Quantized Attention-Gated Kernel Reinforcement Learning for Brain–Machine Interface Decoding

Fang Wang, Yiwen Wang, *Member, IEEE*, Kai Xu, Hongbao Li, Yuxi Liao, Qiaosheng Zhang, *Member, IEEE*, Shaomin Zhang, Xiaoxiang Zheng, *Member, IEEE*, and José C. Principe, *Fellow, IEEE*

***Abstract*— Reinforcement learning (RL)-based decoders in brain–machine interfaces (BMIs) interpret dynamic neural activity without patients' real limb movements. In conventional RL, the goal state is selected by the user or defined by the physics of the problem, and the decoder finds an optimal policy essentially by assigning credit over time, which is normally very time-consuming. However, BMI tasks require finding a good policy in very few trials, which impose a limit on the complexity of the tasks that can be learned before the animal quits. Therefore, this paper explores the possibility of letting the agent infer potential goals through actions over space with multiple objects, using the instantaneous reward to assign credit spatially. A previous method, attention-gated RL employs a multilayer perceptron trained with backpropagation, but it is prone to local minima entrapment. We propose a quantized attention-gated kernel RL (QAGKRL) to avoid the local minima adaptation in spatial credit assignment and sparsify the network topology. The experimental results show that the QAGKRL achieves higher successful rates and more stable performance, indicating its powerful decoding ability for more sophisticated BMI tasks as required in clinical applications.**

***Index Terms*— Attention-gated reinforcement learning (AGREL), brain–machine interfaces (BMIs), kernel adaptive filtering, quantization approach.**

Manuscript received November 16, 2014; revised September 13, 2015 and October 9, 2015; accepted October 10, 2015. Date of publication November 23, 2015; date of current version March 15, 2017. This work was supported in part by the National High Technology Research and Development Program of China under Grant 2012AA011602, in part by the National Basic Research Program of China under Grant 2013CB329506, in part by the Natural Science Foundation of China under Grant 61473261, Grant 61233015, Grant 61305146, and Grant 31371001, in part by the Zhejiang Provincial International Science and Technology Cooperation Program under Grant 2012C24025, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY14F030015 and Grant LZ14F020002, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Yiwen Wang.)*

F. Wang, K. Xu, H. Li, Y. Liao, and Q. Zhang are with the Qiushi Academy for Advanced Studies, Department of Biomedical Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: dreamtraveler.87@gmail.com; xkjadehill@gmail.com; yuxiliao@gmail.com; qiaoshengzhang@gmail.com; lihb@zju.edu.cn).

Y. Wang is with the Key Laboratory of Biomedical Engineering, Qiushi Academy for Advanced Studies, Ministry of Education, Zhejiang University, Hangzhou 310027, China (e-mail: eewangyw@zju.edu.cn).

S. Zhang and X. Zheng are with the Key Laboratory of Biomedical Engineering, Innovation Joint Research Center for Cyber-Physical-Society System, Qiushi Academy for Advanced Studies, Department of Biomedical Engineering, Ministry of Education, Zhejiang University, Hangzhou 310027, China (e-mail: zxx667@gmail.com; shaomin.bme@gmail.com).

J. C. Principe is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: principe@cnel.ufl.edu).

Digital Object Identifier 10.1109/TNNLS.2015.2493079

I. Introduction

BRAIN–MACHINE interfaces (BMIs) seek to decode neural signals directly into control commands for the external devices, indicating a significant potential to help disabled individuals [1]. In the last decade, BMI applications have successfully shown neural control for prosthetic devices [2]–[4], such as a computer cursor or a robotic arm. Recently, BMI researchers have started to focus on the development of a coadaptive BMI architecture that allows interactions with dynamic environments [5]–[7]. Without the knowledge of real limb movements, which is commonly used in the supervised neural decoder [8]–[13], BMI users adjust their brain activities to the decoder through trial-and-error, by observing how the external devices execute the task. Neuroplasticity underlies the ability to modulate the existing neural ensembles to better implement prosthetic control over time [5], [6], [14], [15]. Such a BMI framework enables a naïve user and the decoder to coadapt and maintain the stable performance in spite of the nonstationary nature of neural activity.

This novel BMI framework is mainly based on reinforcement learning (RL) theory [16], which provides a general framework for a BMI system to build a dynamic mapping from neural state to action adapted to the changing environment. Only a scalar evaluative signal (reward) is delivered from the environment to reinforce the decoder according to task accomplishment rather than a specific and permanently available error signal. In this sense, RL is a more realistic learning scheme for those patients who cannot produce explicit limb movements to train the decoder. In such a scenario, they just have to learn which action yields the most reward when reaching the target in the environment. DiGiovanna *et al.* [17] first presented $Q(\lambda)$-learning with temporal difference (TD) error to an RL-based BMI experiment paradigm that rats were trained to brain control a prosthetic arm in a two-target choice task. Mahmoudi and Sanchez [18] further developed the work with an internal reward represented by neural activity in the nucleus accumbens [19], [20]. Sanchez *et al.* [21] applied $Q(\lambda)$-learning to predict one-step movement to extend RL-based BMI framework on primates performing a center-out task. Q-learning techniques in [17], [18], and [21], however, suffer from generalization problems and need long training time that are incompatible with the motivation span of the animal, i.e., if animals are not rewarded in a few trials, they are no longer engaged in performing the task.

DiGiovanna *et al.* [17], Mahmoudi and Sanchez [18], and Sanchez *et al.* [21] explore the neural activity with only one target in the field of view that is reached within one trial, e.g., the left or right lever-press or one-step reaching in a center-out task. Q-learning methods implemented in [17] and [22] find the optimal policy by temporal credit assignment, which can take many steps, even in simple grid-world problems. It is also known that the curse of the dimensionality [16] limits RL's ability to explore effectively in spatiotemporal dimensions. In this paper, we will explore the spatial dimension of the scene to speedup the RL training in realistic BMI applications, e.g., an obstacle avoidance task, where every object in the scene from the agent's perspective is a possible goal. From the perspective of the animal, its cognitive state is selecting the goal, but an external observer (the agent) has to infer the subject's goal from the observation of the cognitive state and the complex scene with multiple objects through trial-and-error [23]. We submit that the solution that has been studied in coadaptive BMIs with one single target is ill-prepared to address the spatial credit assignment problem [24]–[26], because it will take many more trials to learn an optimal trajectory.

A classical and efficient spatial credit assignment method used in supervised learning is based on backpropagation (BP) to update the weights of the neural networks [24]. Attention-gated reinforcement learning (AGREL) eliminates the error signals inherent in BP, but still solves the spatial credit assignment problem by assigning credit among the units in the hidden layer of the neural network in a similar way as BP of TD(λ) error, i.e., only the weights of the output with the highest activation are updated [27], [28]. However, these weights are connected to the action selection in the output space for simple environments without differentiating between targets and wrong choices, so indirectly they are related to the spatial organization of the environment and the structure of goals. Therefore, the system has to solve a credit assignment over space as we discuss. We adopt AGREL that competes over space for multiple actions to map neural states into a seven-action ensemble (four directional movements/two holdings/one resting) for continuous trajectory predictions in a center-out task [29]. The performance improvement mainly relies on two factors:

1) using the softmax policy to grade the probabilities of actions. In case that the optimal action is not selected, the suboptimal action can be still chosen with a higher chance than other actions, e.g., it favors the current performance;
2) defining an expansive function to efficiently intensify the learning when taking an unexpected action but happens to be correct.

AGREL demonstrates better generalization and faster convergence than $Q(\lambda)$-learning in a larger state-action space, where the mapper structure is a multilayer perceptron (MLP) trained using BP [27]. A well-known issue is that BP is sensitive to the initialization and can be stuck in local minima especially for online learning [30]. Tsitsiklis and Van Roy [31] have pointed out that Q-learning could possibly diverge for nonlinear mappers, e.g., neural networks. This paper will improve AGREL with the online kernel structure to reach the global optimum in order to solve the spatial credit assignment problem in an obstacle avoidance task with faster learning.

During the last decade, there has been a growing popularity of kernel adaptive methods, such as kernel principal component analysis [32], support vector machine [33], [34], and kernel-based RL methods [35]–[42]. Kernel adaptive filtering uses Gaussian kernels to project the input data into a reproducing kernel Hilbert space (RKHS) where the linear operation can be performed, with an added advantage that the optimization is convex (no local minima) and efficient to compute by the kernel trick [43]. These methods have shown excellent classification and regression capabilities for nonlinear systems, but they are not online methods. Liu *et al.* [44] proposed the kernel least mean square (KLMS) to implement stochastic gradient with the property of well-posedness to simplify the implementation. Chen *et al.* [45] adopted a quantized KLMS (QKLMS) to compress the growing radial basis function (RBF) structure with each coming training sample, which avoids computational issues and curbs the memory growth. The quantization method does not simply discard the redundant data [46], [47]. The coefficient connected to the closest center is updated locally, or a new center is allocated when there is enough difference. Bae *et al.* [36], [37] improved kernel least square TD(λ)-learning by including this strategy to reach comparable performance in BMI decoding with much smaller kernel filter sizes. However, the use of QKLMS has been untested in more demanding applications as AGREL.

Motivated by the spatial credit assignment problem in the BMI tasks, we propose a quantized attention-gated kernel RL (QAGKRL) as a universal approximator for the nonlinear mapping from neural states to actions, which are discriminated based on the probability distribution over space learned in trial-and-error way. The new learning scheme uses the knowledge of the target as recalibrated feedback intention–trained Kalman filter (ReFIT-KF) and closed-loop decoder adaptation (CLDA) but the updates are conceptually different. ReFIT-KF and CLDA adapt the parameters with the desired velocity rotated directly toward the target as the supervised learning [48]–[50], whereas the target in QAGKRL is extracted from the environment to generate the reward signal to evaluate whether the performance is relatively better, and the agent cannot differentiate from the reward whether the action is suboptimal or optimal. In addition, the agent does not know the target but only tries different trajectories to maximize the total rewards. A quantization technique is incorporated to sparsify the network topology to improve the efficiency in exploring the large neural state-action space as well. To validate its ability to map the nonlinear neural firing patterns into a large action space, we compare QAGKRL with AGREL and $Q(\lambda)$-learning for a continuous and curved trajectory reconstruction imposed by an obstacle using the neural data recorded from dorsal premotor cortex (PMd) and primary motor cortex (M1) of a monkey performing an obstacle avoidance task.

The organization of this paper is as follows. In Section II, we describe the formulation of QAGKRL after a brief introduction to AGREL. In Section III, we present the experimental setup, then analyze and compare the experimental results

of QAGKRL and AGREL, as well as the temporal credit assignment by $Q(\lambda)$-learning. The discussion is given in Section IV. Finally, the conclusion is drawn in Section V.

II. Methods

A. AGREL

AGREL learns the neural state-action mapping with an MLP using an instantaneous reward to evaluate the task achievement [29]. Here, the outputs are Q-values of the action ensemble, $\{Z_k\}$, K is the number of available actions specified by the task. The action, Z_k, is normalized to be interpreted as probability as follows:

$$\mathbf{P}(Z_k = 1) = \frac{\exp(q_k)}{\sum_{k'=1}^{K} \exp(q_{k'})} \tag{1}$$

with

$$q_k = \sum_{m=0}^{M} w_{mk} Y_m \tag{2}$$

and

$$Y_m = \frac{1}{1 + \exp(-\sum_{n=0}^{N} v_{nm} x_n)} \tag{3}$$

where w_{mk} and v_{nm} are the weights of the hidden and input layers, and M and N are the corresponding number of nodes. x_n is the nth element in the input vector, e.g., u_i, which can be neural firing rates. Y_m and q_k are the output of the hidden and the output layers. Obviously, AGREL adopts the softmax policy (especially $\tau = 1$) to determine the probability of the winning unit, Z_k, since the actions described in (1) are engaged in a competition. Only the winning output is set with activity $Z_k = 1$, while the rest of units are forced to be 0, i.e., $Z_k = 0$.

We define an instantaneous reward function to discriminate the multiactions at each time instance

$$r = \frac{1}{1 + \exp(-\rho \Delta d(i))} \tag{4}$$

where $\rho = 20$ and $\Delta d(i) = \text{dist}(i-1) - \text{dist}(i)$ here. dist(i) is the distance between the positions of the cursor at time i and the target. Note that the reward function is defined based on the relative distance to the target but never uses the real trajectory. If the cursor moves toward the target at the current time step compared with the previous one, a positive scalar reward signal otherwise 0 is sent to update the parameters.

AGREL then calculates a global error signal after receiving an instantaneous reward, r, as follows:

$$\delta = r - \mathbf{P}(Z_k = 1). \tag{5}$$

Specifically, on the unrewarded trials, AGREL does not depend on the probability of a wrong action to adjust the synaptic plasticity, but sets $\delta = -1$ directly.

A global error-based expansive function $g(\delta)$ is defined to augment the learning if an unexpectedly rewarded action is chosen, shown as

$$g(\delta) = \begin{cases} \dfrac{\delta}{1 - \delta + \epsilon}, & \delta \geq 0 \\ \delta, & \delta = -1 \end{cases} \tag{6}$$

where ε = 1e-4 here, a small constant to eliminate the singularity when $\delta = 1$.

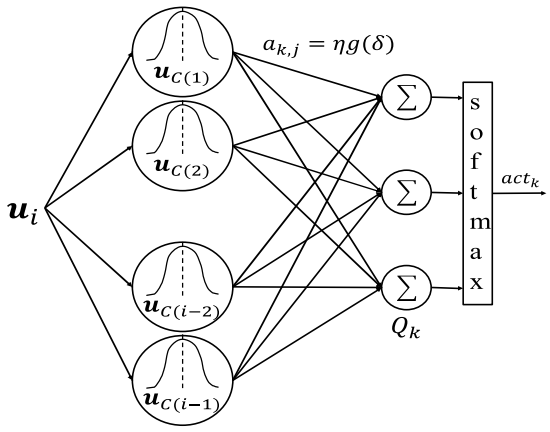


Fig. 1. RBF network structure of QAGKRL. The Gaussian kernel centers are the quantized input vectors, and the coefficient $a_{k,j}$ is the global error-based expansive function, $g(\delta)$, scaled by the learning rate, η. QAGKRL adopts the softmax policy as the action selection rule.

AGREL utilizes a physiologically plausible Hebbian rule for the weights update. Note that only the connections between the winning action and the nodes in the hidden layer are updated because other outputs are set to 0 after the competition [25], [29]

$$\Delta w_{mk} = \beta Y_m Z_k g(\delta) \tag{7}$$

$$\Delta v_{nm} = \beta x_n Y_m g(\delta)(1 - Y_m) \sum_{k=1}^{K} Z_k w'_{km} \tag{8}$$

where β is the leaning rate.

B. QAGKRL

Bae *et al.* [37], Liu *et al.* [44], and Chen *et al.* [45] testified the advantages of kernel methods in solving nonlinear adaptive filtering problems through inner products. It is appealing to embed kernel methods with a Gaussian kernel into RL framework [35], [38]–[41], [51], since the adaptation does not suffer from the local minima but preserve the universal approximation property. We propose a new learning scheme, called QAGKRL, as a state-action value function approximator to replace the MLP structure of AGREL with kernel adaptive filtering, and use a quantization technique to reduce the growth of the RBF structure. The formulation of QAGKRL will be presented next to a brief introduction to kernel adaptive filtering.

For a continuous, positive-definite function κ: $U \times U \to \mathbf{R}$, there exists a Hilbert space $\mathbf{H}$ and a mapping ϕ by the Mercer's theorem: $\kappa(u_i, u_j) = \langle \phi(u_i), \phi(u_j) \rangle$, where the space $\mathbf{H}$ is called RKHS satisfying, $f(u) = \langle f, \phi(u) \rangle = \langle f, \kappa(u, \cdot) \rangle$, $\forall f \in \mathbf{H}$, where f represents the nonlinear mapping function in RKHS. In this paper, we use the Gaussian kernel because it is strictly positive-definite

$$\kappa(u_i, u_j) = \exp\left(-\frac{\|u_i - u_j\|^2}{2h^2}\right) \tag{9}$$

where h is the kernel width, selected based on the distribution of Euclidean distances between pairs of input vectors [43].

The input vector u_i can be built from the neural firing rates for BMIs and transformed into $\mathbf{H}$ as $\phi(u_i)$. The state-action mapping in QAGKRL at the time index i is defined as follows:

$$Q_k(u_i) = \sum_{j=1}^{i-1} a_{k,j} \langle \phi(u_i), \phi(u_j) \rangle = \sum_{j=1}^{i-1} a_{k,j} \kappa(u_i, u_j) \quad (10)$$

where $a_{k,j} = \eta g(\delta)$ is the coefficient connecting from the jth Gaussian kernel center, u_j, in the RBF network to the kth action, $g(\delta)$ is the expansive function defined by (6), and η is the learning rate, chosen according to [43]

$$\eta < \frac{L}{tr[\mathbf{G}\Psi]} = \frac{L}{\sum_{i=1}^{L} \kappa(u_i, u_i)} = 1 \quad (11)$$

where L is the length of samples. The transformed data matrix is denoted as $\Psi = [\phi(u_1), \phi(u_2), \ldots, \phi(u_L)]$ and its Gram matrix is $G_\Psi = \Psi^T \Psi$. The preceding $(i-1)$ input vectors are served as the Gaussian kernel centers forming a dictionary, C. Equation (10) is a typical kernel adaptive filtering formula, and more details about the derivation can be found in [43]. However, it allocates a new kernel center for every new training sample, which results in the linear growth of the computational complexity and memory space issues. An effective quantization approach [45] is used to decrease the number of the kernel centers, which can reduce the growth to logarithmic. A quantization threshold ξ_U is chosen according to the Euclidean distances between the pairs of the input vectors. The final state-action value function approximation can be formulated as follows:

$$Q_k(u_i) = \sum_{j=1}^{|C(i-1)|} a_{k,j} \kappa\left(u_i, u_j^q\right) \quad (12)$$

where u_j^q is the jth input vector, whose minimum distance to the existing centers in the dictionary is larger than the quantization threshold ξ_U, calculated as follows:

$$\text{dist}(u_j, C_{c_j}(j-1)) = \min_{1 \le c_j \le |C(j-1)|} \|u_j - C_{c_j}(j-1)\| \quad (13)$$

where $|C(j-1)|$ is the size of the dictionary including those input patterns of the preceding $(j-1)$ samples already added to the dictionary, usually less than $(j-1)$, and c_j is the index.

If $\text{dist}(u_j, C_{c_j}(j-1))$ is larger than ξ_U, a new kernel center is assigned to this input vector, u_j, with the coefficient $a_{k,j}$. Otherwise the dictionary remains unchanged, u_j is quantized to the closest center rather than purely discarded, and that center's coefficient is locally updated. In this way, the time complexity of QAGKRL is reduced to $O(L|C|)$ from $O(L^2)$, where $|C|$ is the final size of the dictionary, and the coefficient update rule is accordingly as follows:

$$\begin{cases} a_{k,j} = \eta g(\delta), & j = \text{ index to the new center} \\ a_{k,c_j} = a_{k,c_j} + \eta g(\delta), & c_j = \text{ index to the closest center.} \end{cases} \quad (14)$$

QAGKRL chooses the winning action according to the probability distribution just as (1), and Fig. 1 shows its RBF network structure.

Algorithm 1 Quantized Attention-Gated Kernel Reinforcement Learning

Input patterns: $u_i \in U$, $i = 1, 2, \cdots, L$

Variables used in the loop

C:the center dictionary

a:coefficients vector of all the centers

Initialization

η: the learning rate

h: the kernel width parameter

ξ_U: the quantization threshold

$C(1) = [u_1]$

$\mathbf{a}(1) = [\eta \mathbf{0}]$

While (not meet the stop criteria)

{

1) Evaluate outputs of the RBF network:

$$Q_k(u_i) = \sum_{j=1}^{|C(i-1)|} a_{k,j} \kappa(u_i, u_j^q) \quad (1)$$

2) Select an action Z_k using the softmax policy:

$$\mathbf{P}(Z_k = 1) = \frac{\exp(q_k)}{\sum_{k'=1}^{K} \exp(q_{k'})} \quad (2)$$

3) Calculate the reward r and the global error signal, δ:

$$r = \frac{1}{1 + \exp(-\rho \Delta d(i))} \quad (3)$$

$$\delta = r - \mathbf{P}(Z_k = 1) \quad (4)$$

4) Compute the expansive function based on δ:

$$g(\delta) = \begin{cases} \frac{\delta}{1-\delta+\epsilon}, & \delta \ge 0 \\ \delta, & \delta = -1 \end{cases} \quad (5)$$

5) Compute the distance between u_i and C(i-1):

$$\text{dist}(u_j, C_{c_j}(j-1)) = \min_{1 \le c_j \le |C(i-1)|} \|u_i - C_j(i-1)\| \quad (6)$$

If $dist(u_i, C_j(i-1)) \le \xi_U$ then $C(i) = C(i-1)$,
$a_{k,j} = a_{k,j} + \eta g(\delta)$ and $\mathbf{a}(i) = \mathbf{a}(i-1)$,
where $j = \arg \min_{1 \le c_j \le |C(i-1)|} \|u_j - C_j(i)\|$,
else allocate a new center, $C(i) = [C(i-1), u_i]$,
$a_{k,i} = \eta g(\delta)$, $\mathbf{a}(i) = [\mathbf{a}(i-1), a_{k,i}]$

}

The training of the RBF network, which is also the learning process for the agent to interact with the environment through trial-and-error, is summarized in Algorithm 1.

The mean and variance of the coefficient change in QAGKRL are analyzed just as AGREL [25]. Due to the competition, only the coefficient connected to the selected kth output is locally updated. For a rewarded action, Z_{c_k} is chosen with the probability $\mathbf{P}(Z_{c_k} = 1)$, and the corresponding mean of the coefficient change is computed as follows:

$$\begin{aligned} E(\Delta a_{k,j}) &= E(\eta g(\delta)) \\ &= \mathbf{P}(Z_{c_k} = 1) \eta \frac{\delta}{1-\delta} \\ &= \eta [1 - \mathbf{P}(Z_{c_k} = 1)]. \end{aligned} \quad (15)$$

For a wrong action, Z_k, $(k \neq c_k)$ is selected with the probability $\mathbf{P}(Z_k = 1)$, and its average coefficient change

$$\begin{aligned} E(\Delta a_{k,j}) &= E(\eta g(\delta)) \\ &= \mathbf{P}(Z_{c_k} = 1)\eta(-1) \\ &= -\eta \mathbf{P}(Z_{c_k} = 1). \end{aligned} \tag{16}$$

Thus, we conclude

$$E(\Delta a_{k,j}) = \eta[t_k - \mathbf{P}(Z_{c_k} = 1)] \tag{17}$$

where t_k is 1 for the rewarded action and 0 for the erroneous action.

The variance of the coefficient change for a rewarded action is computed as follows:

$$\begin{aligned} \mathrm{Var}(\Delta a_{k,j}) &= E\{(\Delta a_{k,j})^2\} - E\{(\Delta a_{k,j})\}^2 \\ &= \mathbf{P}(Z_{c_k} = 1)[\eta g(\delta)]^2 - [\eta(1 - \mathbf{P}(Z_{c_k} = 1))]^2 \\ &= \eta^2 \frac{(1 - \mathbf{P}(Z_{c_k} = 1))^3}{\mathbf{P}(Z_{c_k} = 1)}. \end{aligned} \tag{18}$$

In addition, for an unrewarded action

$$\begin{aligned} \mathrm{Var}(\Delta a_{k,j}) &= E\{(\Delta a_{k,j})^2\} - E\{(\Delta a_{k,j})\}^2 \\ &= \mathbf{P}(Z_{c_k} = 1)[\eta g(\delta)]^2 - [-\eta \mathbf{P}(Z_{c_k} = 1)]^2 \\ &= \eta^2 \mathbf{P}(Z_k = 1)[1 - \mathbf{P}(Z_k = 1)]. \end{aligned} \tag{19}$$

Note that the coefficient change only depends on the probability of the chosen action, which links directly the kernel function of the input vector u_i and the center u_j to the kth output. When QAGKRL learns the neural state-action mapping, $P(Z_{c_k} = 1) \to 1$ and $P(Z_k = 1) \to 0$, the mean and the variance of the coefficient change get close to 0. That is to say, the coefficients become stable and the mapper reaches the convergence. There are no coefficients connected from the input layer to the hidden layer but only one hyperparameter, the kernel width, h, to control the computation on the hidden layer in QAGKRL, while AGREL depends on the calculation of the output of the hidden layer Y_m (i.e., the dimension of weights is $N * M$, where N and M are the number of input and hidden nodes, respectively). In this sense, QAGKRL could reduce the number of parameters of the regular network for training.

III. Experimental Study

Now, we apply the proposed QAGKRL on the real neural data recorded from the brain of a monkey performing an obstacle avoidance task to test its performance. The agent (QAGKRL) needs to infer the subject's goal among multiple objects in space, and assigns credit over space to avoid obstacle and complete the reaching task through trial-and-error.

A. Obstacle Avoidance Task and Data Collection

The motor BMI experimental paradigm was designed and implemented at Qiushi Academy for Advanced Studies, Zhejiang University. All animal handling procedures were approved by the Animal Care Committee at Zhejiang University, China, strictly complying with the Guide for Care and Use of Laboratory Animals (China Ministry of Health). An adult male rhesus monkey was trained to perform an obstacle avoidance task, as shown in Fig. 2(a). The monkey used its right hand to manipulate a joystick to move a cursor (radius 0.75 cm) to avoid hitting an obstacle and reach the target. As shown in Fig. 2(b), trials were initialized by the monkey holding the joystick projected as the cursor limited in the red square for a certain period of time delay. One target shown as a yellow circle (radius 1.5 cm) randomly appeared in other three potential positions. An obstacle displayed as a green bar was followed lying in the location between the initial position of the cursor and the target after the holding time. If the monkey controlled the joystick to move the cursor to avoid the obstacle and reach the target, it was considered as a successful trial. The computer program automatically sent a signal to give the monkey a drop of water reward. Then, the current position of the cursor was treated as the initial position for the next trial, even if the monkey failed in this trial, or stopped at any position on the monitor. The target in the next trial would appear randomly in other three positions, except for the previous starting position.

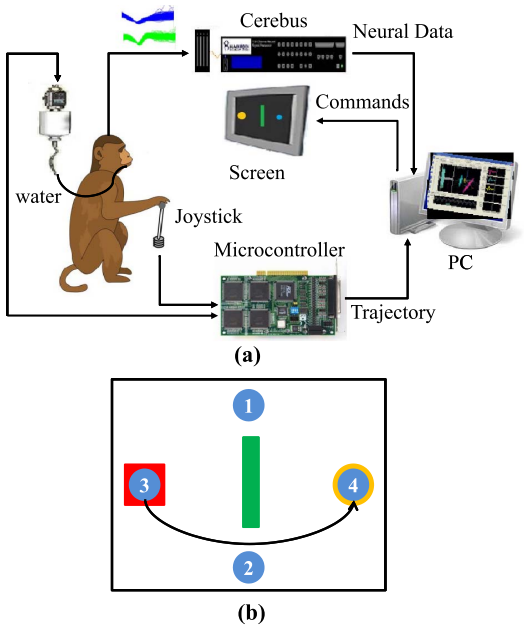


Fig. 2. Obstacle avoidance task. (a) Rhesus monkey sat in a primate chair, using its right hand to control a joystick projected as a computer cursor on a monitor. (b) Cursor appeared randomly in one of the four blue circles labeled as 1/2/3/4, the target (big yellow circle) displayed randomly in other three potential positions, then an obstacle (a green bar), lay in the middle of the cursor and the target. In this trial, the monkey should hold the cursor limited in a red square in position 3 for a certain time period of delay, move to avoid the obstacle to reach the target 4.

To collect the neural data, two Utah arrays (96 channels with ICS-96 connector, Blackrock Microsystems Inc., Salt Lake City, UT, USA) were chronically implanted in the upper limb area of PMd and M1 contralateral to the hand performing the task, respectively. Neural data were recorded by cerebus data acquisition system (Blackrock Microsystems Inc., Salt Lake City, UT, USA) when the monkey performed the task. The sampling rate was 30 kHz and the raw

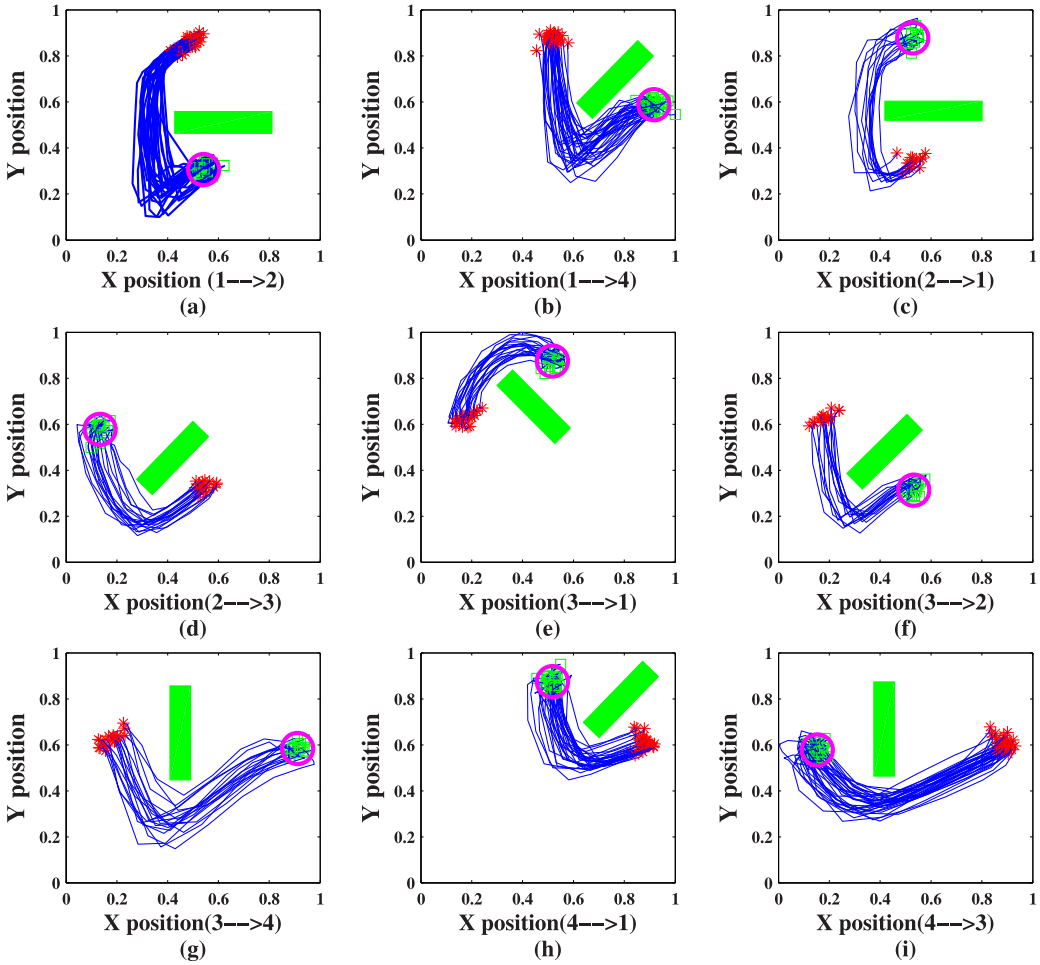


Fig. 3. Real trajectories of all the successful trials over nine scenarios. Red asterisks: start positions of all trials. Green squares: reaching target locations of all trials. Magenta circle: target area. Green bars: obstacles. Each subplot represents one scenario of obstacle avoidance. (a) S1:1→2. (b) S2:1→4. (c) S3:2→1. (d) S4:2→3. (e) S5:3→1. (f) S6:3→2. (g) S7:3→4. (h) S8:4 →1. (i) S9: 4 →3. The other three scenarios (1→3, 2→4, and 4→2) are not shown because the number of successful trials is less than 5.

neural data were amplified and then bandpass filtered from 250 Hz to 7.5 kHz. Spikes were detected using a threshold method and binned by a nonoverlapping 100-ms sliding time window as firing rates without spike sorting [the multiunit activity (MUA)]. The corresponding behavioral data, e.g., 2-D joystick trajectory, were simultaneously collected by a data acquisition card at a sampling rate of 30 Hz, then low-passed and down-sampled to 10 Hz to be synchronized with neural firing rates. A total of two data segments (23 minutes each) during a period of two days were collected for analysis.

B. Neural Network Settings

In this part, we will present how to set the neural networks in terms of the input vector, output actions, and hyperparameters to interpret the neural state-action mapper.

The input vector is specifically a spike firing rates vector, $u_i = \{(x_n)|x_n \in \mathbf{R}^{N\times 1}\}$, (bin size = 100 ms) at time index i, embedded a history of the neural activity (0.5 s) into the current neural firing rates plus a bias, where $N = 6\text{ch} + 1$ and ch is the number of channels containing MUA, here, $\text{ch} = 55$. The outputs are the corresponding actions that the agent will only choose one in a competition. Since AGREL and QAGKRL are both essentially RL-based algorithms which may suffer from the curse of dimensionality, namely, as the state-/action space increase, the computational time and the memory space exponentially grow [16], it is very crucial to determine the number of the discrete output actions. From Fig. 2(b), we can see that the movement trajectory consists of avoiding the obstacle and reaching the target in one trial, which turns out to be far from a straight line. Thus, we investigate the features of the movement trajectories through clustering. Fig. 3 shows nine scenarios of real trajectories for successful trials, in which the red asterisks represent the initial positions of the cursor and the green squares are the reaching target locations, and the magenta circle is the target and the green bar is the obstacle. The other three scenarios (1→3, 2→4, and 4→2) are not shown because the number of successful trials is less than 5. Fig. 4(a) shows the average trajectory of all the trials from position 4 to target 3 (red line) and the corresponding fitting curve (green line). The agent as an external observer has no knowledge of the role of the green bar (obstacle). The agent fails if it hits the obstacle, and only gets rewards when moves toward the target from either side of the

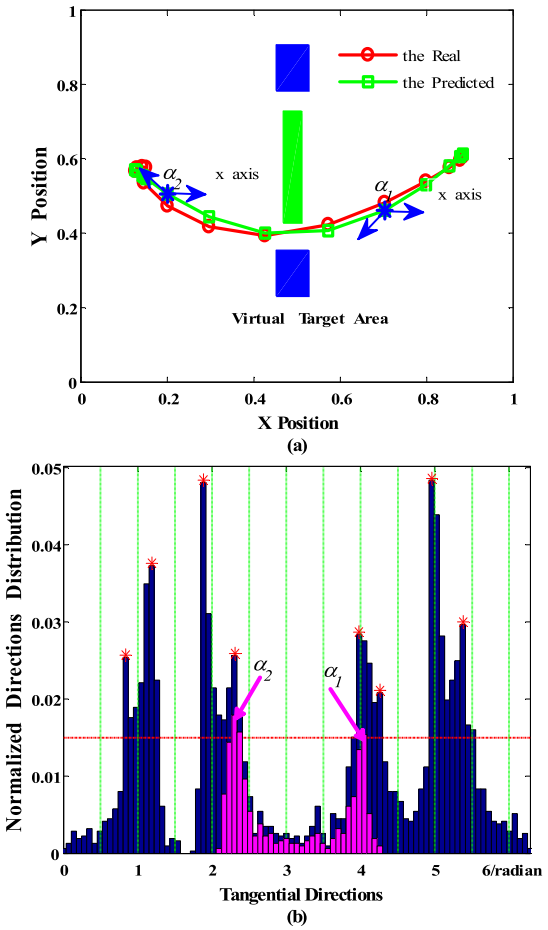


Fig. 4. Illustration of how to determine the output actions of the neural network. (a) Average trajectory of all the trials from position 4 to target 3 (red solid line) and the corresponding fitting curve (green solid line). Green bar: obstacle. Two blue rectangles are set at both sides of the obstacle served as virtual target areas. (b) Normalized directions distribution of Scenario: 4→3 (magenta histogram) and the tangential directions of all the trials at each time step within the nine scenarios (blue histogram). The tangential directions in Scenario: 4→3 are naturally grouped into two subsets whose peak values, α_1 and α_2, are taken as the centers. All the tangential directions are pooled together and divided equally into 12 parts by the green dotted lines, and the highest local angles but larger than a threshold (0.015, the red dashed-dotted line) are picked as the output actions, so the final number of the output actions 8, that is, $\theta = [0.838, 1.19, 1.88, 2.30, 3.98, 4.26, 4.96, 5.38]$ in radian, marked with the red asterisks.

obstacle (virtual target area). The virtual target area is placed at either side of the obstacle (two blue rectangles) to exactly mimic how the animal understands the obstacle avoidance, which is learned by the agent from the interaction with multiple objects through trial-and-error. Note that the targets are not prelabeled to supervise the learning. After the agent has learned avoidance, indicated by passing through the virtual target area, the maximum number of the steps exploration can be further reduced. Moreover, the agent's trajectory is not necessarily on the same side of the monkey's true trajectory and cannot impact the monkey's movement because of the open-loop analysis. Then, each trajectory is fitted with a specific quadratic polynomial equation, and the tangential directions at each time step are calculated and gathered in each scenario. We can infer that all the directions can be naturally grouped into two subsets, which include avoiding and reaching afterward, as shown in Fig. 4(b) (magenta histogram). The peak values, $\alpha_1 = 4$ and $\alpha_2 = 2.30$, are regarded as the centers of the two groups. It is the same circumstance for all the other scenarios. The peak values of some groups appear overlapped, e.g., the first peak values of scenarios 2→3 and 4→1 are the same, i.e., 1.88 radians; or the second peak values of cases 1→4 and 3→2 are quite close, 4.96 radians. All the tangential directions are pooled together to determine the final number of the output actions, as shown in Fig. 4(b) (blue histogram). The distribution is divided equally into 12 parts by the green dotted lines, and the highest local values in each interval but larger than a threshold (0.015 here, the red dashed-dotted line) are picked as the output actions. Such an operation is done to limit the increasing number of output nodes, which can extend the state-action space to impose a penalty on computational cost and brings convergence problems to the network. Taking the tradeoff between accurate estimation and computational complexity into account, the output number is set to 8, whose directions are expressed in radian $\theta = [0.838, 1.19, 1.88, 2.30, 3.98, 4.26, 4.96, 5.38]$ marked with red asterisks.

The predicted trajectory of each trial in each scenario is a direct 2-D reconstruction based on the system's output action, θ, at each time step

$$\begin{cases} x_{\text{pred}}(t) = x_{\text{pred}}(t-1) + \Delta_x \\ y_{\text{pred}}(t) = y_{\text{pred}}(t-1) + \Delta_y \end{cases} \tag{20}$$

with

$$\begin{cases} \Delta_x = l\cos(\theta) \\ \Delta_y = l\sin(\theta) \end{cases} \tag{21}$$

where θ is the output action mapping from the neural firing rates, and t is time index of the trial. $x_{\text{pred}}(0) = x_d(0)$ and $y_{\text{pred}}(0) = y_d(0)$. In addition, $x_d(0)$ and $y_d(0)$ are the desired movements at the initial position of each trial. $l \in \{l1, l2\}$ is the average step size obtained from training data. $l1$ is the one in obstacle avoidance phase, which is calculated by the Euclidean distance between the start position and the virtual target area center divided by the number of steps within the two points; $l2$ is the one in target reaching phase, which is calculated by the Euclidean distance between the virtual target area center and the target divided by the number of the steps within the two points.

Then, the hyperparameters, e.g., the learning rates and the kernel width, are explored for AGREL and QAGKRL. The optimal learning rates, $\beta_{\text{AGREL}} = 0.01$ and $\eta_{\text{QAGKRL}} = 0.05$, are determined according to the average success rates of the last 50 trials across ten initializations with directly setting the kernel width, $h = 2$. Then, the kernel width, h, is searched within the range $[h_s/10, 10h_s]$, where $h_s = 1.06\min\{\sigma, R/1.34\}T^{(-1/5H)}$ with the standard deviation σ, the interquartile R, and the length T of the Euclidean distances between the pairs of the neural data. As 500-ms modulation history is embedded into the current neural input vector, $H = 5$ [43], and $h = [1.0, 1.6, 2.0, 2.4, 2.8, 3.2]$. The quantization size ξ_U is explored within interquartile

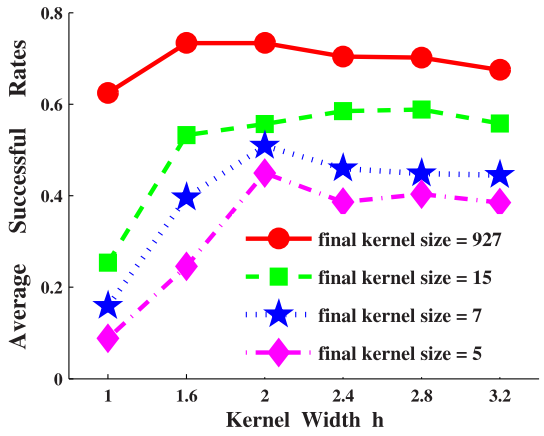


Fig. 5. Average successful rates of QAGKRL within the last 50 trials across ten initializations with six distinctive kernel widths $h = [1.0, 1.6, 2.0, 2.4, 2.8, 3.2]$, and four different quantization thresholds $\xi_U = [3.75, 9.19, 11.07, 12.84]$, the corresponding final kernel filter sizes are [927, 15, 7, 5].

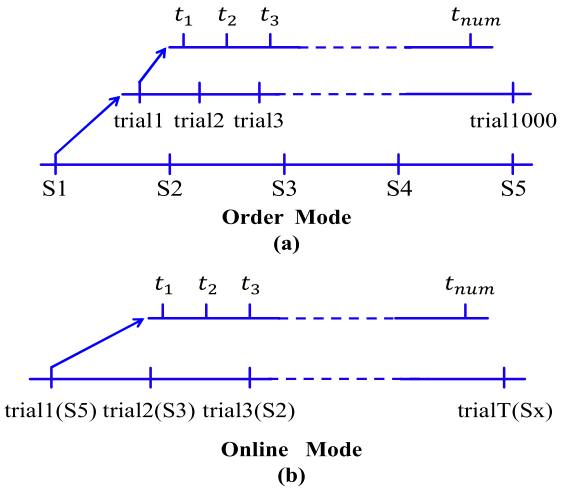


Fig. 6. Illustration of the order mode and the random mode. In both modes, the time index $(t_1, t_2, \ldots, t_{\text{num}}$, where num is the number of steps, usually 14 or 16, for the agent to complete one trial) is in the same sequence as recorded within one trial. (a) Order mode. Scenarios are played in sequence, that is S1:1→2, S2:2→3, S3:3→1, S4:4→1, and S5:4→3, and each includes 1000 trials grouped together. (b) Random mode. The distinctive trials are replayed with random scenarios Sx, which can be any one of the five scenarios.

range [37], [45], i.e., $\xi_U = [3.75, 9.19, 11.07, 12.84]$. Fig. 5 shows the average successful rates of the last 50 trials across ten random initializations with the different combinations of h and ξ_U. To achieve the optimal performance, we select $h = 1.6$ and $\xi_U = 3.75$ with the final kernel size $= 927$ for QAGKRL.

C. Training and Testing

Only the neural data corresponding to the successful trials are extracted for decoding. The weights of AGREL are initialized randomly between ± 1, while the coefficients of the first center are set to 0 for QAGKRL, and all keep updating after receiving an instantaneous reward for the action evaluation at every time step. The go cue signal is detected to initiate the decoding for a trial. If the agent hits the obstacle at any time step, this trial is aborted immediately and the next trial is initiated, or if it can avoid the barrier and reach the target within 2 s, then this trial is successful. Since the experiment is open loop, the decoded actions do not influence the subject's brain activity currently. The decoders in parallel learn to map the coming neural firing rates into actions in a trial-and-error manner. When the adaptive decoders have learned the neural state-action mapping, the parameters are used for testing on the data that have never appeared in the training. In addition, the trajectory reconstructions by a sequence of output actions are used to evaluate the decoding performance.

To validate the efficiency and robustness of QAGKRL for BMI decoding, we employ two kinds of modes: 1) order mode and 2) online mode. The order mode implies that all the trials of one scenario are manually grouped together but randomized within the scenario, and the models are trained by a sequence of the reordered scenarios. As shown in Fig. 6(a), the order mode contains totally 5000 trials over the five scenarios (S1, S2, S3, S4, and S5), and every 1000 trials are from the same scenario. Note that the neural firing rates at every time step, t_n, within each trial are in the same sequence just as recorded. By contrast, the online mode means that the neural data are replayed in the same time sequence as collected within and across trials, where the cases of the task scenes (the positions of the target and the obstacle) vary randomly. Fig. 6(b) shows the online mode that the trials are played back from any one of the five scenarios, Sx (S1:1→2, S2:2→3, S3:3→1, S4:4→1, and S5:4→3). The order mode simplifies the learning since the neural firing patterns in one scenario are much easier to discriminate, while the online mode would not take the advantage of the order of presentations amongst trials [18], and is a good testing strategy for QAGKRL to explore a larger and noisy state-action space initially. The agent makes action selection when each new neural input vector arrives at time t from the signal source in both modes. An online mode is applied right after the order mode without resetting the weights to observe if the decoders are able to retain the proper mapping.

We use threefold cross validation analysis on the real neural data across 30 initializations in the above two modes, and test the models with data. The learning curves across 450 epochs (1epoch $= 20$ trials) and the ranges of the total successful rates within each scenario are used for an evaluation of the policy during the training in terms of decoding stability and accuracy. In the order mode, the final weights of one scenario become the initial values to the next scenario and keep being updated throughout all the scenarios. The average learning curves of AGREL and QAGKRL increase within 50 epochs in each scenario during the training, as shown in Fig. 7(a). As more scenarios are encountered, AGREL shows the relatively slow convergence speed, i.e., the slopes and amplitudes gradually decrease scenario-by-scenario. But the performance of AGREL and QAGKRL drops when it comes to the online mode, which is assessed over eight representative actions. Notice that the QAGKRL recovers the performance progressively back to the one at the 250th epoch with the average successful rate at 0.8282 ± 0.0564 during the following epochs,

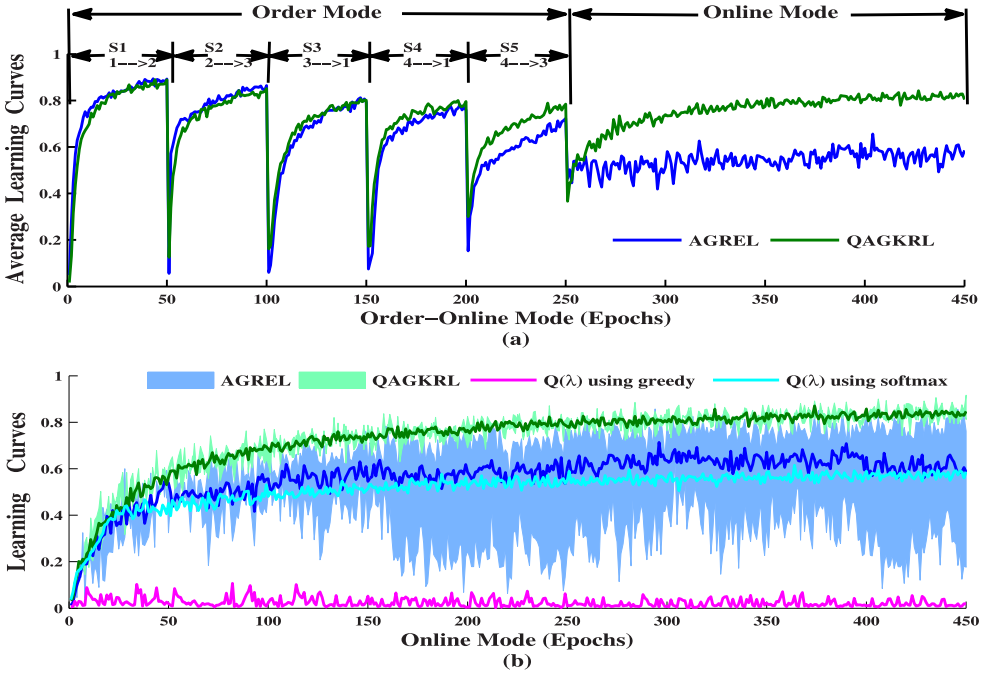


Fig. 7. Average learning curves within the five scenarios using threefold cross-validation across 30 random initializations by AGREL (blue solid lines) and QAGKRL (deep green lines) during the training process in the order mode and online mode. (a) Order-online mode. For clarity, error bars represent standard deviations are not shown. (b) Online mode. The corresponding light color shading areas represent respective regions with the best and the worst performance for AGREL and QAGKRL. For comparison, the magenta line represents the average learning curve of $Q(\lambda)$-learning ($\lambda = 0.2$) using greedy policy, while the cyan line is $Q(\lambda)$-learning ($\lambda = 0$) using the softmax policy.

where the trials vary across the scenarios. This is because the QAGKRL could adaptively follow the nonstationary neural activity, and quickly explore the extending neural state-action space when new scenarios enter.

However, AGREL maintains at 0.5856 ± 0.2373 without further improvement, indicating the disability to explore the larger state-action space and likely forget the past learned actions. Fig. 8(a) shows the ranges of the total successful rates within each scenario only in the order mode in Fig. 7(a). We can see that the successful rate ranges of AGREL are slightly better than those of QAGKRL in S1 and S2, of which the means are 0.7877 versus 0.7608 and 0.7756 versus 0.7417. The performance drops over scenarios when the task becomes more difficult as the neural state-action space extends. In comparison, QAGKRL shows the fairly steady performance across all the scenarios.

To unambiguously illustrate that QAGKRL does not have local minima, AGREL and QAGKRL are then applied in the online mode, where both models have to initially explore a larger and noisy neural state-action space. Thus, the task gets more complicated but realistic in the daily life. Fig. 7(b) shows two regions enclosed by the average learning curves with the best and the worst performance during the training process of AGREL (light blue area) and QAGKRL (light green area). The blue line and the deep green line represent the corresponding average learning curves across 30 initializations. Note that all the initializations for AGREL and QAGKRL are with the respective optimal hyperparameters. The best learning curve of AGREL (the upper boundary) is slightly worse than the average QAGKRL but its lower boundary is much worse which yields a significant difference from the average performances of both algorithms. In contrast, the range of performance across the different initializations of QAGKRL is quite narrow and can be interpreted by stochastic gradient descent. Fig. 8(b) shows the corresponding box-plot distributions of the total successful rates over the five scenarios

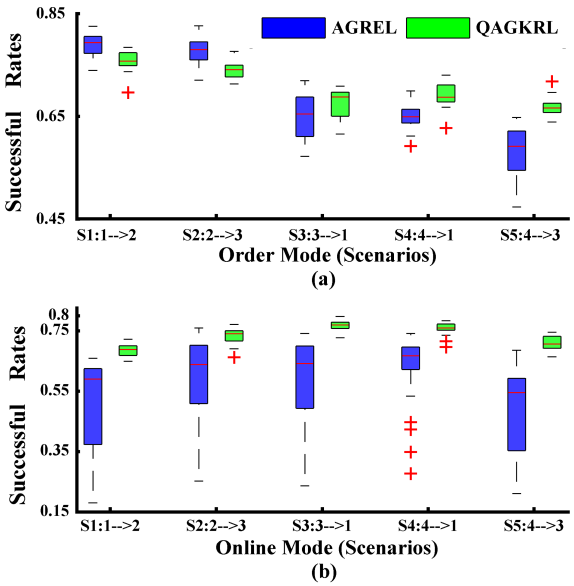


Fig. 8. Ranges of the total successful rates within the five scenarios across 30 random initializations by AGREL (blue boxes) and QAGKRL (green boxes) using threefold cross-validation during the training process. (a) Order mode (not include the random part). (b) Online mode (right tail paired-sample t-test, $p = 2.6965\text{e-}4$). Red plus signs: outliers.

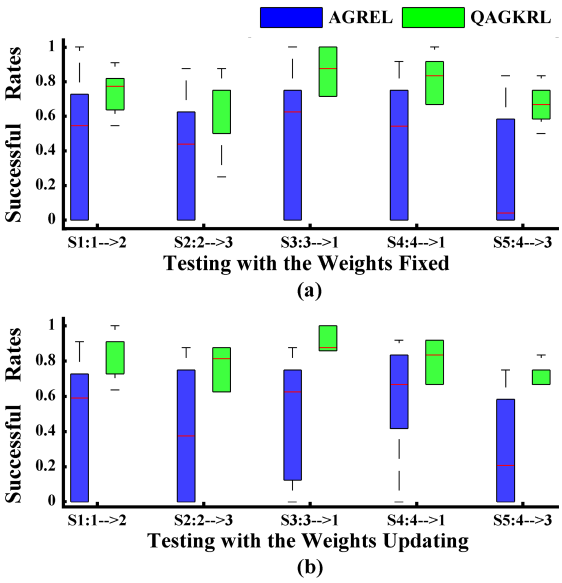


Fig. 9. Ranges of the total successful rates within the five scenarios with the testing data after the parameters trained in the online mode by AGREL (blue boxes) and QAGKRL (green boxes) corresponding to Fig. 8(b). (a) Testing with the parameters fixed (the right tail paired-sample t-test, $p = 4.2296\text{e-}5$). (b) Testing with the parameters updating (the right tail paired-sample t-test, $p = 3.0916\text{e-}4$).

across all initializations. We can find that QAGKRL ranges are much more compact with shorter whiskers, while AGREL ranges have larger distribution. The results indicate that the decoding performance of AGREL is consistent with local minima during the search, whereas QAGKRL is always in a neighborhood of the global optimum. It is worth mentioning that the output action at each time index is approximated by the eight representative directions. The decoding on the corresponding neural data is more complex compared with the four directional center-out tasks [29], thus AGREL fluctuates in the obstacle avoidance task. Furthermore, to illustrate the efficiency of spatial credit assignment versus temporal credit assignment in this task, we also implement $Q(\lambda)$-learning with eligibility trace to compared with AGREL and QAGKRL in the online mode. The magenta line is the average learning curve of $Q(\lambda)$-learning ($\lambda = 0.2$) using greedy policy ($\zeta = 0.01$) as in [17]. It completely fails because it needs tremendous steps to explore over time, while the number of steps to complete the task is limited before animal quits. The average learning curve of $Q(\lambda)$-learning ($\lambda = 0$) using softmax policy (cyan line) is worse than AGREL (blue line) with slightly slower learning speed, which indicates more training trials to explore both time and space.

When the performance reaches the convergence in the online mode, the two models are tested on the data that never appear in training. The trained parameters are either fixed for the normal testing or kept updating due to the nonstationary neural firing patterns [52], [53] to assess the extent to which the network is retaining the information it has learned in the past. Fig. 9 shows the ranges of the total successful rates of AGREL and QAGKRL within the five scenarios in the testing with [Fig. 9(a)] and without [Fig. 9(b)] the parameters fixed. A phenomenon that the minimum values of all the ranges and some first quartiles in AGREL regardless of whether the parameters fixed are 0 is observed, which means at least a quarter of the neural networks do not learn the mapping when initialized with the bad weights (the local minima). But the ranges of QAGKRL are much more consistent with different initializations. The neural networks with the parameters kept updating outperform those with the parameters fixed. One thing should be pointed out that these neural networks are not overtrained since the performance does not decrease in testing, e.g., the successful rates of QAGKRL in S4 are shown as mean $\pm$ standard deviation: 0.7577 ± 0.0189 (training), 0.7972 ± 0.1065 (testing with parameters fixed), and 0.8083 ± 0.103 (testing with the parameters kept updating).

Fig. 10 shows the adaptation process with three examples of trajectory reconstructions during the different training stages and the testing phase of QAGKRL in the online mode. The circles represent the initial positions and the squares are the targets. The green bars represent the obstacles. The green color depth represents the appearance order of obstacles, i.e., the light green bar is the earlier case, and the dark green is the latter one. The solid lines are the real trajectories, and the dashed-dotted lines are the corresponding predictions. In Fig. 10(a), QAGKRL initially is unable to map neural firing rates into the actions leading to stochastically predicted trajectories due to the random initializations (10th epoch). It begins avoiding the obstacle when learning through trial-and-error over time, but still has the limited capability to reach the target, e.g., some unnecessary movements to deviate from the target, in the middle of the training stage [150th epoch in Fig. 10(b)]. Fig. 10(c) and (d) shows the examples in the training phase (when the model converges) and testing phase, respectively. The predictions are indeed two straight lines. The agent's trajectories are actually different from the real ones, which are expected in RL and it is unlike supervised learning where the minimization of errors will tend to provide closer matching that can also bring overtraining. Indeed, the ultimate goal of an agent in RL is to maximize its total amount of rewards [16], so it is reasonable and beneficial for the agent to take the most efficient way (straight line) to get rewards and complete the task.

IV. Discussion

In conventional RL, the user selects the goal state (or the physics of the problem define it). Thus, the problem for the agent is formulated as temporal credit assignment. Once the agent solves the problem, it has also implicitly solved a spatial problem, i.e., the best trajectory in space. However, the issue is that it may take the agent an enormous amount of time to discover the solution. This is incompatible with RL for BMIs because the subjects just quit if they are not rewarded in a few trials. Likewise, the human will get very frustrated with a long training time. Therefore, the goal is to decrease the number of training time in RL for BMIs, and the formulation of the spatial credit assignment problem is a step in this direction. The tasks we are addressing here are quite complex in space, and benefit from a credit assignment over space formulation, where the agent has to infer the potential target or obstacle through interaction with every object in the space scene.

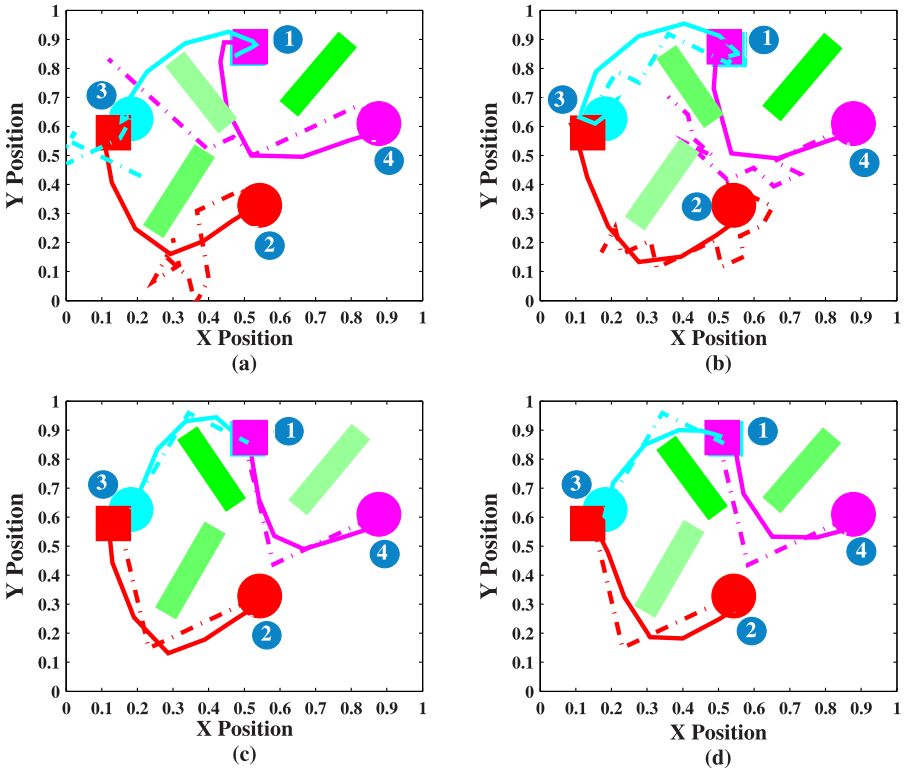


Fig. 10. Successfully reconstructed examples during the different training stages at (a) 10th epoch, (b) 150th epoch, and (c) 400th epoch. (d) Testing phase with all the parameters fixed after QAGKRL is trained in the online mode. Circles: initial positions of the cursor. Squares: targets. Green bar: obstacle. Green color depth: appearance order of obstacles. Solid lines: real trajectories. Dashed-dotted lines: corresponding predications.

This spatial credit assignment problem is not normally studied in the RL literature, but real and unappreciated in BMIs. In this paper, we first propose QAGKRL as an RL method to infer the subject's goal in a complex scene by discriminating actions through trial-and-error. We need to clarify that QAGKRL shares the same structure as the kernel TD learning in [36] and [37]. The main difference is that Bae's work solves the temporal credit assignment problem, which maximizes the expectation of the total rewards over time. Such a cost design derives from the TD error of the Q-values with eligibility traces. This paper focuses on solving the space problem, which maximizes the probability of the correct action chosen in space. The stochastic softmax policy is obtained from such a cost function to generate actions according to the probability distribution on the outputs of the network, which is essentially the same as the classic softmax. In other words, the agent's action is not chosen deterministically as in the supervised learning. The instantaneous reward is used to discriminate actions to reach multiple goals for simplicity, and avoids the exploration in spatial and temporal dimensions at the same time. The implementation of $Q(\lambda)$-learning with eligibility traces also shows that the obstacle avoidance task cannot be simply solved by temporal credit assignment.

Specifically, the structural features of QAGKRL appear in three aspects: 1) the output action is chosen according to the probability distribution of the network to grade the actions, and the corresponding coefficient only connected to the winning output is updated but not the entire network; 2) a quantization approach is used to compress the growing structure of the RBF network; and 3) the error signal is not defined as the exact difference between the output and the monkey's true trajectory, and amplified if the agent chooses an action with a small probability but happens to be correct. In our experiment, we observe that the average total successful rate is 0.7300 ± 0.0337 using $g(\delta)$ compared with 0.6658 ± 0.0412 using δ for the five scenarios in the online mode. Thus, QAGKRL with the expansive function, $g(\delta)$, has higher accuracy and faster learning speed than the one only using the error itself. These features help QAGKRL improve the efficiency and the power in exploring a large state-action space.

We also theoretically analyze the mean and variance of the coefficient change in QAGKRL just as AGREL [25], and prove that the coefficients can reach the convergence when the model learns the neural state-action mapping. Next, we experimentally validate QAGKRL in a complex obstacle avoidance task, and compare the experimental results with AGREL. QAGKRL does not depend on the initializations whereas AGREL fluctuations across 30 initializations. The MLP network of AGREL trained with BP rule on weights update is easy to fall into the local minima, which does not guarantee the optimality for the nonlinear neural decoding. The inherent kernel structure in QAGKRL renders the parameters optimization convex, so it outperforms AGREL in exploring a larger neural state-action space with much higher accuracy and lower variability.

To the agent, the reward signal can be calculated at every time instance according to (4) as long as the information of the

target and the cursor position are available. A series of 1-step reward updates can also be retroactively applied as a batch as long as there is eventually the reward feedback while the system coadapts with the user. This reward is not instantaneous for the subject when applied online. One possible implementation is to use the real-time visual/audio feedback on the monitor to tell the subject how well the task is completed, e.g., the green cursor indicates the movements with positive instantaneous rewards whereas the red cursor for the wrong ones along with a brief tone. If the task is not completed within $\sim$4 s (including the hold-time in the initial and target positions), the trial is ended. Then, everything within one trial is back to an initial state. These steps could reduce the frustration of the subject, and help it learn the task and get rewards for the online decoding. In some BMI applications, the agent cannot receive the reward until it completes one trial that needs multisteps, e.g., a robot has to learn a sequence of movements but the reward is only delivered at the end of the action sequence. Thus, the current QAGKRL algorithm should be further extended to the case with the delayed rewards but with more efficiency. In addition, the kernel operation in QAGKRL requires quite heavy computation to estimate Gaussian kernel function between the current input vector and each existing center, which can be implemented in graphics processing unit in parallel [54] to increase the processing speed for the real-time applications. The current quantization method still lets the filter size grow, which makes real-time implementation challenging. Alternatively, one can keep the dictionary constant by throwing away one sample for every new one that is brought into the dictionary. But this needs to be further researched since the system may progressively forget the past patterns.

V. Conclusion

The ultimate goal of BMI applications is to control external devices directly with brain activity for the motor-impairment patients to improve the quality of their lives. These BMI users cannot generate the limb movements but can adapt to the environment due to neuroplasticity induced by the biofeedback. The RL-based BMI architecture enables adaptive decoders to learn to decode the dynamic neural activity without real limb movement, which is appropriate for these patients. In this paper, we propose QAGKRL as an RL approach to assign credit over space for the complex BMI tasks, e.g., the obstacle avoidance task. The animal needs to select its goal among many potential targets in the scene, and the agent without priori knowledge has to infer the subject's goal through trial-and-error. QAGKRL interpret the nonlinear state-action mapping as the linear combination of the inner products in RKHS via kernel adaptive filtering, where the linear operation can be easily applied to reach the global minima. The output actions of the network are distinguished according to the probability distribution formed by learning with the instantaneous reward, which efficiently explores a large state-action space. Compared with AGREL, it achieves much higher accuracy and more stable performance, indicating its powerful ability for more sophisticated BMI applications as required in clinical applications.

Acknowledgment

The authors would like to thank S. Xiong and K. Yang for their assistance with animal care and training. The corresponding author Y. Wang would like to thank Y. Shen for the assistance in experiments.

References

[1] M. A. Lebedev and M. A. L. Nicolelis, "Brain–machine interfaces: Past, present and future," *Trends Neurosci.*, vol. 29, no. 9, pp. 536–546, Sep. 2006.

[2] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. L. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," *Nature Neurosci.*, vol. 2, no. 7, pp. 664–670, 1999.

[3] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, no. 5681, pp. 258–262, Jul. 2004.

[4] L. R. Hochberg *et al.*, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, no. 7099, pp. 164–171, Jul. 2006.

[5] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, "Direct cortical control of 3D neuroprosthetic devices," *Science*, vol. 296, no. 5574, pp. 1829–1832, 2002.

[6] G. J. Gage, K. A. Ludwig, K. J. Otto, E. L. Ionides, and D. R. Kipke, "Naïve coadaptive cortical control," *J. Neural Eng.*, vol. 2, no. 2, pp. 52–63, 2005.

[7] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, no. 7198, pp. 1098–1101, Jun. 2008.

[8] W. Wu, M. J. Black, D. Mumford, Y. Gao, E. Bienenstock, and J. P. Donoghue, "Modeling and decoding motor cortical activity using a switching Kalman filter," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 933–942, Jun. 2004.

[9] K. H. Kim, S. S. Kim, and S. J. Kim, "Superiority of nonlinear mapping in decoding multiple single-unit neuronal spike trains: A simulation study," *J. Neurosci. Methods*, vol. 150, no. 2, pp. 202–211, Jan. 2006.

[10] G. Santhanam, S. I. Ryu, B. M. Yu, A. Afshar, and K. V. Shenoy, "A high-performance brain–computer interface," *Nature*, vol. 442, no. 7099, pp. 195–198, Jul. 2006.

[11] Y. Wang, A. R. C. Paiva, J. C. Príncipe, and J. C. Sanchez, "Sequential Monte Carlo point-process estimation of kinematics from neural spiking activity for brain-machine interfaces," *Neural Comput.*, vol. 21, no. 10, pp. 2894–2930, 2009.

[12] Y. Wang, J. C. Principe, and J. C. Sanchez, "Ascertaining neuron importance by information theoretical analysis in motor brain–machine interfaces," *Neural Netw.*, vol. 22, nos. 5–6, pp. 781–790, Jul./Aug. 2009.

[13] Y. Wang and J. C. Principe, "Instantaneous estimation of motor cortical neural encoding for online brain–machine interfaces," *J. Neural Eng.*, vol. 7, no. 5, p. 056010, Sep. 2010.

[14] S. I. H. Tillery, D. M. Taylor, and A. B. Schwartz, "Training in cortical control of neuroprosthetic devices improves signal extraction from small neuronal ensembles," *Rev. Neurosci.*, vol. 14, nos. 1–2, pp. 107–120, 2003.

[15] B. Jarosiewicz, S. M. Chase, G. W. Fraser, M. Velliste, R. E. Kass, and A. B. Schwartz, "Functional network reorganization during learning in a brain-computer interface paradigm," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 49, pp. 19486–19491, 2008.

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

[17] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, "Coadaptive brain–machine interface via reinforcement learning," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 1, pp. 54–64, Jan. 2009.

[18] B. Mahmoudi and J. C. Sanchez, "A symbiotic brain-machine interface through value-based decision making," *PLoS One*, vol. 6, no. 3, p. e14760, 2011.

[19] S. C. Tanaka, K. Doya, G. Okada, K. Ueda, Y. Okamoto, and S. Yamawaki, "Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops," *Nature Neurosci.*, vol. 7, no. 8, pp. 887–893, 2004.

[20] K. Doya, "Modulators of decision making," *Nature Neurosci.*, vol. 11, no. 4, pp. 410–416, 2008.

[21] J. C. Sanchez *et al.*, "Control of a center-out reaching task using a reinforcement learning brain-machine interface," in *Proc. 5th Int. IEEE/EMBS Conf. Neural Eng. (NER)*, Cancún, Mexico, Apr./May 2011, pp. 525–528.

[22] B. Mahmoudi, E. A. Pohlmeyer, N. W. Prins, S. Geng, and J. C. Sanchez, "Towards autonomous neuroprosthetic control using Hebbian reinforcement learning," *J. Neural Eng.*, vol. 10, no. 6, p. 066005, Oct. 2013.

[23] J. A. Villacorta-Atienza and V. A. Makarov, "Neural network architecture for cognitive navigation in dynamic environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 12, pp. 2075–2087, Dec. 2013.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 9, pp. 533–536, Oct. 1986.

[25] P. R. Roelfsema and A. van Ooyen, "Attention-gated reinforcement learning of internal representations for classification," *Neural Comput.*, vol. 17, no. 10, pp. 2176–2214, Oct. 2005.

[26] D. Balduzzi, H. Vanchinathan, and J. Buhmann. (2014). "Kickback cuts Backprop's red-tape: Biologically plausible credit assignment in neural networks." [Online]. Available: http://arxiv.org/abs/1411.6191

[27] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[28] R. S. Sutton, "Implementation details of the TD(λ) procedure for the case of vector predictions and backpropagation," GTE Lab., Waltham, MA, USA, Tech. Rep. TN87-509.1, 1989.

[29] Y. Wang, F. Wang, K. Xu, Q. Zhang, S. Zhang, and X. Zheng, "Neural control of a tracking task via attention-gated reinforcement learning for brain-machine interfaces," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 23, no. 3, pp. 458–467, May 2015.

[30] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Hoboken, NJ, USA: MacMillan, 1994.

[31] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.

[32] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[33] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.

[34] S. Dangi *et al.*, "Continuous closed-loop decoder adaptation with a recursive maximum likelihood algorithm allows for rapid performance acquisition in brain-machine interfaces," *Neural Comput.*, vol. 26, no. 9, pp. 1811–1839, Sep. 2014.

[35] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, Jul. 2007.

[36] J. Bae, P. Chhatbar, J. T. Francis, J. C. Sanchez, and J. C. Principe, "Reinforcement learning via kernel temporal difference," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Boston, MA, USA, Aug./Sep. 2011, pp. 5662–5665.

[37] J. Bae, L. S. Giraldo, P. Chhatbar, J. Francis, J. Sanchez, and J. Principe, "Stochastic kernel temporal difference for reinforcement learning," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Cantabria, Spain, Sep. 2011, pp. 1–6.

[38] A. S. Barreto, D. Precup, and J. Pineau, "Reinforcement learning using kernel-based stochastic factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, Dec. 2011, pp. 720–728.

[39] B. Kveton and G. Theocharous, "Kernel-based reinforcement learning on representative states," in *Proc. 26th AAAI Conf. Artif. Intell.*, Toronto, ON, Canada, Jul. 2012, pp. 977–983.

[40] D. Precup, J. Pineau, and A. S. Barreto, "On-line reinforcement learning using incremental kernel-based stochastic factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1484–1492.

[41] B. Kveton and G. Theocharous, "Structured kernel-based reinforcement learning," in *Proc. 27th AAAI Conf. Artif. Intell.*, Bellevue, WA, USA, Jul. 2013, pp. 569–575.

[42] X. Chen, Y. Gao, and R. Wang, "Online selective kernel-based temporal difference learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 12, pp. 1944–1956, Dec. 2013.

[43] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. New York, NY, USA: Wiley, 2011.

[44] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

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

[46] 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.

[47] W. Liu, I. Park, and J. C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.

[48] V. Gilja *et al.*, "A high-performance neural prosthesis enabled by control algorithm design," *Nature Neurosci.*, vol. 15, no. 12, pp. 1752–1757, 2012.

[49] A. L. Orsborn, S. Dangi, H. G. Moorman, and J. M. Carmena, "Closed-loop decoder adaptation on intermediate time-scales facilitates rapid BMI performance improvements independent of decoder initialization conditions," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 4, pp. 468–477, Jul. 2012.

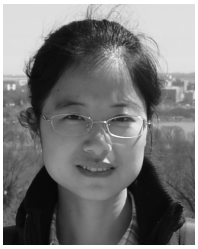
[50] S. Dangi, A. L. Orsborn, H. G. Moorman, and J. M. Carmena, "Design and analysis of closed-loop decoder adaptation algorithms for brain-machine interfaces," *Neural Comput.*, vol. 25, no. 7, pp. 1693–1731, Jul. 2013.

[51] C. E. Rasmussen and M. Kuss, "Gaussian processes in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2003, pp. 1–8.

[52] K. Ganguly, D. F. Dimitrov, J. D. Wallis, and J. M. Carmena, "Reversible large-scale modification of cortical networks during neuroprosthetic control," *Nature Neurosci.*, vol. 14, no. 5, pp. 662–667, 2011.

[53] M. A. Lebedev *et al.*, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *J. Neurosci.*, vol. 25, no. 19, pp. 4681–4693, 2005.

[54] K. Xu *et al.*, "Neural decoding using a parallel sequential Monte Carlo method on point processes with ensemble effect," *BioMed Res. Int.*, vol. 2014, May 2014, Art. ID 685492.



Fang Wang received the B.S. degree in biomedical engineering from Tianjin University, Tianjin, China, in 2010. She is currently pursuing the Ph.D. degree in biomedical engineering with the Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou, China.

Her current research interests include machine learning, adaptive signal processing, and computational neuroscience.

Yiwen Wang (S'05–M'08) received the B.S. and M.S. degrees from the University of Science and Technology of China, Hefei, China, in 2001 and 2004, respectively, and the Ph.D. degree from the University of Florida, Gainesville, FL, USA, in 2008.

She joined the Department of Electronics and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, as a Research Associate. In 2010, she joined as a Faculty Member with the Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou, China, where she is currently an Associate Professor. Her current research interests include neural decoding of brain–machine interfaces, adaptive signal processing, computational neuroscience, and neuromorphic engineering.

Kai Xu received the B.S. and Ph.D. degrees in biomedical engineering from Zhejiang University, Hangzhou, China, in 2009 and 2015, respectively.

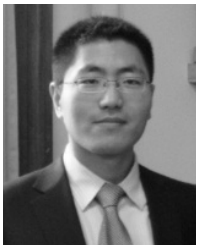
He is currently an Associate Researcher with HULU, Beijing, China, focusing on personalized recommendation. His current research interests include machine learning, recommendation systems, graphics processing unit computing, and computational neuroscience.

Hongbao Li received the B.S. degree in biomedical engineering from Shandong University, Jinan, China, in 2012. He is currently pursuing the Ph.D. degree in biomedical engineering with Zhejiang University, Hangzhou, China.

He has been with the Qiushi Academy for Advanced Studies, Zhejiang University, since 2012. His current research interests include brain–machine interface system construction, signal processing, and computational neuroscience.

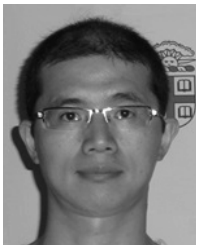
Yuxi Liao received the B.S. and Ph.D. degrees in biomedical engineering from Zhejiang University, Hangzhou, China, in 2009 and 2014, respectively.

She is currently a Research Engineer in large-scale distributed machine learning with Huawei Technologies Company, Ltd., Hangzhou, China. Her current research interests include machine learning, adaptive signal processing, and computational neuroscience.



Qiaosheng Zhang (M'13) received the B.S. degree in biotechnology from Guizhou University, Guiyang, China, in 2006, and the M.S. degree in biophysics and the Ph.D. degree in biomedical engineering from Zhejiang University, Hangzhou, China, in 2008 and 2012, respectively.

He joined the Qiushi Academy for Advanced Studies, Zhejiang University, from 2012 to 2015. He is currently a Post-Doctoral Fellow with the Department of Anesthesiology, New York University, New York City, NY, USA. His current research interests include brain–machine interfaces and pain modulation pathway.



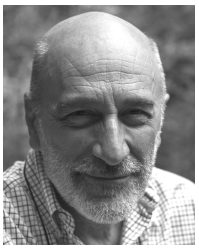
Shaomin Zhang received the B.S. and Ph.D. degrees in biomedical engineering from Zhejiang University, Hangzhou, China, in 2002 and 2007, respectively.

He joined as a Faculty Member with Zhejiang University in 2007, where he is currently an Associate Professor with the Qiushi Academy for Advanced Studies. His current research interests include the chronic neural recording and decoding in large cortical ensembles, and the development of brain–machine interfaces.

Xiaoxiang Zheng (M'13) received the B.S. degree from Zhejiang University, Hangzhou, China, in 1968, and the M.S. and M.D. degrees from Tsukuba University, Tsukuba, Japan, in 1984 and 1993, respectively.

She was the Dean of the College of Biomedical Engineering and Instrument Science from 1999 to 2005. Since 1993, she has been a Full Professor of Biomedical Engineering with Zhejiang University. She has been the Executive Director of the Qiushi Academy for Advanced Studies with Zhejiang University since 2006. Her current research interests include microcirculation, physiology of cell, and neural engineering.

Prof. Zheng is the Board Member of the Chinese Medicine Research Committee and a member of the Chinese Society of Biomedical Engineering.



José C. Principe (F'00) received the master's degree in electrical engineering from the University of Porto, Porto, Portugal, the Ph.D. degree in electrical engineering from the University of Florida, Gainesville, FL, USA, and the Honoris Causa degrees from the Universita Mediterranea in Reggio Calabria, Reggio Calabria, Italy, the Universidade do Maranhao, São Luís, Brazil, and Aalto University, Espoo, Finland.

He has been a Distinguished Professor of Electrical and Biomedical Engineering with the University of Florida since 2002. He is currently a BellSouth Professor and the Founding Director of the Computational NeuroEngineering Laboratory with the University of Florida. He joined the University of Florida in 1987, after an eight year appointment as a Professor with the University of Aveiro, Aveiro, Portugal. He has authored over 600 refereed publications (five books, seven edited books, 19 book chapters, 201 journal papers, and 427 conference proceedings). He holds 22 patents and has submitted seven more.

Dr. Principe was a fellow of the American Institute for Medical and Biological Engineering in 2006 and the International Academy of Medical and Biological Engineering in 2012. He received the INNS Gabor Award, the IEEE Engineering in Medicine and Biology Society Career Achievement Award, and the IEEE Computational Intelligence Society Neural Network Pioneer Award. He served as the President of the International Neural Network Society in 2004, the Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING from 2001 to 2007, and a member of the Advisory Science Board of the Food and Drug Administration from 2001 to 2004. He is the Founding Editor-in-Chief of the IEEE REVIEWS IN BIOMEDICAL ENGINEERING. He has been heavily involved in conference organization and several IEEE society administrative committees. He chaired 78 Ph.D. and 61 master's student committees.