

Simultaneous Prediction of Four Kinematic Variables for a Brain-Machine Interface Using a Single Recurrent Neural Network

J. C. Sanchez¹, J. C. Principe², J. M. Carmena^{3,4}, Mikhail A. Lebedev^{3,4}, M. A. L. Nicolelis^{3,4,5}

¹Department of Biomedical Engineering, ²Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL

³Department of Neurobiology, ⁴Center for Neuroengineering, ⁵Department of Biomedical Engineering Duke University, Durham

Abstract— Implementation of brain-machine interface neural-to-motor mapping algorithms in low-power, portable digital signal processors (DSPs) requires efficient use of model resources especially when predicting signals that show interdependencies. We show here that a single recurrent neural network can simultaneously predict hand position and velocity from the same ensemble of cells using a minimalist topology. Analysis of the trained topology showed that the model learns to concurrently represent multiple kinematic parameters in a single state variable. We further assess the expressive power of the state variables for both large and small topologies.

Keywords—Brain-Machine Interface, recurrent neural network, RMLP, neuroprosthetic, state-space analysis

I. INTRODUCTION

In the design of Brain-Machine interfaces (BMIs), the goal is to generate control signals for a variety of devices using only the subject intent, which is represented in the modulation of cortical neuronal activity. One such application of a BMI could be the control of a prosthetic limb that would enable a disabled individual to reach and grasp a desired object. In this scenario, the individual must indicate the 3-D position to move to, how fast to get to the target, and the amount of force to grasp the object. Ultimately, all of these signals should be derived directly from activity in the motor, premotor, parietal, and somatosensory cortices of the individual and use it to drive the motors of the prosthetic device.

In the BMI literature [1-4], many researchers (including ourselves) have implemented linear vector Wiener filters (finite impulse response, FIR) to successfully predict a variety of hand kinematics in behaving monkeys. In this framework, separate linear models must be used to predict each kinematic variable that included hand position (HP), hand velocity (HV), and hand gripping force (GF). Training separate filters for each kinematic variable requires parallelizing update algorithms. This design comes at the cost of computational complexity and the optimization of thousands of parameters resulting from the fact that the model topology must be copied for the addition of each new kinematic variable. In experiments involving four kinematic variables and FIR filters, a total of up to 8,000 model parameters have to be trained (200 input neurons, 10 tap-delays, and 4 outputs). In order to overcome this limitation, models with states (either linear or nonlinear) must be

utilized. Moreover, we are ultimately interested in implementing neural-to-motor mapping algorithms in low-power, portable digital signal processors (DSPs) that could be coupled with electronics inside a prosthetic device. For this goal to become a reality, it is critical to efficiently use model resources especially when predicting signals that show interdependencies. In the case of using position and velocity information to control a prosthetic arm we may be able to exploit the fact that the velocity is simply the derivative of position. The question now becomes how can a representation space be created that maps neuronal activity to *both* kinematic variables. This direction of research is motivated by the need to understand neural modulation in the context of parsimonious BMI models that can simultaneously predict multiple kinematic variables with the same ensemble of cells.

While for the type of tasks that have been attempted nonlinear models have only provided marginally better performance for BMIs, we show here that a single nonlinear model topology is capable of predicting multiple kinematic parameters from the same ensemble of cells. Our model is the Recurrent Multilayer Perceptron (RMLP) which has been proven to be a powerful topology for both modeling and analysis in BMI applications [5, 6]. In this paper, we would like to describe the “engineering advantages” that can be employed by such a dynamic neural network compared to a linear topology for implementation of algorithms in low-power, portable DSP electronics. In our analysis, we would like to understand how the model effectively uses state variables to represent multiple kinematic parameters to achieve parsimonious neural-to-motor mappings.

II. ADVANTAGES OF THE RMLP MODEL

We first describe the RMLP model used in this study and briefly outline the two most important advantages it has over the linear filters (FIR). First, the RMLP modeling approach uses only the instantaneous neural activity to compute each output. Unlike the FIR filter topology, the RMLP implements a frugal memory structure by eliminating the input tap-delay lines. In Fig., 1, we show how a tap delay memory structure, which scales the number of free parameters by the number of input neurons, can be replaced by an architecture that contains feedback. The memory depth is controlled during training and is specified by the value of the feedback weights indicated by \mathbf{W}_f . This memory structure can implement up to $(N_1)^2$ different

timescales where N_1 is the number of hidden nodes in the topology; comparatively the FIR topology is limited to a single memory depth indicated by the largest delay. Next, the recurrent topology has the ability to simultaneously find the mapping between neural inputs and multiple kinematic variables (HP and HV) with minimal cost in the addition of extra degrees of freedom by adding extra processing elements (PEs) in the output layer. Since in BMIs the number of hidden PEs is much less than the number of inputs this is a great savings in parameters. In fact, the FIR filter requires a complete copy of the entire topology for each additional kinematic variable. We can express the state vector of the RMLP hidden layer in (1) as a nonlinear function of the linear combination of input and previous state plus a bias (\mathbf{b}_1). The output layer of the RMLP can have any number of PEs but in this case there are 4 linear PEs (4 x,y coordinates of HP and HV) The output of the network is given in (2). In summary, each hidden PE is a nonlinear adaptive basis for the output that projects the high dimensional neuronal data. These projections (states) are then linearly combined to form the outputs (predictions) (plus a bias \mathbf{b}_2) of the RMLP.

$$\mathbf{y}_1(t) = f(\mathbf{W}_1 \mathbf{x}(t) + \mathbf{W}_f \mathbf{y}_1(t-1) + \mathbf{b}_1) \quad (1)$$

$$\mathbf{y}_2(t) = \mathbf{W}_2 \mathbf{y}_1(t) + \mathbf{b}_2 \quad (2)$$

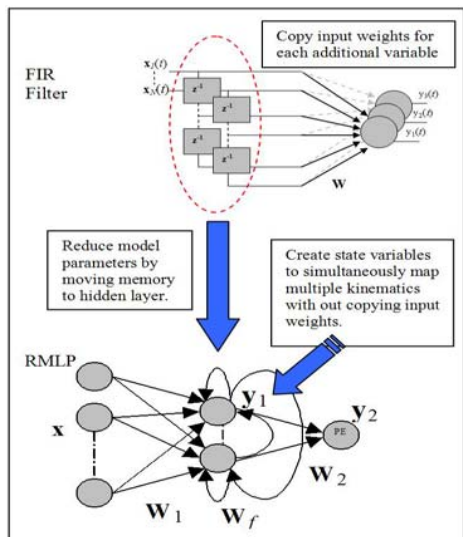


Figure 1. Fully connected, state RMLP

III. EXPERIMENTAL SETUP AND MODEL TRAINING

A. Experimental Setup

The data for the BMI experiments presented here were collected in Nicolelis primate laboratory at Duke University. Microwire electrode arrays [7] were chronically implanted in the dorsal premotor cortex (PMd), supplementary motor area (SMA), primary motor cortex (M1, both hemispheres)

and primary somatosensory cortex (S1) of an adult female monkey (*Macaca mulatta*) which yielded activity from 183 neurons. The animal was performing a cursor hitting task that involved the presentation of a randomly placed target on a computer monitor in front of the monkey. The monkey used a hand-held manipulandum (joystick) to move the computer cursor so that it intersects the target. While the monkey performed the motor task, the hand position, and velocity was recorded in real time along with the corresponding neural activity from multiple channels [4].

The recorded neuronal spike events that were used as inputs to the models were binned (added) in non-overlapping windows of 100ms and the behavioral datasets were downsampled and lowpass filtered to 10Hz. This data set was segmented into two exclusive parts: 5,000 samples for model training and 3,000 samples for model testing. The test data block is contiguous to the training data.

B. Model Training

In this study, a total of three RMLP models were trained which consisted of one model optimized simultaneously for HP and HV while the other two models were optimized independently for each HP, HV. In all models, the optimal weights are determined by minimizing MSE using backpropagation through time (BPTT) [8]. Training was stopped using the method of cross-validation (batch size of 1000 pts.) to maximize the generalization of the network [9]. Even though the RMLP is a parsimonious model, there still are a significant number of weights in the input layer. To further improve the generalization of this model we implement a regularization technique (weight decay) in the input layer by subtracting a small constant (0.00001) from each weight at each update [10]. This procedure ensures that weights that are not being significantly updated will be reduced to zero.

The number of hidden PEs used in each topology was optimized to produce the best testing performance with the smallest number of degrees of freedom. Using an iterative procedure, the number of PEs was incrementally increased from one until performance was maximized. If the addition of one extra PE did not yield an increase in performance, the model topology was fixed to the previous number of PEs.

IV. PERFORMANCE RESULTS

Model performance was quantified using the correlation coefficient CC between the testing model outputs and the known kinematic values. In Table 1, we present the size of the topology (# Hidden PEs) that was required to produce the corresponding CC values. Additionally, we present five representative testing trajectories for the model trained simultaneously for all parameters. On average, the simultaneous model performed as well as any of the individual models. Representative testing HP and HV

trajectories for the simultaneous model presented in Fig. 2 show that the predictions made from only the neuronal activity track the actual hand kinematics well. Moreover, the simultaneous model produced high performance with a minimal number of processing elements. In theory, if the training for each variable were required to be independent (i.e. each variable exists in separate subspaces), we would need a total of 9 PEs (5+4) in the hidden layer to create a representation space. In this particular example, we were able to simultaneously achieve a high-performance while saving 406 extra model parameters. This corresponds to two less function evaluations and 406 fewer multiplications *per iteration* that would be required in a DSP implementation. Comparatively a FIR filter would require 5955 *more* multiplications per iteration.

Table 1. RMLP testing performance.

Model	# Hidden PEs	CC Position (x,y)	CC Velocity (x,y)
HP	5	0.68, 0.70	
HV	4		0.74, 0.70
HP, HV	7	0.76, 0.68	0.72, 0.66

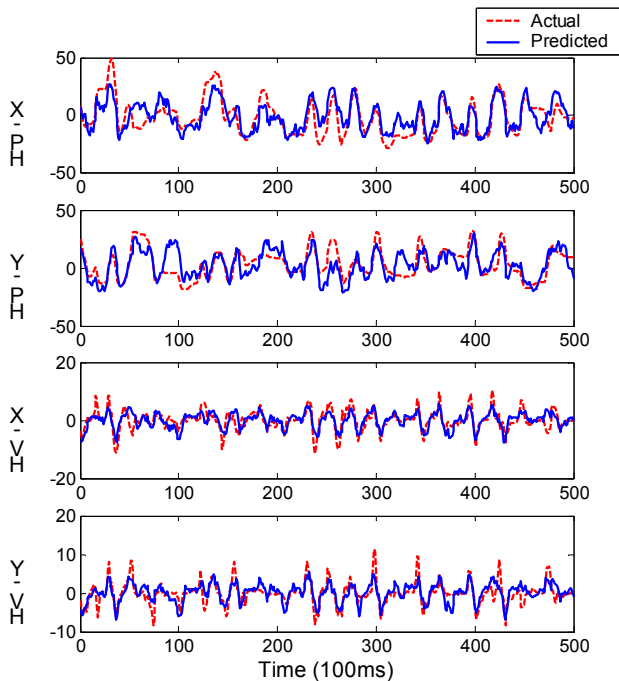


Figure 2. Testing RMLP model predictions of five variables (units: HP mm, HV mm/100ms)

V. ANALYSIS OF RMLP HIDDEN STATES

A. State Variable Representation for Small Topologies

Since the simultaneous RMLP achieved a similar level of performance to the independent networks, we will now ask the question how do the hidden states of the topology represent each of the kinematic variables. We experimentally found the correspondence of each of the hidden states (y_1 , Eq. 1) of the independent models with the states of the simultaneous model by computing the absolute value of the CC between all pairs of states. For example, in Table 2, state #4 for the HV model has a large correlation (0.77) with state #1 of the simultaneous model. From this analysis, we can infer that the simultaneously trained RMLP can concurrently represent multiple kinematic parameters in a single state variable as shown by states #4 (HP, and HV), #5 (GF and HP), and # 6 (HP and HV). The universal approximating ability of the RMLP exploited similar trends in the HP and HV time-series.

Table 2. Correspondence (in CC) between states of trained RMLPs.

		States of Simultaneous Model						
		1	2	3	4	5	6	7
States of Independent Models	1		HP 0.59					
	2				HP 0.66	HP 0.78	HP 0.56	HP 0.57
	3				HV 0.73			
	4	HV 0.77	HP 0.51				HV 0.60	
	5							

B. Tradeoff between Specificity and Generalization

In the previous section, the focus was to design parsimonious models to represent the five kinematic variables; our analysis here is motivated by the recent work of Jaeger who has proposed the use of a special type of recurrent neural network (called Echo State networks, ESN) for function approximation. The fundamental idea is to use a large number of hidden states with fixed connections to produce diversified representations of an input signal, which can then be instantaneously combined in an optimal manner to approximate a desired response [11]. In a sense the RMLP has the same basic architecture as an ESN, except that the number of hidden PEs is much smaller and the weights in the first layer are trained.

We are most interested in the diversity of kinematic signals that can be reconstructed by the adaptive basis functions that make the states of the RMLP network, which is directly related to the number of hidden PEs. Therefore, to explore the representation ability of the RMLP network, we will first train all three weight matrices (\mathbf{W}_1 , \mathbf{W}_f , \mathbf{W}_2) with a variety of hidden PEs (5, 10, 20, 30), using HP as the desired signal. We will then freeze the \mathbf{W}_1 , \mathbf{W}_f matrices and adapt only the \mathbf{W}_2 matrix for another kinematic variable (HV). In theory if the hidden states contain enough diversity, then any desired signal existing in the basis

subspace can be approximated (in this case, HV). In Table 3, we show how the performance of an RMLP with only adaptation of its output layer weight performs for HV. As the number of hidden PEs increases, we can create better predictions of HV as indicated by the increase in CC. This example illustrates that by specifying a small RMLP topology training produces a specific projection of the input; that is a subspace for HP. However if we are more liberal with the size of the topology we can achieve projections that are general to HP and HV.

Table 3. Performance vs. Hidden State Diversity

	# of PEs			
	5	10	20	30
Test HV CC (x,y)	0.18, 0.25	0.24, 0.37	0.23, 0.36	0.45, 0.42
	Specificity ← → Generalization			

VII. DISCUSSION

Real applications of BMIs will require models that generalize to a variety of kinematic variables and can be implemented in portable, low-power DSPs. We seek models that are computationally inexpensive meaning that the number of mathematical operations and the amount of memory required for each predicted output is small. We have shown here that the RMLP topology can efficiently and accurately produce simultaneous predictions of HP and HV from the same ensemble of cortical neurons. This performance was made possible by using a dynamical model that concurrently represents multiple kinematic variables on a small set of state variables. The expressive power of the RMLP comes at a cost though. First, the BPTT training algorithm is more difficult to implement than other linear topologies. Second, for a particular set of data training small topologies produce specific models meaning if we desire to include an additional kinematic variable then the entire topology must be retrained. As an alternative, we could train larger RMLPs that have hundreds of hidden states that could potentially encode many kinematic variables. However, the increase in dimensionality does not contribute to our goal of DSP implementation.

The bottom line is that the state-of-the-art in BMI design is not yet prepared to deal with extremes in generalization and model architecture. It is through the techniques of modeling and analysis as presented here that we will learn the requirements (training, topology, representation) for good engineering design of powerful BMI models suitable for use in clinical applications.

ACKNOWLEDGMENT

This work was supported by DARPA sponsored grant # ONR-450595112 and by the Christopher Reeve Paralysis Foundation (contract number CA2-0308-2) to JMC.

REFERENCES

1. Wessberg, J., et al., *Real-time prediction of hand trajectory by ensembles of cortical neurons in primates*. Nature, 2000. **408**(6810): p. 361-365.
2. Serruya, M.D., et al., *Brain-machine interface: Instant neural control of a movement signal*. Nature, 2002. **416**: p. 141-142.
3. Sanchez, J.C., et al. *Input-output mapping performance of linear and nonlinear models for estimating hand trajectories from cortical neuronal firing patterns*. in *International Work on Neural Networks for Signal Processing*. 2002. Martigny, Switzerland: IEEE.
4. Carmena, J.M., et al., *Learning to control a brain-machine interface for reaching and grasping by primates*. PLoS Biology, 2003. **1**: p. 1-16.
5. Sanchez, J.C., et al. *A comparison between nonlinear mappings and linear state estimation to model the relation from motor cortical neuronal firing to hand movements*. in *SAB Workshop on Motor Control in Humans and Robots: on the Interplay of Real Brains and Artificial Devices*. 2002. University of Edinburgh, Scotland.
6. Sanchez, J.C., et al., *A model based approach to quantify motor cortical neuronal interactions in a brain machine interface*. Neural Computation, 2004. **Submitted**.
7. Nicolelis, M.A.L., et al., *Chronic, multi-site, multi-electrode recordings in macaque monkeys*. Proc. Natl. Acad. Sci. U.S.A., 2003. **100**(19): p. 11041-11046.
8. Haykin, S., *Neural networks: a comprehensive foundation*. 1994, New York: Toronto: Macmillan; Maxwell Macmillan Canada.
9. Vapnik, V., *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. 1999, New York: Springer-Verlag. 304.
10. Principe, J.C., N.R. Euliano, and W.C. Lefebvre, *Neural and adaptive systems: fundamentals through simulations*. 2000, New York: Wiley.
11. Jaeger, H., *The "Echo State" Approach to Analyzing and Training Recurrent Neural Networks*, in *GMD Report 148*. 2001, GMD-German National Research Institute for Computer Science.