Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2008, Article ID 216453, 12 pages
doi:10.1155/2008/216453

Research Article

Boosted and Linked Mixtures of HMMs for Brain-Machine Interfaces

Shalom Darmanjian and Jose C. Principe

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

Correspondence should be addressed to Jose C. Principe, principe@cnel.ufl.edu

Received 9 October 2007; Accepted 26 February 2008

Recommended by Aníbal Figueiras-Vidal

We propose two algorithms that decompose the joint likelihood of observing multidimensional neural input data into marginal likelihoods. The first algorithm, boosted mixtures of hidden Markov chains (BMs-HMM), applies techniques from boosting to create implicit hierarchic dependencies between these marginal subspaces. The second algorithm, linked mixtures of hidden Markov chains (LMs-HMM), uses a graphical modeling framework to explicitly create the hierarchic dependencies between these marginal subspaces. Our results show that these algorithms are very simple to train and computationally efficient, while also reducing the input dimensionality for brain-machine interfaces (BMIs).

1. INTRODUCTION

The field of brain-machine interfaces (BMIs) is devoted to accomplishing the goal of one day restoring paralyzed patients mobility by directly connecting their brain to a machine. A majority of developments in this field have centered around linear and nonlinear models that map neural firing patterns of an animal to a robotic prosthetic [1–3]. Usually, in this type of experiment, a primate or rat engages in a movement task as neural data is recorded from their cortex (as well as the kinematic information). Once a prediction model has been trained with the trajectory/neural data, only neural data is used to control a robotic arm in real-time [2, 3].

As a result of the above-mentioned experiments, our group established that when multiple feed-forward prediction models map discrete portions of the neural data to respective portions of the continuous trajectory, the overall trajectory reconstruction is improved [4, 5]. The classifier that is responsible for switching between these different feed-forward models is the main focus of this paper (see Figure 1 for system overview).

The previous switching classifier was an ensemble method that incorporated multiple independent experts. These experts were single neural-channel HMM chains that formed an independently coupled hidden Markov model (IC-HMM) [5]. Since it is unlikely that the independence assumption applies to all of the neurons, finding dependencies between some of the neurons, could prove beneficial for final kinematic reconstruction or other biologically-inspired modeling. Additionally, one limitation of this algorithm is that as more neurons are sampled from the brain (as is the current trend for BMIs), the number of independent models could grow to an unmanageable level [6]. Although the ICHMM is still computationally more efficient than using a model that has full dependencies, like the coupled hidden Markov model (CHMM), it is still desirable that the input dimensionality be reduced to avoid this pitfall.

In this paper, we first take multidimensional neural input data and decompose the joint likelihood of observing the neural input into marginal likelihoods using boosted mixtures of hidden Markov chains (BMs-HMM). The algorithm applies techniques from boosting to create hierarchical dependencies between these marginal subspaces in an unsupervised manner. Additionally, ideas from mixture of experts (MOEs) are incorporated so that the local information is weighted and integrated into an ensemble decision. Our results show that this algorithm is very simple to train and computationally efficient, while also providing the ability to reduce the input dimensionality for BMIs.

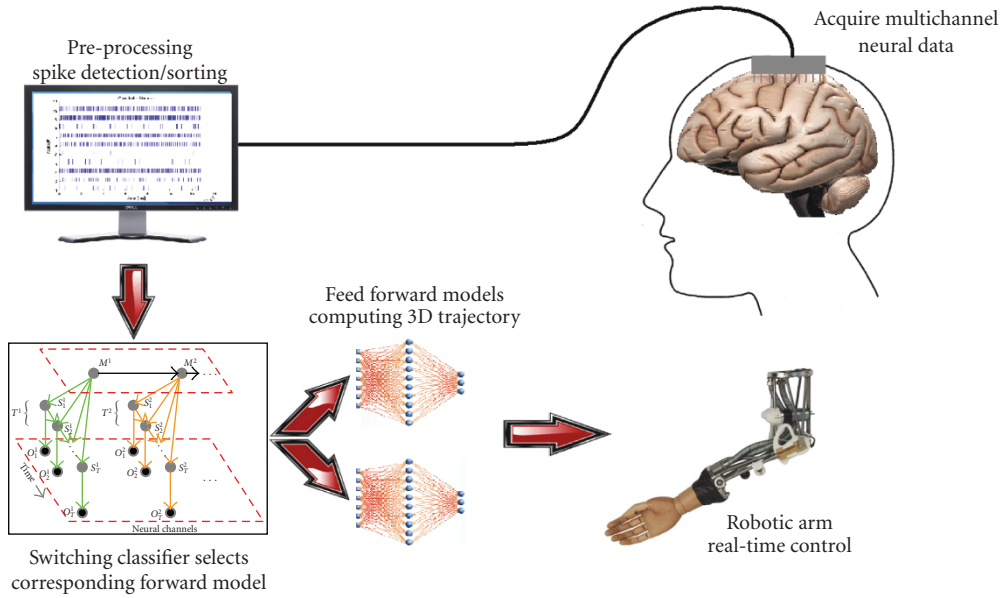

Figure 1: BMI overview.

We then move from this method, which only implicitly exploits the dependencies between the neurons, to a more formal graphical model structure which explicitly models dependencies between neurons. We call this model linked mixture of hidden Markov chains (LM-HMM). Borrowing ideas from the BM-HMM, we show how this semisupervised model explicitly exploits the dependencies between neurons (if they exist).

The development and evaluation of the BM-HMM and LM-HMM serve as the core of this paper and directs the following organization. First, we discuss our motivation and technique for the BM-HMM. Second, we develop the theory for the LM-HMM and explain how it relates to the BM-HMM. Third, we compare results of the new models to our previous models, as well as present some possible interpretation of the results. Finally, we suggest areas of future work.

2. MODELING NEURAL DATA FOR BRAIN-MACHINE INTERFACES

2.1. Experimental animal data

Neural action potentials from two different animal experiments are used to train and test the models in this paper. In the first experiment, neural data was recorded from an owl monkey's cortex as it performed a food reaching task. Specifically, multiple implanted microwire 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, 5].

In the second experiment, a male Sprague-Dawley rat performed a go, no-go lever pressing task as neural data was recorded. This dataset contains 16 neural cells that were collected with microwire arrays implanted in the forelimb region of the left primary cortex (M1) [7]. Subsequently, the data was spike-detected and spike-sorted using thresholds and template matching [7]. The lever presses were recorded simultaneously with the neural activity. The beginning of each trial was signaled to the rat by an LED indicating when to press the lever in order to receive a reward [7].

The data sets resulting from the above experiments were segmented into movement and rest classes for modeling. For the monkey data, all angular velocities greater than 4 mm/s are labeled as part of the movement class. Included in this movement class are the times when the monkey momentarily holds its arm during a reach for food or its mouth [5, 8]. For the rat experiments, since there is no information about the rat moving around the cage (or grooming), only the lever press is included as part of the movement class [7].

For both experiments, the neural data is binned into 100-millisecond counts, which is consistent with the neural science community [8, 9]. Consequently, the movement data is down-sampled to match the 10 Hz neural bin counts. The time recording for the monkey experiment corresponds to a dataset of 23000×104 time bins. For the rat experiment, the dataset consists of 13000×16 time bins [1, 7].

2.2. Motivation

Neuroscientists often treat the multiple channels of neural data acquired from BMI experiments as multivariate observations from a single process [10]. This perspective requires fully coupled statistics across all of the channels at all times irrespective of partial independence among the multiple processes. Additionally, it has been established that

neurons exhibit nonstationary behavior [11]. Our group's own work has shown that modeling this data as a single multivariate process is not the most appropriate [4]. It is also inappropriate to model all of the channels independently since there is a possibility of dependencies between some of the neurons.

From a machine learning perspective, modeling of the observed and hidden neural information can be 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 [9, 12]. We further assume that this compositional representation of the interacting processes occurs through space and time (i.e., between neurons at different times).

We also differ from other work in the BMI field by not taking the traditional regression approach of mapping the neural data directly to the patient hand kinematics with a conventional linear/nonlinear model [10]. Since the final BMI paradigm will not include desired kinematic information from paraplegics, we use generative models to explain the observable neural data. From these generative models, we divide the input space into regions that we call "motion primitives." These structures have been loosely touched upon in other work [13, 14]. The basic idea is that kinematic information or neural data can be decomposed into these "motion primitives" similar to phonemes in speech processing. In speech processing, graphical models (specifically HMMs) are the leading technology because they are able to capture very well the piecewise nonstationarity of speech [15, 16]. Since speech production is ultimately a motor function, graphical models can potentially also be useful for motor BMIs (also nonstationary) [11, 15]. By using smaller simpler components, more complicated arm kinematics can be constructed through the combination of these simple structures.

The ultimate goal is to decipher these underlying structures so that the model may one day be decoupled from the desired kinematics (for unsupervised modeling). It is our current goal to determine these underlying structures through a supervised mode in order to later exploit them in an unsupervised mode.

3. BOOSTED MIXTURES OF HMM CHAINS

3.1. Related work

With IC-HMMs, each neural channel is modeled with a hidden and observable random process (i.e., an HMM chain) [5]. Each neural channel HMM in the IC-HMM is computed independently. To calculate the log likelihood for the IC-HMM (Figure 2) we use

$$\log P(O \mid S, \Theta) = \sum_{i=1}^{N} \log P(O^i \mid S^i, \Theta^i), \tag{1}$$

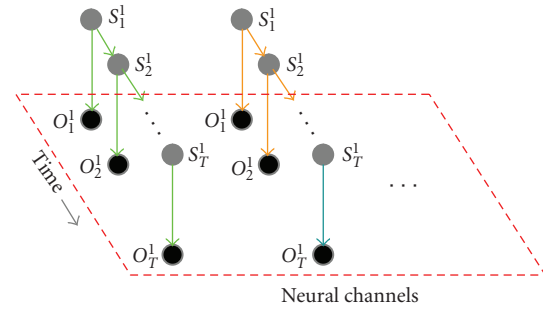

Figure 2: IC-HMM graphical model.

where Θ^i is the set of all the parameters for the model of a particular neural channel (where $i = 1, \ldots, N$ neural channels). Given that, $O^i = (O_1^i, O_2^i, \ldots, O_T^i)$ is a sequence of discrete binned neural firings and $S^i = (S_1^i, S_2^i, \ldots, S_T^i)$ represents the sequence of discrete hidden states or variables (of length T), this joint probability can be further decomposed into

$$\log P(O \mid S, \Theta) = \sum_{i=1}^{N} \left(\log P(S_1^i) + \sum_{t=1}^{T} \log P(O_t^i \mid S_t^i, \Theta^i) + \sum_{t=2}^{T} \log P(S_t^i \mid S_{t-1}^i, \Theta^i) \right). \tag{2}$$

From Figure 2 and (2), we see that the observable variables are IID and the hidden state sequence is Markovian (which is the classical HMM formulation). Essentially, the IC-HMM decomposes the multivariate input into the independent marginals so that only the likelihoods of the individual neurons are needed for the final calculation [5].

In order to move beyond the IC-HMM and exploit the complimentary information provided by the independent HMM chains, we first look to boosting. Boosting is a technique that creates different training distributions from an initial input distribution so that a set of weak classifiers is generated [17, 18]. The generated classifiers then form an ensemble vote for the current data example. It has been shown that these hierarchic combinations of classifiers achieve lower error rates than the individual base classifiers [17, 18].

Adaboost is the most widely used algorithm to evolve from boosting methods [19]. This algorithm sequentially generates weak classifiers based on weighted training examples. Essentially, the initial distribution of training examples is resampled each round (based on the distribution of the weights W_i) in order to train the next classifier up to R rounds [19]. The training examples that fail to be classified on a particular round receive an increased weighting so that the subsequent classifiers are more likely to be trained on these hard examples. With the initial values of the weights being $W_i = 1/n$, for $i = 1, \ldots, N$ samples, the update for each weight is

$$W_i \longleftarrow \frac{W_i \exp\left[\alpha_r \cdot 1_{(y \neq f_r(x))}\right]}{Z_r}, \tag{3}$$

where α_r are the external weights for each rth expert that has been trained for the rth round which act very similar to priors for the respective experts. Additionally, Z_r is a normalization factor in order to make the weights W_i a distribution. Concurrent to the weights W_i for training examples, each α_r is updated so that it can be used in a final ensemble vote (which is a linear combination of the α_r weights and the hypothesis of each expert):

$$\alpha_r = \log \frac{1 - \mathrm{err}_r}{\mathrm{err}_r}, \tag{4}$$

where

$$\mathrm{err}_r = \sum_{i=1}^{N} W_i [y_i \neq h(x_i)] \tag{5}$$

and the final ensemble output becomes

$$H(x) = \mathrm{sign}\left[\sum_{r=1}^{R} \alpha_r f_r(x)\right]. \tag{6}$$

The success of boosting has been attributed to the distribution of the "margins" of the training examples [17]. For further details on Adaboost or boosting with respect to the margin and relationships to support vector machines, see Schapier et al.

With respect to improving IC-HMM, Adaboost offers a promising way to implicitly find dependencies among the channels through the process of boosting classifiers during training. In our case, Adaboost is applied differently to the multidimensional neural data as explained in the next section.

3.2. Modeling framework

BMI data imposes specific constraints that require modifications to the standard procedures. First, our data is high dimensional (104 channels) and the importance of each channel to the mapping can be very different and is unknown. Second, the classes have markedly different prior probabilities. Third, the computational practicality of using thousands of experts to generate a decision is infeasible for hundreds of neural channels. Our approach uses Adaboost as a way to select multichannel experts that contribute the most information in the training set. Although the algorithm starts out with parallel training for the independent experts, gradually a winning expert is chosen for each hierarchic level to combine later into the ensemble. Since each level is formed in an unsupervised way, this algorithm is a hybrid between supervised and unsupervised learning, following a process similar to the mixture of experts framework [20].

Our first major departure from Adaboost results from how the ensemble is generated. Instead of forming one expert at a time, the M independent HMM chains are trained in parallel using the Baum-Welch formulation [21]. As explained, this splits the joint likelihood into marginals so that independent processes are working in simpler subspaces [5]. A ranking is then performed and a winner is chosen based on the classification performance for the current distribution of input examples. Specifically, the winner that minimizes the error with respect to the distribution of samples is chosen. To find the minimal error, we differ from Adaboost and use an Euclidean distance for the classes to avoid biasing class assignments (since classes may not have equal priors) [5]. Next, the remaining experts are trained within their respective subspace but relative to the errors of the previous winner. Finally, the W_i are used to select the next distribution of examples for the remaining experts. Similar to Adaboost, the remaining experts are trained on the hard examples from different subspaces. In turn, a hierarchic structure is formed as the winning experts affect the training on the local subspaces for the subsequent experts. During this process, they are implicitly modeling the dependencies among the channels.

As explained earlier, Adaboost uses α_m's as external weights to the classifiers as opposed to the W_i's which weight the training examples. Our computation of the α_m's is the second major departure from Adaboost since we use a mixture of experts formulation for the external weights or mixture coefficients. To find the mixture coefficients for the local classifiers, we look to the boosted mixture of experts (BME) [22]. With BME, improved performance is gained through the use of a confidence measure for the individual experts [22]. Although many different confidence measures exist, the majority use a scalar function of the expert's output which is then used as a static gating function or mixture coefficient [20, 22]. Our algorithm uses a simple measure for each expert based on the $L2$-Norm of the class errors (instead of the one outlined in (4)):

$$\alpha_m = 1 - \sqrt{\mathrm{err}_M^2 + \mathrm{err}_R^2}, \tag{7}$$

where err_M and err_R are the respective errors of the two classes in our problem, move and rest (which could generalize to more classes). We use β_m to substitute for the normal Adaboost formulation of α_m (4) to update the W_i's in (3).

Since there is a condition placed during the boosting phase to discard experts with less than 50% classification, negative alphas will not occur [19]. Notice that as the errors between the two classes are smaller, the weights for the experts become larger. We present the proposed Adaboost training algorithm for BM-HMM (see Algorithm 1).

The criterion for stopping is based on two conditions. The first stopping condition occurs, if the chosen experts are performing less than 50% classification. The second stopping condition occurs if the cross-validation set shows an increase in error or a plateau in performance for a significant number of rounds.

Since a single HMM chain is trained on a single neural channel, the number of parameters is very small and can support the amount of training data. The individual HMM chains in the BM-HMM contain around 70 parameters for a training set of 10000 examples as opposed to almost 18000 parameters necessary for a comparable CHMM (due to the dependent states) [5].

Given $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in X$, $y_i \in Y = \{-1, +1\}$,
initialize $W_i = 1/N$, $i = 1, \ldots, N$ samples.
For $r = 1, \ldots, R$ rounds,
(i) train all of the HMMs using samples from distribution W_i (with replacement);
(ii) find the mth expert that minimizes the error with respect to the distribution W_i:

$$\text{err}_m = E_W\left[1_{(y \neq f_m(x))}\right];$$

(iii) choose $\alpha_m = 1 - \sqrt{\text{err}_M^2 + \text{err}_R^2}$:

$$\beta_m = \log \frac{1 - \text{err}_m}{\text{err}_m};$$

(iv) update

$$W_i \longleftarrow \frac{W_i \exp\left[\beta_m \cdot 1_{(y \neq f_m(x))}\right]}{Z_r};$$

(v) the ensemble output:

$$H(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m f_m(x)\right],$$

where Z_r is a normalization factor so that W_i will be a distribution and $\sum_i W_i = 1$.

Algorithm 1

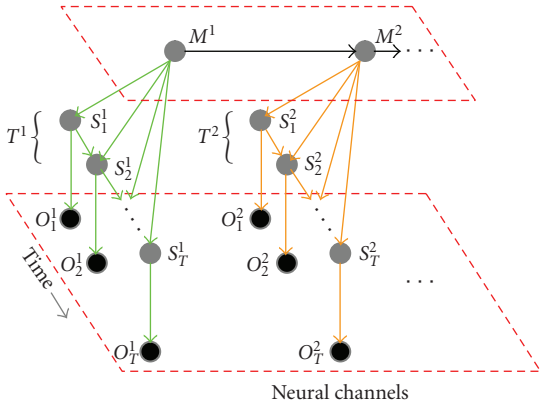

Figure 3: LM-HMM graphical model.

In the next section, we slightly increase the complexity of the BM-HMM to explicitly model dependencies between the neurons.

4. LINKED MIXTURES OF HMM CHAINS

4.1. Modeling framework

Let Z represents the set of variables (both hidden and observed) included in the probabilistic model. A graphical model (or Bayesian network) representation can then be used to provide insight into the probability distributions over Z encoded in a graph structure [23]. With this type of representation, edges of the graph represent direct dependencies between variables. Conversely, and more importantly, the absence of an edge allows the assumption of conditional independence between variables. Ultimately, these conditional independencies allow a more complicated multivariate distribution to be decomposed (or factorized) into simple and tractable distributions [23].

Since there is a variety of graphical model representations that decomposes the joint probability of the hidden and observed variables in Z, choosing the best approximation is overwhelming. To move beyond the IC-HMM and implicit dependencies in the BM-HMM, we establish another layer of hidden or latent variables to link and express the spatial dependencies between the lower level HMM structures (Figure 3), thus creating a clique tree structure T (since there are cycles), where hierarchic links exist between neural channels.

The log likelihood of the dynamic neural firings from all of the neurons for this structure (Figure 3) is

$$\begin{aligned}\log P(O \mid S, M, \Theta) = \log\, & P(M^1) + \sum_{i=2}^{N} \log\, P(M^i \mid M^{i-1}, \Theta) \\ & + \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \log P(O_t^i \mid S_t^i, \Theta^i) \right. \\ & \left. + \sum_{t=1}^{T} \log P(S_t^i \mid S_{t-1}^i, M^i, \Theta^i) \right),\end{aligned} \tag{8}$$

where the dependency between the tree cliques are represented by a hidden variable M in the second layer:

$$P(M^i \mid M^{i-1}, \Theta^i) \tag{9}$$

and the hidden state sequence S also has a dependency on the hidden variable M in the second layer:

$$P(S_t^i \mid S_{t-1}^i, M^i, \Theta^i). \tag{10}$$

This hierarchic model allows us to model data across multiple neural channels while also exploiting possible dependencies. From Figure 3, we see that the lower observable variables O^i are conditionally independent from the

second layer hidden variable M^i as well as the subgraphs of the other neural channels T^j (where $i \neq j$). The hidden variable M in the second layer of (8) can be interpreted as a mixture variable (when excluding the hierarchic links).

The LM-HMM implements a middle ground between making an independence assumption and a full dependence assumption. Since we are adding a layer of hidden variables, we are incurring a computational cost. The next section will show that we can make an approximation to ease computational costs but still bring in the richness of modeling the interrelationships between neurons.

4.2. Training with expectation maximization (EM)

Although using EM with graphical models gives us insight into probability distributions over our observed and hidden variables, some probabilities of interest are intractable to compute. Instead of using brute force methods to evaluate such probabilities, conditional independencies represented in the graphical model can be exploited. Often, approximations like Gibbs sampling, variational methods, and mean field approximations are applied in order to make the problem tractable or computationally efficient [23, 24].

For our model, we use a mean field approximation to allow interactions associated with tractable substructures to be taken into account [24]. The basic idea is to associate with the intractable distribution a simplified distribution that retains certain terms of the original distribution while neglecting others, replacing them with parameters u_i that we will refer to as "variational parameters." Graphically, the method can be viewed as deleting edges from the original graph until a forest of tractable structures is obtained. Edges that remain in the simplified graph correspond to terms that are retained in the original distribution and edges that are deleted correspond to variation parameters [24, 25].

We make approximations when finding the expectation of (8). In particular, we will first approximate $P(S_t^i \mid S_{t-1}^i, M^i, \Theta^i)$ by treating M^i as independent from S making conditional probability equal to the familiar $P(S_t^i \mid S_{t-1}^i, \Theta^i)$. 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 $P(M^i \mid M^{i-1}, \Theta^i)$ which we will address later [20].

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

$$\alpha_j(t) = P(O_1 = o_1, \ldots, O_t = o_t, S_t = j \mid \Theta). \tag{11}$$

We can calculate this quantity recursively by setting

$$\begin{aligned} \alpha_j(1) &= \pi_j b_j(o_1), \\ \alpha_k(t+1) &= \left[\sum_{j=1}^{N} \alpha_j(t) a_{jk}\right] b_k(o_{t+1}). \end{aligned} \tag{12}$$

The well-known backward procedure is similar:

$$\beta_j(t) = P(O_{t+1} = o_{t+1}, \ldots, O_T = o_T \mid S_t = j, \Theta), \tag{13}$$

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

$$\begin{aligned} \beta_j(T) &= 1, \\ \beta_j(t) &= \sum_{k=1}^{N} a_{jk} b_k(o_{t+1}) \beta_k(t+1). \end{aligned} \tag{14}$$

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

$$\gamma_j(t) = P(S_t = j \mid O, \Theta) \tag{15}$$

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

$$P(S_t = j \mid O, \Theta) = \frac{P(O, S_t = j \mid \Theta)}{P(O \mid S, \Theta)} = \frac{P(O, S_t = j \mid \Theta)}{\sum_{k=1}^{N} P(O, S_t = k \mid \Theta)} \tag{16}$$

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

$$\gamma_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}. \tag{17}$$

We also define

$$\xi_{jk}(t) = P(S_t = j, S_{t+1} = k \mid O, \Theta) \tag{18}$$

which can be expanded as

$$\begin{aligned} \xi_{jk}(t) &= \frac{P(S_t = j, S_{t+1} = k, O \mid \Theta)}{P(O \mid S, \Theta)} \\ &= \frac{\alpha_j(t) a_{jk} b_k(o_{t+1}) \beta_k(t+1)}{\sum_{j=1}^{N} \sum_{k=1}^{N} \alpha_j(t) a_{jk} b_k(o_{t+1}) \beta_k(t+1)}. \end{aligned} \tag{19}$$

The M-step departs from the Baum-Welch formulation and introduces the variational parameter [24]. 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

$$\begin{aligned} \hat{\pi}_j^i &= \frac{\sum_{i=1}^{I} u_i \gamma_1^i(j)}{\sum_{i=1}^{I} u_i}, \\ \hat{a}_{jk}^i &= \frac{\sum_{i=1}^{I} u_i \sum_{t=1}^{T-1} \xi_t^i(j,k)}{\sum_{i=1}^{I} u_i \sum_{t=1}^{T-1} \gamma_t^i(j)}, \\ \hat{b}_j^i(L) &= \frac{\sum_{i=1}^{I} u_i \sum_{t=1}^{T} \delta_{o_t, v_L} \gamma_t^i(j)}{\sum_{i=1}^{I} u_i \sum_{t=1}^{T} \gamma_t^i(j)}. \end{aligned} \tag{20}$$

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.

4.3. 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 ith HMM each having a prior probability of p^i. We want to estimate the set of parameters that maximize the likelihood function [25–27]:

$$\prod_{z=1}^{n}\sum_{i=1}^{I} p^i P(O^{zi} \mid S^i, \Theta^i). \tag{21}$$

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

$$u_{zi} = E\left[M^i \mid M^{i-1}, O^{zi} \mid \Theta^i\right] = \Pr\left[M^i = 1 \mid M^{i-1} = 1, O^{zi}, \Theta^i\right]. \tag{22}$$

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

$$u_{zi} = E\left[M^i \mid O^{zi}, \Theta^i\right] E\left[M^i \mid M^{i-1}\right]. \tag{23}$$

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

$$E\left[M^i \mid O^i, \Theta^i\right] = \Pr\left[M^{i=1} \mid O^i, \Theta^i\right] = \frac{p^i P(O^i \mid S^i, \Theta^i)}{\sum_{i=1}^{I} p^i P(O^i \mid S^i, \Theta^i)}. \tag{24}$$

Since the integration for the second term is much harder to compute, we look to 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 [28]. We can approximate the integration with

$$E\left[M^i \mid M^{i-1}\right] = \frac{1}{n}\sum_{z=1}^{n} \frac{P(O^{zi} \mid S^i, \Theta^i)}{P\left(O^{z(i-1)} \mid S^{i-1}, \Theta^{i-1}\right)}, \tag{25}$$

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

$$u_{zi} = \frac{p^i P(O^{zi} \mid S^i, \Theta^i)}{\sum_{i=1}^{I} p^i P(O^{zi} \mid S^i, \Theta^i)} \sum_{z=1}^{n} \frac{P(O^{zi} \mid S^i, \Theta^i)}{nP\left(O^{z(i-1)} \mid S^{i-1}, \Theta^{i-1}\right)}. \tag{26}$$

To compute the M-step,

$$\hat{p}^i = \frac{\sum_{z=1}^{n} u_{zi}}{\sum_{i=1}^{I}\sum_{z=1}^{n} u_{zi}} = \frac{\sum_{z=1}^{n} u_{zi}}{n}. \tag{27}$$

Table 1: Classification results (BM-HMM selected channels).

Model	Channels no.	% correct
	With monkey data	
IC-HMM	104	92.4%
IC-HMM	9	87.1%
BM-HMM	**9**	**92.0**%
Linear	104	88.3%
Linear	9	86.9%
	With rat data	
IC-HMM	16	62.5%
IC-HMM	6	58.3%
BM-HMM	**6**	**64.0**%
Linear	16	61.8%
Linear	6	56.9%

Table 2: Classification results (random LM-HMM selected channels).

Model	Channels no.	% correct
	With monkey data	
IC-HMM	104	92.4%
IC-HMM	9	89.5%
LM-HMM	**9**	**92.1**%
Linear	104	88.3%
Linear	9	87.8%
	With rat data	
IC-HMM	16	62.5%
IC-HMM	6	56.5%
LM-HMM	**6**	**62.3**%
Linear	16	61.8%
Linear	6	55.2%

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.

5. EXPERIMENTAL RESULTS

5.1. Quantitive results

In this section, we first show the results of our model on the two animal experiments. Next, we illustrate the effects of the algorithm by comparing the BM-HMM and LM-HMM chains to the IC-HMM chains. The BM-HMM and LM-HMM chains are ranked in order of the rounds, while the IC-HMM chains are ranked from the best to worst in terms of the individual chain's ability to classify. We conclude with an analysis of the methods by randomly selecting the chains for the LM-HMM and BM-HMM to understand if dependencies are being captured.

In Tables 1 and 2, we see a comparison of the results from the BM-HMM and LM-HMM versus the full IC-HMM

Table 3: Classification results (random BM-HMM selected channels).

Model	channels no.	% correct
	With monkey data	
IC-HMM	104	92.4%
IC-HMM	9	69.1%
BM-HMM	**9**	**69.4**%
Linear	104	88.3%
Linear	9	68.1%
	With rat data	
IC-HMM	16	62.5%
IC-HMM	6	56.5%
BM-HMM	**6**	**56.9**%
Linear	16	61.8%
Linear	6	55.1%

Table 4: Classification results (random LM-HMM selected channels).

Model	channels no.	% correct
	With monkey data	
IC-HMM	104	92.4%
IC-HMM	9	63.5%
LM-HMM	**9**	**65.4**%
Linear	104	88.3%
Linear	9	63.2%
	With rat data	
IC-HMM	16	62.5%
IC-HMM	6	55.4%
LM-HMM	**6**	**54.3**%
Linear	16	61.8%
Linear	6	54.8%

and a simple linear classifier created by a regression model followed by a threshold [5]. For the HMMs, we use three hidden states and an observation sequence length $T = 10$, which corresponds to one second of data (given the 100-millisecond bins). These choices were based on previous efforts to optimize performance [4]. For the linear classifier, a Wiener filter with a 10-tap delay (that corresponds to one second of data) is used. These parameters and thresholds were also chosen based on previous work [4, 13]. The classification results are determined on each data point in the training set. Cross-validation sets, Monte-Carlo runs, leave-K-out methodologies are also employed to make sure that our results are general.

As seen in Tables 1, 2, 3, and 4, the correct classification percentage is much higher for the monkey data than the rat data. This is explained primarily by the experimental conditions. Since the rat experiment is not as controlled as the monkey experiment, the rat dataset is much noisier. In particular, as the rat moves around the cage there are many training and testing samples labeled as nonmovement,

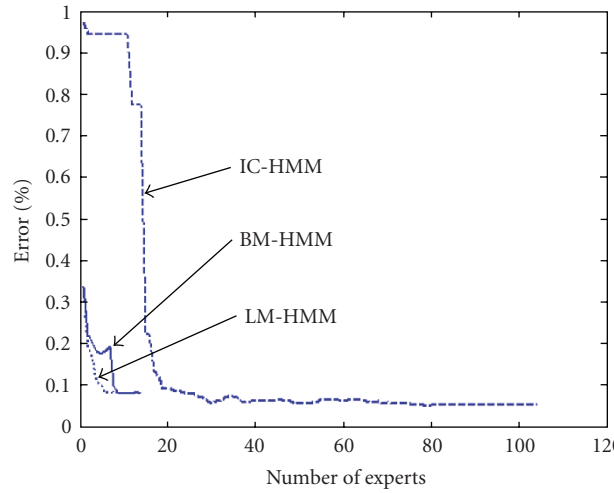

Figure 4: Monkey expert adding experiment.

when the animal is actually moving (only lever presses are counted as movement in this setup). This explains the much lower performance on the rat data. Also, fewer neural channels were collected from the rat than the monkey [1, 7]. Apart from this difference, the same trends are seen in both datasets.

To give a fair comparison between the methods, the same neural channels that were chosen by the BM-HMM are used with the linear model and the IC-HMM. For the LM-HMM, a different set of neurons (although some overlap the BM-HMM) was selected based on the algorithm, but the number of final channels were kept the same. As compared in Tables 1 and 2, the BM-HMM and LM-HMM perform on par with the IC-HMM, but with the added benefit of dimensionality reduction. Effectively, the same performance is obtained with a fraction of the number of neural channels. For this experiment, the performance on the rat data is better with the BM-HMM, and slightly worst with the LM-HMM.

In Tables 3 and 4, we wanted to see the effect of randomly selecting the experts. Notice the results in these tables are poor in comparison to Tables 1 and 2. In particular, Tables 1 and 2 show a significant decrease in performance modeling the monkey data. The performance reduction is less pronounced for the data collected from the rat. We believe this is due to the available number of channels that we can randomly select. Since the monkey data is collected from a greater number of channels, there is an increased possibility of selecting a bad expert for both the BM-HMM and LM-HMM. It is interesting that the α_m for both models on both datasets were approximating similar values (like an averaging), almost as if the neurons randomly selected are more independent than dependent. This is also evident since the result of the IC-HMM using the same channels has similar performance with respect to both models in Tables 3 and 4. Even Figure 8 points to the channels having zero correlation amongst themselves, alluding to independency.

The results demonstrate three interesting points. First, it is significant that on the monkey data, nine BM-HMM

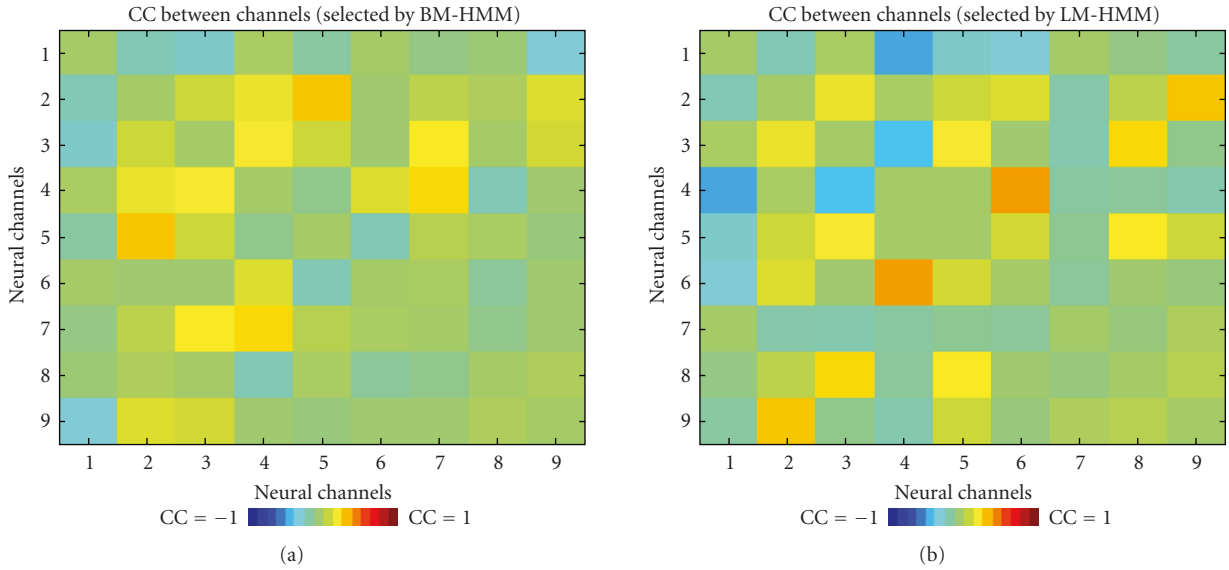

Figure 5: Correlation coefficients between channels (monkey moving).

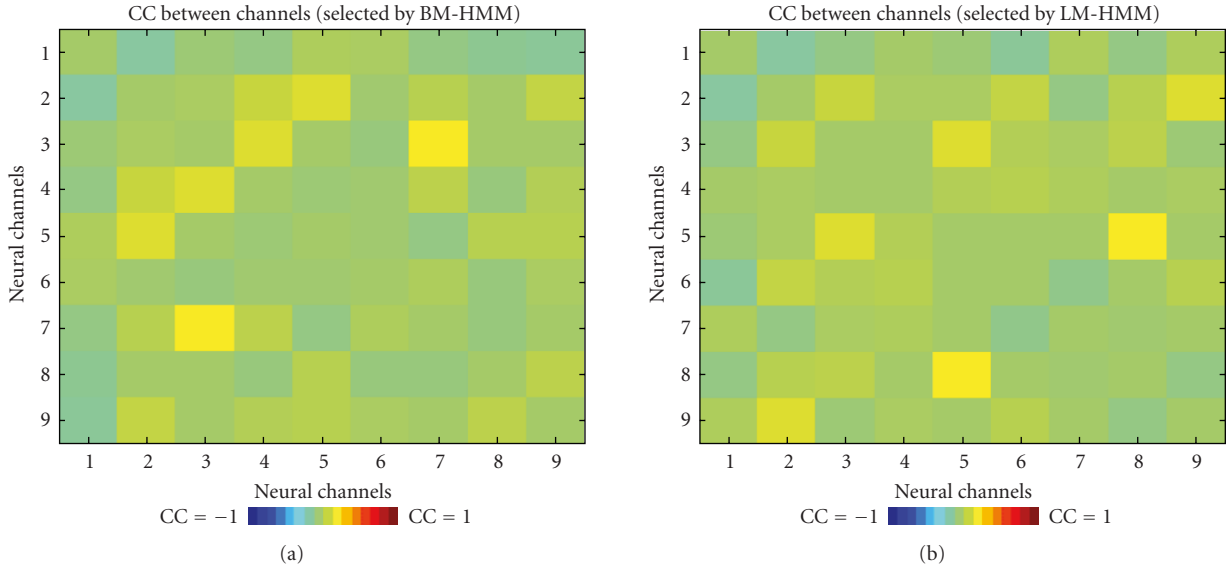

Figure 6: Correlation coefficients between channels (monkey at rest).

and LM-HMM chains outperform the linear classifier that uses the full input space. Second, the subset of experts that are chosen by the BM-HMM seems to perform well on the linear model. This result is expected since the BM-HMM chains select neural channels with important complimentary information. Third, when comparing the BM-HMM and LM-HMM to the Wiener filter and the IC-HMM using the same subset of neural channels, the results show that the hierarchic training of the BM-HMM and LM-HMM provides a significant increase in performance. We believe this is due to the dependencies that are being exploited during each round of training, where the other models simply try to uniformly combine all the neural information into a single hypothesis. This is supported by Tables 3 and 4 since the LM-HMM and BM-HMM appears to default to an independent approximation. Finally, other BMI researchers apply sensitivity analysis to understand the importance of a neural channel respective to the kinematics performed by the subject. In contrast the BM-HMM and LM-HMM channel selection is trying to improve classification results by exploiting dependencies between channels as well as kinematics. Interestingly, some of the channels selected in both datasets do overlap some of the same neurons selected during sensitivity analysis [3, 7].

Additionally, if we do an expert adding experiment in which the best ranked experts are added one by one to the ensemble vote, an interesting result emerges. In Figure 4, as the BM-HMM chains are added, the error rate quickly

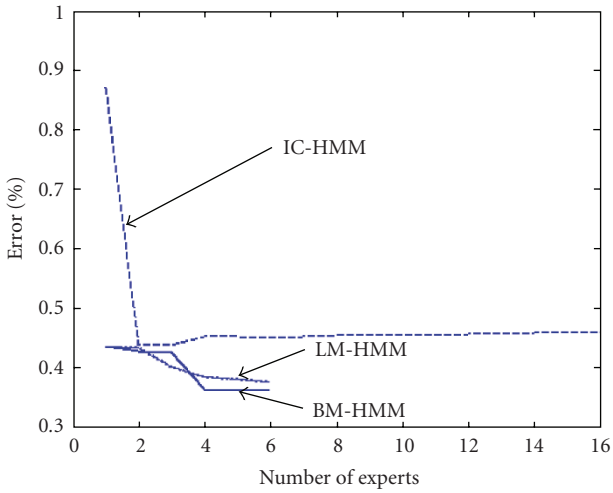

Figure 7: Rat expert adding experiment.

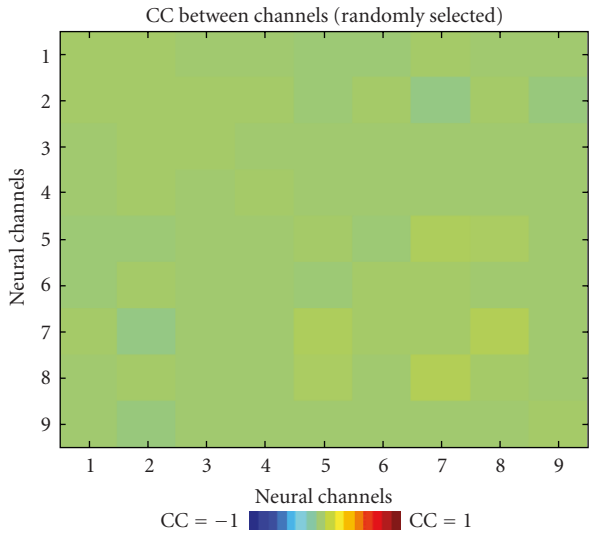

Figure 8: Correlation coefficient between channels (randomly selected for monkey).

decreases below the IC-HMM error when applied to the monkey neural data. The LM-HMM has an even faster drop in error but does not achieve the same result as the BM-HMM. Figure 7 shows a similar result when applied to the rat neural data. Overall, we find that the boosted mixtures or linked mixtures are exploiting more useful and complimentary information for the final ensemble than the simple IC-HMM.

6. DISCUSSION AND FUTURE WORK

Based on the results, the BM-HMM performs better than the LM-HMM due to the fact that the BM-HMM takes advantage of using the errors in classification when determining the α's. In contrast, the LM-HMM only uses the likelihood information for updating the priors and variational parameters in order to create explicit dependencies. Despite the slightly lower performance of the LM-HMM, it is significant that the model can still find and exploit neural channels that are important for movement. Additionally, since the likelihood is only required, perhaps the algorithm could be improved by dynamically updating the prior and variational parameters during the evaluation of the test set. This could allow us to modify the temporal and spatial dynamics in an online fashion.

To understand how the two methods differ in selecting the channels, we plotted in Figures 5 and 6, the correlation coefficients between the subset of channels chosen by the BM-HMM and LM-HMM. Interestingly, we see that more of the channels in the LM-HMM have a large positive and large negative correlation (with respect to the move class). In contrast, the BM-HMM has some positive correlation amongst its subset, but few channels exhibit negative correlation. Figure 8 shows the correlation coefficient between the channels that were randomly selected. Notice how the correlations between channels are near zero. These results suggest that the LM-HMM prefers stronger correlated channels (i.e., dependant) than the BM-HMM and much stronger than randomly selecting channels (which seems to exhibit more independence between channels).

With respect to other algorithms in the machine learning community, there are a few ways to interpret the BM-HMM. BM-HMMs can be thought of as a modification to boosting or even a simpler version of the mixture of trees algorithm, if the HMM chains are interpreted as binary stumps [29]. Additionally, the temporal Markovian dynamics coupled with the hierarchic structure and mixture modeling can be thought of as a simple approximation to tree structured HMMs [23]. Other work has focused on solving this problem of boosting multiple parallel classifiers [30]. Other authors have proposed boosting solutions that reduce the dimensionality of the input data [30–32]. From their perspective, the multidimensional inputs are treated as simple features of a single random process [30]. We differ from this perspective by assuming the input space is composed of multiple random processes that are interacting with each other in some unknown way. By decomposing the input space into multiple random processes, the local contributions of the individual processes are exploited rather than using the global effect of a single process. This type of algorithm also has components similar to the mixture of experts (MOEs) algorithm.

Although some similarities exist between MOE and both of our models, it is important to note that the MOE has the advantages of localization and the use of a dynamic model for combining the outputs from the experts [20, 22]. Others have proposed a similar formulation of building boosted hierarchic structures with the MOE algorithm, but these efforts lack the Markovian dynamics that are inherent with BM-HMMs [20, 22, 23, 33]. Our work also differs from these formulations, since our "gating functions" are static and behave like priors for the experts [5]. Because of the static "gating functions," it could even be closer aligned with an ensemble method, but in future work we hope to exploit a dynamic gating function similar to MOE, where temporal dynamics changes the output of the switching classifier.

Finally, for BMIs, the selection of only 9 neurons (for the monkey data) from the large number of cortical neurons must be interpreted in the proper context. Notice that we are just selecting the neurons that are capable of allowing recognition of movement or rest, which is a very simple task. Our experience with cursor tracking tasks shows that a much larger number of neurons is required, and since the data is nonstationary, neurons are being multiplexed across time to implement tasks. This implies that in principle, the larger the number of neurons the better, provided that one has sufficient data to train the models. Nonetheless, this HMM methodology will be extremely useful for the design of BMIs, especially if ways are found to discover the time varying dependencies.

ACKNOWLEDGMENTS

The authors would like to thank Johan Wessberg and Miguel A. L. Nicolelis for sharing their monkey neural data and experience and Justin Sanchez for his rat data. They also thank Antonio Paiva for his linear classifier code and Michael Nechyba for providing a helpful hand in this research as well as thanking Jeremy Anderson and Grisel Morales for editing the paper. This paper was supported by DARPA Project no. N66001-02-C-8022 and NSF Project no. CNS-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] 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.

[3] 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 12th IEEE Workshop on Neural Networks for Signal Processing*, pp. 139–148, Martigny, Switzerland, September 2002.

[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/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, vol. 3, pp. 3612–3617, Las Vegas, Nev, USA, October 2003.

[5] S. Darmanjian, S.-P. Kim, M. C. Nechyba, J. C. Principe, J. Wessberg, and M. A. L. Nicolelis, "Independently coupled HMM switching classifier for a bimodel brain-machine interface ," in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pp. 379–384, Maynooth, Ireland, September 2006.

[6] S. Darmanjian, A. R. C. Paiva, J. C. Principe, et al., "Hierarchal decomposition of neural data using boosted mixtures of independently coupled hidden Markov chains," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 89–93, Orlando, Fla, USA, August 2007.

[7] J. C. Sanchez, J. C. Principe, and P. R. Carney, "Is neuron discrimination preprocessing necessary for linear and nonlinear brain machine interface models," in *Proceedings of the 11th International Conference on Human-Computer Interaction*, Vegas, Nev, USA, July 2005.

[8] E. Todorov, "On the role of primary motor cortex in arm movement control," in *Progress in Motor Control III*, chapter 6, pp. 125–166, Human Kinetics, Champaign, Ill, USA, 2003.

[9] 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. 701–707, 2001.

[10] M. A. Lebedev and M. A. L. Nicolelis, "Brain-machine interfaces: past, present and future," *Trends in Neurosciences*, vol. 29, no. 9, pp. 536–546, 2006.

[11] R. B. Northrop, *Introduction to Dynamic Modeling of Neurosensory Systems*, CRC Press, Boca Raton, Fla, USA, 2001.

[12] W. J. Freeman, *Mass Action in the Nervous System*, Academic Press, New York, NY, USA, 1975.

[13] 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.

[14] N. G. Hatsopoulos, Q. Xu, and Y. Amit, "Encoding of movement fragments in the motor cortex," *Journal of Neuroscience*, vol. 27, no. 19, pp. 5105–5114, 2007.

[15] 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.

[16] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2001.

[17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," in *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, pp. 322–330, Nashville, Tenn, USA, July 1997.

[18] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[19] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning (ICML '96)*, pp. 148–156, Bari, Italy, July 1996.

[20] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[21] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[22] R. Avnimelech and N. Intrator, "Boosted mixture of experts: an ensemble learning scheme," *Neural Computation*, vol. 11, no. 2, pp. 483–497, 1999.

[23] M. I. Jordan, *Learning in Graphical Models*, MIT Press, Cambridge, Mass, USA, 1999.

[24] 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.

[25] A. Ypma and T. Heskes, "Automatic categorization of web pages and user clustering with mixtures of hidden Markov models," in *Proceedings of the 4th International Workshop on MiningWeb Data for Discovering Usage Pattens and Profiles (WEBKDD '02)*, pp. 35–49, Edmonton, Canada, July 2002.

[26] 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 on Computer Vision and Pattern Recognition (CVPR '03), vol. 1, pp. 375–381, Madison, Wis, USA, June 2003.

[27] 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.

[28] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer, New York, NY, USA, 1995.

[29] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classication and Regression Trees*, Wadsworth International Group, Belmont, Calif, USA, 1984.

[30] Y. Sun and J. Li, "Iterative RELIEF for feature weighting," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 913–920, ACM Press, Pittsburgh, Pa, USA, June 2006.

[31] M. Martínez-Ramón, V. Koltchinskii, G. L. Heileman, and S. Posse, "fMRI pattern classification using neuroanatomically constrained boosting," *NeuroImage*, vol. 31, no. 3, pp. 1129–1141, 2006.

[32] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. 511–518, Kauai, Hawaii, USA, December 2001.

[33] M. Meila and M. I. Jordan, "Learning fine motion by Markov mixtures of experts," in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., vol. 8, MIT Press, Cambridge, Mass, USA, 1996.