35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

A New Method of Concurrently Visualizing States, Values, and Actions in Reinforcement based Brain Machine Interfaces

Jihye Bae,[1] Luis G. Sanchez Giraldo,[1] Eric A. Pohlmeyer,[2]
Justin C. Sanchez,[2] *Senior Member, IEEE*, Jose C. Principe,[1] *Fellow, IEEE*

Abstract— **This paper presents the first attempt to quantify the individual performance of the subject and of the computer agent on a closed loop Reinforcement Learning Brain Machine Interface (RLBMI). The distinctive feature of the RLBMI architecture is the co-adaptation of two systems (a BMI decoder in agent and a BMI user in environment). In this work, an agent implemented using Q-learning via kernel temporal difference (KTD)(λ) decodes the neural states of a monkey and transforms them into action directions of a robotic arm. We analyze how each participant influences the overall performance both in successful and missed trials by visualizing states, corresponding action value Q, and resulting actions in two-dimensional space. With the proposed methodology, we can observe how the decoder effectively learns a good state to action mapping, and how neural states affect the prediction performance.**

I. INTRODUCTION

Research in brain machine interfaces (BMIs) is a multidisciplinary effort involving fields such as neurophysiology and engineering. Developments in this area have a wide range of applications, especially for those with neuromuscular disabilities for whom BMIs may become a significant aid. BMIs allow direct communication between the central nervous system and a computer that controls an external device such as a prosthetic arm for disabled individuals. Neural decoding of motor signals is one of the main tasks that needs to be executed by the BMI and it requires methods that adapt and adjust to subtle neural variations.

Reinforcement learning (RL) is a general framework for system adaptation to novel environments. It is similar to the way biological organisms interact with the environment and learn from experience. In RL, it is possible to learn only from the information given by the environment, and thus there is no need for a desired signal as in supervised learning. Therefore, RL is a well suited candidate for the neural decoding stage in BMI applications.

A BMI architecture based on reinforcement learning (RLBMI) is introduced in [1]; successful applications of this approach can be found in [2], [3], [4]. The key idea of RLBMI is co-adaptation between two intelligent systems: the BMI decoder in the agent and the BMI user in the environment (Figure 1). Both systems learn how to earn rewards based on their joint behavior. The BMI decoder learns a control strategy based on the user's neural state and perform actions in goal directed tasks that affect the state of the external device in the environment. In addition, the user learns the task based on the state of the external device. Notice that both systems act symbiotically by sharing the external device to complete their tasks, and this co-adaptation allows for continuous synergistic adaptation between the BMI decoder and the user even in changing environments.

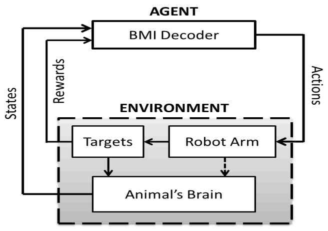

Fig. 1. RLBMI architecture.

Because the two intelligent systems work in closed loop feedback it is not easy to tease apart their influence on the overall performance. Therefore, an important task that has not been addressed so far is the analysis of the effects that one system has on the other. This analysis can provide valuable information to elucidate ways to make the learning algorithm robust to errors incurred by the user, and possibly develop new algorithms to translate brain states and to derive the policy. The performance analysis presented in this paper is a first step towards developing decoding strategies that can judge the reliability of the input states.

For the BMI decoder we chose Q-learning via kernel Temporal Difference (KTD)(λ) [3], [5]. The behavior task is a reaching task using a robotic arm, in which the decoder controls the robot arm's action direction by predicting the monkey's intent based on its neuronal activity. To tease apart the influence of the subject and of the agent in task completion, the distribution of the neural states along with the learned Q function are visualized on a two dimensional subspace using principal component analysis (PCA) [6]. The policy implemented by the BMI decoder, which can be derived from the corresponding Q values, is also displayed.

In the projected state space, we can pinpoint the location of incorrect action predictions as well as the distributions of successful trials to provide an indicator of the reliability of neural states. These results help quantify the effects of each learning system in the RLBMI.

This work is partially supported by DARPA Contract N66001-10-C-2008.
[1] Jihye Bae, Luis G. Sanchez Giraldo, and Jose C. Principe are with Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida 32611 USA jihyebae@ufl.edu, principe@cnel.ufl.edu
[2] Eric A. Pohlmeyer and Justin C. Sanchez are with Department of Biomedical Engineering, University of Miami, Coral Gables, Florida 33146 jcsanchez@miami.edu

II. Reinforcement Learning for Brain Machine Interfaces (RLBMI)

For a given a Markov decision process (MDP) with states $x(t) \in \mathcal{X}$, actions $a(t) \in \mathcal{A}$, a reward function $r(t) \in \mathbb{R}$, and a state transition probability $\mathcal{P}^{a}_{xx'} = P(x(t+1) = x'|x(t) = x, a(t) = a)$, RL finds an optimal policy $\pi^* : \mathcal{X} \rightarrow \mathcal{A}$ which maximizes the expected reward of all future actions given the current knowledge of the environment.At time step t, the agent receives the representation of the environment's state $x(t)$ as input, and it selects an action $a(t)$ accordingly. By performing the selected action $a(t)$, the agents receives a reward $r(t+1)$, and the state of the environment changes from $x(t)$ to $x(t+1)$. The new state $x(t+1)$ evolves based on the state transition probability $\mathcal{P}^{a}_{xx'}$ given the action $a(t)$ and the current state $x(t)$. At the new state $x(t+1)$ the same process repeats.

In the RLBMI architecture, the decoder updates the state of the environment based on the subject's neural activity and the received rewards. The agent updates its decoder of brain activity, and at the same time the subject produces the corresponding brain activity. Through iterations, the controller learns how to correctly translate neural states into action direction pairs. Note that the subject has no direct access to actions, and the decoder must interpret the subject's brain activity correctly to facilitate the rewards [1].

A. Agent

Q-learning via kernel Temporal Difference (KTD)(λ) is applied to the neural decoding. Estimating value functions is a fundamental step to finding an optimal policy. Typically in RL, temporal difference algorithms approximate the value function using a parametrized family of functions. A non-parametric variant, KTD(λ) [3], [5], is obtained by approximating the value function using a function $f \in \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) with reproducing kernel $\kappa(\cdot,\cdot)$, such that $f(x(t)) = \langle f, \phi(x(t))\rangle = \langle f, \kappa(x(t),\cdot)\rangle, \forall f \in \mathcal{H}$ with a mapping function $\phi(x(t)) : \mathcal{X} \rightarrow \mathcal{H}$ and a positive definite function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. The KTD($\lambda$) update rule is

$$f \leftarrow f + \eta \sum_{t=1}^{m} (\langle f, \phi(x(t+1))\rangle - \langle f, \phi(x(t))\rangle) \sum_{k=1}^{t} \lambda^{t-k} \phi(x(k)), \tag{1}$$

where η is the stepsize and m is the number of steps until the target is reached (steps per trial).

We can find an optimal policy π^* based on the action value function $Q(x(t), a(t)) = E[\mathbf{R}(t)|x(t), a(t)]$ where the return is defined by an *infinite-horizon* discounted model $\mathbf{R}(t) = \sum_{k=0}^{\infty} \gamma^k r(t+k+1)$ with discount factor $0 < \gamma < 1$. Since value functions represent the expected cumulative reward given a policy, we can say that the policy π is better than the policy π' when the policy π gives greater expected return than the policy π'. In other words, $\pi \geq \pi'$ if and only if $Q^{\pi}(x) \geq Q^{\pi'}(x)$ for all $x \in \mathcal{X}$. Therefore, the optimal action value function Q can be obtained by $Q^*(x(t), a(t)) = \max_{\pi} Q^{\pi}(x(t), a(t))$. Q-learning [7] estimates the optimal value function online. Based on Q-learning, the update rule for KTD(λ) (1) can be written as

$$\begin{aligned} f \leftarrow f + \eta \sum_{t=1}^{m} (r(t+1) + \gamma \max_{a} y(x(t+1), a) + \\ - y(x(t), a(t))) \sum_{k=1}^{t-1} \lambda^{t-k} \phi(x(k)). \end{aligned} \tag{2}$$

For single-step prediction problems ($m = 1$), (2) yields single updates for Q-learning via KTD(λ) of the form

$$Q_n(x(t)) = \eta \sum_{j=1}^{t-1} e_{TD}(j) I_k(j) \kappa\langle x(t), x(j)\rangle. \tag{3}$$

Here, $Q_n(x(t)) = Q(x(t), a = n)$ denotes discrete actions, $e_{TD}(t)$ denotes the TD error defined as $e_{TD}(t) = r_n + \gamma Q_m(x(t+1)) - Q_n(x(t))$, and $I_k(t)$ is an indicator vector with the same size as the number of outputs. This means that only the kth entry of the vector is set to 1 and the rest of the entries are 0. The selection of the action unit k at time t can be based on a greedy method. Therefore, only the weight (parameter vector) corresponding to the winning action gets updated. Recall, that the reward r_n corresponds to the action selected by the current policy with input $x(t)$ because it is assumed that this action causes the next input state $x(t+1)$.

B. Environment

During pre-training, a marmoset monkey has been trained to perform a reaching task with targets at two spatial locations (A or B trial). After one target is assigned, the trial starts with a beep. To conduct the trial during the user training phase, the monkey is required to steadily place its hand on a touchpad for $700 \sim 1200ms$. This action produces a *go* beep that is followed by one of the two target LEDs being lit on (A trial: red light at left direction or B trial: green light at right direction). The robot arm goes up to home position; i.e. the center position between the two targets. Its gripper shows an object (food reward such as waxworm or marshmallow for A trial and undesirable object, wooden bead, for B trial). For A trial, the monkey should move its arm to a target sensor within $2000ms$, and for B target, the monkey should hold its arm on the initial sensor for $2500ms$. If the monkey successfully conducts the task, the robot arm moves to the assigned location, and the target LED light blinks. The monkey gets a food reward.

After the monkey is trained to perform the assigned task properly, a micro-electrode array (16-channel tungsten microelectrode arrays, Tucker Davis Technologies, FL) is surgically implanted under isoflurane anesthesia and sterile conditions. In the closed loop RLBMI, neural states from the motor cortex (M1) are recorded. These neural states become inputs to the neural decoder. All surgical and animal care procedures are consistent with the National Research Council Guide for the Care and Use of Laboratory Animals and are approved by the University of Miami Institutional Animal Care and Use Committee. In the closed loop experiment, after the initial holding time that produces the *go* beep, the robotic arm's position is updated based solely on the monkey's neural states, and the monkey is not required to perform any movement unlike the user pre-training sessions.

III. METHOD

The decoder provides a mapping between neural states and actions which updates the robot arm's position in the environment. This change will influence the user's subsequent neural states because of the visual feedback involved in the process. In other words, the input to the BMI decoder is the user's neural states, which can be considered to be the user's output. Likewise, the action directions of the external device are the decoder's output and because of the visual feedback they can also be considered as the input to the user.

In the two-target reaching task, the decoder contains two output units representing the functions $Q(x, a = \text{left})$ and $Q(x, a = \text{right})$. The policy is determined by selecting the action associated with one of these units based on their Q values. The performance of the decoder is commonly evaluated in terms of success rate by counting the successful trials which reach the desired targets, along with the changes in the TD error or the Q values. However, these criteria are not well suited to understand how the two intelligent systems interact during learning. For instance, if there is a change in performance or an error in the decoding process it is hard to tell which of the two subsystems is more likely to be responsible for it.

Another added difficulty in evaluating the user's output is that the neural states are high dimensional vectors. In this sense, we want to apply an unsupervised dimensionality reduction technique to produce an output that can be visualized and easily interpreted. We have found out that principal component analysis (PCA) on the set of observed neural states is sufficient for the goal of this analysis. PCA is a well known method to transform data to a new coordinate system based on eigenvalue decomposition of a data covariance. Let $\mathbf{X} = [x(1),\ x(2),\ \cdots,\ x(t)]^\top$ be the data matrix containing the set of observed states during the closed loop experiment until time t. A transformed dataset $\mathbf{Y} = \mathbf{X}\mathbf{W}$ can be obtained by using the transformation matrix $\mathbf{W}$, which corresponds to the matrix of eigenvectors of the covariance matrix $n^{-1}\mathbf{X}^\top\mathbf{X}$. Without loss of generality we assume that the data $\mathbf{X}$ has zero mean. The distribution of states up to time t can be visualized by projecting the high dimensional neural states into two dimensions using the first two largest principal components.

In this two-dimensional space of projected neural states, we can also show the relation with the decoder by computing the outputs of the units associated with each one of actions and displaying them as contour plots. A set of two-dimensional space locations $\mathbf{Y}_{\text{grid}}$ evenly distributed on the plane can be projected in the high dimensional space of neural states as $\hat{\mathbf{X}} = \mathbf{Y}_{\text{grid}}\mathbf{W}^\top$. Let $Q_n^{(t)}$ be the n unit from the decoder updated using (3) at time t. We can compute the estimated Q values at a point y on the two dimensional plane using $\hat{Q}^{(t)}(\hat{x} = \mathbf{W}y, a = n)$. In this way, we can extrapolate the possible outputs that the decoder would produce in the vicinity of the already observed data points. Furthermore, the final estimated policy can be obtained by selecting the action corresponding to the unit with the highest Q value.

Here, we visualize the neural states, corresponding Q values, and the policy π related to the final performance. Thus, the final learned decoder $\hat{Q}^{(T)}$ and all the neural states $\mathbf{X}$ are utilized; that is, $t = T$ and $\mathbf{X}$ is size $T \times d$ where d is the dimension of the neural state vectors. Notice that the proposed method can also be applied at any stage of the learning process; we can observe the behavior of two systems at any intermediate time by using the subset of neural states that have been observed as well as the learned decoder up to this time.

IV. EXPERIMENTAL RESULTS

A closed loop RLBMI using Q-learning via the KTD algorithm is implemented. During the real-time experiment, 14 neurons are obtained from 10 electrodes. The neural states are represented by the firing rates on a $2sec$ window following the *go* signal. The overall performance is evaluated by checking whether the robotic arm reaches the assigned target or not. Once the robot arm reaches the target, the decoder gets a positive reward $+1$, otherwise, it receives negative reward -1.

For KTD(λ), the Gaussian kernel $\kappa(x(i), x(j)) = exp(-\|x(i) - x(j)\|^2/2h^2)$ is employed. The kernel size h is automatically selected based on the history of inputs. Note that in the closed loop experiments, the dynamic range of states varies from experiment to experiment, consequently, the kernel size needs to be readjusted each time a new experiment takes place and cannot be determined before hand. Therefore, we use the values of the distances between all the previously observed states to select the kernel size. At each time, from all squared distances between pairs of previously seen input states, we can obtain an estimate of the mean distance, and this value is also averaged along with past kernel sizes to assign current kernel size. In addition, the initial error is set to zero, and the first input state vector is assigned as the first unit's center. Normalization of the input neural states is not applied, and a stepsize $\eta = 0.5$ is used. Moreover, we consider $\gamma = 1$ and $\lambda = 0$ since our experiment performs single step trials in (2).

Figure 2 shows the performance of decoder for 2 experiments; The first experiment (left column) has a total of 20 trials (10 A trials and 10 B trials); the overall success rate is 90%. Only the first trial for each target is mis-assigned. The second experiment (right column) has a total of 53 trials (27 A trials and 26 B trials), with overall success rate of $41/53$ (around 77%). Although the success rate of the second experiment is not as high as the first experiment, both experiments show that the algorithm provides an appropriate neural state to action map. Even though there is variation among the neural states within each experiment, the decoder adapts well to minimize the TD error, and the Q values converge to the desired values for each action.

Figure 3 presents the visualization of the distribution of the 14 dimensional neural states projected into two dimensions

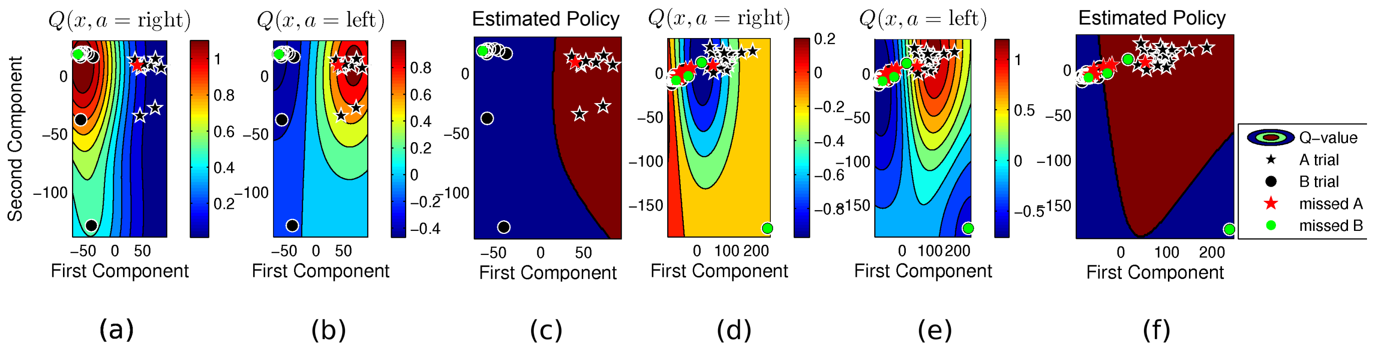

Fig. 3. The estimated Q values (a, b, d, e) and resulting policy (c, f) for the projected neural states using PCA from experiment 1 (a, b, c) and experiment 2 (d, e, f). (a) and (d) show the Q values for "right" direction, and (b) and (e) show the Q values for "left" direction.

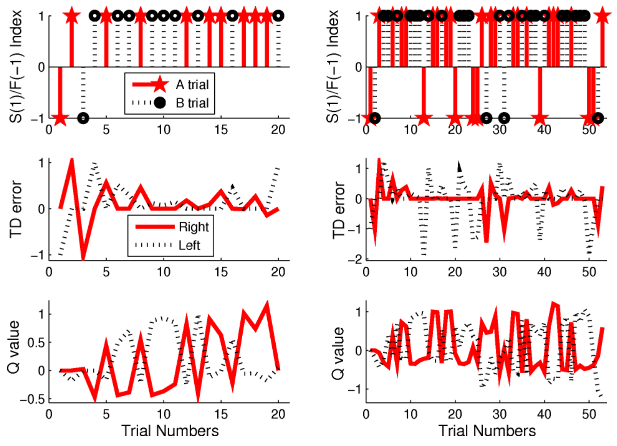

Fig. 2. Performance of Q-learning via KTD in the closed loop RLBMI performed by a monkey for experiment 1 (left) and experiment 2 (right); The success (+1) and failure (-1) index of each trial (top), the change of TD error (middle), and the change of Q values (down).

as described in the previous section. The corresponding contour levels are computed based on the Q values that are estimated using the learned decoder from the closed loop experiment. In addition, we provide the partition for left and right actions in the projected two dimensional space, which corresponds to the final policy derived from the estimated Q values. The projection shows that the neural states from the two classes are separable. As we expected, the Q values for each direction have higher values on regions occupied by the corresponding neural states. For example, the Q values for the right direction have larger values for the areas filled by the states corresponding to B trial. This is confirmed by showing the partitions achieved by the resulting policy.

During the training session, the success rates are highly dependent on the monkey's performance. Most of the times when the agent predicts the wrong target, it is observed that the monkey is distracted, or it is not interacting with the task properly. We are also able to see this phenomenon from the plots: the failed trials during the closed loop experiment are marked as red stars (missed A trials) and green dots (missed B trials). We can see that most of the neural states that are misclassified appear to be closer to the states corresponding to the opposite target in the projected state space. This supports the idea that failure during these trials is mainly due to the monkey's behavior and not to the decoder. From (c) and (f), the decoder can predict nonlinear policies. In addition, (f) shows that the system effectively learns and goes from an initially misclassified A trial (during the online closed loop), which is located near the border and right bottom areas, to a final decoder where the same state would be assigned to the right direction. It is a remarkable fact that the system adapts to the environment on-line.

V. Conclusion

Q-learning via KTD is applied to closed loop RLBMI experiments. For the reaching task described above, the algorithm is able to decode the monkey's neural state into the robot arm's position properly. To check the influences of the BMI decoder and the subject on the overall performance, we propose a methodology to visualize how each trial's neural state and the corresponding Q values are distributed. From the projected neural states, we can see that mislabelled neural states can cause the missed trials. Moreover, from the final estimated policy, we verify that the adaptation process works properly. Using this method, we can also observe how the adaptation process takes place sequentially in the decoder based on the incoming neural states. Note that instead of PCA, other unsupervised dimensionality reduction methods can be employed as long as there exist a suitable way to obtain pre-images that can be input to the decoder.

References

[1] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, "Coadaptive brain-machine interface via reinforcement learning," *IEEE Trans. on Biomedical Engineering*, vol. 56, no. 1, 2009.
[2] B. Mahmoudi, "Integrating robotic action with biologic perception: A brain machine symbiosis theory," Ph.D. dissertation, University of Florida, 2010.
[3] J. Bae, P. Chhatbar, J. T. Francis, J. C. Sanchez, and J. C. Principe, "Reinforcement learning via kernel temporal difference," in *EMBC*, 2011, pp. 5662 – 5665.
[4] E. A. Pohlmeyer, B. Mahmoudi, S. Geng, N. Prins, and J. C. Sanchez, "Brain-machine interface control of a robot arm using actor-critic rainforcement learning," in *EMBC*, 2012, pp. 4108 – 4111.
[5] J. Bae, L. Sanchez Giraldo, P. Chhatbar, J. T. Francis, J. C. Sanchez, and J. C. Principe, "Stochastic kernel temporal difference for reinforcement learning," in *MLSP*, 2011.
[6] I. Jolliffe, *Principal Component Analysis*. Springer, 2002.
[7] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.