot,\cdot)$, the discount factor γ and forgetting factor μ

1: Collect samples $\mathbb{D}=\{s_1,s_2,\ldots,s_n\}$ and $U=(u_1,u_2,\ldots,u_n)$ through the plant dynamics;
2: Generate the feature points $(x_{d1},x_{d2},\ldots,x_{dg})$ from the $Samples=(\mathbb{D},U)^T$;
3: Construct the least square model $y=W\Phi(x)$ and calculate the W using (12);
4: Use nearest neighbor algorithm to generate the kernel dictionary $\mathcal{D}=\{s_1,s_2,\ldots,s_m\}$;
5: Initialize η,Ξ,σ;
6: Let $t=1$;
7: **repeat**
8: Put current control policy $\hat{u}(s_t)$ and current state s_t into the plant and observe the next state s_{t+1} and calculate the reward $r(s_t,u(s_t))$ using (9);
9: Calculate $\hat{\lambda}(s_{t+1})$ through (22) and $u_d(s_t)$ through (35);
10: Update the weights vector $\sigma(s_{t+1})$ by (36) in actor;
11: Calculate $\mathcal{K}(s_t)$, $\mathcal{K}(s_{t+1})$ and $W\partial\Phi/\partial s_t$;
12: Update weights $\Xi(s_{t+1})$ by (30) in critic;
13: $t=t+1$;
14: **until** the convergence condition is reached
15: return $\Xi(s_t)$ and $\sigma(s_t)$

Output: The control policy

$y = s_{t+1}, x = (s_t, u(s_t))^T$ and $W = [w_1, w_2, \ldots, w_n]^T$ is the weight vector. In $\Phi(x)$, $(x_{d1}, x_{d2}, \ldots, x_{dg})$ are the feature points extracted from the $Samples = (\mathbb{D}, U)^T$ using a kernel sparsification method and $\phi(\cdot,\cdot)$ denotes a kernel function.

From (10), we have

$$W = \left(\Phi^T \Phi\right)^{-1} \Phi^T y \tag{12}$$

From the identified model (10), $\frac{\partial s_{t+1}}{\partial s_t}$ and $\frac{\partial s_{t+1}}{\partial u_t}$ can be easily written as follows:

$$\begin{aligned} \frac{\partial s_{t+1}}{\partial s_t} &= W \frac{\partial \Phi}{\partial s_t} \\ \frac{\partial s_{t+1}}{\partial u_t} &= W \frac{\partial \Phi}{\partial u_t} \end{aligned} \tag{13}$$

where

$$\frac{\partial \Phi}{\partial s_t} = \left(\frac{\partial \phi(x, x_{d1})}{\partial s_t}, \frac{\partial \phi(x, x_{d2})}{\partial s_t}, \ldots, \frac{\partial \phi(x, x_{dg})}{\partial s_t} \right) \tag{14}$$

$$\frac{\partial \Phi}{\partial u_t} = \left(\frac{\partial \phi(x, x_{d1})}{\partial u_t}, \frac{\partial \phi(x, x_{d2})}{\partial u_t}, \ldots, \frac{\partial \phi(x, x_{dg})}{\partial u_t} \right) \tag{15}$$

In Critic, the derivative of the value function is the approximated target that can be described as follows:

$$\lambda(s_t) = \frac{\partial V(s_t)}{\partial s_t} \tag{16}$$

So the optimality equation is written as follows:

$$\begin{aligned} \lambda^*(s_t) &= \frac{\partial V^*(s_t)}{\partial s_t} = \frac{\partial \left(r(s_t, u_t) + \gamma E^{\pi^*}\left[V^*(s_{t+1})\right] \right)}{\partial s_t} \\ &= 2Qs_t + \gamma E^{\pi^*}\left[\left\{ W \frac{\partial \Phi}{\partial s_t} \right\}^T \frac{\partial V^*(s_{t+1})}{\partial s_{t+1}} \right] + \left\{ \frac{\partial u^*(s_t)}{\partial s_t} \right\}^T \\ &\quad \times \left\{ 2Ru^*(s_t) + \gamma E^{\pi^*}\left[\left\{ \frac{\partial s_{t+1}}{\partial u^*(s_t)} \right\}^T \frac{\partial V^*(s_{t+1})}{\partial s_{t+1}} \right] \right\} \end{aligned} \tag{17}$$

In the Bellman equation (3), the optimal control policy u^* makes the state value function to be the minimum value, so (18) can be obtained

$$\frac{\partial V^*(s_t)}{\partial u^*(s_t)} = 2Ru^*(s_t) + \gamma \left[\left\{ \frac{\partial s_{t+1}}{\partial u^*(s_t)} \right\}^T \frac{\partial V^*(s_{t+1})}{\partial s_{t+1}} \right] = 0 \tag{18}$$

Then combining (17) and (18), the optimal derivative of the value function is written as follows:

$$\lambda^*(s_t) = 2Qs_t + \gamma E^{\pi^*}\left[\left\{ W \frac{\partial \Phi}{\partial s_t} \right\}^T \lambda^*(s_{t+1}) \right] \tag{19}$$

Therefore, when s_t is applied, the temporal difference error is described as follows:

$$\delta(s_t) = \hat{\lambda}(s_t) - \left[2Qs_t + \gamma E^{\pi(s_t)} \left[\left\{ W \frac{\partial \Phi}{\partial s_t} \right\}^T \hat{\lambda}(s_{t+1}) \right] \right] \tag{20}$$

where $\hat{\lambda}(s)$ is described in Critic as a linear approximation structure (8). Combining (7) and (8), we obtain

$$\begin{aligned} \hat{\lambda}(s_t) &= \sum_{i=1}^{m} \sum_{j=1}^{l} \eta_j k_j(s_t, s_i)\theta_i = \theta_1(\eta_1 k_1(s_t, s_1) \\ &+ \eta_2 k_2(s_t, s_1) + \cdots \eta_l k_l(s_t, s_1)) + \theta_2(\eta_1 k_1(s_t, s_2) \\ &+ \eta_2 k_2(s_t, s_2) + \cdots \eta_l k_l(s_t, s_2)) + \cdots + \theta_m(\eta_1 k_1(s_t, s_m) \\ &+ \eta_2 k_2(s_t, s_m) + \cdots \eta_l k_l(s_t, s_m)) \end{aligned} \tag{21}$$

where m, l, η and θ denote the dimension of dictionary vector, the number of kernel, the weights of different kernels and the weights of the linear approximation structure, respectively. And $k_j(s_t, s_i)$ is the basis kernel respect to the current state s_t and the i-th kernel dictionary.

Finally, the linear approximator in the critic is written as follows:

$$\hat{\lambda}(s_t) = \mathcal{K}^T(s_t)\, \Xi \tag{22}$$

where the multi-kernel function $\mathcal{K}$ and the weights Ξ can be rewritten as (23) and (24), respectively

$$\begin{aligned} \mathcal{K} = [&k_1(s_t, s_1), k_2(s_t, s_1), \ldots, k_l(s_t, s_1) \\ &k_1(s_t, s_2), k_2(s_t, s_2), \ldots, k_l(s_t, s_2) \\ &\vdots \\ &k_1(s_t, s_m), k_2(s_t, s_m), \ldots, k_l(s_t, s_m)]^T \end{aligned} \tag{23}$$

$$\begin{aligned} \Xi = [&\theta_1\eta_1, \theta_1\eta_2, \ldots, \theta_1\eta_l, \\ &\theta_2\eta_1, \theta_2\eta_2, \ldots, \theta_2\eta_l \\ &\vdots \\ &\theta_m\eta_1, \theta_m\eta_2, \ldots, \theta_m\eta_l]^T \end{aligned} \tag{24}$$

Compared with single kernel methods, the ability of feature learning is improved in multi-kernel methods, so MMLC is possible to achieve better performance. Combing (20) and (22), we get

$$\begin{aligned} \delta(s_t) = &\left[\mathcal{K}^T(s_t) - \gamma E^{\pi(s_t)} \left[\left\{ W \frac{\partial \Phi}{\partial s_t} \right\}^T \mathcal{K}^T(s_{t+1}) \right] \right] \\ &\Xi - 2Qs_t \end{aligned} \tag{25}$$

The learning target is to minimize the TD error. Assuming the error is 0, we can get

$$\left[\mathcal{K}^T(s_t) - \gamma E^{\pi(s_t)}\left[\left\{W\frac{\partial\Phi}{\partial s_t}\right\}^T \mathcal{K}^T(s_{t+1})\right]\right]\Xi = 2Qs_t \tag{26}$$

And multiplying $\mathcal{K}$ to both sides of (26), we get

$$\mathcal{K}(s_t)\left[\mathcal{K}^T(s_t) - \gamma E^{\pi(s_t)}\left[\left\{W\frac{\partial\Phi}{\partial s_t}\right\}^T \mathcal{K}^T(s_{t+1})\right]\right] \Xi = 2\mathcal{K}(s_t)Qs_t \tag{27}$$

So the kernel-based least squares fixed point solution to the TD (0) learning is described as follows:

$$\Xi = A_n^{-1}b_n \tag{28}$$

where A_n and b_n are expressed as follows:

$$\begin{aligned} A_n &= \mathcal{K}(s_t)\left[\mathcal{K}^T(s_t) - \gamma E^{\pi(s_t)}\left[\left\{W\frac{\partial\Phi}{\partial s_t}\right\}^T \mathcal{K}^T(s_{t+1})\right]\right] \\ b_n &= 2\mathcal{K}(s_t)Qs_t \end{aligned} \tag{29}$$

Based on the recursive least square temporal difference (RLS-TD) algorithm [39], the update rule in the critic is designed as follows:

$$\begin{aligned} \beta(s_{t+1}) &= \frac{P(s_t)\mathcal{K}(s_t)}{\mu + f_t P(s_t)\mathcal{K}(s_t)} \\ \Xi(s_{t+1}) &= \Xi(s_t) - \beta(s_{t+1})(2Qs_t - f_t\Xi(s_t)) \\ P(s_{t+1}) &= \frac{1}{\mu}\left[P(s_t) - \beta(s_{t+1})f_t P(s_t)\right] \end{aligned} \tag{30}$$

where

$$f_t = \mathcal{K}^T(s_t) - \gamma E^{\pi(s_t)}\left[\left\{W\frac{\partial\Phi}{\partial s_t}\right\}^T \mathcal{K}^T(s_{t+1})\right] \tag{31}$$

and β_t, μ denote the learning step size and the forgetting factor, respectively. $p_0 = \varsigma I$, where ς and I are positive real number and identity matrix, respectively.

At last, the output of the critic is represented by the multi-kernel function in (22).

In the actor, a neural network (NN) is used for approximating the optimal control policy. The NN structure is described as follows:

$$\hat{u}(s_t) = G(s_t, \sigma) \tag{32}$$

where σ denotes the weights vector.

From (18), the optimal control policy can be represented by

$$u^*(s_t) = -\frac{1}{2}\gamma R^{-1}E^{\pi(s_t)}\left[\left\{\frac{\partial s_{t+1}}{\partial u^*(s_t)}\right\}^T \lambda^*(s_{t+1})\right] \tag{33}$$

And the error is defined as

$$e(s_t) = \frac{1}{2}\left(\hat{u}(s_t) - u_d(s_t)\right)^2 \tag{34}$$

where

$$u_d(s_t) = -\frac{1}{2}\gamma R^{-1}E^{\pi(s_t)}\left[\left\{W\frac{\partial\Phi}{\partial\hat{u}(s_t)}\right\}^T \hat{\lambda}(s_{t+1})\right] \tag{35}$$

In (35), $\pi(s_t)$ is the control policy with respect to the state s_t, and $\hat{\lambda}(s_{t+1})$ represents the output of the critic.

The actor update rules are as follows:

$$\sigma(s_{t+1}) = \sigma(s_t) - \alpha(s_t)\left(\hat{u}(s_t) - u_d(s_t)\right)\frac{\partial\hat{u}(s_t)}{\partial\sigma(s_t)} \tag{36}$$

where $0 < \alpha(s_t) \leq 1$ denotes the learning step size and $\partial\hat{u}(s_t)/\partial\sigma(s_t)$ can be calculated from (32).

Under the framework of MKL, the MMLC algorithm is proposed as an improved approach of RL in two aspects. In one aspect, MMLC contains the function of model identification, being more valuable for practical applications. In the second aspect, the multi-kernel feature representation has stronger feature representation than single-kernel and also avoids many of the difficulties, such as the manual parameter setting. The advantage is that the algorithm takes the weights of a series of basis function and multi-kernel function in the critic into consideration together. In other words, we do not need to decide which kernel is the most significant and holds the heaviest weight, as the MMLC algorithm can determine the weights of kernel function through learning.

4. Simulation Studies

In this section, we will discuss online learning control problems to provide an assessment for the proposed MMLC algorithm. Recent years have witnessed some control approaches designed for robotic systems [40]–[42]. In control theory and system community, the inverted pendulum has been studied as a commonly used benchmark for a long time. So, a single-link inverted pendulum will be studied at first, and a double-link pendulum as a further study will be discussed in the next part. Recent studies of the inverted pendulum control problem have been carried out to improve the robustness under uncertainties and turbulence when faced with nonlinearity of the model, instability of the system and some other challenges [43].

In the application of the standard dual heuristic programming two neural networks were applied to the actor and the critic, also known as the actor neural network and the critic neural network. While in the kernel-based algorithm, the critic neural network is replaced with the linearized approximation structure as the formulation (22). For the single-link inverted pendulum, the actor neural network is a 3-layer neural network structure equipped with four nodes in the input layer, 10 nodes in the hidden layer and a node in the output layer. While in the double-link system, the actor neural network is equipped with a 6–10–1 neural network structure. The sigmoid function expressed as $f(x) = 1/(1+e^{-x})$ takes the role of activation function from the input layer to the hidden layer as well as the linear function described as $f(x) = x$ act on the transition from the hidden layer to the output layer. The learning rate is set to be 0.3 in the actor neural network.

The NN structure is randomly initialized in its weights that drop down into the range of $[-0.5, 0.5]$. In the kernel-based methods, the linearized approximator described as (8) is used in the critic. In KDHP, single Gaussian kernel described as $k(s, s_i) = \exp\left(-\|s - s_i\|^2/2\sigma^2\right)$ has been applied as the basis function in the critic and the parameter of kernel function σ is set to be some certain value. In MMLC, the multi-kernel function (7) is used as the basis function in the linearized approximator, and the important parameters include the number of kernel function and σ in every kernel function.

In the simulation, 5,000 pairs of states and actions were randomly collected. Not only are the kernel dictionaries in kernel learning, but also feature points in model identification generated from the samples by a sparsification algorithm. The nearest neighbor algorithm is used in the experiment. The Gaussian kernel $\phi(x, x_{di}) = \exp\left(-\|x - x_{di}\|^2/2\sigma^2\right)$ is chosen as the basis function in $\Phi(x)$. W can be calculated in (12). In the RLS-TD algorithm, the matrix P starts at initial value $0.1I$, I is a unit matrix and forgetting factor μ is set to be 1. A learning trial is defined as 10,000 time steps in learning control, and one run is made up of 500 trials. If the inverted pendulum system can be stabilized near the equilibrium position in one trial, the learned policy is considered as a successful policy. If the inverted pendulum system has not been stabilized near the equilibrium after 500 trials, all parameters of learning need to be reinitialized, and a new trial will start at an initial state.

4.1 Single-Link Inverted Pendulum

The dynamics of a single-link inverted pendulum is described as follows:

$$\begin{aligned} &(M+m)\ddot{x} + b\dot{x} - ml\ddot{\theta}\cos\theta + ml\dot{\theta}\sin\theta = F \\ &\left(I + ml^2\right)\ddot{\theta} + mgl\sin\theta = ml\ddot{x}\cos\theta \end{aligned} \tag{37}$$

where g, M, m, l, b and I denote the acceleration of gravity, the mass of the cart, the mass of the pole, the half-pole length, the friction function between the cart and the flat platform and the inertia of the pole in the single-link inverted pendulum, respectively. The difference equation of the single-link inverted pendulum with parameters being assigned with specific values is described as follows:

$$\begin{aligned} s_1(k+1) &= s_1(k) + \tau s_2(k) \\ s_2(k+1) &= s_2(k) + 3\tau\left(9.8(\sin s_1(k)) + u(k)\cos(s_1(k))\right) \\ s_3(k+1) &= s_3(k) + \tau s_4(k) \\ s_4(k+1) &= s_4(k) + \tau u(k) \end{aligned} \tag{38}$$

where $(s_1, s_2, s_3, s_4)^T = \left(\theta, \dot{\theta}, x, \dot{x}\right)^T$ denotes the state and u denotes the action. In the state space, $x, \dot{x}, \theta$ and $\dot{\theta}$ represent the cart position, cart velocity, pole angle and pole angle velocity, respectively. And action u is the expected force. The simulation time step τ is 0.02s. The boundaries of state variables should be in the following intervals:

$$\begin{aligned} \theta \in [-0.22\,\text{rad},\ 0.22\,\text{rad}], \dot{\theta} \in [-2\,\text{rad/s}, 2\,\text{rad/s}] \\ x \in [-0.5\,\text{m}, -0.5\,\text{m}], \dot{x} \in [-0.8\,\text{m/s}, 0.8\,\text{m/s}] \end{aligned} \tag{39}$$

In the meantime, the boundaries of action variables are:

$$u \in [-10\text{N}, 10\text{N}] \tag{40}$$

In this section, the dynamics is used for sampling the data. The reward is defined as (9) where $Q = diag\{1, 0.2, 1, 0.2\}$ and $R = 0.01$. For the online learning control, the learned policy is considered as a successful policy when the state space of inverted pendulum system can be stabilized near the equilibrium state and hold on at least 10,000 time steps. The discount factor γ is set to be 0.92. In online learning, the initial states are set to be $[0.05, -0.05, 0.02, 0.02]$.

It is worth mentioning that the parameter of kernel function has a significant influence not merely on the critic module, but also on the process and result of learning control. That is to say, the parameters of kernel function need to be tuned manually to adapt to the system model. When it comes to MMLC, the combination of multiple kernels replaces the single kernel Critic dealing with the problem of parameter selection to a great extent. In the initialization of MMLC, several kernel functions with different parameters have been applied in Critic. Then, the weights of multiple kernels can be determined in the learning process, which means that the parameter of kernel function will be adaptive in some known interval.

Figure 2 and Table 1 illustrate that different kernel parameters of KDHP have an obvious effect on the real-time online learning performance. Experimental results show that even small changes in kernel parameters can lead to large effects in learning performance. In Table 1, the RMSE of state and the RMSE of value function are calculated by the mean of RMSE of every state and the RMSE of the approximation of Critic, respectively. Furthermore, the average learning time is the statistical time for multiple runs. Therefore, if the KDHP algorithm needs to give a suitable policy after learning, hundreds of experiments need to settle down to determine which value are proper for the kernel parameter in the system. Nevertheless, the MMLC algorithm would not trap into the dilemma of KDHP. Within the framework of multiple kernels, the problem of choosing parameters in kernel function is transferred to the choice of the weights of basis kernels, and the weights converge to certain values after online actor–critic learning.

Figure 3 shows that the proper policy for the single-link inverted pendulum system can be quickly learned and MMLC has a better performance than KDHP compared with Fig. 2. In the simulation, the parameters of Gaussian basis kernels are set to be [1, 1.5], [1, 1.5, 2], [1, 1.5, 2, 2.5] and [1, 1.5, 2, 2.5, 3] for 2, 3, 4 and 5 kernels, respectively. Table 2 is drawn under the same parameters with Fig. 3. It shows that the selection of the number of the kernel in multi-kernel function is not significant for the performance of learning control and the selection of the parameters of Gaussian basis kernels is quite straightforward and casual. Figure 4 depicts online learning process of MMLC for single-link inverted pendulum from four different initial states. 4 basis kernels with parameters set to be [1,1.5,2,2.5] are combined into a multi-kernel function unit in this simulation. Four initial states $S_1(0), S_2(0), S_3(0), S_4(0)$ are set to be

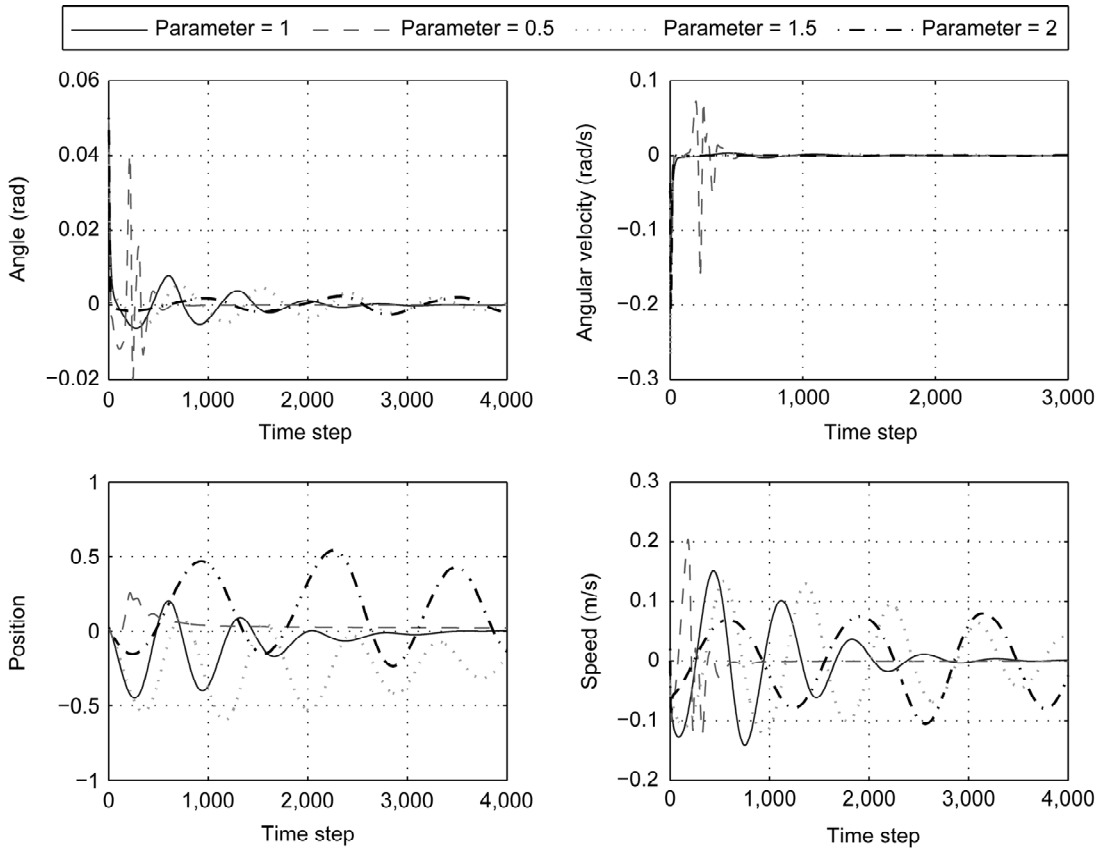

Figure 2. Online learning control process of KDHP for single-link inverted pendulum employing different Gaussian kernel parameters.

Table 1
RMSE of State and RMSE of Value Function of KDHP for Single-Link Inverted Pendulum with Different Gaussian Kernel Parameters

Parameter σ	0.5	1	1.5	2
RMSE of state	0.0177	0.0335	0.0388	0.0589
RMSE of value function	0.0121	0.0135	0.0140	0.0221
Average learning time (s)	217.2063	20.2016	9.1125	7.6531

$[0.05, -0.5, 0.02, 0.02]$, $[0.05, 0.05, 0.02, -0.4]$, $[0.05, 0.05, 0.02, 0.4]$ and $[0.05, 0.5, 0.02, 0.02]$, respectively. This result gives an assurance that MMLC algorithm can converges in a short time; namely, this policy also can quickly produce a controlled quantity after disturbance.

4.2 Double-Link Inverted Pendulum

To make a comprehensive comparison, the double-link inverted pendulum which is more complicated and more challenging is taken as a platform of further experiments.

The dynamics of a double-link inverted pendulum is described as follows:

$$\begin{bmatrix} M+m_1+m_2 & (0.5m_1l_1+m_2l_1)\cos\theta_1 & 0.5m_2l_2\cos\theta_2 \\ (0.5m_1l_1+m_2l_1)\cos\theta_1 & J_1+m_2l_1^2 & 0.5m_2l_1l_2\cos(\theta_1-\theta_2) \\ 0.5m_2l_2\cos\theta_2 & 0.5m_2l_1l_2\cos(\theta_1-\theta_2) & J_2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} b_0 & -(0.5m_1l_1+m_2l_1)\dot{\theta}_1\sin\theta_1 & 0.5m_2l_2\cos\theta_2 \\ 0.5m_1l_1\cos\theta_1 & b_1+b_2 & -b_2-0.5m_2l_1l_2\dot{\theta}_2\sin(\theta_1-\theta_2) \\ 0.5m_2l_2\cos\theta_2 & -b_2-0.5m_2l_1l_2\dot{\theta}_2\sin(\theta_1-\theta_2) & b_2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ -(0.5m_1+m_2)gl_1\sin\theta_1 \\ -0.5m_2gl_2\sin\theta_2 \end{bmatrix} = \begin{bmatrix} F \\ 0 \\ 0 \end{bmatrix} \tag{41}$$

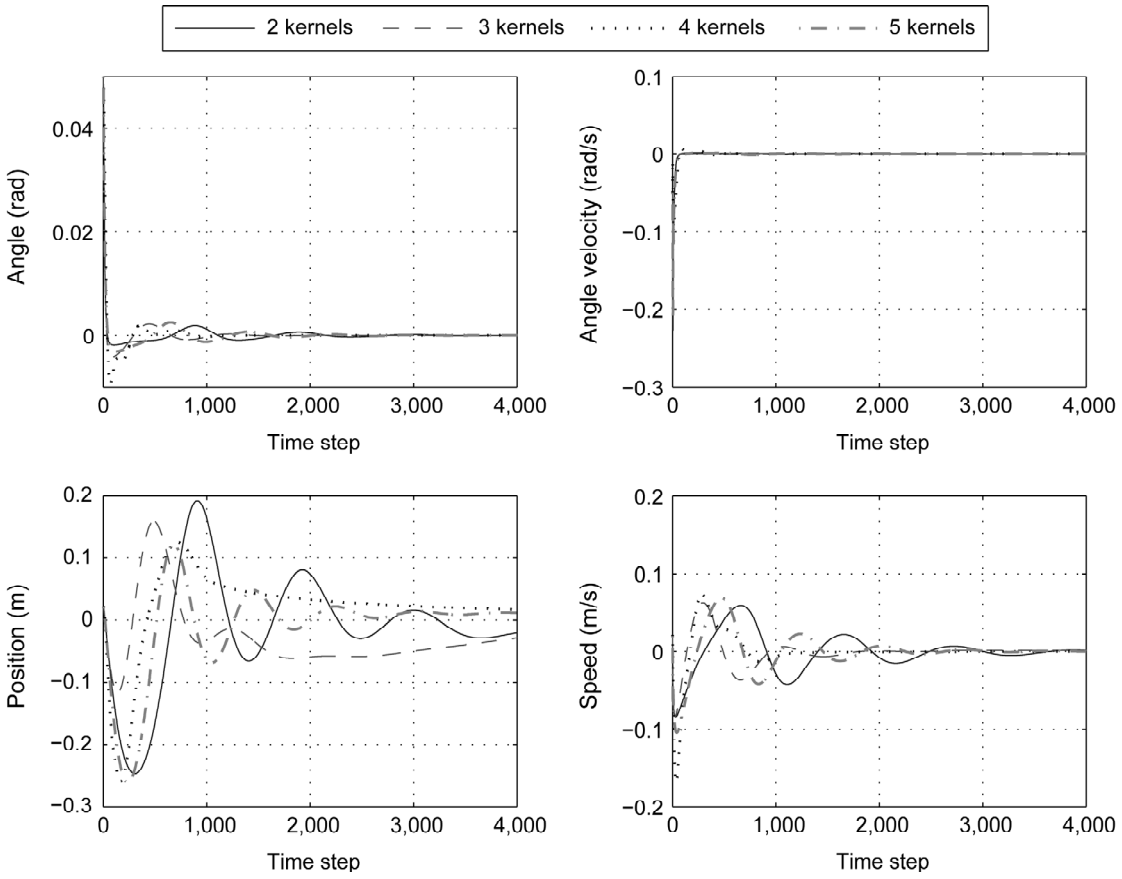

Figure 3. Online learning control process of MMLC for single-link inverted pendulum employing a different number of Gaussian kernel.

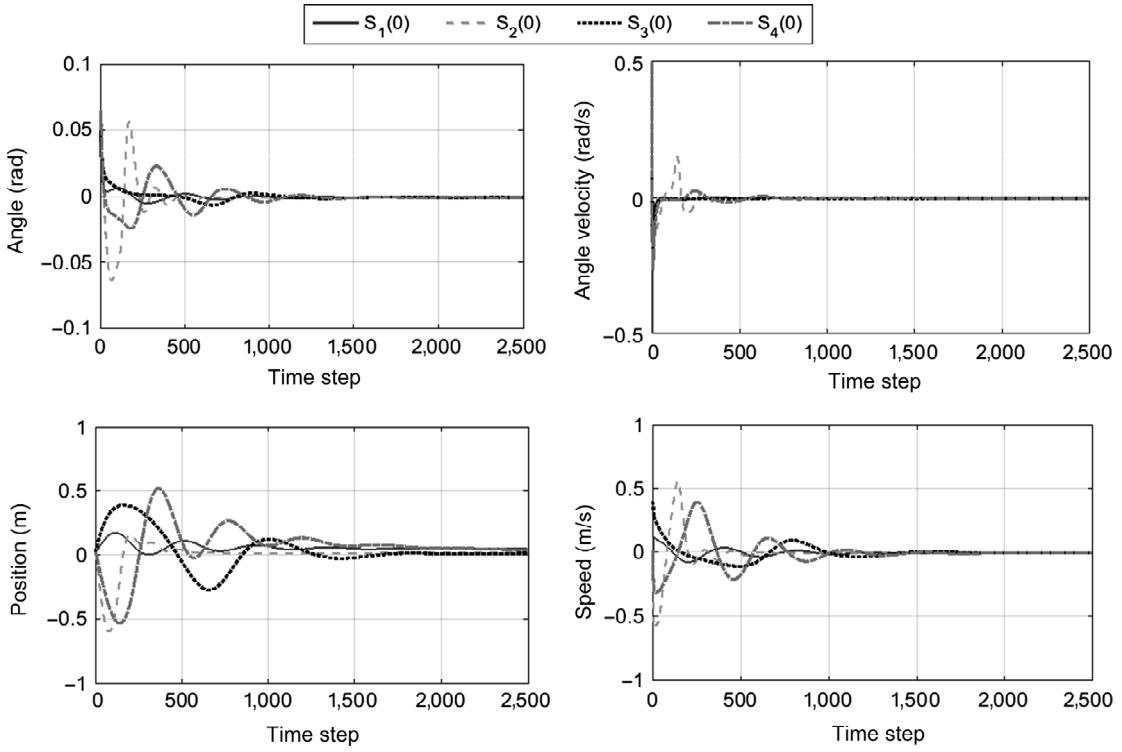

Figure 4. Online learning process of MMLC for single-link inverted pendulum from 4 different initial states.

Table 2
RMSE of State and RMSE of Value Function of MMLC for Single-Link Inverted Pendulum with a Different Number of Gaussian Kernel

Number of kernel	2	3	4	5
RMSE of state	0.0467	0.0312	0.0466	0.0315
RMSE of value function	0.0148	0.0099	0.0138	0.01
Average learning time (s)	15.2672	12.9109	15.9234	15.1969

where g is the acceleration of gravity, M denotes the mass of the cart, m_1 and m_2 denote the masses of the first and second pendulums, l_1 and l_2 denote the lengths of the first pendulum, J_1 and J_2 denote inertia of two pendulums respect to joints, b_1 and b_2 denote the friction coefficients on the joints and b_0 denotes the friction coefficient between cart and rail. Then, after parameters being assigned with specific values, the difference equation of double-link inverted pendulum is described as follows:

$$\begin{aligned}
s_1(k+1) &= x_1(k) + \tau s_4(k) \\
s_2(k+1) &= s_2(k) + \tau s_5(k) \\
s_3(k+1) &= s_3(k) + \tau s_6(k) \\
s_4(k+1) &= s_4(k) + \tau u(k) \\
s_5(k+1) &= s_5(k) + \tau(86.69 s_2(k) - 21.62 s_3(k) \\
&\quad + 6.64 u(k)) \\
s_6(k+1) &= s_6(k) + \tau(-40.31 s_2(k) + 39.45 s_3(k) \\
&\quad - 0.088 u(k))
\end{aligned} \tag{42}$$

where $(s_1, s_2, s_3, s_4, s_5, s_6)^T = \left(x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2\right)^T$ denotes the basis of state space and u denotes the action. In the state space, x, θ_1, θ_2, $\dot{x}$, $\dot{\theta}_1$ and $\dot{\theta}_2$ represent cart position, lower pole angle, upper pole angle, cart velocity, lower pole angle velocity and upper angle velocity, respectively. And action u is the expected force. The simulation time step τ is 0.02 s. The boundaries of state variables are in the following intervals:

$$\begin{aligned}
&x \in [-0.5\,\mathrm{m}, 0.5\,\mathrm{m}], \theta_1 \in [-0.22\,\mathrm{rad}, 0.22\,\mathrm{rad}], \\
&\quad \theta_2 \in [-0.22\,\mathrm{rad}, 0.22\,\mathrm{rad}] \\
&\dot{x} \in [-0.8\,\mathrm{m/s}, 0.8\,\mathrm{m/s}], \dot{\theta}_1 \in [-2\,\mathrm{rad/s}, 2\,\mathrm{rad/s}], \\
&\quad \dot{\theta}_2 \in [-2\,\mathrm{rad/s}, 2\,\mathrm{rad/s}]
\end{aligned} \tag{43}$$

Then, the boundaries of action variables can be described as follows:

$$u \in [-10\,\mathrm{N}, 10\,\mathrm{N}] \tag{44}$$

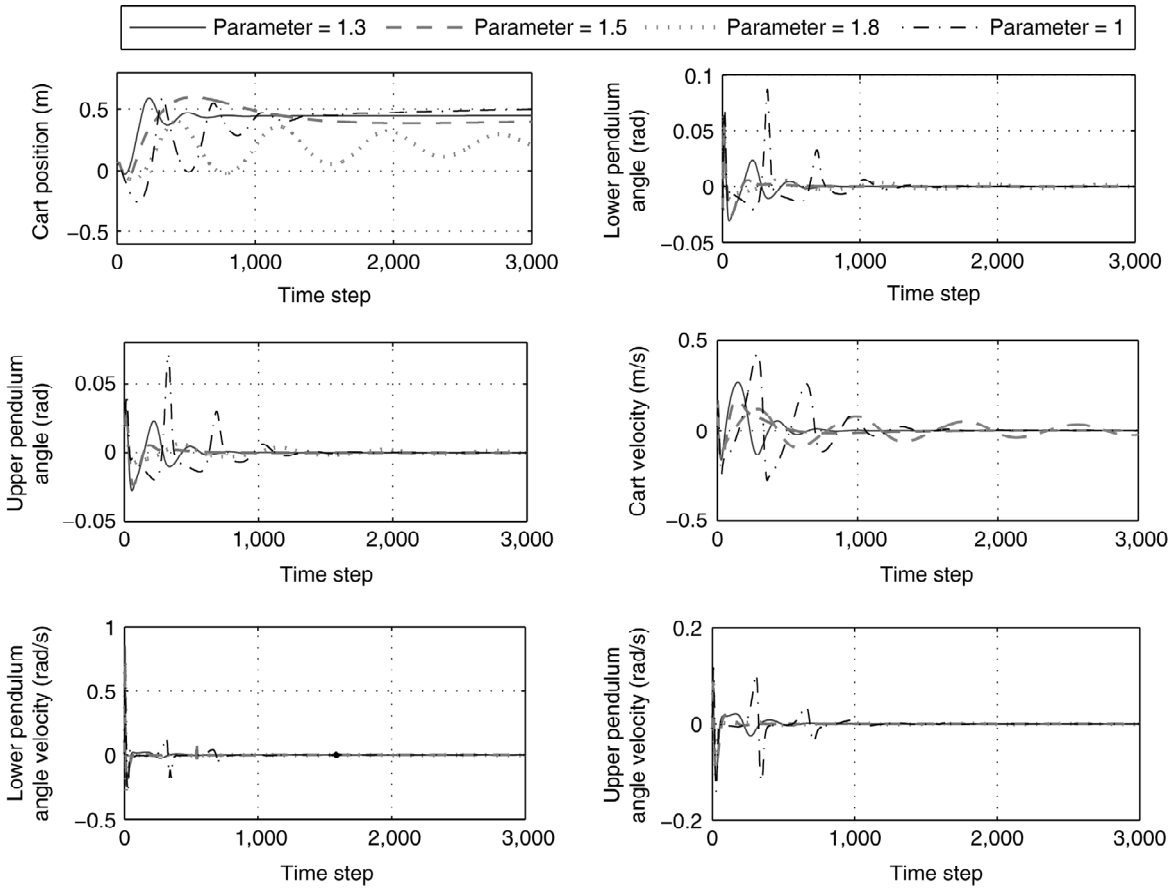

Figure 5. Online learning control process of KDHP for double-link inverted pendulum employing different Gaussian kernel parameters.

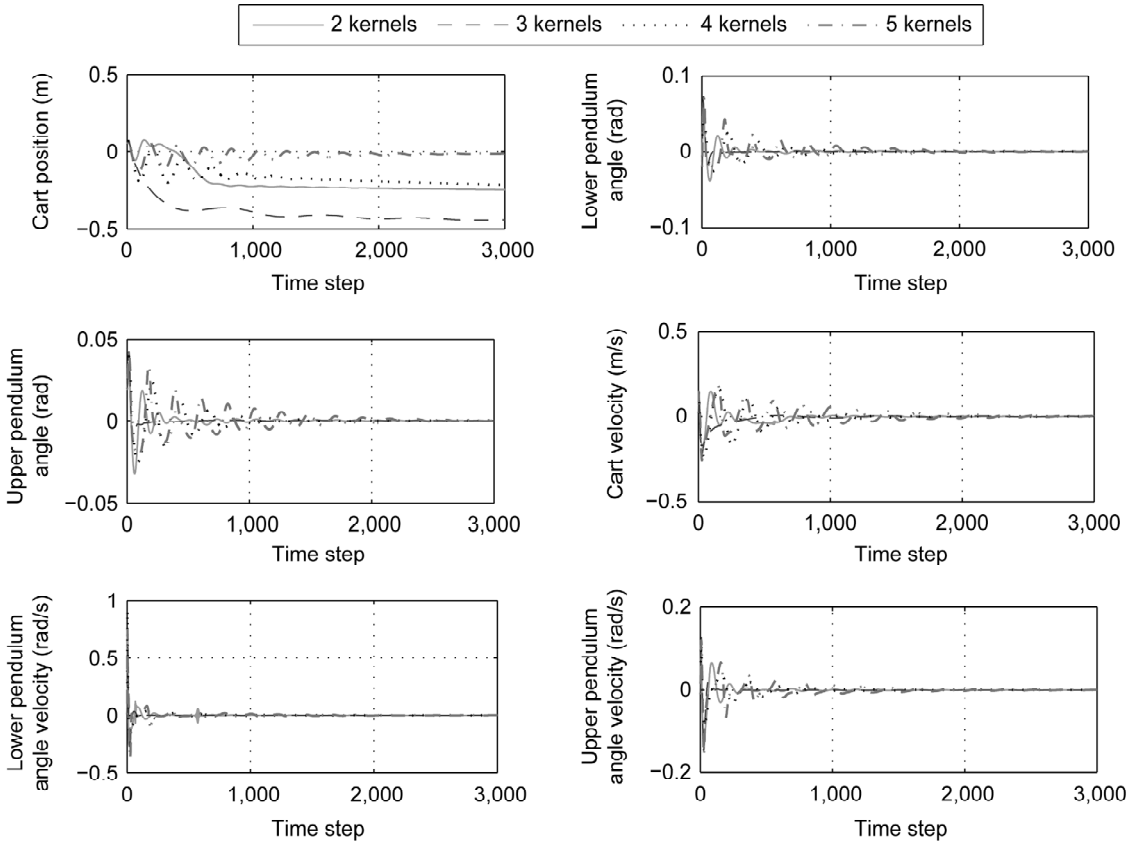

Figure 6. Online learning control process of MMLC for double-link inverted pendulum employing a different number of Gaussian kernel.

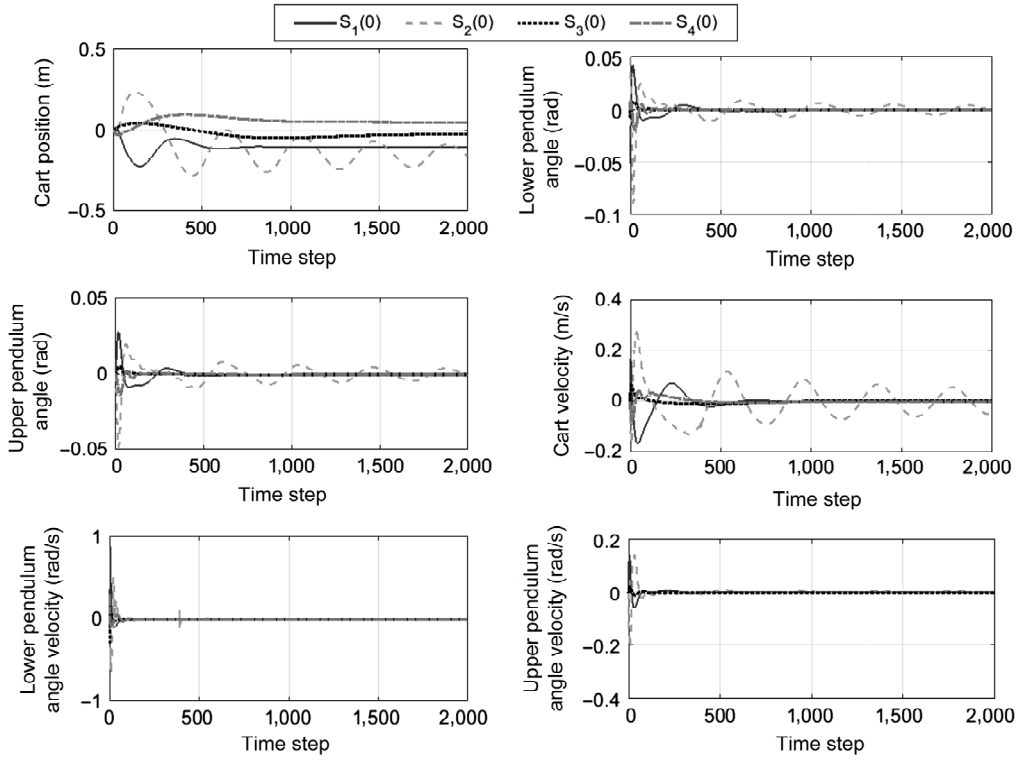

Figure 7. Online learning process of MMLC for double-link inverted pendulum from different initial states.

Table 3
RMSE of State and RMSE of Value Function of KDHP for Double-Link Inverted Pendulum with Different Gaussian Kernel Parameters

Parameter σ	0.9	1	1.1	1.2	1.3
RMSE of state	0.0222	0.0611	0.112	0.0678	0.0463
RMSE of value function	0.0454	0.0139	0.0249	0.0196	0.0174
Average learning time (s)	261.2344	154.1156	90.6625	176.1563	551.8021

Table 4
RMSE of State and RMSE of Value Function of MMLC for Double-Link Inverted Pendulum with a Different Number of Gaussian Kernel

Number of Kernel	2	3	4	5
RMSE of State	0.0669	0.0693	0.056	0.0526
RMSE of Value Function	0.0192	0.0195	0.015	0.0161
Average Learning Time(s)	170.1308	213.6658	133.7801	146.5846

The reward function is defined as (9) where $Q = diag\{0.03, 2, 2, 0.02, 0.02, 0.02\}$ and $R = 0.01$. The discount factor in objective function is set to be 0.9. In the online learning process, the initial states is set to be $[0.05, -0.02, 0.02, 0.02, 0.01, -0.01]$.

In Fig. 5, KDHP with different kernel parameters has been tested in the simulation. That is to say, KDHP has a great limit of choosing parameter in the kernel, and it is easy to fail to get a proper policy in the process. Table 3 is drawn to display the statistical data when the parameter is given as 0.9, 1, 1.1, 1.2, 1.3 with multiple runs. The performance of the KDHP is shown to be affected by the parameter. Figure 6 and Table 4 show the online learning performance of MMLC. The parameters of Gaussian basis kernels are set to be $[1, 1.5]$, $[1, 1.5, 2]$, $[1, 1.5, 2, 2.5]$ and $[1, 1.5, 2, 2.5, 3]$ for 2, 3, 4 and 5 kernels, respectively. From the comparison among Fig. 5 and Fig. 6 as well as Table 3 and Table 4, the selection of parameters in MMLC is easy and the online learning performance of MMLC without precise model information has comparable performance compared to the KDHP with known dynamics. This is because in the framework of MMLC, the algorithm parameters can be adaptively adjusted in the learning procedure. Under the multi-kernel approach, the interpretation of data has more diverse scales and more abundant information. In Fig. 7, the parameter setting in MMLC is similar with MMLC in Fig. 4. Four initial states $S_1(0), S_2(0), S_3(0), S_4(0)$ are set to be $[0, -0.05, 0, 0, 0, 0]$, $[0, 0.05, 0, 0, 0, 0]$, $[0, 0, 0, 0, -0.3, 0]$ and $[0, 0, 0, 0, 0.3, 0]$, respectively. Thus, this policy has enough ability to deal with any disturbance that does not make the states exceed the limits (43).

5. Conclusion and Future Work

In this paper, an MMLC approach is proposed for a class of nonlinear discrete-time systems. The multi-kernel approximation structure is designed in ACDs to choose appropriate kernel-based feature representations by tuning the weights of a series of basis kernels in the learning process. Different from single-kernel-based RL methods, the proposed MMLC approach can learn a near-optimal control policy without model information, and reduce the difficulty of parameter selection by using a more flexible multi-kernel structure. Further work will be conducted in two aspects. In one aspect, more characteristics of multi-kernel-based ACDs will be analysed. In the other aspect, multi-kernel based ADP algorithms will be applied and tested on some real-world problems.

Acknowledgement

The authors would like to thank the Associate Editor and anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grants U1564214 and 61611540348.

References

[1] R.S. Sutton and A.G. Barto, *Reinforcement learning: An introduction* (Cambridge, MA: MIT Press, 1998).

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

[3] F.L. Lewis and D. Vrabie, Reinforcement learning and adaptive dynamic programming for feedback control, *IEEE Circuits and Systems Magazine*, *9*(3), 2009, 32–50.

[4] F.Y. Wang, H. Zhang, and D. Liu, Adaptive dynamic programming: An introduction, *IEEE Computational Intelligence Magazine*, *4*(2), 2009, 39–47.

[5] M.L. Littman, Reinforcement learning improves behaviour from evaluative feedback, *Nature*, *521*(7553), 2015, 445–451.

[6] J. Ni, X. Li, M. Hua, and S.X. Yang, Bioinspired neural network-based q-learning approach for robot path planning in unknown environments, *International Journal of Robotics & Automation*, *31*(6), 2016. DOI: 10.2316/Journal.206.2016.6.206-4526

[7] M.L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming* (New York: Wiley, 2014).

[8] C.J. Watkins and P. Dayan, Q-learning, *Machine Learning*, *8*(3–4), 1992, 279–292.

[9] J. Baxter and P.L. Bartlett, Infinite-horizon policy-gradient estimation, *Journal of Artificial Intelligence Research*, *15*, 2001, 319–350.

[10] R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, Fast gradient-descent methods for temporal-difference learning with linear function approximation, in *Proc. 26th Annu. Int. Conf. on Machine Learning*, Montreal, Canada (New York: ACM, 2009), 993–1000.

[11] M.G. LAGouDAkis, Value function approximation, in C. Sammut and G.I. Webb (eds.) *Encyclopedia of Machine Learning* (Berlin: Springer, 2011), 1011–1021.

[12] M. Geist and O. Pietquin, Algorithmic survey of parametric value function approximation, *IEEE Transactions on Neural Networks and Learning Systems*, *24*(6), 2013, 845–867.

[13] Z. Huang, C. Lian, X. Xu, and J. Wang, Lateral control for autonomous land vehicles via dual heuristic programming, *International Journal of Robotics & Automation*, *31*(6), 2016. DOI: 10.2316/Journal.206.2016.6.206-4878

[14] V.R. Konda and J.N. Tsitsiklis, Actor–critic algorithms, *Proc. Neural Information Processing Systems*, vol. 13, Denver, America, 1999, 1008–1014.

[15] C.J. C.H. Watkins, *Learning from delayed rewards*, Ph.D. dissertation, University of Cambridge, England, 1989.

[16] A.G. Barto, D.A. White and D.A. Sofge, Reinforcement learning and adaptive critic methods, in D.A. White and D.A. Sofge (eds.), *Handbook of intelligent control*, vol. 469, 1992, 491.

[17] S. Bhasin, N. Sharma, P. Patre, and W. Dixon, Asymptotic tracking by a reinforcement learning-based adaptive critic controller, *Journal of Control Theory and Applications*, *9*(3), 2011, 400–409.

[18] K.G. Vamvoudakis and F.L. Lewis, Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem, *Automatica*, *46*(5), 2010, 878–888.

[19] H. Zhang, L. Cui, X. Zhang, and Y. Luo, Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method, *IEEE Transactions on Neural Networks*, *22*(12), 2011, 2226–2236.

[20] T. Dierks and S. Jagannathan, Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update, *IEEE Transactions on Neural Networks and Learning Systems*, *23*(7), 2012, 1118–1129.

[21] D. Ormoneit and Ś. Sen, Kernel-based reinforcement learning, *Machine Learning*, *49*(2–3), 2002, 161–178.

[22] X. Xu, D.W. Hu, and X.C. Lu, Kernel-based least-squares policy iteration for reinforcement learning, *IEEE Transactions on Neural Networks*, *18*(4), 2007, 973–992.

[23] J.C. Santamaría, R.S. Sutton, and A. Ram, Experiments with reinforcement learning in problems with continuous state and action spaces, *Adaptive Behavior*, *6*(2), 1997, 163–217.

[24] X. Xu, Z.S. Hou, C.Q. Lian, and H.B. He, Online learning control using adaptive critic designs with sparse kernel machines, *IEEE Transactions on Neural Networks and Learning System*, *24*(5), 2013, 762–775.

[25] X. Xu, C.Q. Lian, L. Zuo, and H.B. He, Kernel-based approximate dynamic programming for real-time online learning control: An experimental study, *IEEE Transactions on Control and Systems Technology*, *22*(1), 2014, 146–156.

[26] F.R. Bach, G.R. Lanckriet, and M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, *Proc. 21st Int. Conf. Machine Learning*, Banff, Alberta, Canada (New York: ACM, 2004), 6.

[27] M. Gönen and E. Alpaydın, Multiple kernel learning algorithms, *Journal of Machine Learning Research*, *12*(July), 2011, 2211–2268.

[28] G.R. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, and M.I. Jordan, Learning the kernel matrix with semidefinite programming, *Journal of Machine Learning Research*, *5*(January), 2004, 27–72.

[29] B. Schölkopf, K. Tsuda, and J.-P. Vert, *Kernel methods in computational biology* (Cambridge, MA: MIT Press, 2004).

[30] S.S. Bucak, R. Jin, and A.K. Jain, Multiple kernel learning for visual object recognition: A review, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(7), 2014, 1354–1369.

[31] H. Xia, S.C. Hoi, R. Jin, and P. Zhao, Online multiple kernel similarity learning for visual search, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(3), 2014, 536–549.

[32] C. Longworth and M.J. Gales, Multiple kernel learning for speaker verification, *2008 IEEE Int. Conf. Acoustics, Speech and Signal Processing* Las Vegas, America (Piscataway ,NJ: IEEE, 2008), 1581–1584.

[33] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, Multiple kernel learning for dimensionality reduction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(6), 2011, 1147–1160.

[34] A.F. Martins, M.A. Figueiredo, P.M. Aguiar, N.A. Smith, and E.P. Xing, Online multiple kernel learning for structured prediction, *arXiv preprint arXiv:1010.2770*, 2010.

[35] W. Samek, A. Binder, and K.-R. Müller, Multiple kernel learning for brain–computer interfacing, *2013 35th Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan (Piscataway, NJ: IEEE, 2013), 7048–7051.

[36] B. Schölkopf and A.J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond* (Cambridge, MA: MIT Press, 2002).

[37] M. Hu, Y. Chen, and J.T.-Y. Kwok, Building sparse multiple-kernel SVM classifiers, *IEEE Transactions on Neural Networks*, *20*(5), 2009, 827–839.

[38] Z.-Y. Chen and Z.-P. Fan, Dynamic customer lifetime value prediction using longitudinal data: An improved multiple kernel SVR approach, *Knowledge-Based Systems*, *43*, 2013, 123–134.

[39] J.A. Boyan, Technical update: Least-squares temporal difference learning, *Machine Learning*, *49*(2–3), 2002, 233–246.

[40] F. Piltan, N. Sulaiman, A. Gavahian, S. Soltani, and S. Roosta, Design mathematical tunable gain PID-like sliding mode fuzzy controller with minimum rule base, *International Journal of Robotics & Automation*, *2*(2), 2011, 146–156.

[41] H. Šiljak, Inverse matching-based mobile robot following algorithm using fuzzy logic, *International Journal of Robotics & Automation*, *29*(4), 2014, 369–377.

[42] S. Islam, P.X. Liu, and A.E. Saddik, Adaptive sliding mode control of unmanned four rotor flying vehicle, *International Journal of Robotics & Automation*, *30*(2), 2014, 140–148.

[43] S. Jadlovská and J. Sarnovskỳ, Classical double inverted pendulum A complex overview of a system, *2012 IEEE 10th Int. Symp. on Applied Machine Intelligence and Informatics (SAMI)*. (Herl'any, Slovakia: IEEE, 2012), 103–108.

Biographies

Jiahang Liu received the Bachelor's degree in automation from the College of Information Science and Engineering, Northeastern University, Shenyang, China, in 2015. Now, he is pursuing the Master's degree in control science and engineering with the College of Mechatronics and Automation, National University of Defense Technology, Changsha, China.

Xin Xu received the B.S. degree in electrical engineering from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, China, in 1996, where he received the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation, NUDT, in 2002. He has been a visiting professor in Hong Kong Polytechnic University, University of Alberta, University of Guelph, and the University of Strathclyde, UK, respectively. He is currently a Professor with the College of Mechatronics and Automation, NUDT, China. He has co-authored more than 140 papers in international journals and conferences, and co-authored four books. His research interests include intelligent control, reinforcement learning, approximate dynamic programming, machine learning, robotics and autonomous vehicles. He received the Fork Ying Tong Youth Teacher Fund of China in 2008 and the Second-class National Natural Science Award of China, in 2012. He serves as the co-Editor-in-Chief of *Journal of Intelligent Learning Systems and Applications* and the Associate Editor-in-Chief of *CAAI Transactions on Intelligent Technology* (Elsevier). He is an Associate Editor of *Information Sciences*, *Intelligent Automation and Soft Computing* and *Acta Automatica Sinica*. He was a Guest Editor of the *International Journal of Adaptive Control and Signal Processing* and *Mathematical Problems in Engineering*. He is a Senior Member of IEEE and a member of the IEEE CIS Technical Committee on Approximate Dynamic Programming and Reinforcement Learning (ADPRL) and the IEEE RAS Technical Committee on Robot Learning.

Zhenhua Huang received the Bachelor's degree in vehicle engineering from Changsha University of Science and Technology, in 2008, and the Master's degree from the College of Mechatronics and Automation (CMA), National University of Defense Technology (NUDT), P.R. China, in 2010. He is currently a Ph.D. student at the Institute of Unmanned Systems, CMA, NUDT, P.R. China. His current research interests include machine learning, robot control, and autonomous land vehicles. He has co-authored 11 papers in international journals and conferences.

Chuanqiang Lian received the Bachelor's degree in automation from the Department of Automation, Tsinghua University, Beijing, China, in 2008, and the Master's degree in control science and engineering from the College of Mechatronics and Automation, National University of Defense Technology, Changsha, China, in 2010, where he received the Ph.D. degree in control science and engineering with the Institute of Unmanned Systems in 2016. He has authored or co-authored more than 10 papers in international journals and conferences. His current research interests include reinforcement learning, approximate dynamic programming and autonomous vehicles.