Decoding High Level Signals for Asynchronous Brain Machine Interfaces

Byron Olson, Jennie Si, and Jason Silver

***Abstract*— While many brain machine interface (BMI) systems have been presented in the literature, most of these systems present the user with an 'always on' interface with no way to shut the interface down when not needed. This paper proposes two extensions of previous BMI work to create an asynchronous BMI in which the system only produces outputs when needed. The first classifies incoming signals into not only task related states, but also an idle state. A refinement of this system utilizes a Markov Model (MM) of the task to impose order on the sequence of states produced by the system. This MM filter improves the accuracy of the system an average of 16%.**

I. INTRODUCTION

The past decade has seen an explosion in the number and complexity of brain machine interfaces (BMIs) [1]–[4]. These systems, which utilize signals from the brain to directly control some external device have matured to the point where their clinical application seems far more likely. Most of these systems, however, are limited to a highly confined workspace and further have no off state. Realistic systems will not only need to produce task related signals, but also decide to turn those signals off and on. Asynchronous systems only provide task related signals when needed and providing no output otherwise.

A. Previous Work

In previous work [5], our group, motivated by creating devices that could provide patients with tangible increases in quality of life, has focused on the possibility of controlling a vehicle with asynchronous, as-needed, high level commands as opposed to second-to-second updating of a three dimensional trajectory. Using signals from motor cortical areas to decode these high level commands, five rats were able to successfully complete a paddle pressing task using only a small number of neurons.

Specifically, rats used a BMI (Fig. 1) to indicate a desired direction to a task. The BMI looked for the start of a task and then, at a specific time in the future, collected data to create a neural activity vector (NAV). This single NAV per trial was fed into a support vector machine (SVM) and a decision about direction was made. This interface worked well. The problem, however, was that the system needed to know when a trial had begun. The methods outlined in this paper eliminates the need for any non-neural signals, and yet performs as well or better than the previous system.

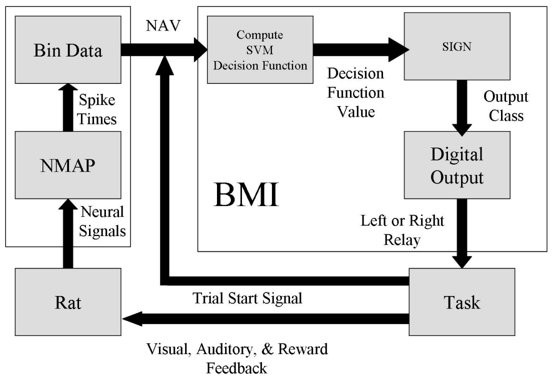

Fig. 1. Original System Diagram

II. METHODS

A. Experimental Setup

The data used in paper was recorded during experiment discussed in [5]. For completeness, the methods used will be summarized here.

Animals (male Sprague-Dawley rats) were implanted with a single 2×4 array of 50μm microwire electrodes spaced 500μm apart in their right hemisphere motor cortical area [1]. Signals from the arrays were amplified, discriminated, and recorded using an NMAP system (Plexon, Inc, TX).

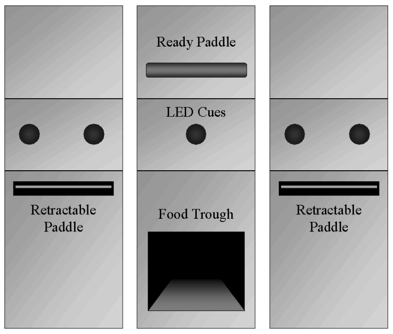

Fig. 2. A Rat's View of the Conditioning Chamber

Data used in this work was obtained via support from DARPA Bio-Info-Micro Grant # MDA972-00-1-0027 and NSF ECS-0233529. The first author completed this work during post-doc at the Center for Computational Intelligence, Learning, and Discovery under a grant from the Iowa State Foundation.

B. Olson (Post-Doc), J. Si (Professor), and J. Silver (Graduate Student) are all with the Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287, USA {Byron.Olson, si, Jason.Silver}@asu.edu

[1]The Institutional Animal Care and Use Committee at Arizona State University approved the experimental protocol for this investigation.

The rats performed a five-light task. For this a rat was placed in an operant conditioning chamber facing a panel (Fig. 2) with a retractable paddle to his left and right and a non-retractable ready paddle in the center. The retractable paddles controlled a series of red LEDs placed horizontally across the panel, two on the left, two on the right, and one in the center. The task would proceed as follows:

- Rat pressed the ready paddle to start a trial.
- A non-center LED would illuminate.
- After two seconds, the retractable paddles were extended.
- Presses on the left/right paddle would 'move' the light to the right/left.
- If the center light was illuminated for more than 1s, a food reward was dispensed.

In the original study, only the 'inside left' and 'inside right' LEDs were used as initial cues. Thus, correct responses would move the light to the center and incorrect responses would move the light away from the center to an 'outside' LED. Each day rats would participate in around 100 trials of such manual interaction while neural signals were recorded. From this data, a classifier was constructed. Trials remaining in the 45 minute experimental session were controlled by the classifier instead of the paddles.

This paper utilizes the same neural and behavioral data as the original system in a simulated on-line fashion. Please note, while some significant adaptation was detected during the original experiments, we will not take that into account in our reexamination. This paper examines open-loop systems created from data derived in a closed-loop experiment, so no adaptation is possible to the new methods. We would expect some increase in performance if such systems were implemented in a closed-loop fashion. Nevertheless, data examination here will take place as a simulation of the original conditions. Data from the calibration phase (first 99-100 trials) will be used to train all methods (see Fig. 3) and data from the brain controlled phase will be input into the systems to test these methods (see Fig. 4) These are natural testing and training datasets with differences in distributional parameters likely to resemble what will be encountered in real closed-loop experiments.

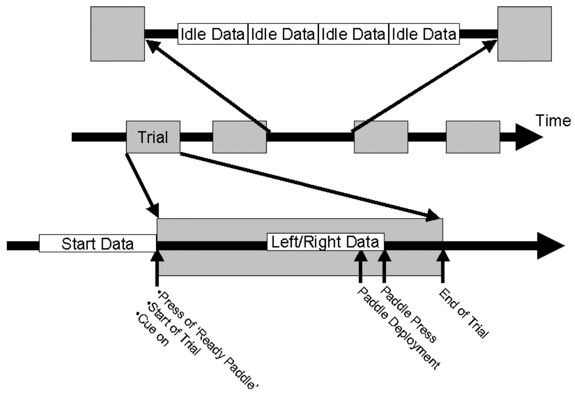

Fig. 3. Graphical Representation of Data used to calibrate BMI systems

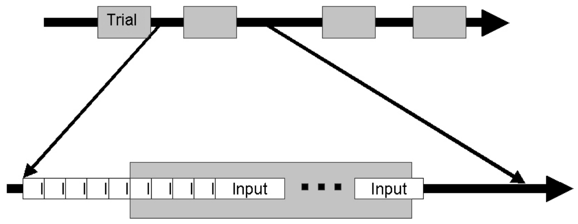

Fig. 4. Graphical Representation of Data used to test BMI systems

B. Neural Activity Vectors

Neural Activity Vectors (NAVs) encode the information observed in the neurons under consideration and serve as inputs to machine learning techniques. Typically NAVs consist of firing rates (frequencies) estimated by counting spikes (action potentials) observed in a window of time. Multiple time windows from multiple neurons are then concatenated into a single vector. The original study attempted to incorporate temporal information by using 10 bins per neuron, with each bin representing 100ms. Since temporal information in this study is incorporated in other ways, only a single one second bin (updated every 250ms) per neuron is considered. This allows the classification methods to simply use the most current version (or estimate) of the firing rates.

C. Support Vector Machines

Support Vector Machines (SVMs) are a class of tool utilized in machine learning problems [6]. Traditionally, support vector machines are used to learn a mapping from some input space to a set of classes. This mapping is computed in some high dimensional space by constructing a hyperplane to separate the classes. Since many such separating hyperplanes may exist, the hyperplane with the maximal margin is selected.

SVMs are formulated as binary classification problems where the training data consists of N pairs $\{(X_t, y_t)\}_{t=1}^N$ with input $X_t \in \Re^p$ and binary class labels $y_t \in \{-1, 1\}$. The classifier takes the form

$$y(X) = sign(W^T \phi(X) + b),$$

where ϕ maps the input space $\Re^p$ to a high-dimensional feature space $\Re^m$ (possibly infinite), and W is a weight vector for these high-dimensional vectors. The optimal (maximum margin) separating hyperplane is equivalent to the minimum $\|W\|$ and thus leads to the following optimization problem:

$$\min_{W,\xi_1,\ldots,\xi_N} \left\{ \frac{1}{2} W^T W + C \sum_{t=1}^{N} \xi_t \right\},$$

subject to

$$y_t(W^T \phi(X_t) + b) \geq 1 - \xi_t$$
$$\xi_t \geq 0, t = 1, \ldots, N.$$

Since the data, even in the higher dimensional space, may not be separable we introduce the slack variables, ξ_t's which reflect the level of misclassifications for these non-separable samples. C is used as a tuning parameter that expresses the relative cost of these misclassifications compared to maximizing the margin.

By constructing the Lagrangian,the dual optimization problem emerges as:

$$\max_{\alpha_1,\ldots,\alpha_N} \left\{ \sum_{t=1}^{N} \alpha_t - \frac{1}{2} \sum_{t=1}^{N} \sum_{s=1}^{N} \alpha_t \alpha_s y_t y_s K(X_t, X_s) \right\},$$

subject to

$$0 \le \alpha_t \le C$$
$$\sum_{t=1}^{N} \alpha_t y_t = 0,$$

Where $K(X_a, X_b) = \phi(X_a)^T \phi(X_b)$ is a kernel function. Various types of kernel functions have been used, but to guarantee a solution they must satisfy the Mercer condition [6]. This study uses Gaussian kernels of the form:

$$K(X_a, X_b) = e^{-\gamma \|X_a - X_b\|^2}$$

Expressed using kernels the decision function takes the form:

$$f(X) = \sum_{t=1}^{N} \alpha_t y_t K(X, X_t) + b.$$

And thus the classifier becomes:

$$y(X) = sign(f(X)),$$

The observations which appear in the decision function, i.e., the data points with non-zero coefficients α_t, are called *support vectors*. Therefore, the complexity of the constructed learning machine depends on the number of support vectors rather than the dimension of the feature space.

All support vector machines in this study were written in MATLAB and based on [7]. The C and γ parameters were optimized using 10-fold cross validation over the training data.

D. Probability Models for SVM

Support Vector Machines are discriminative classifiers that do not attempt to generate a model of an underlying probability distribution, but rather simply attempt to discern examples of the classes from one another. The problem, however, is that a probability is needed to make comparisons and integrate prior knowledge from a variety of sources expressed as probabilities. For example, in a realistic driving scenario, vehicle sensors could express the probability of changes in direction.

Two methods of developing a probability model from SVM decision function values $f(X)$ were considered, with essentially identical results. The first method (similar to [8]) fits a sigmoidal function with parameters a and b such that:

$$P_{platt}(class = 1 | f(X) = D) = \frac{1}{1 + e^{aD+b}}$$

The second method empirically determines this probability by simply resampling the training vectors to create N examples from the positive class under the assumption that the dimensions of the input are independent. The decision values $f(\hat{X}_k)$ with $k = 1 \ldots N$ are then computed. The probability of a decision value given the positive class $P_{emp}(class = 1 | f(X) = D)$, is simply the fraction of $\hat{X}_k$ with decision values less than or equal to D.

E. Markov Chain Model

The important states identified for the system $Idle, Start, Left, Right$, do not occur in a random order. In fact, we expect a $Start$ state to follow an $Idle$. Likewise, we expect $Left$ or $Right$ to follow a $Start$. Assuming the state sequence is from a Markov system implies:

$$P(S_{t+1} | S_t, S_{t-1}, \ldots, S_0) = P(S_{t+1} | S_t)$$

Thus, the next state in a sequence is only dependant on the previous state. We can thus create a state transition matrix A where $A_{ij} = P(S(t) = j | S(t-1) = i)$, which will fully describe the state dynamics of the system. The state transition matrix for the task under consideration is shown graphically in Fig 5. Here arrows pointing from one state to another represent a valid state transition.

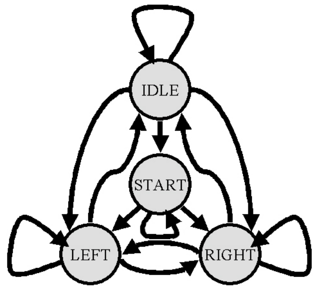

Fig. 5. Graphical representation of Markov model of the BMI system. Nodes Idle, Start, Left, & Right show the states of the system. Edges between nodes indicate valid state transitions.

III. Asynchronous Implementations

Two principled approaches to creating an asynchronous BMI were developed. Both of these approaches take neural signals continuously and provide a paddle pressing signal to the task. Likewise, both approaches consider the task as having four distinct states, $Idle$ indicating the time between trials, $Start$ indicating the start of trial, and $Left$ and $Right$ indicating pressing of the left and right paddles.

The first approach simply attempts to classify every NAV as one of the four classes using an SVM. To do this a $1 - vs - all$ approach is used. This approach creates a single classifier per class in which examples of a given class are pitted against all other examples. The results from the classifiers are compared and the classifier with the highest probability is assigned as the output.

The second approach takes advantage of the orderly succession of states observed in a real task to “filter” the outputs from the support vector machines. By assuming an initial state of the system S_0, the subsequent states can be determined using the both the output of the SVMs and and state transition matrix. Explicitly we attempt to maximum $P(S(t) = k|f(X_t), S(t-1))$ at time t, over all possible $S(t)$. Under a minimum set of assumptions this is equivalent to maximizing

$$P(S(t) = k|f(X_t)) \times P(S(t) = k|S(t-1) = i)$$

where

$$P(S(t) = k|S(t-1) = i) = A_{ik}$$

is from the state transition matrix. While sophisticated methods exist to optimize the state transition matrix. Here, a simple grid search was performed over valid state transition matrices, utilizing the training data.

IV. Results

A. Accuracy

TABLE I
ACCURACY

Rat	Day	4-class SVM	4-class SVM with MM
rat$A$	1	75%	90%
rat$A$	2	86%	92%
rat$A$	3	73%	85%
rat$B$	1	35%	73%
rat$C$	1	71%	81%

Table I presents accuracy computed as the percentage of trials completed correctly using the the 4-class SVM alone versus the 4-class SVM with MAP, Markov Model filter. Trials are considered correct if predicted paddle press times lead to a correct sequence of paddle presses. For this, paddle press predictions outside of the trial period and need for start presses are ignored.

B. Effect of Probability Models

The two methods used to convert SVM decision function values into probabilities, do indeed create slightly different probability models. However, except for a difference in the optimal state transition matrices (shown below), there was no difference in the task completion accuracy.

$$A_{emp} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0.1 & 0.45 & 0.45 \\ 0.8 & 0 & 0.1 & 0.1 \\ 0.8 & 0 & 0.1 & 0.1 \end{bmatrix}$$

$$A_{platt} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0.2 & 0.4 & 0.4 \\ 0.5 & 0 & 0.4 & 0.1 \\ 0.5 & 0 & 0.1 & 0.4 \end{bmatrix}$$

Here rows/columns are assigned as $Idle, Start, Left, Right$.

V. Discussion

The EEG based brain-computer interface (BCI) world has considered asynchronous formulations [9], [10]. Further, finite state machine based systems have also been tried. In [11] for example, a mobile robot was controlled using a Gaussian classifier coupled to a state model. Users would self-select three EEG signals that would indicate a driving direction to the robot, the state model would then interpret the EEG control as well as sensor indications to determine the direction of the robot.

In the BMI world, similar ideas are implemented in Kalman filters. Where observations are combined with state models to provide improved performance. These systems, however, are only concerned with active movement and not with transitioning to and from an idle state.

VI. Conclusions

Here two asynchronous BMI systems are described. The first classifies every NAV as one of four possible states. This allows the system to utilize non-task outputs to put the system into an idle state. The second system refines the approach using a Markov Model (MM) based filter to impose order on the sequence of states. This filter improves the accuracy of the system an average of 16% (min 6% max 38%). This result hold for two separate methods of converting SVM decision function values into probabilities. Such a formulation provides an interesting framework in which to consider the next generation of BMIs.

References

[1] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," *Nature Neuroscience*, vol. 2, pp. 664–670, 1999.

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

[3] D. M. Taylor, S. I. Tillery, and A. B. Schwartz, "Direct cortical control of 3d neuroprosthetic devices," *Science*, vol. 296, pp. 1829–1832, 2002.

[4] 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, 2004.

[5] B. P. Olson, J. Si, J. Hu, , and J. He, "Closed-loop cortical control of direction using support vector machines," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 13, no. 1, pp. 11–18, 2005.

[6] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[7] S. Canu, Y. Grandvalet, and A. Rakotomamonjy, "Svm and kernel methods matlab toolbox," Perception Systmes et Information, INSA de Rouen, Rouen, France, 2003.

[8] J. C. Platt, *Probabilities for SV Machines*. Cambridge, MA: MIT Press, 2000, pp. 61–73.

[9] G. Townsend, B. Graimann, and G. Pfurtscheller, "Continuous eeg classification during motor imagery–simulation of an asynchronous bci." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, no. 2, pp. 258–265, 2004.

[10] J. Millan and J. Mourino, "Asynchronous bci and local neural classifiers: an overview of the adaptive brain interface project." *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 159–161, 2003.

[11] J. Millan, F. Renkens, J. Mourino, and W. Gerstner, "Noninvasive brain-actuated control of a mobile robot by human eeg," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1026 – 1033, 2004.