33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

Reinforcement Learning via Kernel Temporal Difference

Jihye Bae, Pratik Chhatbar, Joseph T. Francis, Justin C. Sanchez, Jose C. Principe

Abstract—**This paper introduces a kernel adaptive filter implemented with stochastic gradient on temporal differences, kernel Temporal Difference (TD)(λ), to estimate the state-action value function in reinforcement learning. The case λ=0 will be studied in this paper. Experimental results show the method's applicability for learning motor state decoding during a center-out reaching task performed by a monkey. The results are compared to the implementation of a time delay neural network (TDNN) trained with backpropagation of the temporal difference error. From the experiments, it is observed that kernel TD(0) allows faster convergence and a better solution than the neural network.**

I. Introduction

BRAIN machine interfaces (BMIs) are an innovation in neurotechnology, and have the potential to be a significant aid for those with neuromuscular disabilities. Neural decoding of sensory-motor signals is a necessary step toward this goal. Initial online implementations of BMIs have demonstrated the ability of control for cursor positions or a single robotic movement depending on sensory-motor signal was reported in [2, 3, 4]. Further developments for controlling a real prosthetic arm or sequence of movements are presented in [5, 6, 7, 8, 9]. Instead of focusing on motor movement reconstruction, current research focuses on the development of a system for real-time interaction with the environment and to achieve higher level performance [10, 11, 12].

Machine learning algorithms have been used for motor movement or state decoding. Important features to evaluate are the learning speed and stability. Supervised learning is one popular approach in BMI [13], but it requires a desired signal in order to learn. Even in this case due to brain plasticity frequent calibration (retraining) is necessary. In contrast, reinforcement learning (RL) algorithms are a general framework for adapting the system to a novel environment, and this aspect is similar to how biological organisms interact with environment and learn from experience. Moreover, RL is able to learn only with rewards from the environment. Unlike supervised learning, RL adapts without the information of desired signal. Therefore, RL is well suited for the BMI decoding paradigm.

Jihye Bae and Jose C. Principe are with the Department of Electrical and Computer Engineering, P.O. Box 116130 NEB 486, Bldg #33, University of Florida, Gainesville, FL 32611 USA (e-mail: jihyebae@ufl.edu, principe@cnel.ufl.edu).

Pratik Chhatbar and Joseph T. Francis are with Department of Physiology, SUNY Downstate Medical Center, Brooklyn, NY 11203 USA (e-mail: pratikchhatbar@gmail.com, joey199us@gmail.com).

Justin C. Sanchez is with Department of Biomedical Engineering, University of Miami, Miami, FL 33146 USA (e-mail: jcsanchez@miami.edu).

Previous work in an RL framework used a time delay neural network (a delay line followed by a multilayer perceptron - MLP) trained with back-propagation (BP) on the time difference error to find the optimal mapping between neural states and movement directions [11, 12, 14]. The application was able to find the optimal neural-to-motor mapping without explicit desired information, i.e., the system successfully learned from interaction with the environment. However, the BP algorithm is sensitive to initialization and can become stuck in local minima of the cost function. This is a well known issue with BP that is particularly concerning for on-line learning, so this application required a search over multiple initial conditions [15].

In the last two decades there has been a growing interest in the area of kernel methods within the machine learning community. The introduction of the support vector machine algorithm for pattern recognition [16] rekindled the interest on this topic. One of the major appeals of kernel methods is the ability to handle nonlinear operations on the data by indirectly computing an underlying nonlinear mapping to a space (Reproducing Kernel Hilbert Spaces (RKHS)) where linear operations can be carried out. The linear solution corresponds to a universal approximation in the input space, and many of the related optimization problems can be posed as convex (no local minima) with algorithms that are still reasonably easy to compute (using the kernel trick [17]). A kernel approach to TD learning in RL employing Gaussian processes and kernel recursive least squares is introduced in [22]. Recent work in adaptive filtering has shown the usefulness of kernel methods in solving nonlinear adaptive filtering problems [18]. Although the standard setting for these algorithms differs from RL, elements such as the adaptive gain on an error term can be well exploited in solving RL problems as we shall see below.

In this paper, we propose a novel approach using a kernel trained with the least mean square (LMS) algorithm and TD learning called Kernel Temporal Difference (TD)(λ) for estimation of state-action or Q values. In this paper only the case λ=0 will be studied in detail. Our algorithm shares many features with the kernel least mean square algorithm (KLMS) presented in [19] except that the error is obtained using the temporal difference framework, i.e. the difference of consecutive outputs is used as the error, unless when the system is rewarded. To test the efficacy of the proposed method, the Kernel TD(0) is applied to a center-out reaching task performed by a monkey, and its performance is compared to a current implementation of RL using TDNN. Results show faster convergence and more stable solution with no initialization issues.

II. Reinforcement Learning for Brain Machine Interfaces

Reinforcement learning (RL) adapts a linear or nonlinear model from the interaction between an agent and the environment. RL maximizes the cumulative sum of future rewards in a sequence of actions [20]. Neural decoding finds an optimal functional mapping between neural states and action directions. To implement neural decoding with RL, the agent learns how to transfer the neural states into actions based on predefined reward values from the environment. Since there are two intelligent agents (the subject and the computer algorithm) in close loop feedback as shown in Figure 1 this is a cooperative game. In fact, the subject has no direct access to actions, and the agent must interpret the subject's brain activity correctly to facilitate the rewards [11].

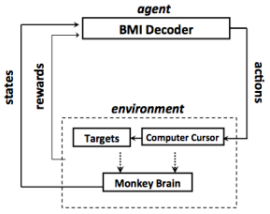

Fig. 1. The decoding structure of RL model in BMI [11].

The agent starts in a naive state (i.e. the parameters are random values), but the subject has been pre trained to receive rewards in the environment. The agent changes the cursor position at prescribed intervals based on the decoding of the subject's neural activity, and once it reaches the assigned target, the system and the subject earn the reward and the agent updates its decoder of brain activity. Through iteration, the agent learns how to correctly translate neural states into action direction (computer cursor position).

III. Kernel Temporal Difference

A. Reproducing Kernels and the Kernel LMS Algorithm

Here we briefly review the main concept of the new approach behind kernel method and the kernel least mean square (KLMS) derivation.

Let $\mathcal{X}$ be a set. For a positive definite function $\kappa: \mathcal{X}\times\mathcal{X}\to\mathbb{R}$ [17], there exists a Hilbert space $\mathcal{H}$ and a mapping $\phi: \mathcal{X}\to\mathcal{H}$ such that $\kappa(x,y)=\langle\phi(x),\phi(y)\rangle$. $\mathcal{H}$ is called a reproducing kernel Hilbert space (RKHS) satisfying

$$f(x)=\langle f,\phi(x)\rangle=\langle f,\kappa(x,\cdot)\rangle, \forall f\in\mathcal{H} . \quad (1)$$

The KLMS algorithm attempts to minimize the risk functional $E\left[(d-f(x))^2\right]$ by minimizing the empirical risk $J(f)=\sum_{i=1}^{N}(d(i)-f(x(i)))^2$ on the space $\mathcal{H}$ defined by the kernel κ. Using (1), we can rewrite

$$J(f)=\sum_{i=1}^{N}\left(d(i)-\langle f,\phi(x(i))\rangle\right)^2 \quad (2)$$

By differentiating the empirical risk $J(f)$ with respect to f and approximating the sum by the current difference (stochastic gradient), we can derive the following update rule:

$$f_i=f_{i-1}+\eta e(i)\phi(x(i)) \quad (3)$$

where $e(i)=d(i)-f_{i-1}(x(i))$ and $f_0=0$ which corresponds to KLMS algorithm [19].

B. Kernel Temporal Difference (TD)(λ)

As in [1], consider the multistep prediction problem in supervised learning, for which we have the observation of input sequence $x(1),x(2),\cdots,x(m)$ and desired d. Then, a system will produce a sequence of predictions $y(1),y(2),\cdots,y(m)$ based on the observed sequence. Notice that in general, the predicted output is a function of all previous inputs:

$$y(i)=f_i(x(1),x(2),\cdots,x(i)) . \quad (4)$$

Here, we assume that $y(i)=f(x(i))$ for simplicity. Let the function f belong to an RKHS $\mathcal{H}$ as in KLMS. By treating the observed input sequence and the desired as a sequence of pairs $(x(1),d),(x(2),d),\cdots,(x(m),d)$, we can obtain the updates of function f after the whole sequence of m inputs has been observed as

$$f\leftarrow f+\eta\sum_{i=1}^{m}\Delta f_i , \quad (5)$$

where $\Delta f_i=\left(d-\langle f,\phi(x(i))\rangle\right)\phi(x(i))$ are the instantaneous updates of the function f from input data based on (1).

By taking $d\triangleq y(m+1)$ and representing the error as $d-y(i)=\sum_{k=i}^{m}\left(y(k+1)-y(k)\right)$, we can arrive at

$$f\leftarrow f+\eta\sum_{i=1}^{m}\left(\langle f,\phi(x(i+1))\rangle-\langle f,\phi(x(i))\rangle\right)\sum_{k=1}^{i}\phi(x(k)) . \quad (6)$$

That is, $\Delta\tilde{f}_i=\langle f,\phi(x(i+1))-\phi(x(i))\rangle\sum_{k=1}^{i}\phi(x(k))$. Thus, generalizing for TD(λ) yields

$$\Delta\tilde{f}_i=\langle f,\phi(x(i+1))-\phi(x(i))\rangle\sum_{k=1}^{i}\lambda^{i-k}\phi(x(k)) , \quad (7)$$

and this approach will be called Kernel TD(λ).

C. Q –learning via Kernel TD(λ)

Off-policy RL using Q-learning [21] updates the state-action value function Q as follows

$$Q(x(i),a(i))\leftarrow Q(x(i),a(i))+\eta[r(i+1)+\gamma\max_{a}Q(x(i+1),a)-Q(x(i),a(i))] \quad (8)$$

to maximize the expected reward

$$R^{a}_{xx'}=E\{r(i+1)\,|\,x(i)=x,a(i)=a,x(i+1)=x'\} . \quad (9)$$

So, we can set the desired output as the cumulative reward. For the optimal trained system the output will equal the desired response, and the following relation will be satisfied:

$$d(i)=\sum_{k=0}^{\infty}\gamma^k r(i+k+1)=y(i)=r(i+1)+\gamma y(i+1) \quad (10)$$

where, γ is discount-rate parameter satisfying $0\le\gamma<1$ and r is a reward value. Therefore, based on TD error $(r(i+1)+\gamma y(i+1))-y(i)$, the generalized update for Q-learning via Kernel TD(λ) has the form

$$\Delta\tilde{f}_i=\left(r(i+1)+\langle f,\gamma\phi(x(i+1))-\phi(x(i))\rangle\right)\sum_{k=1}^{i}\lambda^{i-k}\phi(x(k)) . \quad (11)$$

In the case of λ=0, the contribution of each observed input to the update equation becomes

$$\Delta\tilde{f}_i = \left(r(i+1)+\gamma\left\langle f,\phi(x(i+1))\right\rangle-\left\langle f,\phi(x(i))\right\rangle\right)\phi(x(i)), \quad (12)$$

which in the case of single updates yields

$$Q_n(x(i)) = \eta\sum_{j=1}^{i-1} e_{TD}(j)I_k(j)\kappa(x(i),x(j)). \quad (13)$$

Here, for discrete actions $Q_n(x(i)) = Q(x(i), a=n)$ and $I_k(i)$ is an indicator vector with the same size as the number of outputs which has only the k th value as 1 based on the selected action unit k at time i based on ε-greedy approach

$$I_k(i) = \begin{cases} 1 & , k \text{ th action is chosen} \\ 0 & , \text{otherwise.} \end{cases} \quad (14)$$

Therefore, only the weight corresponding to the winning action gets updated. Also, $e_{TD}(i)$ is TD error defined as

$$e_{TD}(i) = r_n + \gamma Q_m(x(i+1)) - Q_n(x(i)). \quad (15)$$

Recall, that the reward r_n corresponds to the action selected by the current policy with input $x(i)$ because it is assumed that this action causes the next input state $x(i+1)$.

The structure of RLBMI via Kernel TD(λ) is shown in Figure 2.

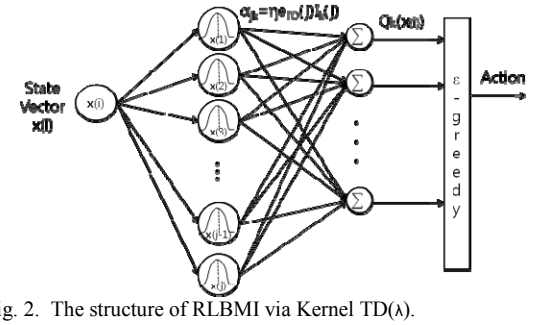

Fig. 2. The structure of RLBMI via Kernel TD(λ).

The input represents a neural state vector. The number of units (kernel evaluations) increase as new training data arrives, and each added unit is centered at the previous inputs $x(1), x(2), \cdots, x(i-1)$.

Each output component represents the possible action directions. Based on the outputs (Q-value), one action which has the maximum value is selected by ε-greedy method [21], and only the selected action unit has the updated weights.

IV. Data Recordings and Neural Decoding

A female bonnet macaque is trained for a center-out reaching task allowing 8 action directions. After about 80% success rate, microelectrode arrays are implanted in motor cortex (M1).

Animal surgery is performed under the Institutional Animal Care and Use Committee (IACUC) regulations and assisted by the Division of Laboratory Animal Resources (DLAR) at SUNY Downstate Medical Center. After that, 185 units obtained from 96 channels are used for the neural decoding.

The neural decoding aims at 2 targets (1 (right (4.4650, 23.3040)) and 5(left (-3.5350, 23.3040)) for 8 possible action directions. Every trial starts at the center point (0.4650, 23.3040). The distance from the center to a target is 4cm with target radius of 0.8~1cm (Figure 3).

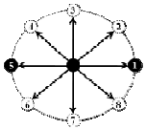

Fig. 3. The center-out reaching task for 2 targets (1 and 5) allowing 8 possible action directions (1~8).

In this implementation, only the successful trials are used for training (total number of 43 trials). The system performance is evaluated by checking whether the changed position reaches the assigned target or not.

V. Experiments and Results

The neural states are represented by the firing rates of 185 units over a 100ms window and their past 6 values, creating a 7 dimensional time embedding per channel; this amounts to an input state vector of 1295 dimensions. After input states are preprocessed by normalizing their dynamic range between -1 and 1, they are input to the system.

The output represents the 8 possible directions, and among the 8 outputs one action is selected by ε-greedy method [21]. Based on the selected direction, the computer cursor position is updated. Here, we test for a 1 step reaching task towards the assigned target. So, after the 1 step change of position, the reward is measured based on the Hamming distance to the target. Once the distance reaches the target, the agent gets rewarded (+0.6), otherwise it receives punishment (-0.6) [14].

Based on the selected action with exploration rate 0.01 and the reward value, the system is adapted by Kernel TD(λ) with $\gamma = 0.9$. In our case, $\lambda = 0$ is specially considered since our experiment performs single step updates per trial. Here, Gaussian kernel

$$\kappa(x,x') = \exp\left(-\|x-x'\|^2 / \left(2h^2\right)\right) \quad (16)$$

is used, and based on the distribution of squared distance between pairs of input states, kernel size $h = 7$ is applied. Also, stepsize $\eta = 0.3$ is selected based on [18]

$$\eta < \frac{N}{tr[G_\varphi]} = \frac{N}{\sum_{j=1}^{N}\kappa(x(j),x(j))} = 1. \quad (17)$$

The initial error is set to zero, and the first input vector is assigned as the first unit's center.

The performance is observed for 20 epochs (Figure 4).

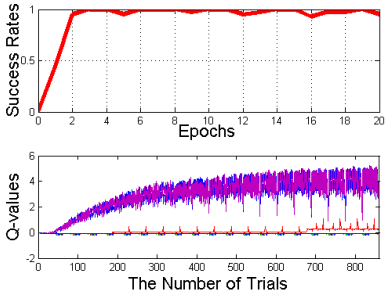

Fig. 4. The change of success rates (up) and Q-values (down) by Kernel TD(λ).

To observe the learning process, the predicted Q-values are displayed after each trial, and for each epoch, success rates are calculated (i.e. 1 epoch containing 43 trials). Notice how after 2 epochs Kernel TD(0) reaches around 100% success rate and the Q-values for the winning units become dominant after around 40 trials.

To compare the performance between Kernel TD(0) and TDNN, the success rates over 50 implementations are measured, and the average learning curves show in Figure 5.

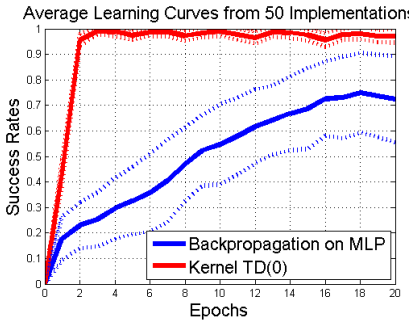

Fig. 5. The comparison of average learning curves from 50 implementations between Kernel TD(λ) and MLPs.

The solid line shows the average success rates and the dashed line shows the standard deviation of the performance. Since all the parameters are fixed over these 50 experiments, the confidence interval for Kernel TD(0) shows the effect of the random action selection for exploration, so the interval is narrow. However, with the TDNN a wider range of performance is observed, which supports the high dependence on initial values. From this result, we can confirm that Kernel TD(0) approach solves the issue of local minima on MLPs depending on the initializations.

Moreover, Kernel TD(0) reaches around 100% success rate after 2 epochs. In contrast, the success rate of TDNN slowly increases and never reaches the same performance.

VI. Conclusions and Future Work

A novel approach using stochastic gradient approximation (LMS) using the TD framework for the approximation of state-action value function Q in RL was applied to a center-out reaching task performed by a monkey. When compared to the TDNN, Kernel TD(0) showed better estimation of Q values along with performance improvements in terms of learning speed and smaller variance.

One characteristic of Kernel TD(λ) is that the size of filter increases as input data comes in. Thus, further experiment will focus on the improvement to control the growing structures of Kernel TD(λ) and the influence of the eligibility trace λ on problems involving the estimation of sequential actions.

Acknowledgment

This work was partially supported by DARPA Contract N66001-10-C-2008.

References

[1] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," Kluwer Academic Publishers, Boston, pp. 9-44, 1988.

[2] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. L. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons inprimates," *Nature*, vol. 408, pp. 361-365, Nov. 2000.

[3] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Brain-machine interface: Instant neural control of a movement signal," *Nature*, vol. 416, pp. 141-142, 2002.

[4] D. M. Taylor, S. I. H. Tilley, and A. B. Schwartz, "Information conveyed through brain-control: Cursor versus robot," *IEEE Trans. Neural Systems and Rehabilitation Eng.*, vol. 11, pp. 107-119, 2003.

[5] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Bio.*, vol. 1, no. 2, Nov. 2003.

[6] J. C. Sanchez, "From Cortical Neural Spike Trains to Behavior: Modeling and Analysis," Ph.D. dissertation, Dept. Biomedical Eng., Univ. of Florida, Gainesville, FL, 2004.

[7] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive Control Signals for Neural Prosthetics," *Science*9, vol. 305, no. 5681, pp. 258-262 Jul. 2004.

[8] M. A. Lebedev, J. M. Carmena, J. E. O`Doherty, M. Zacksenhouse, C. S. Henriquez, J. C. Principe, and M. A. L. Nicolelis, "Cortical Ensemble Adaptation to Represent Velocity of an Artificial Actuator Controlled by a Brain-Machine Interface," *The Journal of Neuroscience*, vol. 25, no. 19, pp. 4681-4693, May 2005.

[9] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, pp. 164-171, Jul. 2006.

[10] 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, pp. 1098-1101, Jun. 2008.

[11] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, "Co-adaptive Brain-Machine Interface via Reinforcement Learning," *IEEE Trans. Biomedical Engineering*, vol. 56, no. 1, Jan, 2009.

[12] B. Mahmoudi, "Integrating Robotic Action with Biologic Perception: A Brain Machine Symbiosis Theory," Ph.D. dissertation, Dept. Biomedical Eng., Univ. of Florida, Gainesville, FL, 2010.

[13] S. Haykin, *Neural Networks: a comprehensive foundation*. Maxwell Macmillan Canada, 1994.

[14] J. C. Sanchez, A. Tarigoppula, J. S. Choi, B. T. Marsh, P. Y. Chhatbar, B. Mahmoudi, and J. T. Francis, "Control of a Center-Out Reaching Task using a Reinforcement Learning Brain-Machine Interface," submitted for publication.

[15] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, NJ: Pearson Education, Inc., 2009.

[16] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*. 5th Annual ACM Workshop on COLT, PA: ACM Press, pp. 144-152, 1992.

[17] B. Scholkopf and A.J. Smola, *Learning with Kernels.* The MIT Press, 2002.

[18] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, 2010.

[19] W. Liu, P. P. Pokharel, and J. C. Principe, "The Kernel Least-Mean Square Algorithm," *IEEE Trans. Signal Processing*, vol. 56, no. 3, Feb. 2008.

[20] R. S. Sutton and A. G. Barto, R*einforcement Learning: An Inroduction*. Cambridge, MA: The MIT Press, 1998.

[21] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, vol. 8, num. 3-4, Springer, 1992, pp.279-292.

[22] Y. Engel, "Algorithms and Representations for Reinforcement Learning," Ph.D. dissertation, School of Eng. and Computer Science, Hebrew Univ., Jerusalem, 2005