Hierarchal Decomposition of Neural Data using Boosted Mixtures of Hidden Markov Chains and its application to a BMI

Shalom Darmanjian[1], Antonio Paiva[1], Jose Principe[2], Justin Sanchez[3]

Abstract—

In this paper, we propose a simple algorithm that takes multidimensional neural input data and decomposes the joint likelihood into marginals using Boosted Mixtures of Hidden Markov Chains (BM-HMM). The algorithm applies techniques from boosting to create hierarchal dependencies between these marginal subspaces. Finally, borrowing ideas from mixture of experts, the local information is weighted and incorporated 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 Brain Machine Interfaces (BMIs).

I. Introduction

The field of Brain Machine Interfaces (BMIs) is devoted to bridging the gap between mind and machine so that paralyzed patients may one day have their mobility restored. One of the focal points in this field has been the development of linear and non-linear models that map neural firing patterns of an animal to a robotic prosthetic [8], [12], [13]. Usually, in this type of experiment, neural data is recorded from a primate or rat cortex as it engages in a movement task. 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 [12], [13].

For the above mentioned experiments, our group established that by using multiple feed-forward prediction models to map discrete portions of the neural data to respective portions of the continuous trajectory, the overall trajectory reconstruction is improved [2], [3]. The classifier that is responsible for switching between these different feed-forward models is the main focus of this paper.

The previous switching classifier was an ensemble method that incorporated multiple independent experts. The experts were single neural-channel HMM chains that formed an Independently Coupled Hidden Markov Model (IC-HMM) [3]. One limitation of this algorithm is that as an increasing number of neurons are sampled from brain (as is the current trend in BMIs), the number of independent models could grow to an unmanageable level. Although it is still computationally more efficient than using a model that has full dependencies, like the Coupled Hidden Markov Model (CHMM), it is still desired that the input dimensionality be reduced to avoid this pitfall. Additionally, since it is likely that the independence assumption is not realistic for all of the neurons, finding dependencies between some of the neurons could prove beneficial for final kinematic reconstruction or other biologically inspired modeling.

In this paper, we take multidimensional neural input data and decompose the joint likelihood into marginals using Boosted Mixtures of Hidden Markov Chains (BM-HMM). The algorithm applies techniques from boosting to create hierarchal dependencies between these marginal subspaces. Finally, borrowing ideas from mixture of experts, the local information is weighted and incorporated 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 Brain Machine Interfaces (BMIs).

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

II. Approach

A. 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 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 [8], [3].

In the second experiment, a male Sprague-Dauley rat performed a go-no go lever pressing task as neural data was recorded. This dataset contains 16 neural cells that were collected with micro-wire arrays implanted in the forelimb region of the left primary cortex (M1) [15]. Subsequently, the data was spike detected and spike sorted using thresholds and template matching [15]. The lever presses were recorded simultaneously with the neural activity. The beginning of each trial was signaled to the rat by a LED indicating when to press the lever in order to receive a reward [15].

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 4mm/s

[1]Student Member, IEEE [2]Fellow, IEEE, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6200. [3]Member, IEEE, Department of Pediatrics, Division of Neurology, University of Florida, Gainesville, FL 32611

are labeled as part of the movement class. Included in this movement class are the times when the monkey momentarily holds it's arm during a reach for food or it's mouth [3], [1]. 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 [15].

For both experiments, the neural data is binned into 100ms counts which is consistent with the neural science community [1], [9]. Consequently, the movement data is down-sampled to match the 10 Hz neural bin counts. Specifically, the time recording for the monkey experiment corresponds to a dataset of 23000×104 time bins. For the rat experiment the dataset consist of 13000×16 time bins [8], [15].

B. Motivation

From a machine learning perspective, the neural data can be viewed as observable and hidden random processes that are interacting with each other in some unknown way. Specifically, 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 [10], [9]. We further assume that this compositional representation of the interacting processes occurs through time and space (i.e. between neurons at different times).

Since the final BMI paradigm will not include desired kinematic information from paraplegics, generative models are used to explain the observable neural data. The ultimate goal is to decipher the underlying structure so that the model may one day be decoupled from the desired kinematics (for unsupervised modeling).

Our first attempt to characterize these interacting processes was to simply compress the multidimensional input space into a single observable process while modeling all the hidden information as a single hidden process. This was accomplished by using vector quantization to compress the multidimensional input into 1D cluster labels for a single discrete HMM chain[2].

Unfortunately, there was a performance plateau since vector quantization assumes synchronous state transitions between all of the neural channels which are known to be non-stationary [2], [16]. By quantizing the non-stationary signals, some of the asynchronous behavior of the neurons is destroyed. Additional factors in the performance plateau were due to the simple distortion experienced when quantizing a multidimensional input space [2]. In turn, this work motivated us to remove the quantization errors and eliminate the assumption that all of the processes have synchronous state transitions.

In our second modeling attempt, we relaxed the requirement that the states of all the neural channels change synchronously and instead modeled them with an Independently Coupled Hidden Markov Model (IC-HMM) [3]. This model removes the need to compress the multidimensional input since it uses the input directly. Consequently, this model

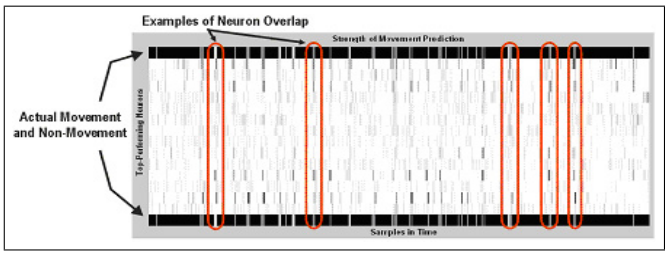

Fig. 1. Probabilistic ratios of 14 neurons

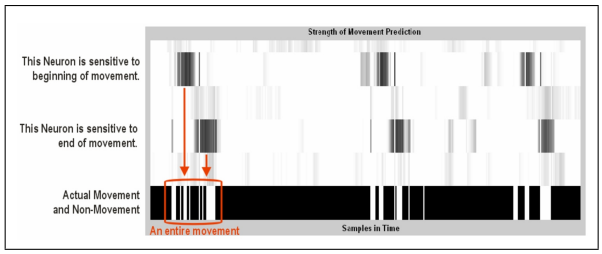

Fig. 2. Zoomed in version of the probabilistic ratios

improved performance and was more computationally appealing than the extreme alternative, a fully Coupled Hidden Markov Model (CHMM) [3].

Unfortunately, there are two drawbacks to the IC-HMM that are addressed in this paper. First, as additional neural channels are sampled from the brain (which is the trend with BMIs), it is not computationally attractive to increase the number of models. Therefore, it would be beneficial to reduce the dimensionality of the dataset so that only the pertinent information is used to classify the data. Further, a reduction in input dimensionality would reduce the number of free parameters a mapping filter would need for final kinematic reconstruction. The second drawback to the IC-HMM stems from its inability to exploit some of the dependency or synchrony information provided by the neural channels. In particular, Figures 1 and 2 show some possible synchrony between the channels or complimentary information at different points in time - respective to each class [3]. Identifying these underling dependencies may help provide a better biological understanding and move us closer to the ultimate goal of unsupervised clustering in the input space. The goal of the work in this paper is to find these dependencies and exploit only the pertinent information (relative to the kinematics) in order to reduce the input dimension without sacrificing classification performance.

C. Related Work

In this section we first detail the IC-HMM and then connect the development of the BM-HMM to related work.

When using IC-HMMs for BMIs, there is a simple assumption that the neurons are acting independently. In turn, each neural channel is then modeled with a hidden and observable random process, i.e. an HMM chain [3]. The independence assumption is supported by neuroscience literature that explains that neurons in the brain are known to modulate independently from other neurons [10] during the

control of movement. Since each neural channel HMM in the IC-HMM is computed independently, the joint likelihood decomposes

$$P(O_T^{(1)}, O_T^{(2)}, ...O_T^{(D)}, |\lambda_{full}) \tag{1}$$

into the product of the marginals

$$\prod_{i=1}^{D} P(O_T^{(i)}|\lambda^{(i)}) \tag{2}$$

of the observation sequences (each length T) for each dth HMM chain λ.

For classification, a likelihood ratio is used in conjunction with a threshold to determine which of the two classes the data is to be classified as.

$$\prod_{i=1}^{D} P(O_T^{(i)}|\lambda_M^{(i)}) > \prod_{i=1}^{D} P(O_T^{(i)}|\lambda_R^{(i)}) \tag{3}$$

or more aptly,

$$l(O) = \frac{\prod_{i=1}^{D} P(O_T^{(i)}|\lambda_M^{(i)})}{\prod_{i=1}^{D} P(O_T^{(i)}|\lambda_R^{(i)})} > \zeta \tag{4}$$

Ratios greater than the threshold ζ are classified as part of the movement class, and those less than ζ as the rest class. This threshold acts as a global weighting for one class or another across all of the chains. The use of thresholds for the likelihood ratio has been used in neural science and other areas of research [7], [11]. Even the idea of partitioning the joint space into marginals has shown up in literature under different names [20], [23], [21].

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 [24], [25]. The generated classifiers then form an ensemble vote for the current data example. It has been shown that these hierarchal combinations of classifiers achieve lower error rates than the individual base classifiers [24], [25].

Adaboost is the most widely used algorithm to evolve from boosting methods. This algorithm sequentially generates weak classifiers based on weighted training examples. Essentially, the initial distribution of training examples is re-sampled each round (based on the distribution of the weights) in order to train the next classifier. 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. Concurrent to the weights for training examples, external weights for each classifier are updated so that they can be used in a final ensemble vote (which is a linear combination of weights and the hypothesis). The success of boosting has been attributed to the distribution of the "margins" of the training examples. For further details on Adaboost or boosting with respect to the margin and relationships to support vector machines see Schapire et al.

With respect to improving IC-HMM, Adaboost offers a promising way to weight and hierarchally split the multidimensional training examples. Unfortunately, to take a 104 HMM chains (as with the IC-HMM) and boost them individually (creating multiples of 104) would be computationally prohibitive. Other work has focused on solving this problem of boosting multiple parallel classifiers. There have even been proposed boosting solutions for reducing the dimensionality of the input data [21], [22], [26]. From their perspective, the multidimensional inputs are treated as simple features of a single random process [26]. 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. In order to exploit the local information and take advantage of the different subspaces of the input we look to the mixture of experts algorithm.

Mixture of experts is a very well established adaptive learning algorithm in which several expert classifiers are trained in conjunction with a gating function [14], [23] . The gating function, in turn, can be thought of as having the task of assigning weights to experts in terms of their contribution to the ensemble vote [14], [23] Although some similarities exist between this method and boosting, it is noted that the mixture of experts has the advantage of localization and the use of a dynamic model for combining the outputs from the experts [14], [23]. Others have proposed a similar formulation of building boosted hierarchal structures with mixture of experts but lack the Markovian dynamics that are inherent with BM-HMMs [14], [18], [17], [23].

In the next section, we give specific details on the structure and training of the BM-HMM and point out the similarities to the above mentioned formulations.

D. Boosting mixtures of IC-HMM Chains

The proposed algorithm is intended for an arbitrary number of HMM chains not to exceed the number of neural inputs. Although the algorithm starts out with parallel training for the independent experts, gradually a winning expert is chosen for each hierarchal level to combine into the ensemble. This algorithm marries some concepts of boosting with concepts from mixture of experts.

In terms of boosting, we borrow some of Schapire & Freund procedures in discrete Adaboost [25].

In this algorithm:

1) Start with weights $w_i = 1/N, i = 1, 2, ..., N$.
2) Repeat for m=1,2,...,M classifiers:
3) Fit each classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
4) Compute $err_m = E_w[1_{(y \neq f_m(x))}]$, where $\alpha_m = log((1 - err_m)/err_m)$
5) Set $w_i \leftarrow w_i exp[\alpha_m \cdot 1_{(y \neq f_m(x))}]$, renormalize so that $\sum_i w_i = 1$
6) The ensemble output $sign[\sum_{m=1}^{M} \alpha_m f_m(x)]$

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 [5]. As explained, this splits the joint likelihood into marginals so that independent processes are working in simpler subspaces [3]. A ranking is then performed and a winner is chosen based on the classification performance for the current distribution of input examples. The ranking is done by looking at the Euclidean distance between the percentage correctly classified for each class. Next, the remaining experts are trained within their respective subspace but relative to the errors of the previous winner. Finally, the w_i is 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 hierarchical structure is formed as the winning experts affect the training on the local subspaces for the subsequent experts.

With Adaboost the α_m's are used as external weights to the classifiers as opposed to the w_i's which weight the training examples. Computing the α_m's is the second major departure from Adaboost since a mixture of experts formulation is used 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) [23]. With BME, improved performance is gained through the use of a confidence measure for the individual experts [23]. Although many different confidence measures exist, the majority are a scalar function of the expert's output which is then used as a gating function or mixture coefficient [14], [23]. Our algorithm uses a simple measure for each expert based on the L2-Norm of the class errors:
Instead of

$$\alpha_m = log((1 - err_m)/err_m) \tag{5}$$

we use,

$$\alpha_m = 1 - \sqrt{err_M^2 + err_R^2} \tag{6}$$

Since there is a condition placed during the boosting phase not to use experts with less than 50% classification, negative alphas will not occur. Notice that as the errors between the two classes are smaller, the weights for the experts become larger.

The following describes how the different procedures are combined to create the BM-HMM:

1) Start with weights $w_i = 1/N, i = 1, 2, ..., N$.
2) Train all of the HMM chains on the full data set
3) Rank and select the best performing HMM chain on current distribution of examples
4) Using the mth winner, compute $err_m = E_w[1_{(y \neq f_m(x))}]$, where $\beta_m = log((1 - err_m)/err_m)$ and $\alpha_m = 1 - \sqrt{err_M^2 + err_R^2}$
5) Set $w_i \leftarrow w_i exp[\beta_m \cdot 1_{(y \neq f_m(x))}]$, renormalize so that $\sum_i w_i = 1$
6) Resample distribution with replacement based on winner

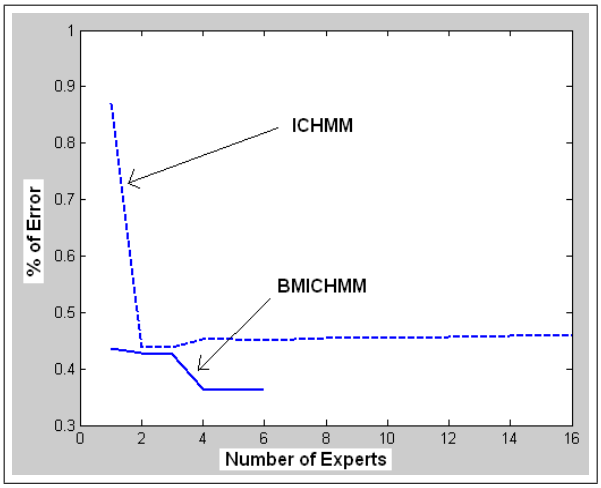

Fig. 3. HMM chain adding experiment for Rat data

7) Train remaining experts on respective subspaces
8) Go to step 3 until stopping criterion is met.
9) The ensemble output $sign[\sum_{m=1}^{M} \alpha_m f_m(x)]$

The criterion for stopping is based on two conditions. The first stopping condition occurs if the individual 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 10,000 examples as opposed to almost 18,000 parameters necessary for a comparable CHMM (due to the dependent states) [3].

III. Results

TABLE I
Classification Performance

Model	*#channels*	*% correct*
With Monkey Data		
IC-HMM	104	92.4%
IC-HMM	9	87.1%
BM-HMM	**9**	**92.0%**
Wiener Filter	104	88.3%
Wiener Filter	9	86.9%
With Rat Data		
IC-HMM	16	62.5%
IC-HMM	6	58.3%
BM-HMM	**6**	**64.0%**
Wiener Filter	16	61.8%
Wiener Filter	6	56.9%

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 chains, ranked from best to worst, to the IC-HMM chains, also ranked from best to worst. Finally, we conclude with an analysis of parallel peri-event time histograms to understand which neural channels the BM-HMM chains are selecting for the ensemble.

In Table 1, we see a comparison of the results from the BM-HMM chains versus the full IC-HMM and a linear classifier. For the HMMs, we use three hidden states and an observation sequence length $T = 10$, which corresponds to a second of data (given the 100ms bins). These choices were based on previous efforts to optimize performance [2]. For the linear classifier, a Wiener filter with a 10-tap delay (that corresponds to a second of data) is used. These parameters were also chosen based on previous work [2], [6].

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. As can be seen from the table, the BM-HMM performs on par with the IC-HMM, but with the added benefit of dimensionality reduction. For this experiment the performance on the rat data is better with the BM-HMM. The improvement could be due to the algorithm's ability to remove noisy or unimportant neural channels.

Furthermore, these results demonstrate three other interesting points. First, it is significant that on the monkey data, nine BM-HMM chains outperform the linear classifier that uses the full input space. Second, the subset of experts that are chosen by the BM-HMM seem to perform well on the linear model. This result is expected since the BM-HMM chains select neural channels with important complimentary information. Lastly, when comparing the BM-HMM to the Wiener filter and the IC-HMM using the same subset of neural channels, the hierarchal training of the BM-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.

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 3, as the BM-HMM chains are added, the error rate quickly decreases below the IC-HMM error when applied to the monkey neural data. Figure 4 shows a similar result when applied to the rat neural data. Overall, we find that the boosted mixtures are exploiting more useful and complimentary information for the final ensemble than the simple IC-HMM.

To understand what possible information the boosted mixtures are selecting from the neural channels, we plot in Figures 5 and 6 the peri-event time histogram for all the neural channels in parallel. These histograms present the firing rate of the different neurons averaged across a single event (i.e the start of a movement trial) [27]. For these figures, the mean firing rate is subtracted so that the light colors illustrate decreased firing rates and the darker colors

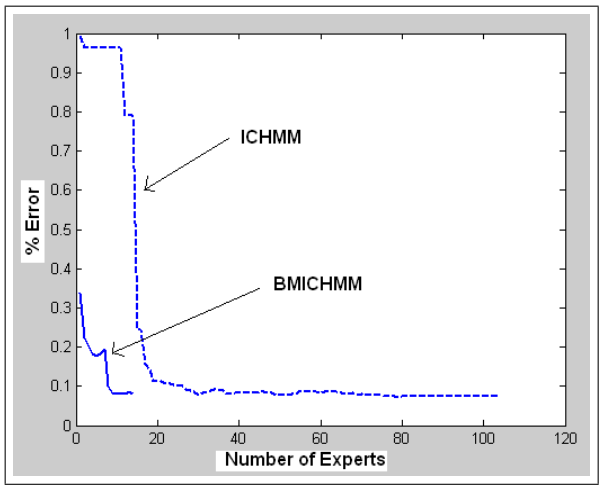

Fig. 4. HMM chain adding experiment for monkey data

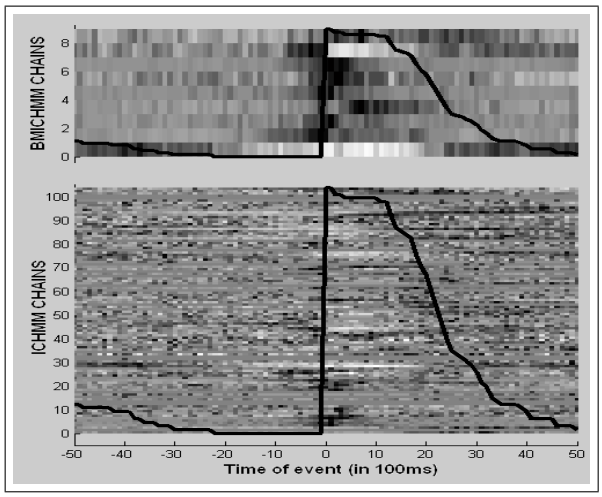

Fig. 5. Parallel peri-event histogram for monkey neural data

indicate increased firing rates. Overlaid and stretched on each image is the average of the movement trials. It is interesting to see how some channels have no pattern associated with them whereas others appear to have a consistent pattern.

Notice how the BM-HMM selects neural channels that display both an increase in firing activity as well as neurons that decrease their firing during the onset of movement. Although we can not biologically substantiate that these neurons are being inhibited, empirically the BM-HMM appears to use this inhibition-like activity.

IV. Discussion and Future Work

Overall, the BM-HMM algorithm is very simple to train and computationally efficient, while also able to reduce the input dimensionality. It accomplishes this by taking the multidimensional neural input and decomposing the joint likelihood into marginals using a BM-HMM. Techniques from boosting are used to create hierarchal dependencies between these marginal subspaces. Then ideas from the mixture of experts weight the local information. The results demonstrate that fewer neurons are needed to achieve similar

or better results than using the IC-HMM and superior to a linear model (both of which use the full input space).

With respect to related algorithms, 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 [19]. Additionally, the temporal Markovian dynamics coupled with the hierarchal structure and mixture modeling can be thought of as a simple approximation to tree structured HMMs [18]. Regardless of the algorithmic interpretation, the BM-HMM algorithm is modeling the neural data as multiple interacting processes where hidden dependencies between neurons are approximated in a simple and efficient way.

In future work, we would like to move to an unsupervised version of the algorithm. As previously mentioned, the final BMI paradigm will not include kinematic information from a paraplegic; therefore it is imperative that BMI algorithms work in an unsupervised or semi-supervised manner. Perhaps we will need to look beyond simple HMM chains to compose the boosted mixtures in order to find a better spatial and temporal representation of the various interacting processes.

V. Acknowledgements

We thank Johan Wessberg and Miguel A. L. Nicolelis for sharing their monkey neural data and experience. We also thank Michael Nechyba for having a helpful hand in this research and Jeremy Anderson for his help in editing. This paper was supported by DARPA project #N66001-02-C-8022 and NSF project #0540304. A. R. C. Paiva was supported by Fundação para a Ciência e a Tecnologia under grant SFRH/BD/18217/2004.

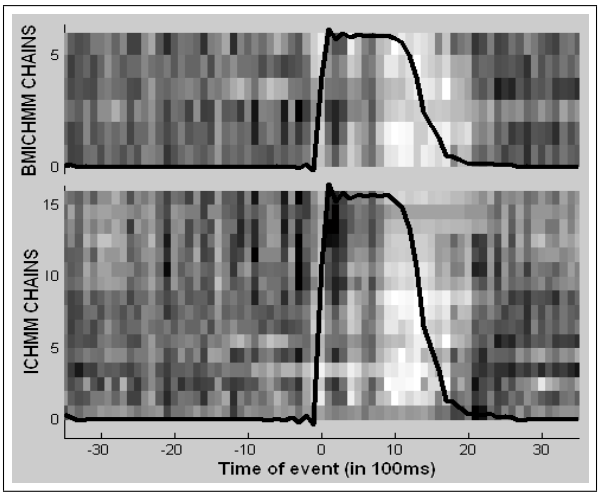

Fig. 6. Parallel peri-event histogram for rat neural data

References

[1] Todorov E., *On the role of primary motor cortex in arm movement control*, To appear in Progress in Motor Control III, 2003.

[2] S. Darmanjian, S. P. Kim, M. C. Nechyba, S. Morrison, J. Principe, J. Wessberg, M. A. L. Nicolelis, *Bimodel Brain-Machine Interface for Motor Control of Robotic Prosthetic*, IEEE Int. Conf. on Intelligent Robots and Systems, Las Vegas, October, 2003.

[3] S. Darmanjian, S. P. Kim, M. C. Nechyba, J. Principe, J. Wessberg, M. A. L. Nicolelis, *Bimodel Brain-Machine Interface for Motor Control of Robotic Prosthetic*, IEEE Machine Learning For Signal Processing, Ireland, September, 2006.

[4] M. Brand. Coupled hidden Markov models for modeling interacting processes. Technical Report 405, MIT Media Lab Perceptual Computing, June 1997.

[5] L. E. Baum, T. Petrie, G. Soules and N. Weiss, A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, Ann. Mathematical Statistics, vol. 41, no. 1, pp. 164-71, 1970.

[6] Sung-Phil Kim, Justin C. Sanchez, Deniz Erdogmus, Yadunandana N. Rao, Jose C. Principe, and Miguel A.L. Nicolelis, *Divide-and-conquer Approach for Brain-Machine Interfaces: Nonlinear Mixture of Competitive Linear Models*, Neural Networks, 16, pp. 865-871, 2003.

[7] Rao RP, "Bayesian computation in recurrent neural circuits. Neural Comput", Vol. 16, No. 1. (January 2004), pp. 1-38.

[8] M. A. L. Nicolelis, D.F. Dimitrov, J.M. Carmena, R.E. Crist, G Lehew, J. D. Kralik, and S.P. Wise, "Chronic, multisite, multielectrode recordings in macaque monkeys," PNAS, vol. 100, no. 19, pp. 11041 - 11046, 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, pp. 701-708, 2001.

[10] Walter J. Freeman, Mass Action in the Nervous System University of California, Berkeley, USA

[11] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, Raj Reddy ,Spoken Language Processing: A Guide to Theory, Algorithm and System Development

[12] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. Nicolelis et al., *Real-time prediction of hand trajectory by ensembles of cortical neurons in primates*, Nature, Volume 408, pp. 361-365, 2000.

[13] J.C. Sanchez, S.-P. Kim, D. Erdogmus, Y.N. Rao, J.C. Principe, J. Wessberg, M. Nicolelis, *Input-Output Mapping Performance of Linear and Nonlinear Models for Estimating Hand Trajectories from Cortical Neuronal Firing Patterns*, 2002 International Workshop for Neural Network Signal Processing, April 2002.

[14] Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E. (1991) , *Adaptive mixtures of local experts*, Neural Computation, 3, 79-87.

[15] J. C. Sanchez, J. C. Principe, and P. R. Carney, "Is Neuron Discrimination Preprocessing Necessary for Linear and Nonlinear Brain Machine Interface Models?," accepted to 11th International Conference on Human-Computer Interaction, 2005.

[16] Z. Ghahramani and M.I. Jordan, "Factorial Hidden Markov Models," Machine Learning, 29, pp. 245-275, 1997.

[17] M. Meila and M. I. Jordan. Learning ne motion by Markov mixtures of experts. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, Advances in Neural Information Processing Systems 8. MIT Press, 1996.

[18] M. I. Jordan, Z. Ghahramani, and L. K. Saul. Hidden Markov decision trees. In M.C. Mozer, M.I Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9. MIT Press, Cambridge, MA, 1997.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classication and Regression Trees.Wadsworth International Group, Belmont, CA, 1984

[20] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 3, pp. 226-239, 1998.

[21] Martinez-Ramon Mane, Koltchinskii V, Heileman Gregory L, Posse S. MRI Pattern Classification Using Neuroanatomically Constrained Boosting. NeuroImage, January 2005.

[22] P.Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In CVPR, volume 1, pages 511-518, 2001.

[23] Ran Avnimelech and Nathan Intrator. Boosted mixture of experts: An ensemble learning scheme. Neural Computation, 11(2):483-497, 1999.

[24] Robert E. Schapire, Yoav Freund, Peter Bartlett, andWee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In Proc. 14th International Conference on Machine Learning, pages 322-330. Morgan Kaufmann, 1997.

[25] Robert E. Schapire. The strength of weak learnability. Machine Learning, 5:197-227, 1990.

[26] Y. Sun and J. Li, "Iterative RELIEF for feature weighting," in Proc. 23rd International Conference on Machine Learning. ACM Press, 2006, pp. 913-920.

[27] Annette Bastian, Gregor Schner, Alexa Riehle (2003) Preshaping and continuous evolution of motor cortical representations during movement preparation European Journal of Neuroscience 18 (7), 2047-2058.