Research Article

Spatial-Temporal Clustering of Neural Data Using Linked-Mixtures of Hidden Markov Models

Shalom Darmanjian and Jose Principe

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6200, USA

Correspondence should be addressed to Shalom Darmanjian, oxygen@cnel.ufl.edu and Jose Principe, principe@cnel.ufl.edu

Received 1 February 2009; Revised 1 September 2009; Accepted 19 November 2009

Recommended by Don Johnson

This paper builds upon the previous Brain Machine Interface (BMI) signal processing models that require apriori knowledge about the patient's arm kinematics. Specifically, we propose an unsupervised hierarchical clustering model that attempts to discover both the interdependencies between neural channels and the self-organized clusters represented in the spatial-temporal neural data. Results from both synthetic data generated with a realistic neural model and real BMI data are used to quantify the performance of the proposed methodology. Since BMIs must work with disabled patients who lack arm kinematic information, the clustering work described within this paper is relevant for future BMIs.

Copyright © 2009 S. Darmanjian and J. Principe. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Introduction

Unfortunately, thousands of people have suffered tragic accidents or debilitating diseases that have either partially or fully removed their ability to effectively interact in the external world. Some devices exist to aid these types of patients, but often lack the requirements to live a normal life. Essentially the idea behind motor Brain Machine Interfaces (BMIs) is to bridge the gap between the brain and the external world so that these patients can achieve effective interaction.

Generally, a BMI is a system that directly retrieves neuronal firing patterns from dozens to hundreds of neurons in the brain and then translates this information into desired actions in the external world. In particular, neural data is recorded from one or more cortices using one or multiple electrode grid arrays [1]. The amplified analog voltages recorded from one or more neurons are then digitally converted and passed to a spike sorting algorithm. Finally, the discrete binned spike counts are fed into a signal processing algorithm [1] and subsequent trajectory/lever predictions are sent to robot arm or display device. All of the processing occurs as an animal engages in a behavioral experiment (lever press, food grasping, finger tracing, or joystick control).

Prior work showed that partitioning the neural input with a switching classifier improves trajectory reconstruction over conventional BMI feed-forward algorithms. Indirectly, the improvement provided evidence that the neural activity in the motor cortex transitions through multiple specific states during movement [2, 3] and that this switching behavior is exploitable for BMIs.

Unfortunately, to partition the neural input space, apriori class labels are needed for separation. However, under real conditions with paraplegics, there are no kinematic clues to separate the neural input into class labels (or clusters). Currently, most of the behaving animals engaged in BMI experiments are not paralyzed, allowing the kinematic information to be used in the training on the models. This Achilles heel plagues most BMI algorithms since they require kinematic training data to find a mapping to the neural data.

Since kinematic clues are not available from paraplegics, neural data must be exclusively used to find a separation. Finding neural structures (i.e., neural assemblies) may offer a solution. The hypothesis argued in this paper is that there are multiple neural structures corresponding to motion primitives. Initial supervised results support this hypothesis [4]. Therefore the goal is to find a model that can learn these temporal-spatial structures or clusters and segment the

neural data without kinematic clues (i.e., unsupervised). In this paper, the Linked-Mixtures of Hidden Markov Models (LMs-HMMs) are combined with a clustering methodology in order to cluster neural data. The LM-HMM model is chosen since it operates solely in the input space and can characterize the temporal spatial space with a reduced computational cost. The methodology described in the next section will explain how the model learns the parameters and structure of the neural data in order to provide a final set of class or cluster labels for segmentation.

2. Generative Model and Clustering Framework

2.1. Related Work. The modeling approaches to BMIs are divided into three categories, supervised, coadaptive, and unsupervised, with the majority of BMI modeling algorithms being supervised. Additionally, most of the supervised algorithms further split into linear modeling, non-linear modeling, and state-space (or generative) modeling, with the majority of BMI algorithms falling under supervised linear modeling.

Supervised linear modeling is traceable to the 1980's and 1990's when neural action potential recordings were taking place with multi-electrode arrays [1, 5, 6]. Essentially, the action potentials (called spikes for short), collected with microelectrode arrays, are sorted by neuron and counted in time windows (called bins) and fed into a linear model (Wiener filter) with a predefined tap delay line depending on the experiment and experimenter [7, 8]. During training the linear model has access to the desired kinematic data, as a desired response, along with the neural data recording. Once a functional mapping is learned between the neural and kinematic training set, a test set containing only neural data is used on a linear model for reconstructing the kinematic trajectory [7].

The Wiener filter is exploited in two ways. First, it serves as a baseline linear classifier to compare results with the models discussed. Second, Wiener filters are used to reconstruct the trajectories from neural data switched by the generative models (see Figure 1). By comparing reconstructions from Wiener filters using our methodology, a reasonable metric can be established for unsupervised clustering in the absence of ground truths. Since the Wiener filter is critical to the paper and serves at the core of many BMI systems today, the details of the Wiener filter are presented in the appendix.

Along with the supervised linear modeling, researchers have also engaged in non-linear supervised learning algorithms for BMIs [1, 8]. Very similar to the paradigm of the linear modeling, the neural data and kinematic data are fed to a non-linear model that finds the relationship between the desired kinematic data and the neural data. Then during testing only neural data is provided to predict kinematic reconstruction.

With respect to state-space models, Kalman filters have been used to reconstruct the trajectory of a behaving monkey's hand [9, 10]. Specifically, they use a generative model for encoding the kinematic state of the hand. For

decoding, the algorithm predicts the state estimates of the hand and then updates this estimate with new neural data to produce a posteriori state estimate. Our group at UF and elsewhere found that the reconstruction was slightly smoother than what input-output models were able to produce [9, 10].

Unfortunately, these supervised BMI models pose many problems. As mentioned, training with desired data is problematic since paralyzed patients cannot provide kinematic data. Second, these models will not capture inhibited neurons, which are known to exist in the brain, since a neuron that fires very little will receive less weighting. Third, there are millions of other neurons not being recorded or accounted for in these models. Including that the missing information into the model would be beneficial. Lastly, all of the BMI models must generalize over a wide range of movements. Normally, generalization is good for a model of the same task, but generalization across tasks produces poor results.

The lack of a desired signal therefore necessitates the need for an unsupervised or coadaptive solution. Recent coadaptive solutions have relied on the test subject to train its brain for goal oriented tasks. Other researchers have used technicians to supply an artificial desired signal as the input-output models learn a functional mapping [11]. Sanchez et al. who used reinforcement learning coadapt their model and the rat's behavior for goal-oriented tasks [12].

Other generative models (even graphical models like Hidden Markov Models—HMMs) [13–15] work on BMIs exploit the hidden state variables to decode possible states taken or transitioned by the behaving animal. Specifically, Kemere et al. found that HMMs can provide representations of movement preparation and execution in the hidden state sequences [15]. Unfortunately, this work also requires the use of supervision or a training data that must first be divided by human intervention (i.e., a user).

Although there is little BMI research into clustering neural data with graphical models, there have been efforts to use HMMs for clustering. The use of hidden Markov models for clustering appears to have first been mentioned in Juang and Rabiner [16] and subsequently used in the context of discovering subfamilies of protein sequences in Cadez and Smyth [17]. Other work use single HMM chains for understanding the transition matrices [17]. The work described in this paper moves significantly beyond prior work since the clustering model finds unsupervised hierarchical dependencies between HMM chains (per neuron) while also clustering the neural data. Essentially the algorithm jointly refines the model parameters and structures as the clustering iterations occur. The hope is that the clustering methodology will serve as a front end for a goal-oriented BMI (with the clusters representing a specific goal, like “forward”) or for a coadaptive algorithm that needs reliable clustering of the neural input (see Figure 1).

2.2. Generative Model. With the absence of information, a probabilistic approach is the best option to model what can be observed from the brain. Modeling the observed and

FIGURE 1: BMI system overview.

hidden neural information is accomplished with observable and hidden random processes that are interacting with each other in some unknown way. To achieve this, we make the assumption that each neuron's output is an observable random process that is affected by hidden information. Since the experiment does not provide detailed biological information about the interactions between the sampled neurons, we use hidden variables to model these hidden interactions [6, 18]. We further assume that the compositional representation of the interacting processes occurs through space and time (i.e., between neurons at different times). Graphical models are the best way to model and observe this interaction between variables in space and time [19]. Another benefit of a state-space generative model over traditional filters is that neurons that fire less during certain movements can be modeled simply as another state rather than a low filter weight value.

Given the need to incorporate spatial dependencies between neural channels (which are known to be important) [20], Linked-Mixture of Hidden Markov Models (LM-HMMs) provides a way to help cluster the class labels while also clustering spatial dependencies between channels.

Let Z represent the set of variables (both hidden and observed) included in the probabilistic model. A graphical model (or Bayesian network) representation provides insight into the probability distributions over Z encoded in a graph structure [19]. With this type of representation, edges of the graph represent direct dependencies between variables. Conversely, and more importantly, the absence of an edge allows for the assumption of conditional independence

FIGURE 2: LM-HMM graphical model.

between variables. Ultimately, these conditional independencies allow a more complicated multivariate distribution to be decomposed (or factorized) into simple and tractable distributions [19].

Since there are a variety of graphical model representations that decompose the joint probability of the hidden and observed variables in Z , choosing the best approximation is overwhelming. For modeling the dependencies between multiple HMMs, the LM-HMM establishes another layer of hidden or latent variables to link and express the spatial dependencies between the lower level HMM structures (Figure 2), thus creating a clique tree structure T (since there are cycles), where hierarchic links exist between neural channels.

Given the model parameters for a given HMM chain $\Theta = \{A, B, \pi\}$, the log probability for this structure (Figure 2) is

$$\begin{aligned} \log P(O, Q, M | \Theta) \\ = \log P(M^1) + \sum_{i=2}^N \log P(M^i | M^{i-1}, \Theta^i) \\ + \sum_{i=1}^N \left(\sum_{t=1}^T \log P(O_t^i | Q_t^i, \Theta^i) \right. \\ \left. + \sum_{t=1}^T \log P(Q_t^i | Q_{t-1}^i, M^i, \Theta^i) \right), \end{aligned} \quad (1)$$

where the dependency between the tree cliques are represented by a hidden variable M^i (corresponding to the i th neuron) in the second layer

$$P(M^i | M^{i-1}, \Theta^i), \quad (2)$$

and the hidden state Q also has a dependency on the hidden variable M in the second layer

$$P(Q_t^i | Q_{t-1}^i, M^i, \Theta^i). \quad (3)$$

The lower observable variables O^i (shown in Figure 2) are conditionally independent from the second hidden layer variable M^i , as well as the sub-graphs of the other neural channels T^j (where $i \neq j$). The variables O^i are the actual bin counts observed from the i th neuron through time. The hidden variable M in the second layer of (2) can be interpreted as a mixture variable (when excluding the hierarchic links).

The LM-HMM is a compromise between making an independence assumption and a full dependence assumption. For further understanding and EM implementations please see the appendix and [20]. Although this hierarchical model can find dependencies between channels in an unsupervised way, we still need to address how to obtain the class (or clustering) labels without user intervention.

2.3. Clustering Framework. This section establishes a model-based method for clustering the spatial-temporal neural signals using the LM-HMM. In effect, the clustering method tries to discover a natural grouping of the exemplar S (i.e., window of multidimensional data or sequences of the neurons) into K clusters. A discriminate (distance) metric similar to K-means is used except that the vector centroids are now probabilistic models (LM-HMMs) representing dynamic temporal data [21].

The bipartite graph view (Figure 3) assumes a set of N data objects D (e.g., exemplars, represented by $S_1, S_2, \dots, S_N$), and K probabilistic generative models (e.g., HMMs), $\lambda_1, \lambda_2, \dots, \lambda_K$, each corresponding to a cluster of exemplars (i.e., windows of data) [22]. The bipartite graph is formed by connections between the data and model spaces. The model space usually contains members of a

FIGURE 3: Bipartite graph of exemplars (x) and models.

specific family of probabilistic models. A model λ_y can be viewed as the generalized “centroid” of cluster y , though it typically provides a much richer description of the cluster than a centroid in the data space. A connection between an object S and a model λ_y indicates that the object S is being associated with cluster y , with the connection weight (closeness) between them given by the log-likelihood $\log p(S | \lambda_y)$.

A straightforward design of a model-based clustering algorithm is to iteratively retrain models and repartition data objects. This can be achieved by applying the EM algorithm to iteratively compute the (hidden) cluster identities of data exemplars in the E-step and estimate the model parameters in the M-step. Although the model parameters start out as poor estimates, eventually the parameters hover around their true values as the iterations progress. The log-likelihoods are a natural way to provide distances between models as opposed to clustering in the parameter space (which is unknown). Basically, during each round, each training exemplar is re-labeled by the winning model with the final outcome retaining a set of labels that relate to a particular cluster or neural state structure (i.e., neural assembly) for which spatial dependencies have also been learned.

Setting the parameters can be daunting since the experimenter must choose the number of states, the length of the exemplar (window size) and the distance metric. To alleviate some of these model initialization problems, previous parameter settings found during early work are used for these experiments [4]. Specifically, an a-priori assumption is made that the neural channels are of the same window size and same number of hidden states. The clustering framework is outlined below.

Let data set D consist of N sequences for J neural channels, $D = S_1^1, \dots, S_N^J$, where $S_n^j = (O_{n_1^j}, \dots, O_{n_T^j})$ is a sequences of observables length T and $\Lambda = (\lambda_1, \dots, \lambda_K)$ a set of Models. We will refer to the multiple sequences in a window of time (size T) as an exemplar. The goal is to locally maximize the log-likelihood function:

$$\log P(D | \Lambda) = \sum_{S_n^j \in D} \log P(S_n^j | \lambda_{y(S_n^j)}). \quad (4)$$

- (1) Randomly assign K labels (with $K < N$), one for each windowed exemplar S_n , $1 \leq n \leq N$. The LM-HMM parameters are initialized randomly.
- (2) Train each assigned model with the respective exemplars using the LM-HMM procedure discussed in the appendix a and [20]. During this step the model learns the dependency structure for the current cluster of exemplars.
- (3) For each model, evaluate the log-likelihood of each of the N exemplars given model λ_i , that is, calculate $L_{in} = \log L(S_n | \lambda_i)$, $1 \leq n \leq N$ and $1 \leq i \leq K$. $y(S_n) = \operatorname{argmax}_y \log L(S_n | \lambda_i)$ is the cluster identity of the exemplar. Then re-label all the exemplars based on cluster identity to maximize (4).
- (4) Repeat steps (2) and (3) until convergence occurs or until a percentage of labeled exemplars does not change (i.e., set a threshold for changing exemplars). More advanced metrics for deciding when to stop cluster could be used (like KL divergence etc.).

3. Simulations

3.1. Simulated Data Generation. Since there are no ground truths to label real BMI neural data, simulations on plausible artificial data will help support the results found by the clustering framework on real data. Neurophysiologic knowledge provides an avenue to create a realistic BMI simulation since there is some evidence to suggest that neurons encode the direction of hand movements with cosine shaped tuning curves [5, 23]. These tuning curves provide a reference of activity for different neurons. In turn, this neural activity potentially relates to a kinematic vector, such as hand position, hand velocity, or hand acceleration, often using a direction or angle between 0 and 360 degrees. A discrete number of bins are chosen to coarsely classify all the movement directions. For the polar plots in this paper we chose 20 bins that each account for 18 degrees of the 360 degree space. For each direction, the average neural firing rate is obtained by using a non-overlapping window of 100 ms. Subsequently this average firing rate on the polar plot indicates the direction of velocities, and the magnitude of the vector is the average firing rate, marked as a blue circle, for each direction. The preferred direction is computed using circular statistics as

$$\text{circular mean} = \arg \left(\sum_N r_N e^{i\Theta_N} \right), \quad (5)$$

where r_N is the neuron's average firing rate for angle Θ_N , and N covers all the angle range. Figure 4 shows the polar plot of four simulated neurons and the average tuning information with standard deviation across 100 Monte Carlo trials evaluated for 16 minutes duration. The computed circular mean, estimated as the firing rate weighted direction, is shown as a solid red line on the polar plot. The figure clearly indicates that the different neurons fired more frequently toward the preferred direction. Additionally, in order to get the statistical evaluation between Monte Carlo runs, the

traditional tuning depth were not normalized to (0, 1) for each realization as normally done in real data. To calculate the tuning depth:

$$\text{Tuning Depth} = \frac{\max(r_N) - \min(r_N)}{\text{std}(r_N)}. \quad (6)$$

A neural model must be selected in order to generate realistic simulated neurons. Although multiple models have been proposed [24, 25], we select the Linear-Nonlinear-Poisson (LNP) model since we can change different tuning properties to generate more realistic neural data. The LNP model consists of three stages. The first is a linear transformation that is then fed into a static non-linearity to provide the conditional firing rate for a Poisson spike generating model at the third stage [24, 25].

Two simulated neural data sets are generated in the following experiments. One data set contains four neurons tuned to two classes (Figure 4) and a second data set contains eight neurons tuned to four classes. We first generate a velocity time series with 100 Hz sampling frequency and 16 minutes duration (1000000 samples totally). Specifically, a simple 2.5 kHz cosine and sine function are used to emulate the kinematics (X-Y Velocities) for the simulation experiments. For both data sets the entire velocity time series is passed through a (LNP) model with the assumed nonlinear tuning function in (7):

$$\lambda_t = \exp(\mu + \beta \vec{v}_t \vec{D}_{\text{prefer}}), \quad (7)$$

where λ_t is the instantaneous firing probability, μ is the background firing rate (set to .00001). The variable β represents the modulation factor for a preferred direction which we set monotonically from 1 to 4 for the four neurons in the two class simulation and a value of 3 for the eight neurons in the four class simulation. The unit vector $\vec{D}_{\text{prefer}}$ is the preferred angular direction of the kinematics which we set to $\pi/4$ and $5\pi/4$ for the two class simulation and $\pi/4$, $3\pi/4$, $5\pi/4$, $7\pi/4$ for the four class simulation (each direction is assigned two neurons for both simulations). The spike train is generated by an inhomogeneous Poisson spike generator using a Bernoulli random variable with probability $\lambda(t)\Delta_t$ within each 1 ms time window. Once the spike trains are generated, we bin them into 100 ms bins and down sample the velocity $\vec{v}_t$ data accordingly.

For each data set, an additional 100 randomly distributed spike trains (also 16 minutes each) are combined with the two data sets to create an artificial neural data set with a total of 104 and 108 neurons. Essentially, this allows for less than an 8% chance for the desirable channels to be randomly selected by the LM-HMMs.

3.2. Simulation Results. Figure 5 demonstrates the clustering results using the LM-HMM on the two-class simulated data set. For this particular experiment, the model parameters are set (during training) for two classes ($k = 2$) which is equal to the true number of classes in the simulation data. Additionally, the class labels alternate since they represent the alternating kinematics (shown at the bottom of figure).

FIGURE 4: Neural Tuning depth of four simulated neurons.

As seen from the figure, the model is able to correctly cluster the data in a relatively small number of iterations (three to four). For the first iteration, each exemplar in the full data set is randomly assigned to one of the clusters (indicated by green and blue colors). For the remaining iterations, a pattern starts to emerge that looks similar to the alternating kinematics. Although the kinematics (cosine and sine wave) are shown below the class labels, the clustering results were acquired solely from the input space.

Figure 6 shows the class tuning preference when the model is initialized with random data. The term class tuning refers to the angular preference of the particular class (or cluster) label. It is calculated the same way as in neural tuning, except that the data is collected from the samples

that have been labeled by a particular class (i.e., circular statistics are calculated on the kinematics from class 3 rather than a neuron). Figure 7 shows the angular preference of the classes after clustering. Overlaid in blue is the original angular tunings of some of the neurons. We see that the model is able to successfully find the separation in neural firings.

For the simulation in Figure 8, the LM-HMM is used to cluster a four-class simulated data set ($k = 4$). Again the correct number of clusters $k = 4$ is set during training to match the true number of classes in the input data (another oscillating pattern). There are a few issues with shrinkage and expansion with respect to the class labels but is due to the temporal-spatial data (not static classification). Overall the

FIGURE 5: LM-HMM cluster iterations (Two classes, $k = 2$).

final result demonstrates that the clustering model with the LM-HMM is able to discover the underlying clusters present in the simulated neural data. Figure 9 shows the clustering on the initial random labels while Figure 10 shows the tuned preference of the four classes after clustering. Remarkably it is able to determine the separation from the four classes using the neural input only. Next, the clustering model is tested for robustness when the number of clusters is unknown or increased noise is added to the neural data.

As with all clustering algorithms, choosing the correct number of underlying clusters is difficult. Choosing the number of clusters for BMI data is even more difficult since there are no known or established ground truths (with respect to motion primitives). To see the effect of defining too many clusters for the model, Figure 11 illustrates when the clustering model is initialized with four classes ($k = 4$) despite the simulation only containing two underlying classes (or clusters) for the input space. Again, the results are generated within a relatively small number of iterations. Notice from the figure that the extra two class labels are absorbed into the two classes shown in the previous Figure 5 (also shown below the four class labels). Interestingly, a repeated pattern of consistent switching occurs with the class labels (as indicated by the pattern of color blocks). Specifically, Figure 11 shows that class 1 precedes class 3 and class 2, when combined, they correspond to class 1 in Figure 5, while class 4 in Figure 11 corresponds to class 2 in Figure 5. This overlapping of clusters is common in clustering methods when the labels outnumber the true underlying number of classes [26]. Even the neural data from such a simple simulation is complicated yet remarkably the clustering method finds the consistent pattern of switching (perhaps indicating that the simple classes for further divisibility).

For the final simulation, random spikes are added to the unbinned spoked trains of the earlier tuned neurons (Figure 4). Specifically, uniformly random spikes are generated with a probability of spiking every 1 ms. Figure 12 shows the classification performance as the probability of

firing is increased from a 1% chance of spiking to 16% chance of spiking in 1 ms. Interestingly, the classification decreases but not significantly. The robustness is due to the tuned neurons still maintaining their underlying temporal structure. Figure 13 shows the tuning polar plots of the four original neurons with the added random spikes. Although this figure shows that tuning seems to broaden across many angular bins, the random spikes do not have a temporal structure. Therefore they do not displace the temporal structure of the tuned neurons significantly (as indicated by only a small change in performance). Please note that increasing the probability of random spikes to 16% every 1 ms puts the spiking beyond the realistic firing rate of real neurons.

As a final note, the different artificial neurons are modulated so that their tuning depth monotonically increased (i.e., β set from 1 to 4). The LM-HMM clustering successfully selects the neurons in the correct order (relative to tuning depth) from the 100 random neural channels. The result is the same when the tuned neurons are corrupted with random spike noise.

4. Experimental Animal Data

4.1. Animal Data Collection. The animal data for these experiments were collected in the primate laboratory at Duke University. Using micro wire electrode arrays chronically implanted in the dorsal premotor cortex (PMd), supplementary motor area (SMA), primary motor cortex (M1, both hemispheres) and primary somatosensory cortex (S1), the firing times of up to 185 cells were simultaneously collected in an adult female monkey (*Macaca mulatta*) while performing a manipulandum behavioral tasks (cursor control) [1, 27]. The monkey used a hand-held manipulandum (joystick) to move the cursor (smaller circle) so that it intersects the target. Upon intersecting the target with the cursor, the monkey received a juice reward. While the monkey performed the motor task, the hand position and velocity for each coordinate direction were recorded in real time along with the corresponding neural activity. Figure 14 illustrates that the majority of the monkey's movement occurs diagonally.

In the second experiment, neural data was recorded from an owl monkey's cortex as it performed a food reaching task. Specifically, multiple implanted micro-wire arrays recorded this data from 104 neural cells in the following cortical areas: posterior parietal cortex (PP), left and right primary motor cortex (M1), and dorsal premotor cortex (PMd). Concurrently with the neural data recording, the 3D hand position was recorded as the monkey made three repeated movements: rest to food, food to mouth, and mouth-to-rest [1, 28].

For both data set, each firing count represents the number of neural firings in a 100-millisecond span of time, which is consistent with methods used within the neurological community [2, 6, 23]. The 2D cursor control monkey data set described in this paper contains 185 neural channels recorded for 43.33 minutes. This time recording

FIGURE 6: Tuning Preference for two classes (initialized).

FIGURE 7: Tuned classes after clustering (two classes).

corresponds to a dataset of 26000×185 time bins. The time recording for the 3D food grasping monkey experiment corresponds to a dataset of 23000×104 time bins.

4.2. Experimental Results

4.2.1. 2D Monkey Cursor Control. Two important questions must be answered with the following experiments. First, what type of clustering results are obtained, that is, are there repeating patterns corresponding to the kinematics. Second, how does the trajectory reconstruction from the unsupervised clustering compare against the trajectory

reconstruction from supervised BMI algorithms. Classification performance is not considered in these experiments since there are no known classes by which to test. Although angular bins might serve the purpose for single neurons, the literature in this area is hotly contested. Therefore, correlation coefficient will serve as the metric and allow a consistent comparison between the results in this paper and previous work.

Multiple Monte Carlo simulations were computed to eliminate spurious effects from initial random conditions (class labels, parameters, etc.). The parameters that require initialization include

FIGURE 8: LM-HMM cluster iterations (Four classes, $k = 4$).

- (1) observation exemplar length (i.e., window size),
- (2) number of states,
- (3) number of clustering rounds,
- (4) number of classes.

To determine the parameters, the observation length (window size) was varied from 5 to 15 time bins (corresponding to .5 seconds to 1.5 seconds). The number of hidden states was varied from 3 to 5. While the number of clustering iterations varied from 4 to 10. After exhausting the number of possible combinations, the parameters were set to: an observation length equal to 5 time bins, 3 hidden states and 6 clustering iterations (since less than 5% of the labels changed). These parameters are set the same for each neural channel.

The model was initialized with four classes after an empirical search (using different parameter sets). Since ground truths are unknown, trajectory reconstruction serves as the basis for how many clusters to select. Specifically, if reconstruction improves or diminishes then an adjustment to the number of clusters is made. Qualitatively, Figure 15 shows the labeling results from the LM-HMM clustering. The Y-Axis represents the number of iterations from random labels, to the final clustering iteration. Each color in the clustering results corresponds to a different class (four in all). The kinematics (x and y velocities) are overplayed at the bottom of the image for this cursor control experiment. Figure 15 shows repetitive labeling for similar kinematic profiles. These repetitive class transitions were also observed in the simulated data. Figure 16 shows trajectory reconstruction matches very closely to the original trajectory thereby indirectly validating the segmentation produced by the clustering method.

For a quantitative understanding, the correlation coefficient (CC) is a way to show if the clustering results have merit. Interestingly, the CC results for this unsupervised clustering are slightly better than the supervised echo state network and Wiener filter (Table 1). As expected random

TABLE 1: Correlation coefficient using LM-HMM on 2D monkey data.

Experiment	CC(X)	CC(Y)
LM-HMM (unsupervised)	.80	.69
LM-HMM (supervised)	.82	.83
Echo State (supervised)	.64	.78
NMCLM (FIR)	.67	.50
NMCLM (Gamma)	.67	.47
NLMS	.68	.50
Gamma	.70	.53
Subspace	.70	.58
Weight Decay	.71	.57
Kalman	.71	.58
TDNN	.65	.51
RMLP	.66	.46
Random Labels	.71	.67

labeling of the classes produces poor results compared to actual clustering. Additionally the random labeling results are similar to other supervised BMI models. As discussed earlier the similar results are due to the random clusters providing generalization of the full space for each filter (thereby becoming equivalent to a single Wiener filter). Table 1 shows that the correlation coefficient produced by the unsupervised LM-HMM clustering is slightly less than the correlation coefficient produced with the supervised version of the LM-HMM. This result is understandable since the supervised version of the LM-HMM can consistently isolate the neural data based on kinematic clues therefore improve reconstruction.

4.2.2. 3D Monkey Food Grasping Task. In this section, the LM-HMM is used to cluster the 3D monkey food grasping task. As with the cursor control experiment, Wiener filters are selected based on the class labels in order reconstruct the trajectory. This trajectory reconstruction is then compared to the supervised reconstruction. Qualitatively, Figure 17, shows that there is a corresponding clustering pattern to the kinematics. Although there are areas of error or perceived errors (since there are no ground truths) since the most obvious kinematic feature (movement) is sometime missed in the figure.

Table 2 shows that the correlation coefficient on the unsupervised LM-HMM clustering reconstruction is better than using random labeling or a single Wiener filter. Although the unsupervised results are not as good as the supervised version of the LM-HMM, it is remarkable that the unsupervised clustering can still outperform the majority of supervised BMI algorithms. This serves to validate that the segmentation is successful.

5. Discussion

Brain machine interfaces have the potential to restore movement to patients subjected to paralysis. Although great progress has been made towards BMI's there is still much

FIGURE 9: Tuning Preference for four classes (initialized).

TABLE 2: Correlation coefficient using LM-HMM on 3D monkey data.

Experiment	CC
LM-HMM (unsupervised)	.78
LM-HMM (supervised)	.86
Single Wiener (supervised)	.68
TDNN (supervised)	.65
RMLP (supervised)	.69
Kalman (supervised)	.68
Random Labels	.69

work to be done. This paper addressed one of the main problems associated with the signal processing side of BMIs

by using generative models with hidden variables to help model the multiple interacting processes (both hidden and observable). A probabilistic model was argued as the best approach since there is a lot of information missing in BMIs.

The clustering model discussed in this paper demonstrated the ability to discover useful clusters while operating solely in the neural input space. The results were first justified with realistic neural simulations that also included noisy and fake neurons. Despite the added noise, the clustering method is able to successfully determine the underlying separation. The division of neural input space was based on the conjecture that animals transition between neural state structures during goal seeking analogous to the motion primitives exhibited during the kinematics [3]. Then the clustering method was compared to conventional BMI

FIGURE 10: Tuned classes after clustering (four classes).

signal processing algorithms on real neural data. Although, trajectory reconstruction was used to show the validity of the clusters, the model could be used as front end for a co-adaptive algorithm or goal-oriented tasks (simple classification that paraplegics could select, i.e., move forward).

Despite these encouraging results, improvements in performance are possible for the hierarchical clustering. For example, the LM-HMM in the hierarchical clustering framework may not be taking full advantage of the dynamic spatial relationships. Specifically, spatial relationships may be evolving through time between the neurons. Although the hierarchical training methodology does create dependencies

between the HMM experts, perhaps there are better ways to exploit the dependencies or aggregate the local information. There may be important dependencies since different neural processes are interacting with other neural processes in an asynchronous fashion and that underlying structure could provide insight into the intrinsic communications occurring between neurons.

As a final point, there was an interesting effect from the experiments (simulated and real neural data). Looking closely at some of the results, consistent transitions occur from different classes to other classes. For example there may be a consistent transition from class 1 to class 3 and class 2

FIGURE 11: LM-HMM cluster iterations (Two classes, $k = 4$).

to class 1. It would be interesting to investigate this further and see if perhaps there is a switching behavior between stationary points in the input space. Perhaps we can ascertain when a stationary switching point has occurred and exploit that information for modeling.

Appendix

A. LM-HMM Framework

A.1. LM-HMM Training with EM. We make approximations when finding the expectation of (1). In particular, we will first approximate $P(S_t^i | S_{t-1}^i, M^i, \Theta^i)$ by treating M^i as independent from S making conditional probability equal to the familiar $P(S_t^i | S_{t-1}^i, \Theta^i)$. Two important features can be seen in this type of approximation. First we have decoupled the simple lower-level HMM chains from the higher-level M^i variables. Second, M^i can now be regarded as a linked mixture variable for the HMM chains since $P(M^i | M^{i-1}, \Theta^i)$ which we will address later [29].

Because the lower-level HMMs have been decoupled, we are able to use the Baum-Welch formulation to compute some of the calculations in the E-step, leaving estimation of the variational parameter for later. As a result, we can calculate the forward pass.

E Step:

$$\alpha_j(t) = P(O_1 = o_1, \dots, O_t = o_t, S_t = j | \Theta). \quad (\text{A.1})$$

We can calculate this quantity recursively by setting

$$\begin{aligned} \alpha_j(1) &= \pi_j b_j(o_1), \\ \alpha_k(t+1) &= \left[\sum_{j=1}^N \alpha_j(t) a_{jk} \right] b_k(o_{t+1}). \end{aligned} \quad (\text{A.2})$$

FIGURE 12: Classification degradation with increased random firings.

The well known backward procedure is similar:

$$\beta_j(t) = P(O_{t+1} = o_{t+1}, \dots, O_T = o_T | S_t = j, \Theta). \quad (\text{A.3})$$

This computes the probability of the ending partial sequence $o_{t+1}, \dots, o_T$ given the start at state j at time t . Recursively, we can define $\beta_j(t)$ as

$$\begin{aligned} \beta_j(T) &= 1, \\ \beta_j(t) &= \sum_{k=1}^N a_{jk} b_k(o_{t+1}) \beta_k(t+1). \end{aligned} \quad (\text{A.4})$$

Additionally, the a_{jk} and $b_j(o_t)$ matrices are the transition and emission matrices defined for the model which are updated in the M-step. Continuing in the E-step we will rearrange posteriors in terms of the forward and backward variables. Let

$$\gamma_j(t) = P(S_t = j | O, \Theta), \quad (\text{A.5})$$

which is the posterior distribution. We can rearrange the equations to quantities we have

$$\begin{aligned} P(S_t = j | O, \Theta) &= \frac{P(O, S_t = j | \Theta)}{P(O | S, \Theta)} \\ &= \frac{P(O, S_t = j | \Theta)}{\sum_{k=1}^N P(O, S_t = k | \Theta)}, \end{aligned} \quad (\text{A.6})$$

and now with the conditional independencies we can define the posterior in terms of α 's and β 's:

$$\gamma_j(t) = \frac{\alpha_j(t) \beta_j(t)}{\sum_{k=1}^N \alpha_k(t) \beta_k(t)}. \quad (\text{A.7})$$

FIGURE 13: Neural Tuning depth with high random firing rate.

We also define

$$\xi_{jk}(t) = P(S_t = j, S_{t+1} = k \mid O, \Theta) \quad (\text{A.8})$$

which can be expanded:

$$\begin{aligned} \xi_{jk}(t) &= \frac{P(S_t = j, S_{t+1} = k, O \mid \Theta)}{P(O \mid S, \Theta)} \\ &= \frac{\alpha_j(t) a_{jk} b_k(o_{t+1}) \beta_k(t+1)}{\sum_{j=1}^N \sum_{k=1}^N \alpha_j(t) a_{jk} b_k(o_{t+1}) \beta_k(t+1)}. \end{aligned} \quad (\text{A.9})$$

The M-step departs from the Baum-Welch formulation and introduces the variational parameter [30]. Specifically, the M-step involves the update of the parameters π_j , a_{jk} , b_L (we will save u_i for later)

M Step:

$$\begin{aligned} \hat{\pi}_j^i &= \frac{\sum_{i=1}^I u_i \gamma_1^i(j)}{\sum_{i=1}^I u_i}, \\ \hat{a}_{jk}^i &= \frac{\sum_{i=1}^I u_i \sum_{t=1}^{T-1} \xi_t^i(j, k)}{\sum_{i=1}^I u_i \sum_{t=1}^{T-1} \gamma_t^i(j)}, \\ \hat{b}_j^i(L) &= \frac{\sum_{i=1}^I u_i \sum_{t=1}^T \delta_{o_t, v_L} \gamma_t^i(j)}{\sum_{i=1}^I u_i \sum_{t=1}^{T-1} \gamma_t^i(j)}. \end{aligned} \quad (\text{A.10})$$

There are two issues left to resolve. First, how can the variational parameter be estimated and maximized given the dependencies. Second, if experimentally it is not known which neurons are affecting other neurons (if at all), how can the dependencies between neurons be defined in the model.

FIGURE 14: 2D Angular Velocities.

FIGURE 15: LM-HMM cluster iterations (Ivy 2D dataset, $k = 4$).

FIGURE 16: Reconstruction using unsupervised LM-HMM clusters (blue) versus Real Trajectory (red).

FIGURE 17: LM-HMM Clustering on monkey food grasping task (2 classes).

A.2. Updating Variational Parameter via Importance Sampling. While still working within the EM framework, we treat the variational parameters u_i as mixture variables generated by the i th HMM each having a prior probability of p^i . We want to estimate the set of parameters that maximize the likelihood function [17, 31, 32]:

$$\prod_{z=1}^n \sum_{i=1}^I p^i P(O^{zi} | S^i, \Theta^i). \quad (\text{A.11})$$

Given the set of exemplars and current estimates of the parameters, the E-step consists of computing the conditional expectation of hidden variable M :

$$\begin{aligned} u_{zi} &= E[M^i | M^{i-1}, O^{zi} | \Theta^i] \\ &= \Pr[M^i = 1 | M^{i-1} = 1, O^{zi}, \Theta^i]. \end{aligned} \quad (\text{A.12})$$

The problem with this conditional expectation is the dependency on M^{i-1} . Since M^{i-1} is independent from O^i and Θ^i we can decompose this into

$$u_{zi} = E[M^i | O^{zi}, \Theta^i] E[M^i | M^{i-1}]. \quad (\text{A.13})$$

The first term, a well-known expectation for Mixture of Experts, is calculated by using Bayes rule and the priori probability that $M = 1$:

$$E[M^i | O^i, \Theta^i] = \Pr[M^{i-1} | O^i, \Theta^i] = \frac{p^i P(O^i | S^i, \Theta^i)}{\sum_{i=1}^I p^i P(O^i | S^i, \Theta^i)}, \quad (\text{A.14})$$

Since the integration for the second term is much harder to compute, we look for an integration approximation that will maintain the dependencies. Importance sampling is a well-known method that is capable of approximating

the integration with a lower variance than Monte-Carlo integration [33]. We can approximate the integration with

$$E[M^i | M^{i-1}] = \frac{1}{n} \sum_{z=1}^n \frac{P(O^{zi} | S^i, \Theta^i)}{P(O^{z(i-1)} | S^{i-1}, \Theta^{i-1})}. \quad (\text{A.15})$$

where the n samples have been drawn from the proposal distribution $P(O^{z(i-1)} | S^{i-1}, \Theta)$. For the estimation of u_i we need to combine the two terms:

$$u_{zi} = \frac{p^i P(O^{zi} | S^i, \Theta^i)}{\sum_{i=1}^I p^i P(O^{zi} | S^i, \Theta^i)} \sum_{z=1}^n \frac{P(O^{zi} | S^i, \Theta^i)}{nP(O^{z(i-1)} | S^{i-1}, \Theta^{i-1})}. \quad (\text{A.16})$$

To compute the M-step,

$$\hat{p}^i = \frac{\sum_{z=1}^n u_{zi}}{\sum_{i=1}^I \sum_{z=1}^n u_{zi}} = \frac{\sum_{z=1}^n u_{zi}}{n}. \quad (\text{A.17})$$

Borrowing from the competitive nature of the BM-HMM, we choose winners based on the same criterion of minimizing the Euclidean distance for the classes for the LM-HMM.

Acknowledgments

The authors thank Johan Wessberg and Miguel A. L. Nicolelis for sharing their monkey data and experience. They also thank Yiwen Wang for providing the code for the neural model used in their simulations. This paper was supported by DARPA project no. N66001-02-C-8022 and NSF project no. 0540304.

References

- [1] M. A. L. Nicolelis, D. Dimitrov, J. M. Carmena, et al., "Chronic, multisite, multielectrode recordings in macaque monkeys," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 19, pp. 11041–11046, 2003.
- [2] E. Todorov, "On the role of primary motor cortex in arm movement control," in *Progress in Motor Control III*, M. L. Latash and M. F. Levin, Eds., chapter 6, pp. 125–166, Human Kinetics, 2001.
- [3] L. Goncalves, E. D. Bernardo, and P. Perona, "Movemes for modeling biological motion perception," in *Seeing, Thinking and Knowing*, vol. 38 of *Theory and Decision Library A*, Springer, Dordrecht, The Netherlands, 2004.
- [4] S. Darmanjian, S. P. Kim, M. C. Nechyba, et al., "Bimodal brain-machine interface for motor control of robotic prosthetic," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 4, pp. 3612–3617, 2003.
- [5] A. P. Georgopoulos, J. T. Lurito, M. Petrides, A. B. Schwartz, and J. T. Massey, "Mental rotation of the neuronal population vector," *Science*, vol. 243, no. 4888, pp. 234–236, 1989.
- [6] A. B. Schwartz, D. M. Taylor, and S. I. H. Tillery, "Extraction algorithms for cortical control of arm prosthetics," *Current Opinion in Neurobiology*, vol. 11, no. 6, pp. 701–708, 2001.
- [7] S.-P. Kim, J. C. Sanchez, D. Erdogmus, et al., "Divide-and-conquer approach for brain machine interfaces: nonlinear mixture of competitive linear models," *Neural Networks*, vol. 16, no. 5-6, pp. 865–871, 2003.
- [8] J. C. Sanchez, S.-P. Kim, D. Erdogmus, et al., "Input-output mapping performance of linear and nonlinear models for estimating hand trajectories from cortical neuronal firing patterns," in *Proceedings of the International Workshop for Neural Network Signal Processing*, pp. 139–148, 2002.
- [9] W. Wu, M. J. Black, Y. Gao, et al., "Neural decoding of cursor motion using a Kalman filter," in *Advances in Neural Information Processing Systems*, vol. 15, pp. 133–140, MIT Press, Cambridge, Mass, USA, 2003.
- [10] Z. Li, J. E. O'Doherty, T. L. Hanson, M. A. Lebedev, C. S. Henriquez, and M. A. L. Nicolelis, "Unscented Kalman filter for brain-machine interfaces," *PLoS ONE*, vol. 4, no. 7, article e6243, 2009.
- [11] L. R. Hochberg, M. D. Serruya, G. M. Friehs, et al., "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, no. 7099, pp. 164–171, 2006.
- [12] J. C. Sanchez, B. Mahmoudi, J. DiGiovanna, and J. C. Principe, "Exploiting co-adaptation for the design of symbiotic neuro-prosthetic assistants," *Neural Networks*, vol. 22, no. 3, pp. 305–315, 2009, special issue on Goal-Directed Neural Systems.
- [13] G. Radons, J. D. Becker, B. Dülfer, and J. Krüger, "Analysis, classification, and coding of multielectrode spike trains with hidden Markov models," *Biological Cybernetics*, vol. 71, no. 4, pp. 359–373, 1994.
- [14] I. Gat, *Unsupervised learning of cell activities in the associative cortex of behaving monkeys, using hidden Markov models*, M.S. thesis, Hebrew University, Jerusalem, Israel, 1994.
- [15] C. Kemere, G. Santhanam, B. M. Yu, et al., "Detecting neural-state transitions using hidden Markov models for motor cortical prostheses," *Journal of Neurophysiology*, vol. 100, no. 4, pp. 2441–2452, 2008.
- [16] B. H. Juang and L. R. Rabiner, "Issues in using hidden Markov models for speech recognition," in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, Eds., pp. 509–553, Marcel Dekker, New York, NY, USA, 1992.
- [17] I. Cadez and P. Smyth, "Probabilistic clustering using hierarchical models," Tech. Rep. 99-16, Department of Information and Computer Science University of California, Irvine, Calif, USA, 1999.
- [18] W. J. Freeman, *Mass Action in the Nervous System*, University of California, Berkeley, Calif, USA, 1975.
- [19] M. I. Jordan, Z. Ghahramani, and L. K. Saul, "Hidden Markov decision trees," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., vol. 9, MIT Press, Cambridge, Mass, USA, 1997.
- [20] S. Darmanjian and J. C. Principe, "Boosted and linked mixtures of HMMs for brain-machine interfaces," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, Article ID 216453, 12 pages, 2008.
- [21] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," in *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, pp. 849–856, MIT Press, Cambridge, Mass, USA, 2002.
- [22] S. Zhong and J. Ghosh, "A unified framework for model-based clustering," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1001–1037, 2004.
- [23] A. P. Georgopoulos, J. F. Kalaska, R. Caminiti, and J. T. Massey, "On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex," *Journal of Neuroscience*, vol. 2, no. 11, pp. 1527–1537, 1982.

- [24] E. P. Simoncelli, L. Paninski, J. Pillow, and O. Schwartz, "Characterization of neural responses with stochastic stimuli," in *The New Cognitive Neurosci*, MIT Press, Cambridge, Mass, USA, 3rd edition, 2004.
- [25] Y. Wang, J. C. Sanchez, and J. C. Principe, "Information theoretical estimators of tuning depth and time delay for motor cortex neurons," in *Proceedings of the 3rd International IEEE/EMBS Conference on Neural Engineering*, pp. 502–505, 2007.
- [26] T. Lange, M. Braun, V. Roth, and J. M. Buhmann, "Stability-based model selection," to appear in *Advances in Neural Information Processing Systems*.
- [27] J. Wessberg, C. R. Stambaugh, J. D. Kralik, et al., "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, no. 6810, pp. 361–365, 2000.
- [28] S. Darmanjian, S. P. Kim, M. C. Nechyba, et al., "Bimodal brain-machine interface for motor control of robotic prosthetic," *IEEE Machine Learning for Signal Processing*, vol. 4, pp. 379–384, 2006.
- [29] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.
- [30] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, MIT Press, Cambridge, Mass, USA, 1998.
- [31] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic, "Discovering clusters in motion time-series data computer vision and pattern recognition," in *Proceedings of the IEEE Computer Society Conference*, vol. 1, pp. 375–381, 2003.
- [32] A. Ypma and T. Heskes, "Categorization of web pages and user clustering with mixtures of hidden Markov models," in *Proceedings of the International Workshop on Web Knowledge Discovery and Data Mining*, pp. 31–43, Edmonton, Canada, 2002.
- [33] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer, Dordrecht, The Netherlands, 1995.