Proceedings of the 5th International
IEEE EMBS Conference on Neural Engineering
Cancun, Mexico, April 27 - May 1, 2011

Adaptive Kalman Filtering for Closed-Loop Brain-Machine Interface Systems

Siddharth Dangi, Suraj Gowda, Rodolphe Héliot, and Jose M. Carmena, *Senior Member, IEEE*

***Abstract*— Brain-Machine Interface (BMI) decoding algorithms are often trained offline, but this paradigm ignores both the non-stationarity of neural signals and the feedback that exists in online, closed-loop control. To address these problems, we have developed an Adaptive Kalman Filter (AKF), a Kalman filter variant that adaptively updates its model parameters during training. For a Kalman filter decoder, batch retraining methods require completely re-estimating the parameter matrices from sufficient data to perform regression accurately, even if only small changes are necessary. Conversely, the AKF is designed to update the decoder parameters continuously and more intelligently. We simulated a population of 41 neurons learning to control a 2D computer cursor. The AKF yielded significantly faster skill acquisition and better robustness to perturbation and neuron loss than a standard Kalman filter with periodic batch retraining.**

I. INTRODUCTION

Brain-Machine Interfaces (BMIs) aim to help severely disabled patients suffering from neurological injuries and diseases by decoding neural activity into control signals for assistive devices. Several groups have exhibited compelling proof-of-concept demonstrations of BMIs in monkeys and humans [1]–[5]. Significant challenges remain before we can achieve clinical viability for humans, but the potential benefits are enormous. For instance, amputees could be outfitted with prosthetic arms – directly controlled by neural signals – enabling them to effortlessly perform everyday reaching and grasping movements that would otherwise be impossible.

BMI decoding algorithms are often initially trained offline by recording neural measurements without the user in the loop, and fitting them against natural or imagined movements. However, this open-loop perspective ignores the visual biofeedback that is present in closed-loop control. Thus, decoders that make good offline predictions often do not perform well online [6].

In this paper, we introduce the Adaptive Kalman Filter (AKF), an innovative online training variant for a Kalman filter (KF) decoder that is designed from a closed-loop perspective and addresses some key outstanding problems in online BMI operation. For instance, KF decoders are often trained by collecting data offline and then performing a batch estimation of the KF parameters. Rather than wait for a batch of data to be collected, the AKF utilizes a stochastic gradient descent approach to update the filter parameters at each iteration. As a result, the AKF has the potential to dramatically decrease the training time required for a KF BMI decoder to achieve high performance online.

Furthermore, the AKF can adapt continuously and intelligently to changing neural statistics. For instance, implanted electrodes in proposed systems are expected to shift over both short and long time scales [7], and often cause glial scaring [8]. These phenomena and others make neural signals non-stationary, necessitating periodic decoder retraining. With a batch approach, retraining a KF requires completely re-estimating the filter matrices, even if the change in neural statistics is small. However, the AKF solves this problem by providing a method to adjust the filter matrices without performing complete re-estimation.

Like any other adaptive filtering method, the AKF requires some measurement of "error" in order to adjust the filter accordingly. In our center-out experiment, the error is the discrepancy between the actual cursor kinematics and the intended cursor kinematics. To generate the latter, we used "cursorGoal" [9], [10], which estimates intended kinematics at each time step by assuming that the user's intended velocity is towards the current target.

This paper is organized as follows. In section II, we review the Kalman filter model and the standard batch training paradigm. In sections III and IV, we derive the AKF update equations and introduce some enhancements and implementation details. In section V, we compare the AKF's performance to that of a standard Kalman filter with batch retraining in a center-out task, using a BMI simulator that utilizes an empirically verified model of the user's learning process during operation of a closed-loop BMI [11].

II. STANDARD KALMAN FILTER

The Kalman filter is a common decoding algorithm choice for BMI applications [12]. Let x_t be a vector representing the kinematic state of a computer cursor or prosthetic limb, and let y_t be a vector of neuron firing rates. Our inference problem is to estimate the state x_t from the observations $\{y_t, y_{t-1}, y_{t-2}, \dots\}$. If we assume the following state evolution and state observation models

$$x_t = Ax_{t-1} + w_t$$
$$y_t = Cx_t + q_t$$

This work was supported in part by the Multiscale Systems Center, and the National Science Foundation CAREER Award to JMC.

Siddharth Dangi and Suraj Gowda are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA. {sdangi,sgowda}@eecs.berkeley.edu

R. Héliot was with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA, and is now with CEA-LETI, Minatec. rodolphe.heliot@cea.fr

J. M. Carmena is with the Department of Electrical Engineering and Computer Sciences, Helen Wills Neuroscience Institute and Program in Cognitive Science, University of California, Berkeley, CA 94720 USA. carmena@eecs.berkeley.edu

where $w_t \sim \mathcal{N}(0, W)$ and $q_t \sim \mathcal{N}(0, Q)$ are noise terms, then a Kalman filter is not only optimal for minimizing the expected mean squared state prediction error, but is also computationally efficient because it is a linear, recursive estimator.

The Kalman filter uses its previous state estimate $\hat{x}_{t-1}$ (and estimate covariance P_{t-1}) to generate the *prior* estimate $\hat{x}_{t|t-1}$ (and estimate covariance $P_{t|t-1}$) of x_t:

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1}$$
$$P_{t|t-1} = AP_{t-1}A^T + W$$

It then calculates the "Kalman gain" K_t, and uses the observation y_t to generate the *posterior* estimate $\hat{x}_t$ (and estimate covariance P_t):

$$K_t = P_{t|t-1}C^T(CP_{t|t-1}C^T + Q)^{-1}$$
$$\hat{x}_t = \hat{x}_{t|t-1} + K_t(y_t - C\hat{x}_{t|t-1})$$
$$P_t = (I - K_tC)P_{t|t-1}$$

To train a Kalman filter, the parameters of the model – the matrices A, C, W, and Q – need to be estimated. As a specific example, consider the state observation matrix, C. In the absence of an explicit observation model of the system, and provided that both state and observation data are available, a standard batch method for estimating C is using simple linear regression:

$$C = YX^T(XX^T)^{-1} \tag{1}$$

where the Y and X matrices are formed by tiling recorded neural activity and measured kinematics, respectively. The other matrices can similarly be estimated from training data:

$$A = X_2X_1^T(X_1X_1^T)^{-1} \tag{2}$$
$$W = \frac{1}{N-1}(X_2 - AX_1)(X_2 - AX_1)^T \tag{3}$$
$$Q = \frac{1}{N}(Y - CX)(Y - CX)^T \tag{4}$$

where $X_1 \triangleq X_{1:\text{end-1}}$, $X_2 \triangleq X_{2:\text{end}}$, and N is the number of data points x_t and y_t [13].

III. ADAPTIVE KALMAN FILTER (AKF)

The standard Kalman filter equations (1)-(4) suggest no obvious way to adjust the filter parameters with additional training observations. Given the non-stationarity of neural recordings, one might propose to periodically re-estimate these matrices, but this leaves periods of time between re-estimations during which the matrices may become suboptimal. Thus, it is necessary to develop a method to allow the Kalman filter matrices to continuously adapt to changing neural firing statistics during online BMI control. In the following sections, we derive an Adaptive Kalman Filter (AKF) with update equations that accomplish exactly that. It is important to note that the x_t terms that appear in the parameter update equations are not recorded kinematics, but rather the cursorGoal estimate of intended kinematics [9], [10].

A. State Evolution (A) and Observation (C) matrices

Starting from first principles, we can write C as the solution to an optimization problem:

$$C = \arg\min_C \underbrace{\mathbb{E}\left[\|y_t - Cx_t\|^2\right]}_{g(C)}$$

where we have defined $g(C) \triangleq \mathbb{E}\left[\|y_t - Cx_t\|^2\right]$. We expect the statistics of the recorded neural signals – and thus, the solution to the optimization problem above – to change over time.

Based on this observation, we propose to update C using gradient descent. We can calculate the true gradient of $g(C)$, and then obtain a stochastic approximation at each time step by removing the expectation operators:

$$\nabla g(C) = 2\left(C\mathbb{E}\left[x_tx_t^T\right] - \mathbb{E}\left[y_tx_t^T\right]\right)$$
$$\approx 2\left(Cx_tx_t^T - y_tx_t^T\right)$$

Standard gradient descent would involve updates to C of the form

$$C^{(i+1)} = C^{(i)} - \mu_C\nabla g(C^{(i)})$$

but using our stochastic gradient method, we update C as

$$C^{(i+1)} = C^{(i)} - \mu_C\left(C^{(i)}x_t - y_t\right)x_t^T \tag{5}$$

where μ_C is the step-size. This update equation is simple to implement, computationally inexpensive, and allows C to adapt over time to the changing value of the expectation $g(C)$.

As we did with C, we can also write the state evolution matrix A as the solution to an optimization problem:

$$A = \arg\min_A \mathbb{E}\left[\|x_t - Ax_{t-1}\|^2\right]$$

A similar derivation yields an update equation for A:

$$A^{(i+1)} = A^{(i)} - \mu_A\left(A^{(i)}x_{t-1} - x_t\right)x_{t-1}^T \tag{6}$$

where μ_A is the step-size.

B. Noise Covariance matrices (W and Q)

We heuristically update W and Q in the following way:

$$W^{(i+1)} = \alpha_W W^{(i)} + (1 - \alpha_W)w_tw_t^T \tag{7}$$
$$Q^{(i+1)} = \alpha_Q Q^{(i)} + (1 - \alpha_Q)q_tq_t^T \tag{8}$$

Here, α_W and α_Q are $\in [0, 1]$, and are typically chosen to be closer to 1. While these updates are suboptimal because they do not represent gradient ascent on the log-likelihood, they are simple to implement and still have a theoretical grounding. Indeed, these equations update W and Q using an exponentially-weighted moving average of the empirical noise covariance matrices. An implementation advantage of this method is that, for example, it is not necessary to store $\{w_t, w_{t-1}, \ldots, w_{t-N+1}\}$, as it would be for a standard N-pt. moving average.

Since we do not measure w_t or q_t, but rather only x_t and y_t, the actual update equations can be written by replacing w_t with $x_t - A^{(i+1)}x_{t-1}$ in (7) and replacing q_t with $y_t - C^{(i+1)}x_t$ in (8).

IV. AKF ENHANCEMENTS AND IMPLEMENTATION DETAILS

A. Normalized step-sizes for A and C

The only free parameters in the update equations (5) and (6) are the step-sizes μ_C and μ_A. Choosing appropriate step-sizes is important to allow these matrices not only to adapt properly to changing signal statistics, but also to converge to the optimal values in the stationary case if signal statistics remain relatively constant. However, choosing fixed step-sizes is difficult because these updates are sensitive to the scaling of the terms x_{t-1}, x_t, and y_t.

One may recall that a similar issue arises with a LMS filter. To motivate an analogous solution for our problem, let us revisit the LMS problem.

The standard LMS filter assumes a linear relationship between the input x_n and the output y_n:

$$y_n = \theta^T x_n + \epsilon_n$$

where ϵ_n is the filter error [13]. The basic LMS update equation for θ is given by

$$\theta^{(t+1)} = \theta^{(t)} + \mu(y_n - \theta^{(t)T} x_n)x_n$$

In a specific implementation known as "Normalized LMS" (NLMS), the step-size is chosen so as to normalize by the power of the input:

$$\mu = \frac{1}{\|x_n\|^2}$$

One can verify that, in addition to possessing some favorable convergence properties, this choice of the step-size updates θ such that if on the next iteration there is a "repeated presentation" $(x_{n+1}, y_{n+1}) = (x_n, y_n)$, then no update will occur because the filter model will be exact.

Motivated by the NLMS's choice of step-size, we find analogous step-sizes for the update equations of the Kalman filter matrices A and C:

$$\mu_A = \frac{1}{\|x_{t-1}\|^2} \quad \text{and} \quad \mu_C = \frac{1}{\|x_t\|^2}$$

Indeed, one can verify that with these choices, if (x_{t+1}, x_t, y_{t+1}) is a repeated presentation of (x_t, x_{t-1}, y_t), then A and C will not be updated because those parts of the Kalman filter model will be exact.

For our application, it is not necessary to update A and C to become exact for the data points at every iteration. Rather, we simply want to take a small step towards "exactness", and thus we instead choose step-sizes

$$\mu_A = \frac{\rho_A}{\|x_{t-1}\|^2 + \epsilon_A} \quad \text{and} \quad \mu_C = \frac{\rho_C}{\|x_t\|^2 + \epsilon_C}$$

where ρ_A and ρ_C are adjustable parameters $\in [0, 1]$ that we typically choose to be closer to 0 (and small epsilon terms are added to avoid divide-by-zero errors in implementation).

B. Restricted updates for A and W

For the example of a BMI controlling a computer cursor, the kinematic state x_t might be

$$x_t = [p_{t,x}, p_{t,y}, v_{t,x}, v_{t,y}, 1]^T$$

where $p_t = (p_{t,x}, p_{t,y})$ and $v_t = (v_{t,x}, v_{t,y})$ represent cursor position and velocity. (The constant "1" is simply added to account for the fact that the observations $\{y_t\}$ may not be zero-mean.) Since we know the relationship between position and velocity, it makes sense to constrain A a priori as

$$A = \begin{bmatrix} 1 & 0 & \text{dt} & 0 & 0 \\ 0 & 1 & 0 & \text{dt} & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

and to constrain W to have zeros in its last row and column.

Updating all elements of A could introduce an unwanted discrepancy into the relationship between position and velocity. Thus, we only update the elements of these matrices that aren't fixed to a certain value. Defining $\hat{A} \triangleq A_{3:4,3:4}$, one can show that the update equation for A becomes:

$$\hat{A}^{(i+1)} = \hat{A}^{(i)} - \frac{\rho_A}{\|v_{t-1}\|^2 + \epsilon_A} \left(\hat{A}^{(i)} v_{t-1} - v_t \right) v_{t-1}^T$$

Likewise for the W matrix, only the top-left 4x4 block is updated using (7).

V. RESULTS

Using our BMI simulator [11], we compared the performance of 1) the AKF with 2) a standard KF with periodic batch retraining and 3) a standard KF with no retraining. The simulator model consists of simulated cortical neurons, a decoder that transforms neural activity into motor output, a feedback controller that reduces the error using an error-descent algorithm, and an open-loop controller whose parameters are updated based on the feedback controller's corrections. Past work has demonstrated high accordance of the model's predictions with actual experimental data from nonhuman primates operating a BMI.

For all simulations, A and C were initialized randomly (A was constrained as in (9)), and W and Q were initialized to scalar multiples of the identity matrix. For the standard KF with periodic batch retraining, KF parameters were re-estimated every 200 trials using (1)-(4). For the AKF, we used (5) and (6) with $\rho_C = \rho_A = 0.05$ and $\epsilon_C = \epsilon_A = 0.001$, and (7) and (8) with $\alpha_W = \alpha_Q = 0.999$. Intended kinematics were generated using "cursorGoal" [9], [10], for use as the x_t terms in our update equations.

We then simulated a BMI center-out task with 8 radial targets over 1400 trials, where we defined performance to be the fraction of successful trials. As shown in Fig. 1, with a standard KF with no retraining, the simulated brain is able to build an inverse model of the decoder and gradually improve its performance. As expected, periodic batch retraining slightly sped up the learning process. However, with the AKF, the learning process occurred much faster.

In this case, not only is the simulated brain adapting to the random decoder, but the decoder is also constantly adapting its parameters based on the brain's firing rates, thus allowing the brain and the decoder to "meet in the middle".

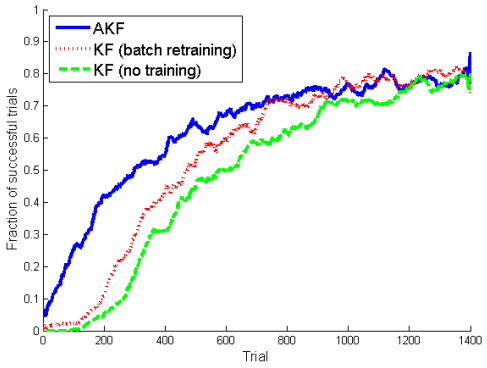

Fig. 1. Simulated brain's performance while learning random KF decoders.

We then tested the AKF's adaptive abilities even further by 1) introducing a sudden change in the base firing rates to simulate an electrode shift, and 2) suddenly dropping 5 neurons (out of 41). As shown in Figs. 2 and 3, using a standard Kalman filter – with or without batch retraining – performance suffers initially, but the brain is gradually able to adapt to the change. The AKF, however, recovers more quickly to these sudden changes.

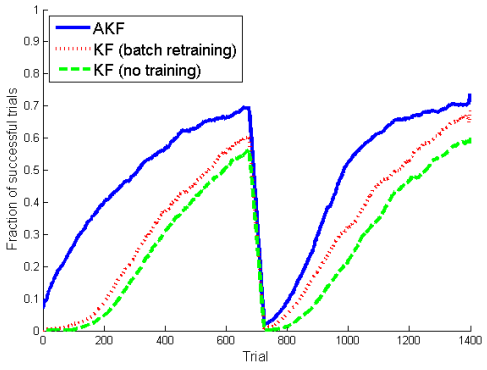

Fig. 2. Simulated brain's performance while learning random KF decoders, with a sudden electrode shift perturbation on trial 700.

VI. Conclusion and Future Work

In this paper, we developed an Adaptive Kalman Filter, a KF decoder training variant designed to significantly reduce the time required for the brain to learn an arbitrary decoder, compared to the standard KF batch retraining paradigm. We simulated the AKF's performance in a 2D center-out experiment with problems that plague real BMI systems, such as electrode array shifts and neuron loss. While it has been shown that the brain can adapt to an arbitrary BMI decoder [5], the skill acquisition time is decreased

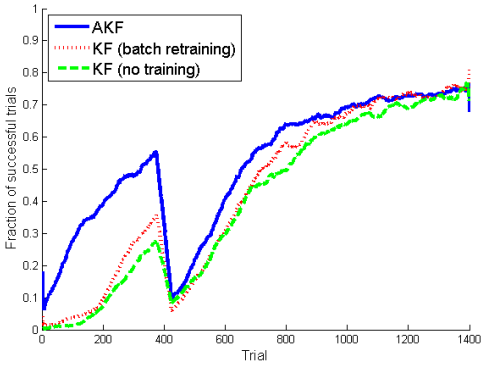

Fig. 3. Simulated brain's performance while learning random KF decoders, with sudden loss of 5 neurons (out of 41) on trial 400.

significantly by adapting the decoder as well, thus placing part of the burden of the learning process on the decoder.

While the AKF's simulated performance is promising, in future work we will further validate our results with experiments involving real animals. In addition, we will derive optimal updates for the matrices W and Q in place of the suboptimal, heuristic updates in (7) and (8).

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 Biol*, vol. 1, no. 2, p. e42, 10 2003.

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

[3] 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, 2006.

[4] 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, May 2008.

[5] K. Ganguly and J. M. Carmena, "Emergence of a stable cortical map for neuroprosthetic control," *PLoS Biol*, vol. 7, p. e1000153, 07 2009.

[6] S. Koyama, S. Chase, A. Whitford, M. Velliste, A. Schwartz, and R. Kass, "Comparison of braincomputer interface decoding algorithms in open-loop and closed-loop control," *Journal of Computational Neuroscience*, vol. 29, pp. 73–87, 2010, 10.1007/s10827-009-0196-9.

[7] M. Linderman *et al.*, "Signal processing challenges for neural prostheses," *Signal Processing Magazine, IEEE*, vol. 25, no. 1, pp. 18 –28, 2008.

[8] V. S. Polikov, P. A. Tresco, and W. M. Reichert, "Response of brain tissue to chronically implanted neural electrodes," *Journal of Neuroscience Methods*, vol. 148, no. 1, pp. 1–18, 2005.

[9] V. Gilja *et al.*, "A high-performance continuous cortically-controlled prosthesis enabled by feedback control design," *2010 Neuroscience Meeting Planner. San Diego, CA*, 2010.

[10] V. Gilja, P. Nuyujukian, C. Chestek, J. Cunningham, B. Yu, S. Ryu, and K. Shenoy, "High-performance continuous neural cursor control enabled by feedback control perspective." Computational and Systems Neuroscience (COSYNE 2010).

[11] R. Héliot, K. Ganguly, J. Jimenez, and J. M. Carmena, "Learning in closed-loop brain-machine interfaces: modeling and experimental validation," *IEEE Trans. Sys. Man Cyber. Part B*, vol. 40, pp. 1387–1397, October 2010.

[12] W. Wu *et al.*, "Bayesian population decoding of motor cortical activity using a Kalman filter." *Neural Computation*, vol. 18, no. 1, pp. 80–118, January 2006.

[13] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, March 1996.