Learning Mappings in Brain Machine Interfaces with Echo State Networks

Yadunandana N. Rao, Sung-Phil Kim, Justin C. Sanchez[1], Deniz Erdogmus, Jose C. Principe,
Jose M. Carmena[2,3], Mikhail A. Lebedev[2,3], Miguel A.L. Nicolelis[2,3,4]
Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering
Department of Biomedical Engineering[1]
University of Florida, Gainesville, FL-32611
Department of Neurobiology[2], Center for Neuroengineering[3], Department of Biomedical Engineering[4]
Duke University, Durham, NC-27710

Abstract- Brain Machine Interfaces (BMI) have been designed using linear or non-linear models that are trained with cortical neuronal firing activity to learn the transfer function to associated behavior, most often the hand position of a primate. BMIs with linear models have shown good performance in modeling the transfer function, however they are not free of shortcomings; one of the important ones is the massive disparity between the dimensionality of the input (number of channels times the number of delays) and output (2 or 3-D hand coordinates) leading to poor generalization. On the other hand, non-linear models like the Recurrent Multi-Layer Perceptron (RMLP) can potentially provide parsimonious mapping functions that generalize better because the input dimensionality is just the number of channels. However, the caveat is that the simplicity of the training mechanism is lost, which can be critical for practical use of a BMI. This paper bridges the gap between superior performance per trained weight and model learning complexity. Towards this end, we propose to use Echo State Networks (ESN) to transform the neuronal firing activity into a higher dimensional space and then derive an optimal sparse linear mapping in the transformed space to match the hand position. The sparse mapping is obtained using a weight constrained cost function whose optimal solution is determined using a stochastic gradient algorithm. We will show correlation coefficient performance comparable to the Wiener filter with tremendous savings in the training complexity, thereby paving the way for easier hardware implementation.

I. INTRODUCTION

Design of optimal Brain Machine Interfaces is a challenging problem that has attracted the attention of several research groups throughout the world. In their widely acclaimed article, Wessberg *et al* showed that, it is possible to predict the hand position of a primate by using linear and non-linear models on the cortical neuronal firing activity [1]. Since then, over the past few years, BMI research groups have adopted diverse modeling frameworks [1-8] with the ultimate goal of translating brain activity into a prediction of animal behavior. The models are usually derived using classical Mean-Squared Error (MSE) criterion, where the error is measured as the Euclidean distance between the model outputs and behavior output (hand position). The inputs to these models are usually multidimensional neural recordings collected from selected regions of a monkey's brain. In the linear modeling category, researchers have used Wiener filters [9] that are estimated using regression methods [1,6,8]. Linear models have been quite popular owing to their simplicity and the availability of simple and robust learning algorithms. However, the limitation of linear mappings may hinder performance, especially in cases where the transfer function between brain activity and hand position becomes nonlinear. Furthermore, these models have huge number of parameters (in the thousands) and the cost function is unable to properly select inputs, thereby degrading their generalization ability. An obvious solution would be to use non-linear models (neural networks) for BMIs. Typical non-linear models that have been used are non-linear Kalman filters [3], Time-Delay Neural Networks (TDNN), Recurrent Multi-Layer Perceptrons (RMLP) [6], particle filters [3] and nonlinear mixture models [7]. Remarkably, these models show only a slight improvement in performance for the experimental paradigms tested, but they can be designed with more parsimonious architectures (fewer weights) [10]. However, the improvement in performance is gained at the expense of a significant leap in the computational costs of training these models. The problem is accentuated by the fact that these models may have to be re-trained occasionally to adjust to the changing statistics and environment. This can severely restrict practical, real-world application of BMIs.

In this paper, we propose a different architecture that has performance similar to that of an RMLP, but the training has linear complexity as opposed to the back-propagation through time (BPTT) RMLP training algorithm [11]. With respect to the Wiener filter it has fewer weights, however this new architecture opens the way to easily extract in parallel different output variables (e.g. hand position, velocity, griping force) from the same system state, creating a more flexible BMI system. At the core of the architecture is an Echo State Network (ESN) [12], which will be described briefly in the next section. The outputs of the ESN are then linearly combined by a *sparse matrix*. This will be discussed in section III of the paper followed by the experimental

section in section IV, where we will design a BMI using this architecture. We will briefly show comparisons with other BMIs reported in literature.

II. Echo State Networks: A Brief Overview

Echo State Networks (ESN) are recurrent networks with fixed weights but nonconvergent dynamics. First proposed by Jaeger [12], ESNs are appealing because of their computational abilities and simplified learning mechanisms. ESNs exhibit some similarities to the "Liquid State Machines (LSM)" proposed by Maas *et al* [13], which possess universal approximation capabilities in myopic functional spaces. In this section, we will briefly summarize the basic principles of an ESN. Detailed material on ESNs can be found in [12]. The fundamental idea of an ESN is to use a "large reservoir" recurrent neural network (RNN) that can produce diversified representations of an input signal, which can then be instantaneously combined in an optimal manner to approximate a desired response. Fig. 1 shows the block diagram of an ESN. A set of input nodes denoted by the vector $\mathbf{u}_n \in \Re^{M\times 1}$ is connected to a "reservoir" of N discrete-time recurrent networks by a connection matrix $\mathbf{W}_{in} \in \Re^{M\times N}$. At any time instant n, the readout (state output) from the RNN reservoir is a column vector denoted by $\mathbf{x}_n \in \Re^{N\times 1}$. Additionally, an ESN can have feedback connections from the output to the RNN reservoir. In fig.1, we show two outputs (representing the X-Y Cartesian coordinates of the primate's hand) and the associated feedback connection matrix, $\mathbf{W}_b \in \Re^{N\times 2} = [\mathbf{w}_{b1}, \mathbf{w}_{b2}]$. The desired outputs form a 2-D column vector $\mathbf{d}_n = [d_{xn}; d_{yn}]$. The reservoir states are transformed by a static linear mapper that can additionally receive contributions from the input $\mathbf{u}_n$. Each processing element (PE) in the reservoir can be implemented as a leaky integrator (first order gamma memory [14]) and the state output or the readout is given by the difference equation in (1).

$$\mathbf{x}_{n+1} = (1-\mu Ca)\mathbf{x}_n + \mu C(f(\mathbf{W}_{in}^T\mathbf{u}_{n+1} + \mathbf{W}\mathbf{x}_n + \mathbf{W}_b\mathbf{d}_n)) \qquad (1)$$

where, $0 < \mu < 1$ is the step-size used for converting a continuous-time leaky integrator into a discrete-time difference equation, C is the time constant and a is the decay rate [12]. The point-wise non-linear function $f(.)$ is chosen to be the standard *tanh* sigmoid, i.e.,f(.) = *tanh*(.). Note that, if μ, C, and a are all equal to unity, the RNNs default to the conventional non-linear PE [14] without memory. From a signal processing point of view, the reservoir creates a set of bases functions to represent the input, while the static mapper finds the optimal projection in this space. There are obvious similarities of this architecture to kernel machines, except that the kernels here are time functions (Hilbert spaces).

We will now give the conditions under which an ESN can be "useful," which Jaeger aptly calls as the "Echo State Property." Loosely stated, the Echo State Property says that the current state is uniquely defined by the past values of the inputs and also the desired outputs if there is feedback. A weaker condition for the existence of echo states is to have the spectral radius of the matrix $\hat{\mathbf{W}} = \mu C\mathbf{W} + (1-\mu Ca)\mathbf{I}$ less than unity [12]. Another aspect critical for the success of ESN is to construct a sparse matrix $\mathbf{W}$. This will ensure that the individual state outputs have different representations of the inputs and desired outputs or in other words, the span of the representation space is sufficiently rich to construct the mapping to the desired response.

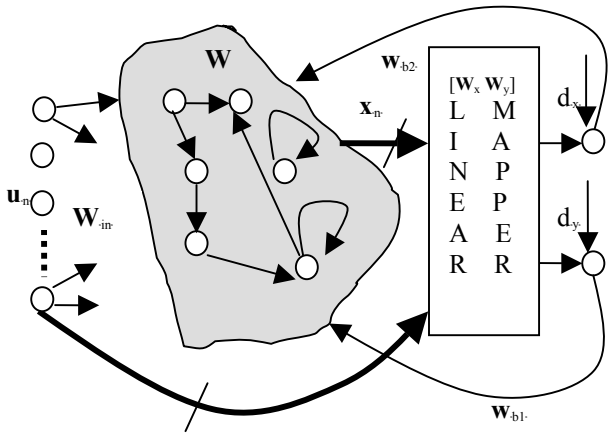

Figure 1. Block diagram of an Echo State Network

We will now shift our focus to the last block of the ESN, i.e., the linear mapper and defer the discussion on the aspects related to training and testing with ESNs to the experimental section IV.

In most engineering applications of ESN, e.g. system identification and prediction [15], the optimal linear mapper (see fig.1) is obtained using standard recursive algorithms for MSE minimization. However, when the number of RNNs (N) in the reservoir is very high, the dimensionality of the input to the linear mapper increases. The problem will only be worsened if we allow connections between the input nodes and the linear mapper. Additionally, if the desired signals are in a lower dimensional space as in our case, increasing N will easily result in over-fitting. This calls for explicit regularization in the estimation of the optimal linear mapper. It is known that L_1 norms are preferable to L_2 norms for this goal [11]. Adaptive weight decay using cross-validation is also widely used for zeroing the weights that make little or no contribution to the cost. However, having a separate cross-validation set may not be feasible for real world applications that require repeated training. In the next section, we will present an *on-line scheme* that introduces constraints on the MSE criterion to result in an optimal and sparse linear mapper.

III. Sparse-lms Algorithm

Let the linear mapper (assume single output for now) be

denoted by the weight vector $\mathbf{w} = \{w_i\}_{i=1\ldots L}$. Consider the cost function in equation (2).

$$J(\mathbf{w}) = E(e_k^2) + \lambda\left[\sum_{i=1}^{L}|w_i|^p - \alpha\right] \quad (2)$$

The first term is the regular Mean-Squared Error (MSE) and the second term is the constraint. The error e_k is the difference between the desired signal d_k and the output $y_k = \mathbf{w}^T\mathbf{x}_k$. The constraint fixes the sum of the p^{th} powers of the absolute values of the individual elements w_i of the weight vector $\mathbf{w}$ to a constant denoted by α. Note that, with $p=1$, we will be constraining the L_1 norm of the weight vector $\mathbf{w}$. This is by far the most widely used explicit sparseness measure. The penalty factor is denoted by λ. Instead of fixing the penalty term, we can include λ as an *adaptive parameter* in the cost function. In order to estimate λ, we modify the cost function in (2) as,

$$J(\mathbf{w},\lambda) = E(e_k^2) + \beta\lambda\left[\sum_{i=1}^{L}|w_i|^p - \alpha\right] - \beta\lambda^2 \quad (3)$$

where, β is a positive stabilization constant that keeps the penalty factor λ bounded. The cost function in (3) is often known as the *augmented Lagrangian* in optimization literature [16]. The stationary points are obtained by using standard gradient methods. The stochastic gradients of the cost are,

$$\begin{aligned}\frac{\partial J(\mathbf{w},\lambda)}{\partial w_i} &\approx -2e_k x_{ki} + \lambda\beta[pw_i^{p-1}sign(w_i^p)] \\ \frac{\partial J(\mathbf{w},\lambda)}{\partial \lambda} &= \beta\left(\sum_{i=1}^{L}|w_i|^p - \alpha\right) - 2\beta\lambda\end{aligned} \quad (4)$$

We will simultaneously minimize and maximize (3) with respect to $\mathbf{w}$ and λ respectively. The stochastic gradient algorithm for online adaptation of the free parameters is then given by the pair of equations,

$$w_i(k+1) = w_i(k) + \eta_w[2e_k x_{ki} - \lambda_k\beta pw_i^{p-1}sign(w_i^p)] \quad (5)$$

$$\lambda_{k+1} = \lambda_k + \eta_\lambda\beta\left(\sum_{i=1}^{L}|w_{ki}|^p - \alpha - 2\lambda_k\right) \quad (6)$$

where, η_w and η_λ are small positive step-sizes, w_{ki} denotes the i^{th} element in the vector $\mathbf{w}_k$ and x_{ki} denotes the i^{th} element in the vector $\mathbf{x}_k$. Allowing, $\lambda_{k+1} = \lambda_k$ as $k \to \infty$, it is easy to see that the sequence $\{\lambda_k\}$ converges to a value λ^* given by,

$$\lambda^* = \frac{1}{2}\left(\sum_{i=1}^{L}|w_i^*|^p - \alpha\right) \quad (7)$$

where, $\mathbf{w}^*$ is the asymptotic weight vector obtained from (5).

A. *Convergence of* λ: For stable asymptotic convergence of the penalty sequence to zero, i.e., $\lim_{k\to\infty}\lambda_k = 0$, a necessary condition is $0 < 2\beta\eta_\lambda << 1$.

Proof: Equation (6) can be rewritten as, $\lambda_{k+1} = \lambda_k(1-2\eta_\lambda\beta) + \eta_\lambda\beta\left(\sum_{i=1}^{L}|w_{ki}|^p - \alpha_k\right)$. If $0 < 2\beta\eta_\lambda << 1$, the equation becomes $\lambda_{k+1} = \lambda_k + \eta_\lambda\beta c_k$, where c_k is the expression for the constraint. By letting $\lambda_{k+1} = \lambda_k$, as $k\to\infty$, it is obvious that $\lim_{k\to\infty} c_k = 0$ and consequently, the Lagrangian λ^* converges to zero.

Proofs of convergence of $\mathbf{w}$ are fairly involved and are beyond the scope of this paper. It will suffice to say that the step-size parameter η_w plays a crucial role in the tradeoff between the speed of convergence and misadjustement in the final estimate. A more robust update for $\mathbf{w}$ can be obtained by including a normalization term (similar to the NLMS algorithm) as shown in (8). Note that the normalization is just done on the gradient of the MSE part of the cost function.

$$w_i(k+1) = w_i(k) + \eta_w\left[\frac{2e_k x_{ki}}{\sigma + \mathbf{x}_k^T\mathbf{x}_k} - \lambda_k\beta pw_i^{p-1}sign(w_i^p)\right] \quad (8)$$

The constant σ is a small positive number used to avoid possible divisions by zero.

B. *Selection of β, α and p*

β is a positive stabilization constant that affects the convergence of the Lagrangian λ. It also acts as a balancing factor, weighing the constraint term against the MSE. A very high β will prioritize the constraint part of the cost function and the resulting residual MSE will be fairly high. On the other hand, a smaller β will tend to emphasize the MSE part and the constraint will not be strictly satisfied in the final estimate of $\mathbf{w}$. In the experiments, we usually choose β in the interval [1-5].

The remaining constants α and p are tied with the definition of sparseness. Typically, both p and α are chosen to be unity. If α is made zero, there is a good chance of the weights converging to an all zero vector. To prevent this, β should be scaled down appropriately. Also, the selection of α should be based on the dimensionality of the input. In our experiments, we choose α in the range [0.5,1], for input data dimensions less than 200 and up to 1.5 if the dimensionality exceeds 200.

IV. Brain Machine Interface Design

In this section, we will combine ESN with the sparse-LMS algorithm (eqs. 6 and 8) to decode the neural activity of a primate onto 2-D hand positions.

A. *Experimental Setup*

The data for these experiments were collected in the Nicolelis primate laboratory at Duke University. Microwire electrode arrays [17] 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*) while performing a hand-reaching task. The task 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 was recorded in real time along with the corresponding neural activity from multiple channels [8].

In the modeling analysis presented here, 185 neurons were monitored; the neuronal spike events were then 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: 5,400 samples for model training and 3,000 samples for model testing. The test data block is contiguous to the training data.

B. *Design of the Echo State Network*

One of the important parameters in the design of the ESN is the number of RNN units in the reservoir. We performed several experiments and finally decided to choose N=800. Increasing N further did not result in any significant improvement in performance. The input weight matrix $\mathbf{W}_{in}$ (185x800) was fully connected with all the weights fixed to unity. The recurrent connection matrix $\mathbf{W}$ was sparse with only 1% of the weights (randomly chosen) being non-zero. Moreover, we fix all the non-zero weights to a value 0.5. Further, each RNN is a gamma delay operator (see eqn. 1) with parameters $\{a, C, \mu\} = \{1,0.7,1\}$. The next aspect is to set the spectral radius, which is crucial for this problem as it controls the dynamics and memory of the echo-states. Higher values are required for slow output dynamics and vice-versa [12]. We observed that the trajectories in the X direction were faster than those in Y by a simple zero-crossing test. This would motivate the use of two different ESNs specifically chosen to satisfy the individual dynamics. The fact that we have two models learning X and Y directions independent of each other makes us believe that, it would be a feasible approach. The only drawback however, is the requirement of additional memory and hardware to accommodate two ESNs. For the experiments in this paper, we utilized a single ESN whose spectral radius was tuned to 0.79. Marginal changes (<1%) in performance (both X & Y) were observed when this parameter was altered by $\pm 10\%$. We also turned off the connections from the output to the RNN reservoir and the direct connections between the inputs and the linear mapper.

The network state is set to zero initially. The training inputs were forced through the network and the states were updated using (1). The first 400 echo state outputs were discarded as transients. The remaining 5000 state outputs were used to train the linear mapper.

C. *Design of the Linear Mapper Using Sparse-LMS*

The outputs from the RNN reservoir and the corresponding hand positions were used as training inputs and desired outputs respectively. The linear mapper is a matrix comprising of two column vectors $\mathbf{w}_x$, $\mathbf{w}_y$, each of length 800. The constraint parameters p, α were set to 1 and 1.5 respectively. The step-sizes for the weight (8) and the Lagrangian updates (6) were both chosen to be $1e$-3 and β was set to unity. The training was done for 20 epochs (an epoch is one presentation of the whole training) after which we did not observe any significant reduction in the MSE.

Once the training was completed, the weights were fixed and the BMI was tested on novel data. The test data is once again driven through the echo state network and the state outputs are fed to the linear mapper.

D. *Results and Discussions*

Fig.2 shows the evolution of the Lagrangian term λ_k. Clearly, they converge to zero, which implies that the final weight vectors satisfy the imposed L_1 norm constraints. Fig.3 shows the learning curve or the plot of the MSE versus the number of training epochs. The residual MSE for the X direction is slightly more than that of the Y direction. A histogram of the weight vectors is plotted in fig. 4. Observe that most of the weights are concentrated around zero, which demonstrates the sparseness of the weight vectors. Fig.5 shows the performance of the models on test data. Only the first 500 samples out of 3000 are shown here for clarity. The outputs of the linear mapper were then lowpass filtered by a 4^{th} order Butterworth filter with normalized cutoff frequency of 0.2. The low pass filtering smoothens the output. This is a requirement for the BMI in [8] as it interfaces to a robot arm. The two subplots show the outputs in the X and Y direction respectively. We see that, in this particular epoch, the performance along Y is better and is evident from the overall correlation coefficients of {0.64, 0.78} for X and Y directions respectively. In order to get a closer view of the matching between model outputs and the desired signals, we calculated the short-term correlation coefficients over non-overlapping windows of 100 samples. These are shown in fig. 6 for both X and Y directions. It is clear that the short-term correlation coefficients are time varying for X and the model performs better only in short patches. However, the same behavior has been reported for other linear and non-linear models applied to this data [8]. Further research is required to uncover the reasons for this behavior, which might lie within the data.

In order to compare the performance of the proposed BMI modeling scheme, we trained a linear filter on the delayed versions of the neuronal data. Ten tap delays were used for each channel (totally 185x10) resulting in an overall input dimensionality of 1850. The same data sets were used for both training and testing. The correlation coefficients for the X and Y directions with this model were {0.64, 0.75}. The RMLP has fewer free parameters (185 inputs-5 hidden nodes-2 output), but the BPTT training algorithm is involved. The simple linear model has 3700 trainable parameters, the proposed approach has roughly 800 (of the total 1600 parameters, more than 50% were very close to zero) and the

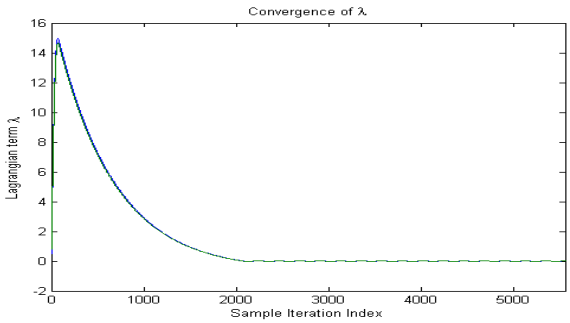

Figure 2. Convergence of the Lagrangian terms λ

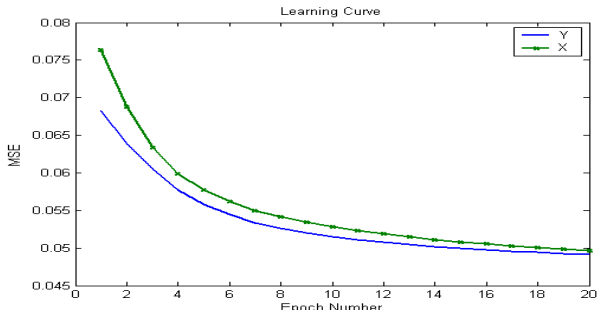

Figure 3. Learning curve

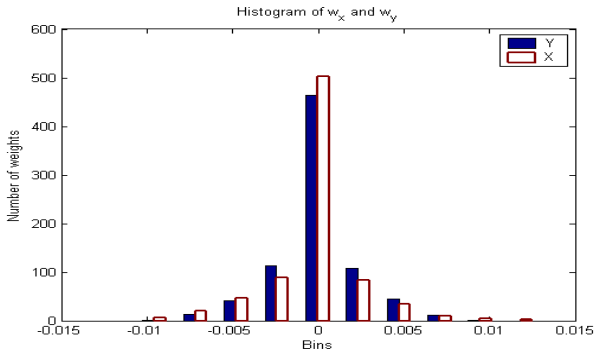

Figure 4. Histogram of the weights

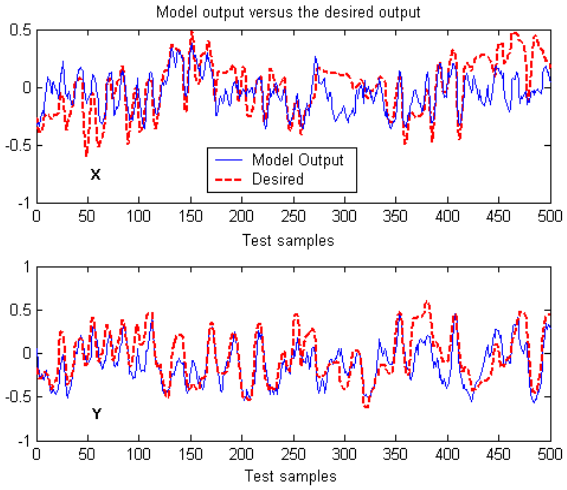

Figure 5. Model output and the desired output

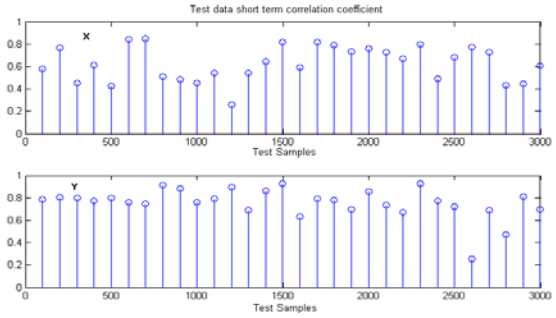

Figure 6. Windowed correlation coefficients

RMLP has 960. By using standard weight decay, it is possible to reduce the number of parameters in the linear and RMLP models, but choosing the decay parameter is not trivial (requires a cross validation set for optimality), and the norm used is an L_2 norm instead of an L_1 norm, penalizing also large weights. The proposed sparse LMS algorithm will do the job of zeroing out the undesired parameters automatically with an L_1 norm.

V. Conclusions

This paper presents the use of Echo State Networks combined with static optimal and sparse linear mappers as a decoding algorithm for a BMI. ESNs represent a different type of architecture when compared with the previous linear or nonlinear designs. The big appeal of ESNs is a representational recurrent infrastructure that brings the information from the past of the input into the present sample *without* adapting parameters. The short-term memory of the ESNs is designed *apriori* and the neuronal data is processed through a reservoir of recurrent networks that are linked by a random, sparse matrix. Therefore, only an adaptive linear or nonlinear regressor (static mapper) is needed to implement functional mappings. As demonstrated here, they perform at the same level as the Wiener filter. However, the training can be done on-line in O(N) time, unlike the Wiener or the RMLP that have O(N^2) training algorithms. The number of trainable weights is also lower than each of these networks, and more importantly, ESNs will scale up much better. In fact, we believe that the recurrent reservoir also contains information for velocity and force mappings, which means that different regressors can capture these mappings in parallel. All these features (performance, scalability, data requirements, and algorithmic complexity) will become very important for a practical BMI implementation. We carefully designed the read out portion of the ESN to guarantee maximal generalization. We implemented an on-line procedure that adaptively creates a sparse interconnection matrix based on a L_1 norm penalty. This is novel, because weight decay uses an L_2 norm and moreover, one has to optimally set the forgetting factor on a cross validation data set. Furthermore, the savings in the training time and computational complexity with the proposed approach are tremendous. ESNs hold the promise of better performance and also of better implementation

characteristics for real-time BMIs. However, the claim has to be quantified on different data sets and the overall generalization ability has to be observed. Additionally, the selection of optimal ESN parameters is critical and there are no systematic procedures currently available. Future work will be focused on these issues and also on exploring other statistical mechanisms of optimally combining the echo state outputs.

Lastly, we would like to mention the appeal of the ESN as a model for biologically plausible computation. If one thinks of the echo states as neuronal states, we can see how a distributed, recurrent topology is capable of representing information about past inputs into a diffuse set of states (neurons). For the most part, the interconnectivity and the value of the weights (synapses) are immaterial for representation. They become however critical for readout (approximation). In this respect, there are very close ties between ESN and liquid state machines, as already mentioned by Maas. These ideas may become useful in developing new distributed paradigms for plasticity and characterization of neuronal response in motor cortex.

Acknowledgements: This work was supported by DARPA sponsored grant # ONR-450595112.

REFERENCES

1. J. Wessberg, C.R. Stambaugh, J.D. Kralik, P.D. Beck, M. Laubach, J.K. Chapin, J. Kim, S.J. Biggs, M.A. Srinivasan, M.A.L. Nicolelis, "Real-time Prediction of Hand Trajectory by Ensembles of Cortical Neurons in Primates," *Nature*, vol. 408, pp. 361-365, 2000.
2. J.K. Chapin, K.A. Moxon, R.S. Markowitz, M.A.L. Nicolelis, "Real-time Control of a Robot Arm Using Simultaneously Recorded Neurons in the Motor Cortex," *Nature Neuroscience*, vol. 2, pp. 664-670, 1999.
3. Y. Gao, M.J. Black, E. Bienenstock, W. Wu, J.P. Donoghue, "A Quantitative Comparison of Linear and Non-linear Models of Motor Cortical Activity for the Encoding and Decoding of Arm Motions," *IEEE EMBS CNE*, Capri, Italy, 2003.
4. A.B. Schwartz, D.M. Taylor, S.I.H. Tillery, "Extraction Algorithms for Cortical Control of Arm Prosthetics," *Current Opinion in Neurobiology*, vol. 11, pp. 701-708, 2001.
5. M.D. Serruya, N.G. Hatsopoulos, L. Paninski, M.R. Fellows, J.P. Donoghue, "Brain-Machine Interface: Instant Neural Control of a Movement Signal," *Nature*, vol. 416, pp. 141-142, 2002.
6. J.C. Sanchez, S.P. Kim, D. Erdogmus, Y.N. Rao, J.C. Principe, J. Wessberg, M.A.L. Nicolelis, "Input-Output Mapping Performance of Linear and Nonlinear Models for Estimating Hand Trajectories from Cortical Neuronal Firing Patterns," *Proc. Neural Networks for Signal Processing*, pp. 139-148, Martigny, Switzerland, 2002.
7. S.P. Kim, J.C. Sanchez, D. Erdogmus, Y.N. Rao, J.C. Principe, M.A.L. Nicolelis, "Divide-and-Conquer Approach for Brain Machine Interfaces: Nonlinear Mixture of Competitive Linear Models," *Neural Networks*, vol. 16, no. 5-6, pp. 865-871, Jun 2003.
8. J.M. Carmena, M.A. Lebedev, R.E. Crist, J.E. O'Doherty, D.M. Santucci, D.F. Dimitrov, P.G. Patil, C.S. Henriquez, M.A.L. Nicolelis, "Learning to Control a Brain-Machine Interface for Reaching and Grasping by Primates," PLoS Biology, vol. 1, pp. 193-208, 2003.
9. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, New Jersey, 1996.
10. J.C. Sanchez, D. Erdogmus, J.C. Principe, J. Wessberg, M.A.L. Nicolelis, "A Comparison Between Nonlinear Mappings and Linear State Estimation to Model the Relation from Motor Cortical Neuronal Firing to Hand Movements," SAB'02-Workshop on Motor Control in Humans and Robots, pp. 59-65, University of Edinburgh, Scotland, Aug 2002.
11. Haykin, S. *Neural Networks, A Comprehensive Foundation*, Prentice Hall, New Jersey, 1999.
12. H. Jaeger, "The "Echo State" Approach to Analyzing and Training Recurrent Neural Networks," GMD Report 148, *GMD-German National Research Institute for Computer Science*, 2001.
13. W. Maas, T. Natschläger, H. Markram, "Real-time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, 14(11):2531-2560, 2002.
14. J.C. Principe, N. Euliano, C. Lefebrve, *Neural systems: Fundamentals through Simulations*, Wiley, 1999.
15. H. Jaeger, "Adaptive Nonlinear System Identification with Echo State Networks," Proceedings of *Neural Information Processing Systems* (NIPS) 2002, http://www.nips.cc/NIPS2002/nips-papers.html.
16. D.G. Luenberger, *Linear and Non-linear Programming, Addison Wesley*, 1989.
17. M.A.L. Nicolelis, D. Dimitrov, J.M. Carmena, R. Crist, G. Lehew, J.D. Kralik, S.P. Wise, "Chronic, Multi-Site, Multi-Electrode Recordings in Macaque Monkeys," *Proc. National Academy of Sciences of the U.S.A.*, vol. 100, no. 19, pp. 11041-11046, Sep 2003.