# Correntropy Kernel Temporal Differences for Reinforcement Learning Brain Machine Interfaces

Jihye Bae,[1] Luis G. Sanchez Giraldo,[1] Joseph T. Francis,[2] Jose C. Principe[1]

*Abstract*— **This paper introduces a novel temporal difference algorithm to estimate a value function in reinforcement learning. This is a kernel adaptive system using a robust cost function called correntropy. We call this system correntropy kernel temporal difference (CKTD). This algorithm integrates $Q$-learning with CKTD to find a proper policy introducing $Q$-learning via CKTD. This method was tested with a synthetic problem and its robustness under a changing policy was quantified. The same algorithm was applied to the decoding of a monkey's neural state in a reinforcement learning BMI (RLBMI) in a center-out reaching task. The results show the potential advantage of the proposed algorithm in the RLBMI framework.**

## I. INTRODUCTION

Brain machine interfaces have been a subject of active research because of their potential for a wide range of applications once they reach maturity. Special efforts have been directed towards developing technologies that will ultimately help overcome neuromuscular disabilities [1], [2]. Neural decoding of motor signals is one of the main tasks that needs to be executed by BMIs. The main goal of neural decoding is to characterize the electrical activity of groups of neurons, that is, to identify neural patterns that correlate with a given behavior task. For this, we need a learning system which can identify the patterns in an autonomous way. This process is a fundamental step towards the design of prosthetic devices that are directly controlled by brain.

RL is one of the representative learning schemes which can adapt and adjust to subtle neural variations. In RL, there is no need for off line training sessions since the system learns directly and continuously from the environment through reinforcement, so RL is well suited for the neural decoding stage of a BMI application.

A BMI architecture based on reinforcement learning (RLBMI) was introduced in [3], and successful applications of this approach can be found in [4], [5]. In the RLBMI structure (Figure 1), the agent learns how to translate the neural states into actions based on reward values from the environment. In fact, the BMI user has no direct access to the actions, and the agent, or bmi decoder, must interpret the user's brain activity correctly to facilitate the rewards [3]. 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.

For the agent, the proper decoding of the neural states is essential to accurately control the external device that interacts with the physical environment. $Q$-learning via kernel temporal difference (Q-KTD)($\lambda$) [4] has been applied to the agent for a neural decoding problem in reaching tasks, and the algorithm's capability to find the proper neural state to action directions has been quantified. In Q-KTD($\lambda$), temporal difference (TD) learning integrated with kernel methods or kernel temporal difference (KTD)($\lambda$) [4] is used to estimate an action value function $Q$, and the estimated value function is used to find a proper policy based on $Q$-learning.

Note that one of the most popularly utilized figures of merit in TD methods is the mean square error (MSE), in fact, KTD($\lambda$) uses this criterion. However, in RL, the choice of the cost function has been under appreciated, because it depends on the role of the value function; when the goal is to obtain a desired controller, and the value estimation is only used as a sub-routine of the algorithm, the accuracy in the approximation of the value function may not be as important as the ordering for action selection. Therefore, MSE may not be the most meaningful choice [6].

In addition, since the policy is not fixed in Q-KTD($\lambda$), it is required that the system explores the environment and learns under changing policies. The system should respond accordingly and be able to disregard large changes that may result from exploration. Therefore, a cost function which can bring about a robust system is preferred.

Correntropy is a generalized correlation measure between two random variables first introduced in [7]. Maximum correntropy criterion (MCC) has been applied to obtain robust methods for adaptive systems in supervised learning [8], [9]. Using MCC, a system can be adapted in such a way that a similarity measure between desired and predicted signals is maximized. In particular, kernel maximum correntropy (KMC), which is a blend between kernel least mean square (KLMS) [10] and MCC, was proposed in [9]. The basic idea of KMC is that input data is transferred to an RKHS using a nonlinear mapping function, and MCC is applied as a cost function to minimize the error. It was shown that the KMC accurately approximates nonlinear systems, and it was able to reduce the detrimental effects of various types of noise in comparison with the MSE criterion.

In this paper, we introduce a novel value function approximation algorithm by integrating the KTD algorithm with the

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`, `sanchez@cnel.ufl.edu`, `principe@cnel.ufl.edu`

[2] Joseph T. Francis is with Department of Physiology and Pharmacology and the Robert F. Furchgott Center for Neural & Behavioral Science, SUNY Downstate Medical Center, Brooklyn, NY 11203 USA `joe.francis@downstate.edu`

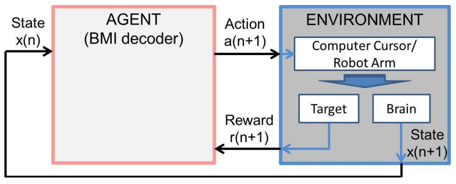


Fig. 1. The decoding structure of RLBMI [3].

maximum correntropy criterion. We call this algorithm the correntropy kernel temporal difference (CKTD). We extend CKTD to estimate a policy based on $Q$-learning, and this gives rise to the $Q$-learning via correntropy kernel temporal difference (Q-CKTD) algorithm. We explore how the new cost function influences the performance on RL problems including open loop RLBMI experiments.

## II. CORRENTROPY KERNEL TEMPORAL DIFFERENCES

### A. *Maximum Correntropy Criterion*

Correntropy is defined in terms of inner products of vectors in a kernel feature space $C(X,Y) = E[\mathcal{G}(X-Y)]$ where $X$ and $Y$ are two random variables, and $\mathcal{G}$ is a shift invariant kernel [7]. Here, we employ the Gaussian kernel. Correntropy can be set as a cost function [8]

$$J = E[\mathcal{G}(e)] \approx \frac{1}{N}\sum_{n=1}^{N}\mathcal{G}(e(n)), \tag{1}$$

where $e$ represents an error signal: $e(n) = d(n) - y(n)$ in supervised learning. For a system described by parametric mapping $y = f(x|\theta)$, the parameter $\theta$ can be adapted such that the correntropy of the error signal is maximized. This is called the maximum correntropy criterion (MCC)

$$\text{MCC} = \max_{\theta}\sum_{n=1}^{N}\mathcal{G}(e(n)). \tag{2}$$

### B. *Correntropy Kernel Temporal Differences*

Using the ideas of both kernel least mean square (KLMS) [10] and maximum correntropy criterion [8], kernel maximum correntropy (KMC) is introduced in supervised learning [9]. To maximize the error correntropy, we can use stochastic gradient ascent on the new cost function in the feature space. In KMC, the gradient of the cost function can be expressed as follows:

$$\nabla J_n = \frac{\partial C(d(n),y(n))}{\partial f} = \frac{1}{h_c^2}e(n)\mathcal{G}(e(n))\phi(x(n)), \tag{3}$$

$h_c$ is the correntropy kernel size. Also, the estimated KMC function at time $n+1$ is obtained as

$$f \leftarrow f + \eta\sum_{i=1}^{n}\left[exp\left(\frac{-e(i)^2}{2h_c^2}\right)e(i)\phi(x(i))\right]. \tag{4}$$

In a multi-step prediction problem, given the observed sequence of input-output pairs $(x(1),d)$, $(x(2),d)$, $\cdots$, $(x(m),d)$, we look for a function $f \in \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), such that $f(x(n)) = \langle f, \phi(x(n))\rangle = \langle f, \kappa(x(n),\cdot)\rangle, \forall f \in \mathcal{H}$ approximating the desired output $d \approx y(n) = f(x(n))$.

In the supervised setting, the function $f$ can only be updated once we have observed the whole sequence of $m$ inputs because $d$ only becomes available at time $m$. This yields updates for $f$ of the form $f \leftarrow f + \sum_{n=1}^{m}\Delta f_n$, where $\Delta f_n = \eta(d - \langle f, \phi(x(n))\rangle)\phi(x(n))$. By taking $d \triangleq y(m+1)$, the error can be written as

$$d - y(n) = \sum_{k=n}^{m}(y(k+1) - y(k)), \tag{5}$$

and we can obtain an update for $f$ without having observed the desired signal $d$ [11].

Thus, in the multistep prediction problem, the temporal difference (TD) error can be linked to the KMC algorithm by using the recursion (5);

$$f \leftarrow f + \eta\sum_{n=1}^{m}\left[exp\left(\frac{-(\sum_{k=n}^{m}e_{TD}(k))^2}{2h_c^2}\right)\right.$$
$$\left.\times\sum_{k=n}^{m}e_{TD}(k)\phi(x(n))\right], \tag{6}$$

where the $e_{TD}(k)$ represents the temporal difference (TD) error defined as $y(k+1) - y(k)$. In the case of $\lambda = 0$, the correntropy kernel temporal difference (CKTD) update rule can be derived as follows

$$f \leftarrow f + \eta\sum_{n=1}^{m}\left[exp\left(\frac{-e_{TD}(n)^2}{2h_c^2}\right)e_{TD}(n)\phi(x(n))\right]. \tag{7}$$

This equation also satisfies (6) in the case of single step predictions ($m = 1$). Note that compared to the KTD(0) update rule in [4], CKTD contains an extra weighting term which is the exponential of the negative squared error.

### C. *Q-learning via Correntropy Kernel Temporal Differences*

The basic idea of $Q$-learning is that when the action value $Q$ is close to the optimal action value $Q^*$, the policy, which is greedy with respect to all action values for a given state, is close to optimal; that is, $\pi \geq \pi'$ if and only if $Q^{\pi}(x,a) \geq Q^{\pi'}(x,a)$ for all states $x \in \mathcal{X}$ and all actions $a \in \mathcal{A}$. Therefore, the optimal action value function $Q$ can be obtained by $Q^*(x(n),a(n)) = \max_{\pi} Q^{\pi}(x(n),a(n))$.

Here, we can employ CKTD to estimate the action value function $Q$ which is defined as $Q(x(n),a(n)) = E[\mathbf{R}(n)|x(n),a(n)]$, where $\mathbf{R}(n)$ is the *infinite-horizon* discounted model defined as $\sum_{k=0}^{\infty}\gamma^k r(n+k+1)$ for $0 < \gamma < 1$. Therefore, the update rule for $Q$-learning via CKTD (Q-CKTD) is given by

$$f \leftarrow f + \eta\sum_{n=1}^{m}[r(n+1) + \gamma\max_{a}Q(x(n+1),a)$$
$$- Q(x(n),a(n))\phi(x(n))], \tag{8}$$

where the $Q$ represents the estimated action value by CKTD; a discrete action $Q(x(n), a = l)$ can be computed as

$$Q(x(n), a = l) = \eta \sum_{i=1}^{n-1} [exp\left(-e_{TDr}(i)^2/2h_c^2\right) \\ \times e_{TDr}(i) I_k(i) \kappa\langle x(n), x(i)\rangle], \quad (9)$$

where $e_{TDr}(i)$ denotes a TD error defined as $e_{TDr}(i) = r(i+1) + \gamma \max_a Q(x(i+1), a) - Q(x(i), a(i) = k)$. Recall that the reward $r(i+1)$ 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 $x(i+1)$.

Here, $I_k(i)$ is an indicator vector with the same size as the number of outputs; only the $k$th 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 $i$ can be based on an $\epsilon$-greedy method. Therefore, only the weight (parameter vector) corresponding to the winning action gets updated.

## III. EXPERIMENTAL RESULTS

### A. Mountain Car Task

We first carry out experiments on a famous episodic task in control problems known as the "Mountain-car task." There is a car driving along a mountain track, and the goal of this task is to reach the top of the right side hill. The challenge in this task is that there are regions near the center of the hill where maximum acceleration of the car is not enough to overcome the force imposed by gravity. Thus, if the system simply tries to maximize short term rewards, it would fail to reach the goal. In this case, the only way to reach the goal is to first accelerate backwards, even though it is further away from the goal, and then drive forward with full acceleration. This is a representative example to evaluate the system's capability in finding a proper policy to achieve a goal in RL.

The details of the model are based on [12]. The system undergoes 30 trials to learn the policy. At each trial, the initial states are drawn randomly from $-1.2 \leq p \leq 0.5$ and $-0.07 \leq v \leq 0.07$ where $p$ and $v$ are car position and velocity respectively. The system is initialized when the first trial starts, and each trial has $10^4$ maximum number of steps. At each trial, the number of steps is counted, and it is averaged over the 30 trials and 50 Monte Carlo runs. For each Monte Carlo run, the same set of 30 initial values are used. For the $\epsilon$-greedy method, we vary the exploration rate to confirm how the system learns under changing policy. We start with a totally random policy $\epsilon = 1$. This value is kept until 200th step, and then it is switched to $\epsilon = 0$. When the exploration rate $\epsilon$ is 0, the performance shows exactly what the system has been able to learn from random exploration.

With the optimal parameters $\eta = 0.3$, $h = 0.2$, and $h_c = 3$, Q-CKTD shows a mean and standard deviation of $349.87 \pm 368.07$, whereas Q-KTD shows $558.57 \pm 1012.3$. This observation reveals the positive effect that robustness of correntropy as a cost function brings to learning under changing policies. We further observe the average step number at each trial over 50 Monte Carlo runs (Figure 2). Q-CKTD takes a larger number of steps at the beginning, but as learning progresses (trial number increases), it requires a significantly fewer steps per trial. We can also see that the system adapts to the environment and is able to find a better policy. Note that until the 200th step, the policy is completely random, and thus, both algorithms show an average number of steps larger than 200. The trials that reach the goal even under a random policy are able to do so because their initial positions are close enough to the goal.

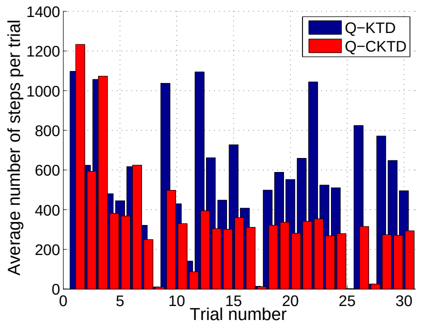


Fig. 2. Average number of steps per trial. Comparison of Q-KTD and Q-CKTD. Average number of steps per trial. Note that the same 30 initial states are applied for 50 Monte Carlo runs.

### B. Open Loop RLBMI Experiment

We apply Q-CKTD to neural decoding of a monkey's neural states on a center-out reaching task aiming at 4 targets (right, up, left, and down). The data employed in this experiment is provided by JoeFrancisLab at SUNY Downstate Medical Center. A female bonnet macaque is trained for a center-out reaching task allowing 8 action directions. The monkey only observes how the position of a computer cursor on a screen changes over time, and its neural states from the motor cortex (M1) are recorded while the monkey is watching the screen change through the duration of the experiment. Every trial starts at the center point, and the distance from the center to each target is $4cm$; anything within a radius of $1cm$ from the target point is considered as a valid reach.

Here, open loop experiments are conducted, so only 144 successful trials are used for neural decoding. Spike times from 49 units are converted to firing rates using a $100ms$ window, and a 9th order tap delay line is applied; hence, 490 dimensions are used to represent the neural states. Each trial allows 2 steps to approach the task. The distance between the center and the target is covered in 1 step. 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 $\epsilon$-greedy method [13]. Based on the selected direction, the computer cursor position is updated. A positive reward value $+1.5$ is assigned to the system when the cursor reaches the reward zone (anywhere within a 0.5 radius from an assigned target). Otherwise, the system earns negative reward $-0.6$. A trial is terminated once it passes

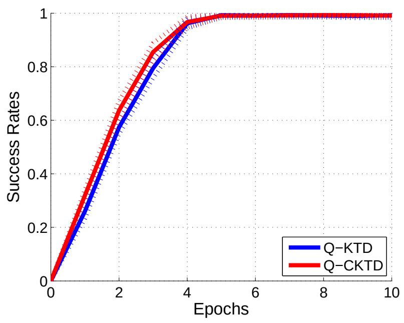


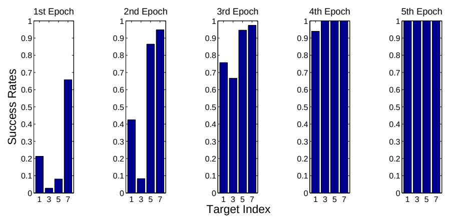
Fig. 3. Comparison of the average success rates of Q-KTD and Q-CKTD. The solid line shows the mean success rates, and the dashed line shows the standard deviation over 50 Monte Carlo runs.

2 steps or receives the positive reward. We apply $\gamma = 0.9$, $\epsilon = 0.01$, and $\eta = 0.5$. The filter kernel size $h$ for Q-CKTD is chosen based on the average of pair wise distances of the input states. With $h = 1$, Q-CKTD shows improvement over KTD when the correntropy kernel size $h_c = 1$ is applied to the first two epochs and changed to $h_c = 0.8$ at the third epoch. The success rates are obtained at each epoch as the average number of successful trials over the 4 targets. These success rates are further averaged over 50 Monte Carlo runs (Figure 3).

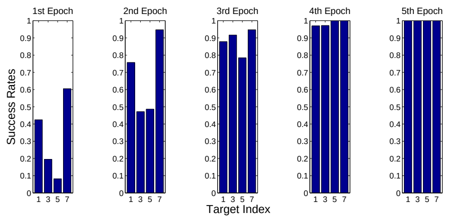
As with KTD, CKTD exploits the advantages of both TD learning and kernel methods. Since the same data is repeated in each epoch, it is expected that the two algorithms converge and find a proper policy. Additionally, because in the open loop experiment, neural states are not changing in response to task execution, the improvement of Q-CKTD is not dominant here. However, individual success rates of Q-CKTD for each target shows improvements versus Q-KTD (Figure 4). The figure employs the same parameter set as in Figure 3, but the results are obtained from a single run. In the beginning of learning, Q-KTD focus only on certain directions, so it does not explore the full repertoire of options (Figure 4(a)). However, the learning variation over each direction in Q-CKTD is smaller in comparison with Q-KTD; in Q-CKTD the system attempts to explore more directions during learning (Figure 4 (b)). When the task requires multiple steps and the search space is high dimensional, this property will be very useful because it should improve convergence time.

## IV. CONCLUSIONS

We introduced a novel policy estimation algorithm: Q-learning via correntropy KTD (Q-CKTD). The main idea of CKTD is to use correntropy instead of mean squared error as an objective function which guides the learning process in KTD. We compared the performance between Q-CKTD and Q-KTD first on a simulated control problem. The results showed the robustness of Q-CKTD under changing policy. In addition, we applied the proposed algorithm to open loop RLBMI experiments. The experimental results show similar convergence to Q-KTD in this simple task, but we can see that the Q-CKTD explores the action space better.



(a) Q-KTD.



(b) Q-CKTD.

Fig. 4. The success rates of each target over 1 through 5 epochs. Target indices 1, 3, 5, and 7 represent right, up, left, and down respectively.

We expect that the true advantage of Q-CKTD will occur in closed-loop experiments of more realistic tasks, which require multisteps to reach the target. The results imply the developed methodology can be useful in relevant practical scenarios.

Further work on fine tuning the correntropy kernel size is still necessary; a principled method to select the correntropy kernel size is under development. Furthermore, further theoretical analysis of cost functions in RL is planned.

## REFERENCES

[1] 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. L. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Bio.*, vol. 1, no. 2, November 2003.

[2] V. Gilja, P. Nuyujukian, C. A. Chestek, J. P. Cunningham, B. M. Yu, J. M. Fan, S. I. Ryu, and K. V. Shenoy, "A brain machine interface control algorithm designed from a feedback control perspective," in *IEEE EMBS*, 2012, pp. 1318–1322.

[3] 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.

[4] J. Bae, P. Chhatbar, J. T. Francis, J. C. Sanchez, and J. C. Principe, "Reinforcement learning via kernel temporal difference," in *IEEE EMBS*, 2011, pp. 5662–5665.

[5] 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 *IEEE EMBS*, 2012, pp. 4108–4111.

[6] C. Szepesvari, *Algorithms for Reinforcement Learning*, R. J. Branchman and T. Dietterich, Eds. Morgan & Slaypool, 2010.

[7] I. Santamaria, P. P. Pokharel, and J. C. Principe, "Generalized correlation function: Definition, properties, and application to blind equalization," *IEEE Trans. on Signal Processing*, vol. 54, no. 6, 2006.

[8] A. Singh and J. C. Principe, "Using correntropy as a cost function in linear adaptive filters," in *IJCNN*, 2009, pp. 2950–2955.

[9] S. Zhao, B. Chen, and J. C. Principe, "Kernel adaptive filtering with maximum correntropy criterion," in *IJCNN*, 2011, pp. 2012–2017.

[10] W. Liu, puskal P. Pokharel, and J. C. Principe, "The kernel least mean square algorithm," *IEEE Trans. on Signal Processing*, vol. 56, no. 2, 2008.

[11] R. S. Sutton, "Learning to predict by the methods of temporal differences," in *Machine Learning*, 1988, pp. 9–44.

[12] S. P. Singh and R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, 1996.

[13] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.