Smooth Deep Reinforcement Learning for Power Control for Spectrum Sharing in Cognitive Radios

Wanli Wang
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hong Kong
1155160517@link.cuhk.edu.hk

Abstract

The spectrum sharing in a cognitive radio system is related with the power transmit of the secondary user by sharing common spectrum with the primary user without harmful inference. The reinforcement learning has been considered as an intelligent power control method with the agent interacting with the environment. Traditional deep reinforcement learning considers a deep neural network for learning a nonlinear map from the state or observation to the Q-value function. The observations in the radio system are collected from wireless network and corrupted by noises with varying magnitude. The deep neural network may therefore yield undesirable result due to the presence of noises and induced degraded network weights. Considering that the kernel-based adaptive filter is beneficial for dealing with the corruptive signal, we aim to apply the kernel-based adaptive filter into the traditional deep reinforcement learning for further smoothing network outputs, yielding smooth deep reinforcement learning (SDRL). Simulation results have shown the efficiency of the proposed smooth deep reinforcement learning in spectrum sharing in cognitive radios in comparison with traditional deep reinforcement learning.

1 Introduction

There is an urgent need to enhance the spectrum efficiency caused by increasing demand for spectrum resources or low utilization rate of some bands in the cognitive radio system. For example, the utilization rate of some bands is only as low as 15% as shown in [1], which motivates the study of spectrum sharing through cognitive radios [2]. Consider a cognitive radio system consisting of the primary user and secondary network [3,4]. A passive primary user model and an active primary user model are common operation of spectrum sharing [5]. For the passive primary model, the secondary user performs spectrum sensing to explore idle spectrum and there is no need for the primary user updating its transmission parameters. In the active model, however, a dynamic power control strategy is required for all users in the network such that a minimum quality of service (QoS) for successful data transmission is fulfilled for both primary and secondary users, which is considered in our work.

Apart from solving the power control issue from an optimization perspective [6, 7], the reinforcement learning [8, 9, 10] as a subfield of machine learning [11] has gained popularity and applied in intelligent power control due to its theoretical and technical achievements in efficiency and generalization. The reinforcement learning aims for behavioural decision making via interacting with the real world and receiving reward feedback. The reinforcement learning problem is in general modeled as a Markov decision process (MDP) which is defined over the state transition probabilities and rewards depend only on the state of the environment and the action taken by the agent [12]. The key point of reinforcement learning is about learning an optimal policy mapping from the states to

the actions so that the accumulated reward in the future is maximized. More concretely, the agent in reinforcement learning chooses an action on the basis of its current state and then receives feedback and reached new state from the environment.

Common reinforcement learning algorithms are categorized into model-based and model-free methods according to whether a complete knowledge of the MDP model can be specified a priori [13]. Model-based methods, also referred to as planning methods, require a complete description of the model in terms of the transition and reward functions, while model-free methods, also referred to as learning methods, learn an optimal policy simply based on received observations and rewards. The model-based techniques aim for finding an optimal policy given a complete description of an MDP model [14, 15] and include two main different approaches, i.e., value iteration and policy iteration [16]. However, the mathematic model is not always tractable in most applications. For solving this issue, model-free methods are developed for learning optimal policy via interacting with environment. Commonly used mode-free approaches include Monte Carlo [13], Temporal difference (TD) [13], SARSA [13, 17] and Q-learning [18, 19]. In particular, Q-learning is one of the most popular model-free approach updating the Q-values in the form of a table due to its simplicity and efficiency. However, it is not feasible for using a look-up table for listing the expected Q-values for a pair of state and action with large number of states and actions. A deep neural network is beneficial for dealing with issue where the network parameter is optimized via nonlinear function of target Q-value and network output, yielding deep Q-network reinforcement learning [20, 21, 22].

The deep Q-network reinforcement learning indeed behaves high efficiency in finding optimal policy but suffers from Gaussian noises in the process of updating the weight parameters of deep network, may yielding degraded network output. This motivates us to combine the deep neural network with the kernel-based adaptive filter [23]. Kernel methods aim to solve nonlinear problems in a linear form in the reproduce Kernel Hilbert space (RKHS) [24, 25, 26]. Commonly used kernel adaptive filters (KAFs) include the kernel least mean square algorithm (KLMS) [27], kernel affine projection algorithm (KAPA) [28] and kernel recursive least squares (KRLS) [29]. Among these algorithms, the KLMS performs desired filtering performance with the lowest computational cost which is considered in our work. A novel smooth reinforcement learning is proposed by incorporating the KLMS algorithm into the deep reinforcement learning for further smoothing the output produced by the deep neural network. In smooth deep reinforcement learning, the kernel-based method works with the deep neural network, beneficial for yielding desired network output in comparison with traditional deep reinforcement learning. Simulation results have shown the efficiency of the proposed smooth deep neural network in the application of spectrum sharing in cognitive radio.

The remaining part of the paper proceeds as follows. Section 2 gives a brief overview of spectrum sharing in the cognitive radio system. Section 3 describes a traditional deep reinforcement learning. Section 4 proposes a novel smooth deep reinforcement learning. Section 5 investigates the efficiency of the proposed smooth deep reinforcement learning in spectrum sharing in the cognitive radio system. Section 6 gives a conclusion.

2 Review of Spectrum Sharing in Cognitive Radio Networks

Consider a cognitive radio network which is comprised of a primary user and a secondary user, respectively. The secondary user is expected to share a common spectrum resource with the primary user without inducing harmful inference to the primary user. The primary user adjusts its transmit power only on the basis of its own power control policy. Assume that there is no communication between the primary user and secondary user. This means that the secondary user has no knowledge about the power control policy of the primary user.

An intelligent secondary user is expected to learn power control policy so that primary and secondary users are all capable of transmitting data with required Qos. The measurements related to the primary user and secondary user like the received signal strength (RSS) are collected from numerous sensors at different locations in the wireless environment. Define the transmit power of the primary and secondary user by p_1 and p_2, respectively. The Qos is measured in terms of the SINR for the primary user or secondary user receivers which is defined by

$$\text{SINR}_i = \frac{|h_{ii}|^2 p_i}{\sum\limits_{j \neq i} |h_{ji}|^2 p_j + \sigma_i}, i = 1, 2 \tag{1}$$

with the noise power σ_i at the receiver Rx_i and channel gain h_{ij} from the transmitter Tx_i to the receiver Rx_j. Consider a minimum SINR requirement for reception, i.e., $\text{SINR}_i \geq \eta_i, i = 1, 2$.

For the primary user, the power is adaptively adjusted by its own power control policy in consideration of the Qos requirement. Two common power control strategies are considered. In [30], the transmit power is updated as follows:

$$p_1(k+1) = D\left(\frac{\eta_1 p_1(k)}{\text{SINR}_1(k)}\right) \tag{2}$$

where $\text{SINR}_1(k)$ and $p_1(k)$ represent the SINR measured at the primary receiver and transmit power st the k-th time frame. The notation $D(\cdot)$ is used for discretizing continuous values into following discrete values

$$P_1 = \{p_1^p, \cdots, p_{L_1}^p\}, p_1^p \leq \cdots \leq p_{L_1}^p. \tag{3}$$

In addition, $D(x)$ is equal to the nearest discrete level no less than x. Assume the transmit power at the k-th time frame as $p_1(k) = p_j^p \in P_1$. The second strategy in contrast updates the transmit power by the following piece-wise function.

$$p_1(k+1) = \begin{cases} p_{j+1}^p & \text{if } p_j^p \leq \tau \leq p_{j+1}^p \text{ and } j+1 \leq L_1 \\ p_{j-1}^p & \text{if } \tau \leq p_{j-1}^p \text{ and } j-1 \geq 1 \\ p_j^p & \text{otherwise} \end{cases} \tag{4}$$

where $\tau = \eta_1 p_1(k)/\text{SINR}_1(k)$.

Denote the path loss between the primary transmitter, secondary transmitter and sensor n by g_{1n} and g_{2n}, formulated as

$$g_{1n} = \left(\frac{\lambda}{4\pi d_{1n}}\right), g_{2n} = \left(\frac{\lambda}{4\pi d_{2n}}\right) \tag{5}$$

with the signal wavelength λ and d_{1n}/d_{2n} representing the distance between the primary/secondary transmitter and node n. Consider the following model for simulating the RSS measurements.

$$P_n^r(k) = p_1(k)g_{1n} + p_2(k)g_{2n} + w_n(k) \tag{6}$$

where $p_1(k)$ and $p_2(k)$ denote the transmit power of the primary and secondary user, respectively. Notation $w_n(k)$ represents a zero mean Gaussian noise with variance δ_n^2.

The secondary user takes the transmit power from the following finite set

$$P_2 = \{p_1^s, \cdots, p_{L_2}^s\}, p_1^s \leq \cdots \leq p_{L_2}^s. \tag{7}$$

The secondary user is expected to learn adjust its own transmit power on the basis of available RSS information $\{P_n^r(k)\}_n$ so that both primary and secondary users meet their Qos requirements for power transmission.

3 Review of Deep Reinforcement Learning

Reinforcement learning is in fact described as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ consisting of the agent's state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, $\mathcal{P}$ describing the transition density $p(s^{'}|s, a)$ taking the action a from the current s to the next state $s^{'}$, $\mathcal{R}$ describing the instantaneous reward $r(s, a)$ and the discount factor γ. Since the mathematical model described by $\mathcal{P}$ is not always tractable in practice, the quadruples $(s, a, r, s^{'})$ in general characterizes one interaction between agent and environment. More concretely, the policy π provides a guideline for allowing the agent to choose an action given a state s. The agent then returns the reward r as the feedback from the environment and updates the state from s to $s^{'}$.

There is no doubt that the key of reinforcement learning is about the design of the optimal policy mapping the state s to the expected action a. The expected long term reward given current state and action also called Q-value is expected to be maximized in the framework of the reinforcement

learning. Consider the Q-value given state s, action a and implicit policy π starting from the discrete time t [31].

$$Q_\pi(s,a) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right] \tag{8}$$

with the expectation operator $E(\cdot)$. Notation r_{t+k+1} represents the instantaneous reward at the discrete time $t+k+1$. The optimal Q-value is defined by $Q^*(s,a) = \max_\pi Q_\pi(s,a)$ and fulfills the following Bellman equation [32]

$$Q^*(s,a) = E\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a')|s_t = s, a_t = a\right] \tag{9}$$

Commonly used Q-learning updates the Q-values in the form of Q-table as follows:

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s', a') - Q(s,a)\right) \tag{10}$$

However, it is not feasible for using a look-up table for listing the expected Q-values for a pair of state and action with large number of states and actions. The loop-up table can be replaced by a deep neural network(DNN) parameterized by θ, i.e., $Q(s,a;\theta)$ which is called deep Q network (DQN). The DQN aims to optimize the parameter θ by minimizing the output of the DNN and target, i.e., [30]

$$L(\theta) = \frac{1}{|\Omega_k|}\sum_{i\in\Omega_k}\left(Q_{ta}(i) - Q(s(i), a(i); \theta)\right)^2 \tag{11}$$

where $|\Omega_k|$ represents the index set of the random minibatch used at the kth iteration and target value $Q_{ta}(i)$ defined by

$$Q_{ta}(i) = r(i) + \gamma Q^m(i), \forall i \in \Omega_k \tag{12}$$

where $Q^m(i) = \max_{a'} Q(s'(i+1), a'; \theta)$.

The deep neural network as an off-policy deep reinforcement learning learns the optimal policy once a batch size of samples is randomly chosen from the experience pool.

4 Smooth Deep Reinforcement Learning

In deep reinforcement learning, the target Q-value and estimated Q-value are calculated by the deep neural network directly. However, the neural network may produce undesirable output due to the presence of noises with different magnitude. In the proposed smooth deep reinforcement learning, we aim to incorporate kernel method-based adaptive filter for smoothing the output of neural network. The kernel least mean square algorithm is a commonly used kernel adaptive filter due to its simplicity and desired filtering precision, which is considered in our work.

As shown in (11), a batch of samples with size $|\Omega_k|$ is sampled randomly from the experience pool for training deep neural network. Meanwhile, each sample is equipped with a KLMS filter for further smoothing the undesired output of the neural network caused by noises. As a result, the target Q-value in (11) is reformulated as

$$Q_{ta} = r + \gamma \max_{a'}(\rho Q(s', a'; \theta) + (1-\rho)Q_{klms}(s', a'; \theta')) \tag{13}$$

where notation $Q_{klms}(s', a'; \theta'))$ represents the estimated Q-values generated by the adaptive kernel least mean square. The kernel least mean square algorithm is a supervised machine learning and applied for smoothing the output of the deep neural network. This means that desired targets are also required for the kernel least mean square as that of the deep neural network. Therefore, it is reasonable for the kernel least mean square for constructing the desired target or label for yielding weight parameter. Considering that the dimension of the weight parameter of the KLMS grows linearly with the discrete time, this may be not practical for the KLMS retaining the past weight

parameters for yielding its own desired target as in (12). One way to solve this issue is about using the desired target yielded by the neural network is efficient. Consider the training set including target generated by the deep neural network and weight parameter of the KLMS, the KLMS learns the policy repeated once a batch size of samples is chosen from the experience pool.

It is necessary to note the design of the balanced parameter ρ and the update form of the KLMS. The parameter ρ is mainly used for balancing the output generated by neural network and the one yielded by the KLMS, which is essential for yielding a desired smooth output. This motivates us to use the covariance function $\boldsymbol{\phi}$ for designing the parameter ρ. Consider an error variable $\varsigma = x - y$ with random variables x and y. The nonlinear stationary covariance function $\boldsymbol{\phi}$ takes the form of a squared covariance function, i.e., [33]

$$\ell^2(\varsigma) = \exp\left[-\left(\frac{\varsigma}{\sigma}\right)^2\right] \tag{14}$$

where σ is the kernel size. Define a vector $\boldsymbol{e}$, where the j-th entry $\ell^2_{\text{iso}}(\boldsymbol{e})$ is expressed as the squared covariance of the j-th dimensional $\boldsymbol{e}$, i.e.,

$$\ell^2_{\text{iso}}(\boldsymbol{e}) = \exp\left[-\left(\frac{-\boldsymbol{e}\circ\boldsymbol{e}}{\sigma^2}\right)\right] \tag{15}$$

The notation $\circ$ in (15) denotes the dot product operation. For example, the j-th entry of $\boldsymbol{e}\circ\boldsymbol{e}$ is equal to e_j^2.

As the squared covariance function in (15) may be sensitive for squared kernel width σ^2, a squared covariance function with kernel width σ can be considered, i.e.,

$$\ell^2_{\text{iso}}(\boldsymbol{e}) = \exp\left(-\frac{\boldsymbol{e}\circ\boldsymbol{e}}{\sigma}\right) \tag{16}$$

which represents an *isotropic* covariance function with only one kernel width σ adopted for weighting $\boldsymbol{e}\circ\boldsymbol{e}$. Comparing with the isotropic covariance function, an *anisotropic* covariance function is much more beneficial since δ exhibits varying sensitivity to $\boldsymbol{e}\circ\boldsymbol{e}$. An anisotropic squared covariance function $\ell^2_{\text{aniso}}(\boldsymbol{e})$ can thus be used to deal with the varying magnitude of $\boldsymbol{e}\circ\boldsymbol{e}$, where the kernel width σ is controlled by a secondary squared covariance function, i.e.,

$$\ell^2_{\text{ani}}(\boldsymbol{e}) = \exp\left(-\frac{\boldsymbol{e}\circ\boldsymbol{e}}{\gamma\ell^2_{\text{iso}}(\boldsymbol{e})}\right) \tag{17}$$

where $\ell^2_{\text{iso}}(\boldsymbol{e})$ is the isotropic squared covariance function with kernel width σ. The parameter γ in (17) plays a similar role with σ in

5 Simulation Result

In this section, we compare the performance of the proposed smooth deep reinforcement learning with that of traditional deep reinforcement learning in spectrum sharing in cognitive radio.

In all experiments, both primary and secondary users choose transmit power from a pre-designed set $\mathcal{P}_1 = \mathcal{P}_2 = \{0.05, 0.1, 0.15, \cdots, 0.35, 0.4\}$ and noise power at the receiver j, $j = 1, 2$ is set to $\sigma_1 = \sigma_2 = 0.01$. The channel gains from the primary user (secondary user) transmitter to the primary user (secondary user) receiver is set as $h_{ij} = 1, \forall i, j$. For guaranteeing the successful reception for primary user and secondary user, the minimum SINR requirements for the primary user and the secondary are set as $\eta_1 = 1.2$ and $\eta_2 = 0.7$, respectively. It can be found that there exists a pair of transmit power $\{p1, p2\}$ ensuring the requirement of the Qos for both primary and secondary users. In addition, a total number of N sensors are considered for collecting the RSS information to assist the secondary user to learn a power control policy. The distance d_{ij} between the transmitter Tx_i and the sensor node Θ_j is uniformly distributed in the interval $[100; 300]$ (in meters).

In all experiments, the deep neural network (DNN) is configured as three hidden layers equipped with the number of neurons 256, 256 and 512, respectively. Rectified linear units (ReLUs) are considered as the activation function for the first and the second hidden layers. The tanh function is used as the activation function for the last hidden layer. In our work, the Adam algorithm [40] is

Algorithm 1 Smooth Deep Reinforcement Learning In Spectrum Sharing In Cognitive Radio

Initialize replay memory D with buffer capacity O; network $Q(s;a;\theta)$ with random weights $\theta = \theta_0$
Initialize $p_1(1)$ and $p_2(1)$, then obtain $s(1)$
for $k = 1, K$ **do**
  ▷ Update $p_1(k+1)$ via the primary user's power control strategy (2) or (4).
  ▷ With probability ε_k select a random action $a(k)$, and otherwise select $a(k) = \max_a Q(s(k), a; \theta_0)$.
  ▷ Obtain $s(k+1)$ via the random observation model (5) and observe reward $r(k)$.
  ▷ Store transition $d(k) = \{s(k); a(k); r(k); s(k+1))\}$ and current state $C(k) = \{s(k)\}$ in D.
  if $k \geq O$ **then**
    ▷ Perform the deep neural network for calculating the $Q^m(i)$ in (12), $\forall i \in |\Omega_k|$ where $|\Omega_k|$ represents the index set of the random minibatch used at the kth iteration
    ▷ Perform the KLMS for calculating $Q^m(i)$ in (12) with supervised label provided by the deep neural network.
    ▷ Calculate Q-target by (14).
    ▷ Update θ by minimizing the loss function (12).
    ▷ Set $\theta_0 = \text{argmin}_\theta L(\theta)$.
  end if
  if $s(k)$ is a goal state **then**
    Initialize $p_1(k+1)$ and $p_2(k+1)$, then obtain $s(k+1)$.
  end if
end for

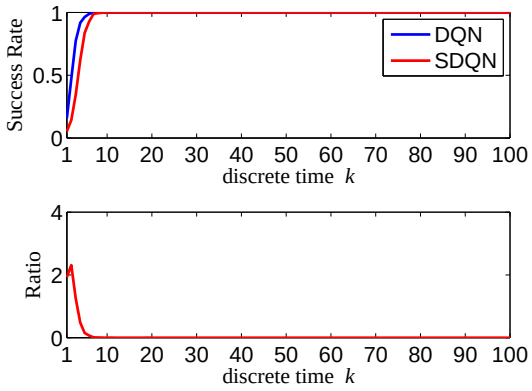

Figure 1: Success Rate of DQN and SDQN and Ratio with Policy1 for Primary User.

adopted for updating the weights θ, where the size of a minibatch is set to 256. We assume that the replay memory D contains $N_D = 300$ most recent transitions, and in each iteration, the training of θ begins only when D stores more than $O = 200$ transitions. The total number of iterations is set to $K = 10^4$. The probability of exploring new actions is set as 0.8.

We use Algorithm 1 to train the network to check its performance. The performance is evaluated via two metrics, namely, the success rate and the ratio. The success rate is computed as the ratio of the number of successful trials to the total number of independent runs. A trial is considered successful if s moves to a goal state within 20 time frames. The Ratio is calculated on the basis of the success rate of the DQN and SDQN as follows:

$$\text{Ratio} = \frac{\text{SS}_{\text{SDQN}} - \text{SS}_{\text{DQN}}}{\text{SS}_{\text{DQN}}} \tag{18}$$

We now study the performance of the deep reinforcement learning approach. Specifically, we examine the success rate, and the corresponding ratio as a function of the number of iterations $k^{'}$ used for

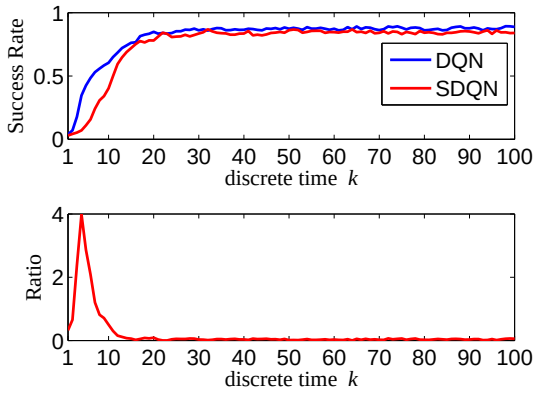

Figure 2: Success Rate of DQN and SDQN and Ratio with Policy2 for Primary User.

training where algorithms are tested each 100 iteration for computational efficiency,i.e., $k = 100k^{'}$ in Figs. 1 and 2. After k iterations of training, the secondary user can use the trained network to interact with the primary user. The success rate and the corresponding ratio are used to evaluate how well the network is trained. Results are averaged over 100 independent runs, in which a random initial state is selected for each run. Fig. 1 plots the the success rate and corresponding ratio vs. the number of iterations k, where we set $N = 10$, the standard deviation of the random variable used to account for the shadowing effect and measurement errors is set to $\delta_n = \left(p_1^p g_{1n} + p_1^s g_{2n}\right)/10$, and the primary user employs (2) to update its transmit power. We see that the proposed SDQN enables the secondary user to still learn an efficient power control policy. In addition, Fig. 2 presents the performance of the SDQN for learning a power control method when the primary user employs the second power control policy (4) to update its transmit power. Same as the conclusion of the Fig. 1, the SDQN still behaves an significantly improved performance over the traditional DQN.

6 Conclusion

The spectrum sharing in a cognitive radio network is about letting the secondary user share common spectrum with the primary user without harmful inference to the primary user. The reinforcement learning has been an intelligent power applied in cognitive radio network for spectrum sharing. The traditional deep reinforcement learning may yield undesired network output due to noises with different magnitude and degraded network weights. A novel deep reinforcement learning called smooth deep reinforcement learning is therefore presented in this paper for improving the efficiency of the traditional deep reinforcement learning. In the proposed smooth reinforcement learning, the kernel-based nonlinear filter is considered for further smoothing the output of deep neural network, beneficial for the agent taking optimal action in the process of interacting with environments. In addition, different scenarios for reducing computational burden are also proposed for improving learning efficiency. Simulation results have shown the efficiency of the proposed smooth deep neural network in the application of spectrum sharing in cognitive radio.

7 Reference

[1] P. Kolodzy and I. Avoidance, "Spectrum policy task force," Federal Commun. Commission, Washington, DC, USA, Tech. Rep. 02–135, 2002

[2] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, pp. 201–220, Feb. 2005

[3] D. Niyato and E. Hossain, "Competitive spectrum sharing in cognitive radio networks: A dynamic game approach," *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, pp. 2651–2660, Jul. 2008.

[4] L. Le and E. Hossain, "Resource allocation for spectrum underlay in cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, Dec. 2008.

[5] D. I. Kim, L. B. Le, and E. Hossain, "Joint rate and power allocation for cognitive radios in dynamic spectrum access environment," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 5517–5527, Dec. 2008.

[6] M. Xiao, N. B. Shroff, and E. K. P. Chong, "A utility-based power-control scheme in wireless cellular systems,"' *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 210–221, Apr. 2003.

[7] Y.-F. Liu, Y.-H. Dai, and S. Ma, "Joint power and admission control: Non-convex Lq approximation and an effective polynomial time deflation approach," *IEEE Trans. Signal Process.*, vol. 63, no. 14, pp. 3641–3656, Jul. 2015.

[8] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 4157–4162, Oct. 2010.

[9] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features, and open issues," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 253–267, 2012.

[10] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybernet.- Part C: Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[11] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEEWireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[12] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Trans. Games*, vol. 12, no. 1, pp. 1–20, Mar. 2020

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[14] D. P. Bertsekas, Dynamic Programming and Optimal Control, vol. 1. Belmont, MA, USA: Athena Sci., 2005.

[15] R. Bellman, Dynamic Programming. Mineola, NY, USA: Courier Corporat., 2013.

[16] Z. M. Fadlullah, B. Mao, F. Tang, and N. Kato, "Value iteration architecture based deep learning for intelligent routing exploiting heterogeneous computing platforms," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 939–950, Jan. 2019

[17] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, "An improved Sarsaλ reinforcement learning algorithm for wireless communication systems," *IEEE Access*, vol. 7, pp. 115418–115427, 2019.

[18] H. V. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 2613–2621, 2010.

[19] H. Ye, G. Y. Li and B. H. F. Juang, "Deep reinforcement learning for resource allocation in V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[20] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[21] S. Liu, X. Hu, and W. Wang, "Deep reinforcement learning based dynamic channel allocation algorithm in multibeam satellite systems," *IEEE Access*, vol. 6, pp. 15733–15742, 2018.

[22] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[23] W. Liu, J. C. J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley, 2010.

[24] J. W. Xu, A. R. C. Paiva, I. Park, and J. C. Príncipe, "A reproducing kernel Hilbert space framework for information-theoretic learning," *IEEE Trans. Signal Process.*, vol. 56, no. 12, pp. 5891-5902, Dec. 2008.

[25] I. Steinwart, D. Hush, and C. Scovel, "An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4635-4643, Oct. 2006.

[26] M. Takizawa and M. Yukawa,"Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4257-4269, Aug. 2015.

[27] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22-32, Jan. 2012.

[28] S. Wang, J. Feng, and C. K. Tse, "Kernel affine projection sign algorithms for combating impulse interference," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 11, pp. 811-815, Nov. 2013.

[29] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275-2285, Aug. 2004.

[30] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, vol. 6, pp. 25463–25473, Apr. 2018.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] R. S. Sutton and A. G. Barto, Reinforcement Learning–An Introduction. Cambridge, MA, USA: MIT Press, 1998.

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.