INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF NEURAL ENGINEERING
J. Neural Eng. **3** (2006) 145–161 doi:10.1088/1741-2560/3/2/009

A comparison of optimal MIMO linear and nonlinear models for brain–machine interfaces

S-P Kim[1], J C Sanchez[2], Y N Rao[3], D Erdogmus[4], J M Carmena[5], M A Lebedev[5], M A L Nicolelis[5] and J C Principe[1]

[1] Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA
[2] Department of Pediatrics, Division of Neurology, University of Florida, Gainesville, FL 32611, USA
[3] Motorola Inc. Plantation, FL 33322, USA
[4] Departments of Computer Science and Biomedical Engineering, Oregon Health & Science University, Beaverton, OR 97006, USA
[5] Department of Neurobiology and the Center for Neuroengineering, Duke University, Durham, NC 27710, USA

E-mail: phil@cnel.ufl.edu

Received 30 August 2005
Accepted for publication 20 March 2006
Published 17 May 2006
Online at stacks.iop.org/JNE/3/145

Abstract
The field of brain–machine interfaces requires the estimation of a mapping from spike trains collected in motor cortex areas to the hand kinematics of the behaving animal. This paper presents a systematic investigation of several linear (Wiener filter, LMS adaptive filters, gamma filter, subspace Wiener filters) and nonlinear models (time-delay neural network and local linear switching models) applied to datasets from two experiments in monkeys performing motor tasks (reaching for food and target hitting). Ensembles of 100–200 cortical neurons were simultaneously recorded in these experiments, and even larger neuronal samples are anticipated in the future. Due to the large size of the models (thousands of parameters), the major issue studied was the generalization performance. Every parameter of the models (not only the weights) was selected optimally using signal processing and machine learning techniques. The models were also compared statistically with respect to the Wiener filter as the baseline. Each of the optimization procedures produced improvements over that baseline for either one of the two datasets or both.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

The field of brain–machine interfaces (BMIs) has recently reached prominence because of the success in producing real-time estimations of movement parameters from neuronal activity recorded in multiple cortical areas [1–5]. This accomplishment has led to remarkable advances in decoding for a wide range of BMIs. For example, Taylor *et al* have measured how much information was conveyed throughout the center-out target reaching movements [6]. In the study by Carmena *et al*, multiple motor parameters could be estimated simultaneously; hand position, velocity and gripping force were estimated, yielding a neural prosthetic control of reaching and grasping using a robotic arm [7]. Wu *et al* have applied Bayesian generative models to infer motor parameters from neuronal activity during the 2D target acquisition tasks, where the temporal prior of motor parameters could be estimated through the use of these models [8]. Gage *et al* have introduced naïve co-adaptive strategies for both the subjects and the decoding filters (the Kalman filters in their works) for the 1D audio cursor control tasks without prior motor training [9]. Musallam *et al* have demonstrated decoding of the goals and expected value signals used to position computer cursors through neuronal activity recorded from the parietal

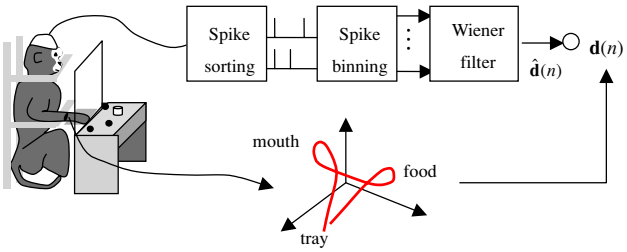

Figure 1. A system identification block diagram of BMIs. The Wiener filter approximates the neural system from spike counts' inputs and the desired 3D hand position, velocity and grasping force.

reach region [10]. It is noteworthy, however, that reasonably accurate estimations of movement parameters were obtained with relatively simple Wiener filters [3, 4, 7].

If one considers the complexity of the motor system, starting from the intricate firing modulation of millions of cells in the cortex, passing through the added complexity of the spinal cord functionality up to the firing of motor neurons that control muscles which also have nontrivial contractile properties and are richly innervated with sensory endings, it is rather surprising that a simple linear projection in the input space is able to capture the parameters of arm movements with correlation coefficients around 0.8 between the desired and actual trajectories. The major factor contributing to the success of linear models in these estimations is the fact that the projection is done on a set of bases that are continuously following the input signal (the past samples of the input), and the linear manifold created in high-dimensional spaces very likely will provide a good projection for the low-dimensional output trajectory. However, the huge number of coefficients to be trained is an issue that must be properly addressed. This paper looks from an optimal signal processing framework at the challenges and opportunities of linear models for BMIs and evaluates the steps that are still needed to go beyond the present level of performance.

Let us describe briefly the goals of BMIs to articulate better the structure of this paper. In Nicolelis' primate laboratory at Duke University, hundreds of electrodes are chronically implanted in frontal and parietal cortical areas of owl [3] and macaque [7, 11] monkeys. At the same time, the 3D position and velocity of the animal's hand and, in addition, hand gripping force are measured and sampled at 10 Hz or higher. Therefore, the implementation of the Wiener filter in order to learn the mapping of neuronal activity to hand movement parameters matches the well-known system identification block diagram of figure 1 [12]: the spike trains are the input to the multiple input–multiple output (MIMO) Wiener filter, and the desired response is the 3D hand position, velocity and gripping force. The ultimate goal is to substitute the use of the animal's arm by a robotic device controlled directly by the outputs of the trained Wiener filter. The applications in clinical situations involving handicapped individuals demand portability emphasizing signal processing models that can be implemented in low-power (eventually fixed point) digital signal processors.

Now that we have an idea of the goals and setup, let us turn our attention to the challenges of the application. One challenge is that the spatio-temporal spike trains' data have highly complex dynamics and therefore cannot be effectively used to guide the model design. Another challenge is the MIMO mapping problem with a large input dimensionality (i.e., for a 100 neuronal input vector, the Wiener filter with three outputs has 3000 free parameters). In addition, the statistics vary both in time and in the space of the electrodes. Some neuronal firings are not related to the task and constitute therefore noise in the data. Finally, because the true mapping of the sampled neuronal ensemble to movement parameters is unknown, it is also unknown if linear model would describe it the best.

The straight Wiener filter algorithm [12] is applied and evaluated in two types of tasks: food reaching and target hitting tasks. This algorithm which has already been proven to work well in BMI tasks will be used as a golden standard for model comparisons. Several modifications will be implemented to evaluate the following issues: the nonstationarity assumption is addressed with the normalized form of the least square algorithm (NLMS) [12] that is known to reach the same solution on average for stationary data, but may handle the nonstationary nature of the data better with its time-varying learning rate.

The issue of the number of free model parameters will be handled by three different techniques. The first is the subspace Wiener filter, which first projects the input data and then derives a Wiener filter to the desired response. In the field of signal processing, principal component analysis (PCA) has been widely used as a major subspace projection method [13], but it does not orient the projection to take advantage of the desired response structure. As an alternative, we propose a new idea of seeking subspace decomposition in the joint space through a hybrid subspace method, which combines the criterion of PCA and partial least squares (PLS) [14]. We also implement a reduction in the number of degrees of freedom of the model by using a generalized feedforward filter based on the gamma tap delay line [15], which has the ability to cover the same memory depth of the tap delay line with fewer taps. The third method implemented uses online regularization based on weight decay, which decreases the weight values of unimportant weights through training.

The final issue covered in this paper relates to the adequacy of the linear modeling. We design a nonlinear mixture of switching, competitive linear models that implement a locally linear but globally nonlinear model [16]. This structure can be thought as a time-delay neural network (TDNN) [13] that is trained in a different way to conquer the difficulty of training thousands of parameters with relatively small datasets. In order to place this class of models versus other alternatives, a Kalman filter and the population vector algorithm will also be compared in this paper.

2. Data collection

The performance of the models in BMI was evaluated in two experimental setups. In the first, the firings of 104 cells were collected by microwire arrays implanted in several cortical areas—posterior parietal cortex (PP), left and right primary

motor cortex (M1), and dorsal premotor cortex (PMd)—while an owl monkey (*Aotus trivirgatus*) performed the reaching task. In this task, the monkey was trained to reach for food placed in a tray. This movement consisted of four phases: reaching for food, grasping food, taking food to mouth and returning to the resting position. Both 3D hand positions and neural activity were recorded simultaneously during the task [3].

In the second experiment, the firing times of 192 cells were collected from three cortical areas, dorsal premotor cortex (PMd), primary motor cortex (M1) and supplementary motor area (SMA), while a Rhesus monkey performed the target hitting task: the monkey was trained to move a cursor controlled by a hand-held pole to hit a randomly located target on the computer screen. If the monkey intersected the target, a liquid reward was given.

A multichannel acquisition processor (MAP, Plexon, Dallas, TX) cluster was used to collect electrical activity from the implanted microwires. The spike was detected by time–amplitude discriminators and sorted by an off-line analysis based on a modified version of PCA [3, 11]. The accuracy of spike sorting can greatly impact BMI modeling that occurs at a later stage [17, 18]. Spike sorting is affected by both the applied detection/discrimination technique and subjective decisions of the neurophysiologists. To reduce the errors in the spike sorting procedure, the data were checked for artifacts and spikes that did not meet normal physiological parameters [11]. Neuron firings were counted (binned) in non-overlapping 100 ms time windows, and a 1 s time window was selected as an appropriate memory depth to derive the best linear projector [3]. The primate's hand position, used as the network desired signal, was also recorded (with a time-shared clock) and digitized with 200 Hz sampling rate. The desired hand position signal was then downsampled to 10 Hz to be aligned synchronously with firing count data. The data were collected by Dr Nicolelis' group at Duke University, and in this paper data from the sessions of one animal per experiment were utilized. Further details about the data collection and experiments can be found in [3, 7, 11].

3. BMI modeling using linear filters

Consider a set of spike counts of M neurons and a hand position vector $\mathbf{d} \in \Re^C$ (C is an output dimension, e.g. $C = 2$ or 3) for a given time instance (sampled at 10 Hz). The spike count for each neuron is embedded by an L-tap time-delay line. Then, the input vector for a given time instance n, $\mathbf{x}(n) = [x_1(n), x_1(n-1), \ldots, x_1(n-L+1), x_2(n), \ldots, x_M(n-L+1)]^T$, has the length of $L \cdot M$, where $x_i(n-l)$ is the spike count of neuron i delayed by l samples. A linear model estimating hand position at time instance n from the embedded spike counts can be described as

$$y^c(n) = \sum_{l=0}^{L-1} \sum_{i=1}^{M} x_i(n-l) w_{il}^c + b^c, \tag{1}$$

where $y^c(n)$ is the model output for the c-coordinate (c can be x, y or z), w_{il}^c is a weight on $x_i(n-l)$ for the c-coordinate output and b^c is a bias for the c-coordinate output. In a matrix form, we may rewrite (1) as

$$\mathbf{y}(n) = \mathbf{W}^T \mathbf{x}(n), \tag{2}$$

where $\mathbf{y}(n)$ is a C-dimensional output vector. The dimension of a weight matrix $\mathbf{W}$ is $(L \cdot M) \times C$. Each column of $\mathbf{W}$ consists of a vector, $\left[w_{10}^c w_{11}^c \cdots w_{1L-1}^c \cdots w_{ML-1}^c\right]^T$. Note that the bias terms can be removed by zeroing the mean of the input and output.

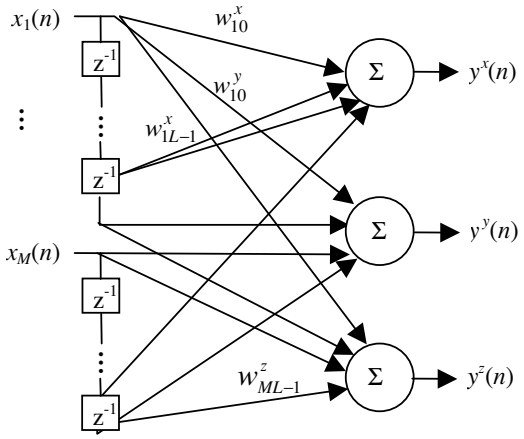

Figure 2. The topology of the linear filter designed for BMIs in the case of the 3D reaching task. w_{il}^c is a weight connecting $x_i(n-l)$ and $y^c(n)$, and z^{-1} is a discrete delay operator.

Figure 2 shows the topology of the linear model for the BMI application, which will be kept basically unchanged throughout these studies. The most significant differences will be in the number of parameters and in the way the parameters of the linear model will be estimated from the data.

In our study, all the models are applied to estimate the 3D or 2D hand positions using $M = 99$ neurons for the reaching task and $M = 192$ for the target hitting task. The number of taps in the delay line, L, is 10 unless otherwise stated. The size of the training and the testing set is 10 000 samples (∼16.7 min) and 3000 samples (∼5 min), respectively. The weights of the models are fixed after adaptation and the outputs are produced for novel testing samples. The output trajectories of the different models for the same segment of test data (the x, y and z coordinates for the reaching task and the x and y coordinates for the target hitting task) will be shown in figures 9 and 10 for visual comparison purpose.

The following quantitative performance measures are used to evaluate the accuracy of the estimation: correlation coefficient (CC) quantifies the linear relationship between estimated and actual hand trajectories defined as

$$\mathrm{CC} \equiv \frac{C_{dy}}{s_d s_y}, \tag{3}$$

where C_{dy} denotes the covariance between estimated (y) and actual (d) hand trajectories, and s_d (or s_y) denotes the standard deviation. The signal-to-error ratio (SER) is the ratio of the powers of the actual hand trajectories and the errors, defined as

$$\mathrm{SER} \equiv \frac{\sum_{k=1}^{K} |d(k)|^2}{\sum_{k=1}^{K} |e(k)|^2}, \tag{4}$$

where $d(k)$ and $e(k)$ are the actual hand signal and the error at time instance k and K is the size of the window in which SER is measured. The SER and CC are computed over short-time sliding windows (the size of windows is determined by the duration of the movement). We measure the performance of all models as the empirical distribution of CC and the SER evaluations: mean and standard deviation of CC (and the SER) over test data are quantified to describe the distribution. Further, we divide evaluation results for food reaching into two modes: movement and rest. Then the average of CC and the SER over three coordinates for each mode are used to estimate the empirical distribution. For target hitting, the distribution is estimated separately for each coordinate. The evaluations of these performance measures for every model we design in this paper will be summarized in tables 1 and 2.

3.1. Wiener filter

The transfer function from the neural spike counts to the hand position can be estimated by linear adaptive filters, among which the Wiener filter plays a central role [12]. The weights of the Wiener filter for the MIMO system are estimated by the Wiener–Hopf solution as

$$\mathbf{W}_{\text{Wiener}} = \mathbf{R}^{-1}\mathbf{P}. \tag{5}$$

$\mathbf{R}$ is the correlation matrix of neural spike inputs with the dimension of $(L \cdot M) \times (L \cdot M)$,

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & \cdots & \mathbf{r}_{1M} \\ \mathbf{r}_{21} & \mathbf{r}_{22} & \cdots & \mathbf{r}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{r}_{M1} & \mathbf{r}_{M2} & \cdots & \mathbf{r}_{MM} \end{bmatrix}, \tag{6}$$

where $\mathbf{r}_{ij}$ is an $L \times L$ cross-correlation matrix between different neurons i and j, and $\mathbf{r}_{ii}$ is an $L \times L$ autocorrelation matrix of neuron i. $\mathbf{P}$ is a cross-correlation matrix between each neuron and hand positions,

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{11} & \cdots & \mathbf{p}_{1C} \\ \mathbf{p}_{21} & \cdots & \mathbf{p}_{2C} \\ \vdots & \ddots & \vdots \\ \mathbf{p}_{M1} & \cdots & \mathbf{p}_{MC} \end{bmatrix}, \tag{7}$$

where $\mathbf{p}_{ic}$ is a cross-correlation vector between neuron i and c-coordinate of hand positions. It is straightforward to see that $\mathbf{r}_{ji}^T = \mathbf{r}_{ij}$, which leads to a symmetric $\mathbf{R}$. Therefore, the Cholesky factorization can be used to invert $\mathbf{R}$ to reduce the computational complexity [19].

Note that $\mathbf{R}$ must be a nonsingular matrix to obtain the solution from (5). However, if the condition number of $\mathbf{R}$ is very large, then $\mathbf{W}_{\text{Wiener}}$ may be inadequately determined. In that case, we can reduce the condition number by adding an identity matrix multiplied by some constant to $\mathbf{R}$ before inversion. This procedure is the well-known ridge regression in statistics [20]. The details will be discussed in section 4.3.

3.2. Normalized LMS adaptation

The underlying assumption of the Wiener filter is that statistics of the data are time invariant. However, in the nonstationary environment where statistics of the data vary over time, the Wiener filter uses only the average statistics to determine weights.

The normalized least mean squares (NLMS) algorithm, a modified version of the least mean squares (LMS) algorithm, can train weights effectively for nonstationary inputs by varying the learning rate [12]. It utilizes an estimation of the input power to adjust the learning rate at each time instance. The update rule of the NLMS for the weight vector connecting the c-coordinate of hand position, at time instance n, is given by

$$\mathbf{w}_{\text{NLMS}}^c(n+1) = \mathbf{w}_{\text{NLMS}}^c(n) + \frac{\eta}{\gamma + \|\mathbf{x}(n)\|^2} e^c(n)\mathbf{x}(n), \tag{8}$$

where η satisfies $0 < \eta < 2$, γ is a small positive constant and $e^c(n)$ is the error for the c-coordinate. Although the weights in the NLMS asymptotically converge in a statistical sense to the Wiener filter for stationary data, the solution can be different for nonstationary data. In our analysis, the weights of the linear filter are trained by NLMS with the settings of $\eta = 0.01$ and $\gamma = 1$.

4. Performance enhancement using regularization

4.1. Spatial regularization by subspace projection

One of the challenges in the BMI application is that some neurons' firings are not substantially modulated during task performance and they only add noise to the multichannel data. In addition, some neurons' firings are correlated, and it may be advantageous to blend their inputs to improve model performance. Subspace projection, which can reduce noise and blend together correlated input signals, also reduces the number of degrees of freedom in the multichannel data, and consequently decreases the variance of the fitted model. In this section, we introduce the hybrid subspace projection method which is derived by combining criteria of principal component analysis (PCA) and partial least squares (PLS). Then we will design the subspace Wiener filters based on the hybrid subspace projection.

PCA, which preserves maximum variance in the projected data, has been widely adopted as a projection method [13]. The projection vector is determined by maximizing the variance of the projection outputs as

$$\mathbf{w}_{\text{PCA}} = \arg\max_{\mathbf{w}} \quad J^{\text{PCA}}(\mathbf{w}) = E[\|\mathbf{x}\mathbf{w}\|^2] = \mathbf{w}^T\mathbf{R}_s\mathbf{w}. \tag{9}$$

where $\mathbf{R}_s$ is the input multichannel correlation matrix but now computed over space only. The columns of the PCA projection matrix are eigenvectors of $\mathbf{R}_s$ corresponding to the largest eigenvalues. However, PCA does not exploit information in the joint space of both input and desired signals. This means that there may be directions with large variance that are not important to describe the desired response (for example, neuronal modulations related to the monkey's anticipation of reward; they might be substantial, but useless for estimation

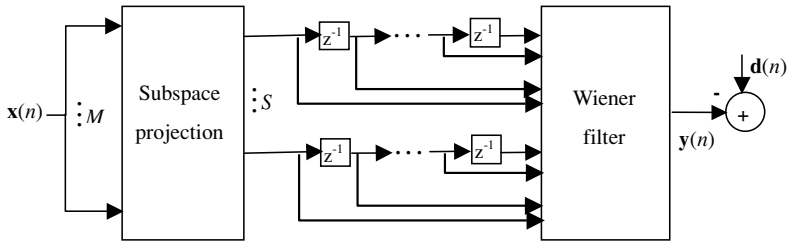

Figure 3. An overall diagram of the subspace Wiener filter. M is the number of input neurons and S is the subspace dimension, where $S < M$. $\mathbf{y}(n)$ is the estimated hand position vector for given spike count input and desired output vectors, $\{\mathbf{x}(n), \mathbf{d}(n)\}$.

of movement parameters), but will be preserved by the PCA decomposition. One of the subspace projection methods to construct the subspace in the joint space is the PLS, which seeks the projection maximizing the cross-correlation between the projection outputs and desired signals [14]. Given an input matrix $\mathbf{X}$ and a desired response vector $\mathbf{d}$, a projection vector of PLS, $\mathbf{w}_{\mathrm{PLS}}$, maximizes the following criterion:

$$\begin{aligned}\mathbf{w}_{\mathrm{PLS}} &= \arg\max_{\mathbf{w}} J^{\mathrm{PLS}}(\mathbf{w}) \\ &= E[(\mathbf{X}\mathbf{w})^T\mathbf{d}] = E[\mathbf{w}^T\mathbf{X}^T\mathbf{d}] = \mathbf{w}^T\mathbf{p},\end{aligned} \tag{10}$$

where $\mathbf{p}$ is defined as a cross-correlation vector between $\mathbf{X}$ and $\mathbf{d}$. The consecutive orthogonal PLS projection vectors are computed using the deflation method [13]. Also, the continuum regression (CR), introduced by Stone and Brooks [21], blends the ordinary least square, PCA and PLS, and provides a projection matrix that maximizes a weighted average of both variance and cross-correlation. Recently, we have proposed a hybrid criterion function similar to CR, together with a sample by sample learning algorithm to estimate the projection matrix that maximizes the criterion [22]. The learned projection can be either PCA, PLS or an arbitrary combination of them. The hybrid criterion function combining PCA and PLS is given by

$$J(\mathbf{w}, \lambda) = \frac{(\mathbf{w}^T\mathbf{p})^{2\lambda}(\mathbf{w}^T\mathbf{R}_s\mathbf{w})^{1-\lambda}}{\mathbf{w}^T\mathbf{w}}. \tag{11}$$

By taking the logarithm, the criterion can be rewritten as

$$\begin{aligned}\log(\hat{J}(\mathbf{w}, \lambda)) &= \lambda \log(\mathbf{w}^T\mathbf{p})^2 \\ &+ (1-\lambda)\log(\mathbf{w}^T\mathbf{R}_s\mathbf{w}) - \log(\mathbf{w}^T\mathbf{w}).\end{aligned} \tag{12}$$

We seek to maximize this criterion for $0 \leqslant \lambda \leqslant 1$. There are two learning algorithms derived in [22] to find $\mathbf{w}$, but we use the fixed-point algorithm in this paper due to its fast convergence and independence of learning rate. The estimation of $\mathbf{w}$ at the $(k+1)$th iteration with the fixed-point algorithm is given by

$$\mathbf{w}(k+1) = (1-T)\mathbf{w}(k) + T\left[\frac{\lambda\mathbf{p}}{\mathbf{w}(k)^T\mathbf{p}} + \frac{(1-\lambda)\mathbf{R}_s\mathbf{w}(k)}{\mathbf{w}(k)^T\mathbf{R}_s\mathbf{w}(k)}\right] \tag{13}$$

with a random initial vector $\mathbf{w}(0)$, where T $(0 < T < 1)$ is a balancing parameter to remove the oscillating behavior near convergence. The convergence rate is affected by T, which produces a tradeoff between the convergence speed and the accuracy. We obtain the fastest convergence for $T \approx 1$. The consecutive projection vectors are also learned by the deflation method, forming the projection matrix $\mathbf{W}$. After projection onto the subspace by $\mathbf{W}$, we embed each channel with L-tap ($L = 10$ here) delays and design the Wiener filter to estimate the hand positions. Figure 3 illustrates an overall diagram of the subspace Wiener filter.

The hold-out cross-validation method [23] is utilized to determine both the optimal subspace dimension (S) and λ. 10 000 samples are divided into 9000 training samples and 1000 validation samples to estimate the generalization error, for both the food reaching and target hitting tasks. The MSE for the validation set at each set of (S_i, λ_j), where $S_i \in \{20, 21, \ldots, 60\}$ and $\lambda_j \in \{0, 0.1, \ldots, 1\}$, is computed to find the optimal set at which the MSE is minimized. In figure 4, the contour map of the MSE for the validation set is depicted. The minimum MSE is found at (37, 0.9) for the food reaching task and (44, 0.6) for the target hitting task. The MSE also tends to be smaller for larger λ in the lower subspace dimensions while the MSE levels are rather flat in the higher subspace dimensions. This indicates that PLS plays a more important role in building a better subspace Wiener filter for smaller subspace dimensions.

4.2. Parsimonious modeling in time using the gamma filter

The large number of parameters in the linear model is caused not only by the number of neurons but also by the number of tap delays required to capture the history of the neuron firings over time. A generalized feedforward filter with the gamma tap delay line can reduce the number of taps by blending features of finite impulse response (FIR) and infinite impulse response (IIR) filters [15]. It has been shown that a generalized feedforward filter can provide trivial stability conditions and easy adaptation while decoupling the memory depth from the filter order [15]. Figure 5 illustrates the architecture of the generalized feedforward filter, where an input signal is delayed through each tap by an operator defined by a specific transfer function $G(z)$. Note that when $G(z) = z^{-1}$, it becomes an FIR filter. The gamma filter is a special case of the generalized feedforward filter with $G(z) = \mu/(z - (1-\mu))$, where μ is a feedback parameter. In this case, the signal from the kth tap is determined by the following difference equation:

$$x_k(n) = (1-\mu)x_k(n-1) + \mu x_{k-1}(n-1), \tag{14}$$

where $x(n)$ is a given input signal at time instance n.

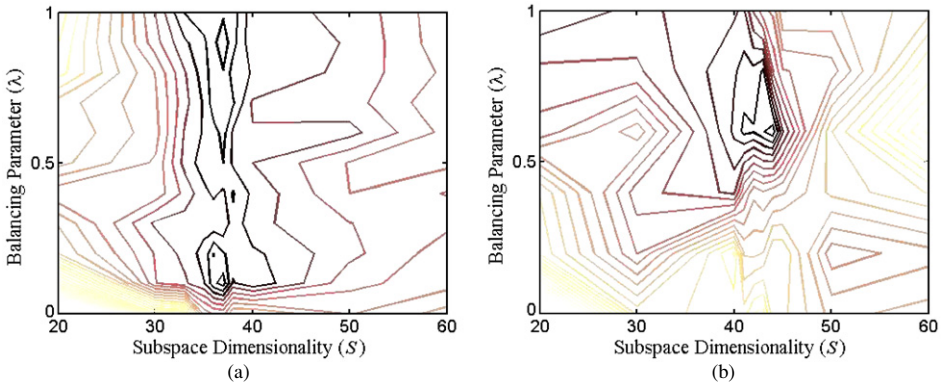

Figure 4. The contour of the validation MSE from the subspace Wiener filter for two tasks: (a) the food reaching task and (b) the target hitting task. The darker contour lines denote lower MSE level.

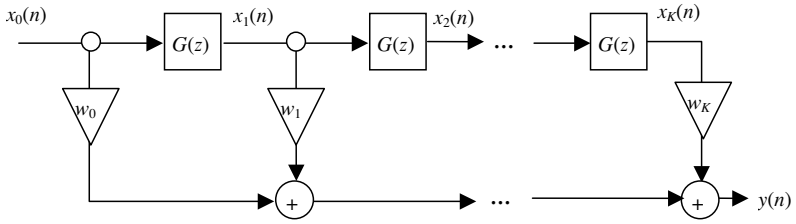

Figure 5. The architecture of the generalized feedforward filter.

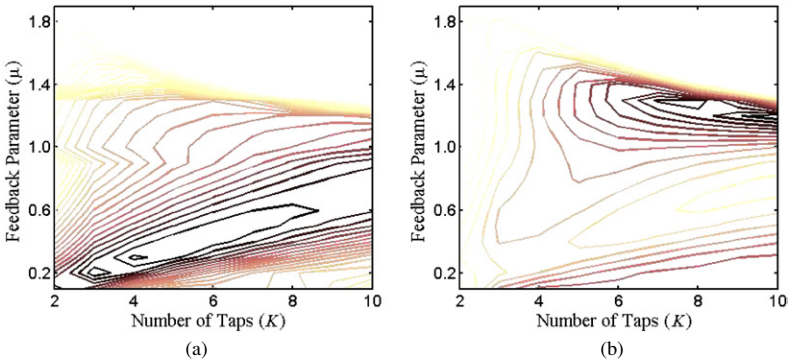

Figure 6. The contour of the validation MSE from the gamma filter for two tasks: (a) the food reaching task and (b) the target hitting task. The darker lines denote lower MSE level.

The memory depth D of the gamma filter consisting of K taps is given by

$$D = \frac{K}{\mu} \quad \text{for} \quad \mu < 1 \qquad \text{or} \qquad D = \frac{K}{2-\mu} \quad \text{for} \quad \mu > 1. \tag{15}$$

This shows that the gamma filter decouples the memory depth from the filter order by adjusting a feedback parameter (μ). In the case of $\mu = 1$ (i.e., the FIR filter), the resolution is maximized whereas the memory depth is minimized for a given filter order. But this choice sometimes results in overfitting when a signal to be modeled requires more time delays than the number of descriptive parameters. On the other hand, the gamma filter with the proper choice of a feedback parameter can avoid overfitting by the decoupled memory structure.

The tap weights can be updated using the NLMS, and therefore the computational complexity is similar to that of FIR filters. A feedback parameter μ can also be adapted from the data. Instead of adaptively learning μ from data, however, we can determine the *best* combination of K and μ in terms of the generalization error, by estimating the cross-validation error for each set of K_j and μ_i, where $K_j \in \{2, 3, \ldots, 10\}$ (ignoring the case of $K_j = 1$, which implements memoryless process) and $\mu_i \in \{0.1, 0.2, \ldots, 1.9\}$.

The same cross-validation procedure is applied for the gamma filter to find the optimal values of K and μ. In figure 6, the contour map of the MSE for the validation set is depicted. The minimum MSE values are found at $(K, \mu) = (4, 0.3)$ for the food reaching task and $(K, \mu) = (10, 1.2)$ for the target hitting task. The memory depth estimated by this empirical

method becomes $D \approx 13$ for the food reaching task and $D \approx 12.5$ for the target hitting task. The savings in the number of parameters are 60% (3120 → 1248) for the food reaching task. The temporal resolution of the gamma filter (which is defined as $R \equiv K/D$) is different for the two tasks: $R = 0.31$ for food reaching and $R = 0.8$ for target hitting. This might indicate that more irregular target hitting movement requires finer temporal resolution in the neural input data.

4.3. Adaptive regularization with pruning

Since the neural firing data are highly variable creating a large dynamic range of bin counts, the condition number of an input correlation matrix may be relatively large. To reduce the condition number, we can add an identity matrix multiplied by a white noise variance to the correlation matrix, which is known as ridge regression (RR) in statistics [20]. The criterion function of RR is given by

$$J(\mathbf{w}) = E[\|\mathbf{e}\|^2] + \delta\|\mathbf{w}\|^2, \tag{16}$$

where the additional term $\delta\|\mathbf{w}\|^2$ smoothes the cost function. The weight decay regularization can be viewed as a simple online method to minimize the RR criterion function using the stochastic gradient, updating the weights by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\hat{\nabla}\zeta(n) - \delta\mathbf{w}(n), \tag{17}$$

where $\hat{\nabla}\zeta(n) = \partial E[\|\mathbf{e}(n)\|^2]/\partial\mathbf{w}(n)$. Both RR and weight decay can be viewed as implementations of a Bayesian approach to complexity control in supervised learning using a zero-mean Gaussian prior [24].

The choice of the amount of regularization (δ) may play an important role in the generalization performance, since there is a tradeoff between the condition number and the achievable MSE for a particular δ. A larger δ can decrease the condition number at the expense of increasing the MSE, while a smaller δ can decrease the MSE but also increase the condition number. Larsen *et al* [25] proposed that δ can be optimized by minimizing the generalization error with respect to δ. Following this procedure, we utilize the K-fold cross-validation [26], which divides the data into K randomly chosen disjoint sets, to estimate the average generalization error empirically,

$$\hat{\xi} = \frac{1}{K}\sum_{k=1}^{K}\varepsilon_k, \tag{18}$$

where ε_k is MSE of the validation for the kth set. Then, the optimal regularization parameter is learned by using gradient descent,

$$\delta(n+1) = \delta(n) - \eta\frac{\partial\hat{\xi}(n)}{\partial\delta}, \tag{19}$$

where $\hat{\xi}(n)$ is an estimate computed with $\delta(n)$ and $\eta > 0$ is a learning rate. See [25] for the procedure of estimation of $\partial\hat{\xi}(n)/\partial\delta$ using weight decay.

In the experiments, we set $K = 10$, $\eta = 10^{-6}$ and update δ until the difference $|\hat{\xi}(n+1) - \hat{\xi}(n)|$ is less than 10^{-3}. The term $\hat{\nabla}\zeta(n)$ in (17) is estimated by the NLMS. In experimental results, δ converges to 1.36×10^{-5} for the food reaching task and 1.02×10^{-5} for the target hitting task. Then, we train the filter using fixed δ with the entire training samples (10 000) to obtain the regularized model.

4.4. Generative model: the Kalman filter

The linear models described so far attempt to directly approximate the input–output mapping from neural firing rates to the hand kinematics such as

$$\mathbf{d}(n) = f(\mathbf{x}(n)) + \omega(n), \tag{20}$$

with the assumption of an additive white Gaussian noise $\omega(n)$. However, there have been different approaches from a generative modeling viewpoint [8, 27, 28]. In these approaches, neural firing rates are assumed to be stochastically generated from the hand kinematics with some noise $\upsilon(n)$ as

$$\mathbf{x}(n) = f(\mathbf{d}(n), \upsilon(n)). \tag{21}$$

This model leads to the likelihood of the observation of firing rates given the state of hand kinematics, $p(\mathbf{x}(n)|\mathbf{d}(n))$. Since we seek to decode the hand kinematics given the observation of firing rates, Bayesian inference can be used to infer the posterior probability, $p(\mathbf{d}(n)|\mathbf{x}(n))$. This inference can be recursively performed by some assumptions of independence [8]. There are many ways to model $p(\mathbf{d}(n)|\mathbf{x}(n))$, but the Kalman filter has been a primary choice since it is optimal with certain parametric assumptions and easy to implement. In the Kalman filter, a linear Gaussian model is employed to approximate the likelihood and a prior such that

$$\mathbf{x}(n) = \mathbf{H}\mathbf{d}(n) + \upsilon(n), \tag{22}$$

$$\mathbf{d}(n+1) = \mathbf{A}\mathbf{d}(n) + \omega(n), \tag{23}$$

where both $\upsilon(n)$ and $\omega(n)$ are Gaussian noises, and $\mathbf{H}$ and $\mathbf{A}$ are linear coefficient matrices. Since, $p(\mathbf{d}(n)|\mathbf{x}(n))$ is also Gaussian, we can infer it by recursively estimating the mean and covariance. The mean of $p(\mathbf{d}(n)|\mathbf{x}(n))$ will be our decoded hand kinematics. More details about parameter estimation, algorithms and discussions of the Kalman filter for BMIs can be found in [27].

In our experiments, hand position, velocity and acceleration are included in the hand kinematic states, and neural spike counts in 100 ms bins are accounted for the observation of firing rates. The empirical test results of the Kalman filter are compared with other models as presented in section 6.2.

5. Nonlinear mixture of competitive linear models

Although it is shown in previous sections that a number of regularization methods can improve the generalization performance for a linear model, there exists a fundamental assumption of linearity that may impact performance. To overcome this limitation, here we compare two nonlinear models such as the straight time-delay neural network (TDNN) [13] and a nonlinear mixture of competitive multiple linear models (NMCLM) [16].

In the TDNN, the mapping between neural activity and motor parameters (hand positions, velocities and griping forces) is estimated by nonlinearly combining spike counts (and their past values) from each neuron. The tap delay lines in the input layer preset the memory to account for temporal

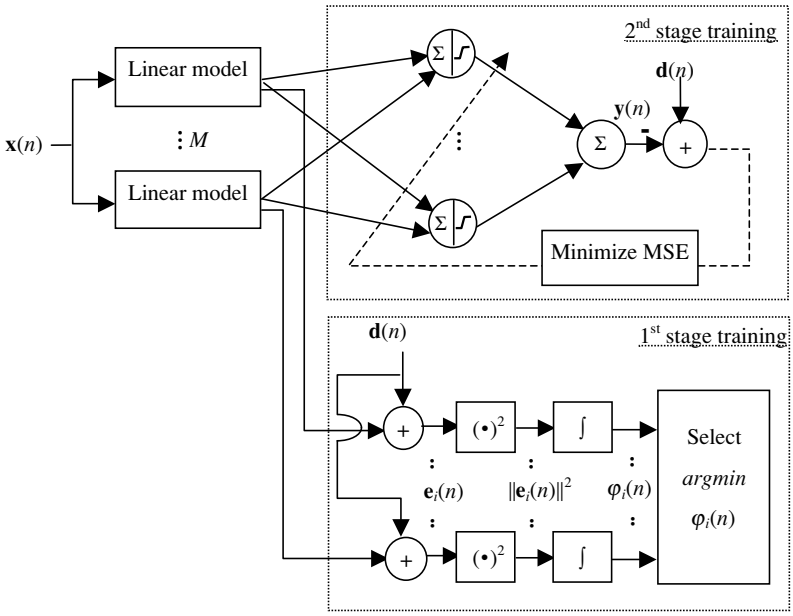

Figure 7. An overall diagram of nonlinear mixture of competitive linear models.

dependencies in neural activity. This architecture has a single hidden layer with sigmoid nonlinearities and the output layer with linear processing elements (PEs). The output of the TDNN is given by $\mathbf{y}(n) = \mathbf{W}_2 f\left(\mathbf{W}_1^T \mathbf{x}(n) + \mathbf{b}_1\right) + \mathbf{b}_2$, where the weights and biases $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are trained by the error backpropagation algorithm [13].

NMCLM consists of a bank of multiple linear models and a multilayer perceptron (MLP) with single hidden layer. The overall architecture of NMCLM is equivalent to the TDNN, but the training procedure is different as depicted in figure 7. A two-stage training procedure that is performed sequentially includes the competitive learning with the NLMS for the linear models and the error backpropagation for the MLP. Note that the same desired response is used in both training stages. The underlying reasoning in the multiple local linear models is that a complex nonlinear mapping can be approximated by dividing it into simpler linear mappings and combining them properly. Fancourt and Principe successfully designed a gated competitive system consisting of multiple linear models for nonstationary signal segmentation based on this approach [29]. If we assume that for different local regions in the hand trajectory space the mapping from neural activity to motor variables differs from the others, linear models each specializing in a local region will enhance the prediction performance significantly.

The multiple linear models are trained by competitively updating weights using the NLMS. The criterion to be used in the competition is the integrated squared error (ISE) that each competing model produces, given by

$$\varphi_i(n) = (1-\alpha)\varphi_i(n-1) + \alpha\|\mathbf{e}_i(n)\|^2, \qquad i = 1, \ldots, M, \tag{24}$$

where M is the number of models and α is the time constant of the leaky integrator. A linear model that exhibits the least ISE wins the competition at a given time instance, and only the weights of the winning model are updated (hard competition) [29]. With this training procedure, each model can specialize in local regions in the joint input/desired signal space. Figure 8 demonstrates the specialization of ten trained models by plotting their outputs with the common input data (40 s long) in the 3D hand trajectory space. This figure shows that the input–output mappings learned by each model display some degree of localization, although overlaps are still present. These overlaps may be consistent with a neuronal multiplexing effect as depicted in [7], which suggests that the same neurons modulate for more than one motor parameter (x and y coordinates of hand position, velocity and griping force).

The competitive local linear models, however, require additional information for switching when applied in BMIs, since the desired signal that is necessary to select a local model is not available in practice. A gate function as in the mixture of experts [30] utilizing input signals needs to be trained to select a local model. Here we opted for a MLP that directly combines the predictions of all models. Therefore, the overall architecture can be conceived as a nonlinear mixture of competitive linear models (NMCLM). This procedure facilitates training of each model compared to the TDNN, since only one linear model is trained at a time in the first stage, while only a relatively small number of weights are trained by error backpropagation in the second stage. It has been shown in [16] that the mutual information [31] between the desired output and the competitive model outputs is larger than the first layer outputs of the equivalent TDNN (all the weights are trained only by error backpropagation), which shows that the training in NMCLM is more efficient.

In the experiments, the topology of NMCLM consists of ten competitive linear models for each coordinate and a single hidden layer MLP with M inputs ($M = 10 \times C$, C is the output dimension: 2 or 3), 30 hidden PEs with hyper-tangent

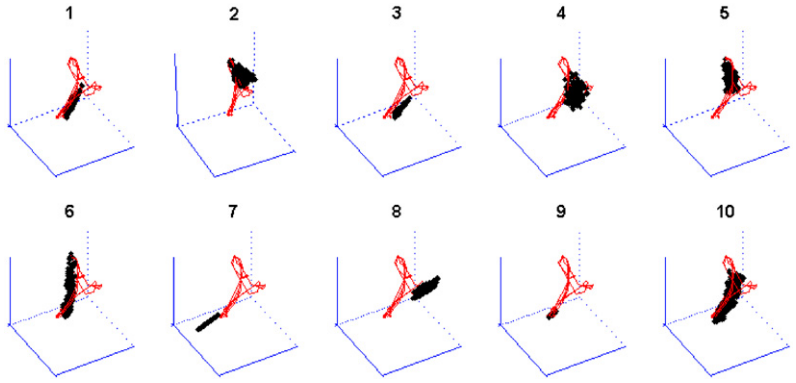

Figure 8. Demonstration of the localization of multiple linear models. The common input data are fed to ten competitively trained models. Then, the 3D outputs from each model (dark dots) are plotted on top of the common actual hand trajectory (thin lines).

function and C linear output PEs to predict each hand position coordinate. Each linear model has the same topology as that used in section 3. The number of multiple models and the number of hidden PEs were chosen empirically (they was not optimized) by varying their number in multiple runs. The hard competition learning rule is utilized along with the NLMS for the training of linear models and the conjugate gradient algorithm is used to train the MLP.

Training of the MLP is repeated with 100 random initial conditions and the solution with the least MSE is selected. The time constant of the leaky integrator (α) is determined by the hold-out cross-validation method utilized in section 4.1. The data are divided into a 9000-sample training set and a 1000-sample validation set. The resulting values of α are 0.3 for the food reaching task and 0.6 for the target hitting task.

The TDNN is trained with the same input and desired response as in NMCLM. The 30 PEs in the hidden layer use tanh nonlinearities. All the weights and biases are trained by the error backpropagation with the MSE criterion.

Even with the simpler training approach, there are over 30 000 parameters in the NMCLM to be trained. Each linear model with around 3000 parameters is trained with a fraction of the total number of samples (only those pertaining to its local area of the space), which is considered too high for the restricted number of training samples. With linear models built from gamma filters, we can significantly reduce the number of parameters in the first layer of NMCLM, while preserving the same level of computational complexity in training.

6. Comparison of models

In this section, we summarize the evaluation of the performance for all models introduced in this paper. We emphasize, however, that the comparison is done for datasets containing 100–200 simultaneously recorded neurons for which the standard Wiener filter algorithm yielded very good performance. With the increase of the number simultaneously recorded neurons, task complexity and complexity of predicted motor parameters, what we see only as tendencies in these comparisons, may become important features of BMI design.

Before presenting quantitative results, we first demonstrate the outputs of every model along with the actual hand trajectories for food reaching in figure 9 and for target hitting in figure 10. Since our approaches have been developed by assigning the Wiener filter as a golden standard, observations in these figures will be made mainly by comparing trajectories of models with that of the Wiener filter.

First, we can observe that NLMS can predict better rest positions than the Wiener filter. The empirical results indicate that a linear filter trained by the NLMS improves the accuracy of the estimation especially during rest for the reaching task (see table 1), meaning that the weights found a better compromise between the two very different characteristics of movement and rest.

This improvement has been achieved because of the NLMS update rule, where the weights are updated with a relatively higher learning rate during rest due to the fact that total firing count relatively decreases when monkeys do not move arms. Thus, for the class of motor behaviors in which movement periods are separated by periods of rest, the NLMS algorithm can capture more information about rest positions than the Wiener filter.

Next, we can see that regularized models yield smoother output trajectories than the Wiener filter especially during rest. Also, NMCLM provides the most accurate movement prediction as shown in figure 9(f). NMCLM shows its ability to follow the rest position with little jitter and to track rapid changes of hand trajectory during movements. This is due to the nonlinear global structure of the NMCLM. On the other hand, in figure 10, all models show similar prediction performance for the target hitting task. Performance measures presented later will demonstrate this performance similarity (although there are statistical differences between models).

6.1. Comparison of the parameters of linear filters

We now compare the parameters of the four linear models: the Wiener filter, the linear model trained by NLMS, the gamma filter and the linear model regularized by weight decay. Note that other models (i.e. the Kalman filter, the

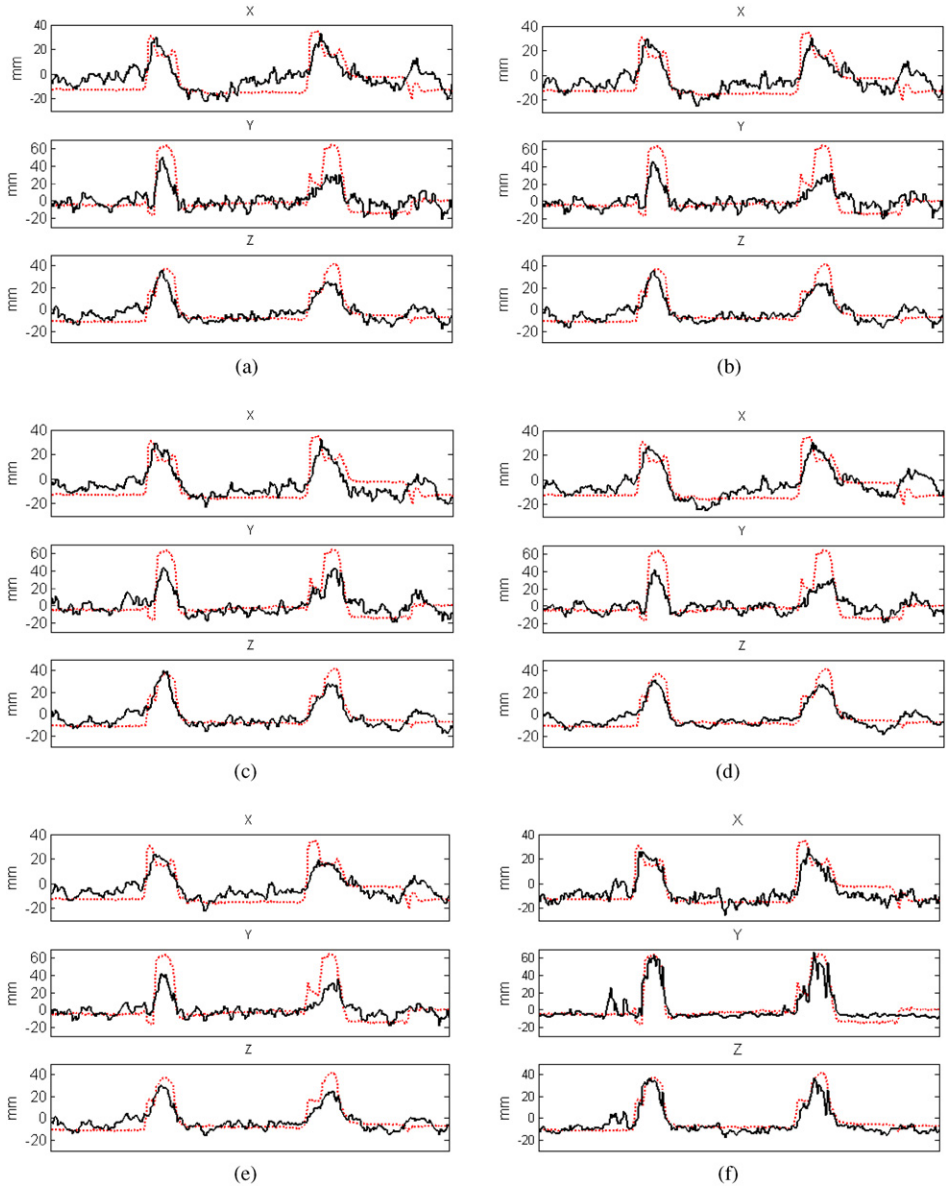

Figure 9. The actual hand trajectory (dotted line) and the estimated hand trajectory (solid line) in the x, y and z coordinates for the 3D food reaching task on a sample part of the test data: (a) the Wiener filter, (b) the linear filter with NLMS, (c) the subspace Wiener filter, (d) the gamma filter, (e) the linear filter regularized by weight decay and (f) NMCLM.

subspace Wiener filter or nonlinear models) are not included here due to the complexity of extracting individual neuronal contributions to model output. We first average the weight magnitudes over time lags of individual neuronal channels and over the output dimension in order to represent the composite weight associated with single neurons since the number of delays in each model is different. Then, the average magnitude per neuron is multiplied by the standard deviation of individual neuron's bin count to obtain a measure of neuronal contribution; this is defined as the average sensitivity of the output to each individual neuron [32]. Figure 11 shows individual neuron sensitivity for the four models. Note that for visual purposes we scale the sensitivity values such that the maximum sensitivity becomes unity.

It can be observed in figure 11(a) that the normalized weight magnitude distributions are similar among models except for the gamma filter. The distribution of the NLMS-trained model follows that of the Wiener filter, but decreases the smaller magnitude weights, which might explain the higher performance during rest. Weight decay further prunes weights, therefore generating sparse distribution of weights, which can help generalization. The distribution of the gamma filter's weights differs as can be expected due to the different time scale. It weights more neurons 57, 84, 87 and 94, where neuron 57 is the neuron with highest firing rate and neuron 94 has one of the highest sensitivities according to the analysis in [32]. For the target hitting task as shown in figure 12(b), all models present similar weight magnitude

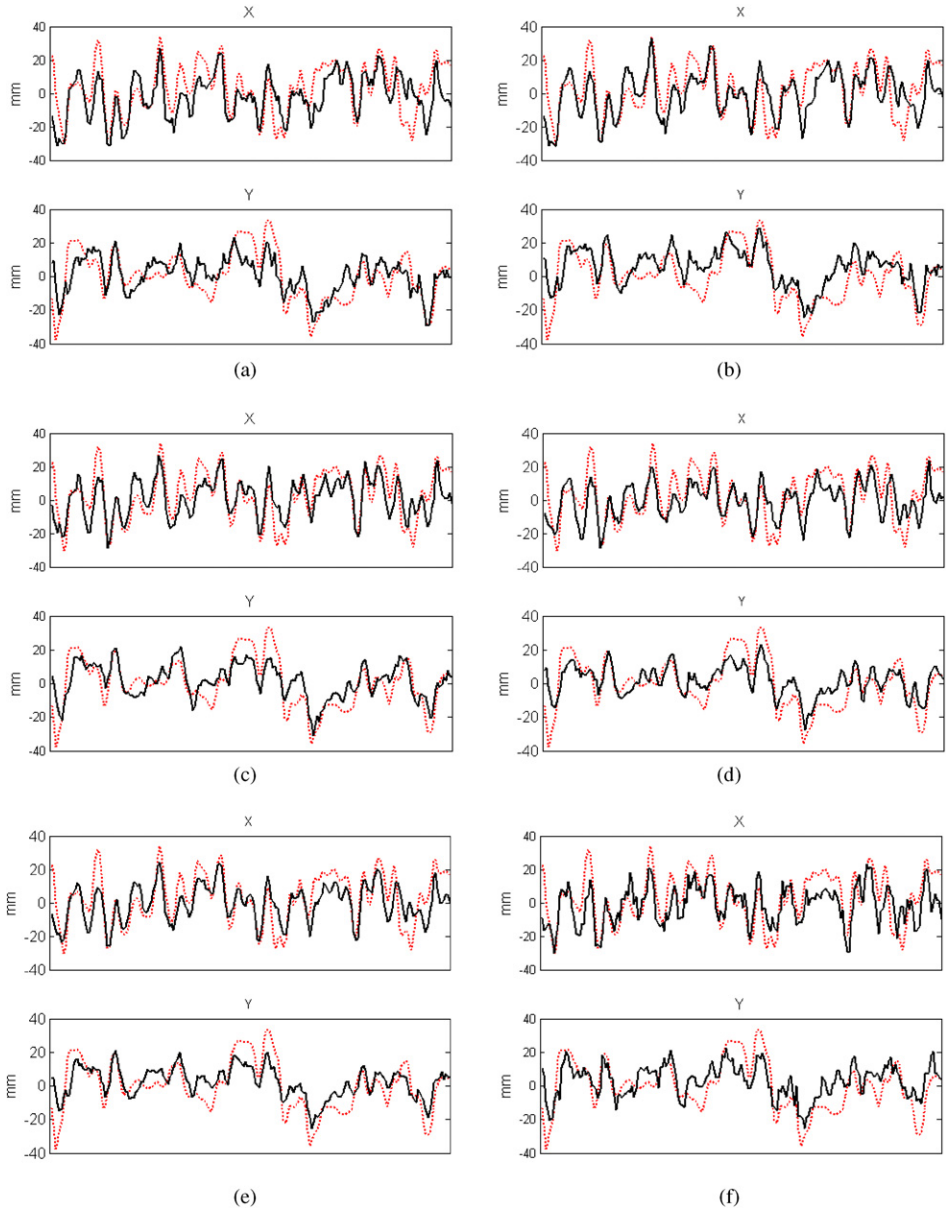

Figure 10. The actual hand trajectory (dotted line) and the estimated hand trajectory (solid line) in the x and y coordinates for the 2D target hitting task on a sample part of the test data: (a) the Wiener filter, (b) the linear filter with NLMS, (c) the subspace Wiener filter, (d) the gamma filter, (e) the linear filter regularized by weight decay and (f) NMCLM.

distributions, which likely explains the similar performance of all models.

6.2. Statistical performance comparison

Tables 1 and 2 summarize the generalization performances of all models using measures introduced in section 3. There are ten food reaching movements in the test data for which the performances are measured. We start by pointing out that the performance of the Wiener filter, our baseline, is relatively similar to all the other implementations, which is surprising. It is interesting to compare the performance versus the number of degrees of freedom of each model. As is well known, the number of parameters improves the bias of the model, but penalizes the variance, so the overall performance is a compromise. We can see that the gamma filter appreciably decreases the number of parameters of the base linear model and slightly improves the performance with the best combination of K and μ, as shown in tables 1 and 2. The performance of the subspace Wiener filter for both tasks also reaches slightly higher level than those of the Wiener filter or the NLMS, which indicates that generalization was improved by the subspace projection. Note, however, that the number of free parameters is not the only requirement for good performance. The NMCLM has many more parameters, but since it has an internal competitive topology, performance

Table 1. The generalization performances of linear and nonlinear models for the 3D food reaching task.

Measures	No of weights	CC (movement)	SER (movement) (dB)	CC (rest)	SER (rest) (dB)
Wiener	2 973	0.76 ± 0.19	4.76 ± 1.87	0.03 ± 0.22	2.40 ± 2.80
NLMS	2 973	0.75 ± 0.20	4.85 ± 2.11	0.06 ± 0.22	3.40 ± 2.76
Gamma	1 191	0.78 ± 0.19	5.25 ± 1.97	0.07 ± 0.21	3.59 ± 3.11
Subspace	1 113	0.77 ± 0.18	4.84 ± 2.06	0.09 ± 0.20	3.78 ± 2.57
Weight decay	<2 973	0.77 ± 0.18	4.73 ± 2.04	0.07 ± 0.22	3.76 ± 2.78
Kalman	1 017	0.78 ± 0.20	4.32 ± 1.97	0.05 ± 0.25	2.26 ± 3.85
TDNN	29 823	0.77 ± 0.17	4.87 ± 2.56	0.02 ± 0.22	3.29 ± 5.67
NMCLM (FIR)	30 753	0.81 ± 0.15	5.90 ± 3.00	0.03 ± 0.22	5.64 ± 4.00
NMCLM (gamma)	12 933	0.81 ± 0.19	6.08 ± 3.19	0.06 ± 0.23	6.23 ± 5.23

Table 2. The generalization performances of linear and nonlinear models for the 2D target hitting task.

Measures	No of weights	CC (x)	SER (x) (dB)	CC (y)	SER (y) (dB)
Wiener	3 842	0.66 ± 0.02	2.42 ± 0.54	0.48 ± 0.10	1.08 ± 0.52
NLMS	3 842	0.68 ± 0.03	2.42 ± 0.55	0.50 ± 0.08	0.90 ± 0.49
Gamma	3 842	0.70 ± 0.02	2.81 ± 0.69	0.53 ± 0.09	1.55 ± 0.43
Subspace	882	0.70 ± 0.03	2.80 ± 0.83	0.58 ± 0.08	1.90 ± 0.57
Weight decay	<3 842	0.71 ± 0.03	2.79 ± 0.92	0.57 ± 0.08	1.75 ± 0.46
Kalman	1 188	0.71 ± 0.03	2.77 ± 0.65	0.58 ± 0.10	1.63 ± 0.76
TDNN	57 691	0.65 ± 0.03	2.24 ± 0.59	0.51 ± 0.08	1.10 ± 0.39
NMCLM (FIR)	58 622	0.67 ± 0.03	2.62 ± 0.53	0.50 ± 0.07	1.23 ± 0.40
NMCLM (gamma)	58 622	0.67 ± 0.02	2.55 ± 0.61	0.47 ± 0.07	0.95 ± 0.40

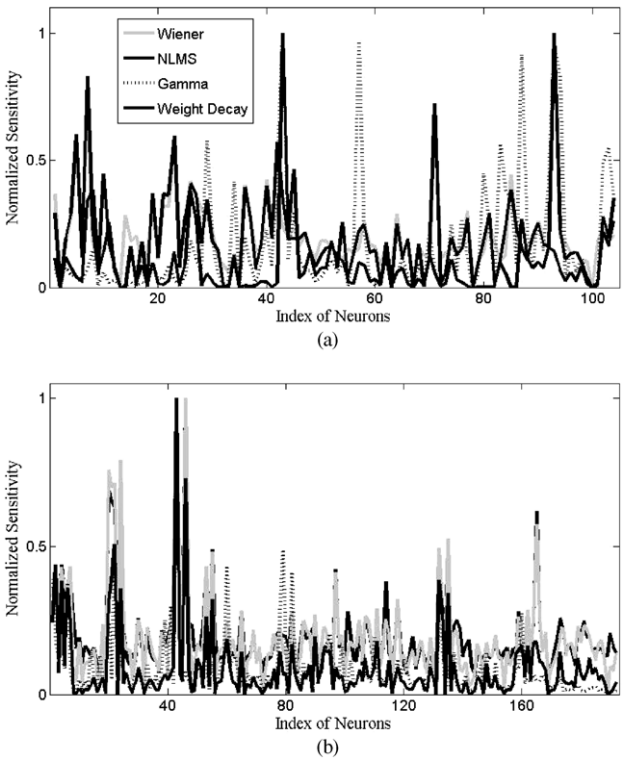

Figure 11. The average sensitivity of the Wiener filter (thin dotted lines), NLMS (thin solid lines), the gamma filter (thick dotted lines) and weight decay (thick solid lines) to neuronal inputs in the BMI data of two tasks: (a) the food reaching task and (b) the target hitting task.

is more accurate than any of the others. Substitution of the gamma filters for the FIR filters also improves the performance further. We also see that the TDNN suffers from poor training, since it has the same number of free parameters as the NMCLM but performs worse than this model.

Table 1 also shows that presenting results for the overall trajectory may be misleading. Indeed, the resting task is much harder to predict than when the animal is moving the arm. This table also shows that the CC and the SER are measuring two slight different things, since the numbers do not vary consistently amongst the models. The target hitting task seems harder to predict than the food reaching task as we can conclude from the CC and SER. It is also interesting to note that the x direction is considerably harder to predict than the y direction for reasons that are not clear at this point.

To quantify more precisely these differences, we test the statistical difference between the Wiener filter and all the other models. We first assume that the average Euclidean distance between the actual and the estimated position in the test data is a sufficient measure of model performance. This measure can be denoted by the error vector $E[|\mathbf{e}|]$.

To compare the performances of different models, we test the difference between the distributions of $E[|\mathbf{e}|]$. The samples of $|\mathbf{e}|$ from non-overlapping time windows with the lengths of 4 s (same as for estimation of CC and the SER) are collected, and $E[|\mathbf{e}|]$ is estimated from samples for each window. A summation used to estimate the mean leads to Gaussian random variables by virtue of the central limit theorem (CLT), by which we can approximately satisfy the normality and independence condition for the application of the t-test to the samples of $E[|\mathbf{e}|]$.

We first define Δ as the difference between $E[|\mathbf{e}|]$ for the Wiener filter and for each of the other models as

$$\Delta(k) = E[|\mathbf{e}|]_M(k) - E[|\mathbf{e}|]_W(k), \tag{25}$$

where $E[|\mathbf{e}|]_M(k)$ denotes the evaluation of $E[|\mathbf{e}|]$ in the kth window for the model under comparison and $E[|\mathbf{e}|]_W(k)$ does

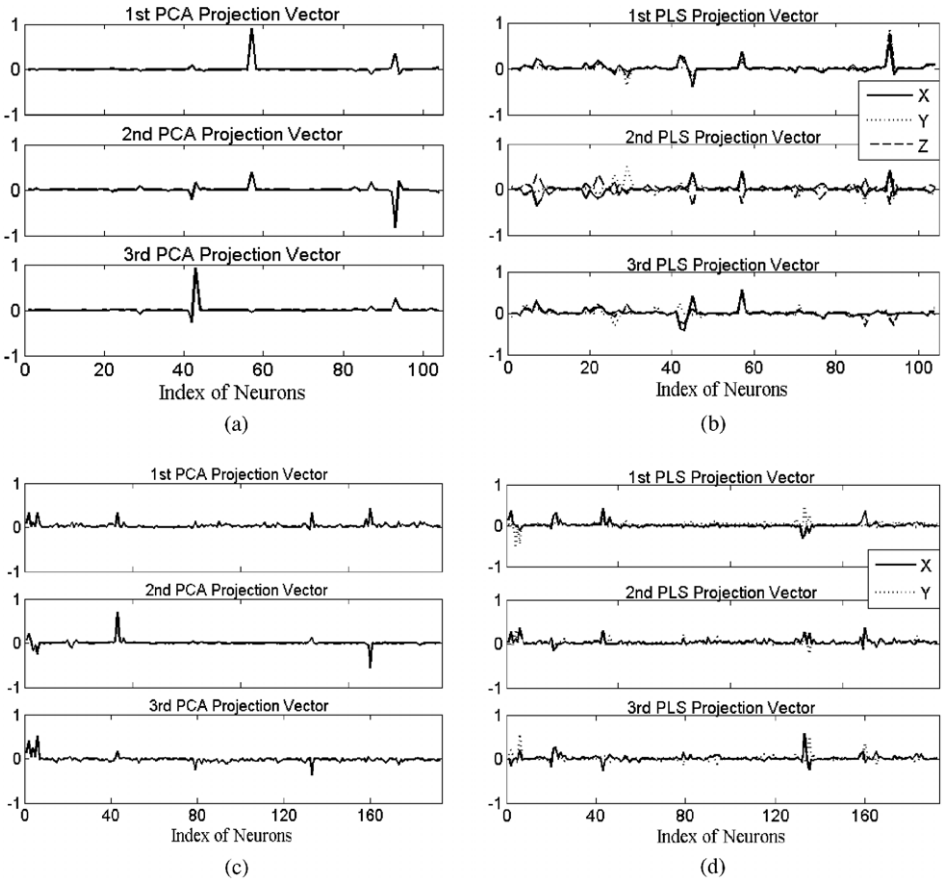

Figure 12. The first three projection vectors in PCA for (a) the food reaching task and (c) the target hitting task, and in PLS for (b) the food reaching, and (d) the target hitting task, respectively. Note that PCA provides single projection vector for all coordinates while PLS separates vectors for each coordinate.

in the same window for the Wiener filter. Then, $\Delta(k)$ is given by

$$\Delta(k) = \frac{1}{K}\sum_{n\in\Omega_k} |\mathbf{e}(n)|_M - \frac{1}{K}\sum_{n\in\Omega_k} |\mathbf{e}(n)|_W = \frac{1}{K}\sum_{n\in\Omega_k} |\mathbf{e}(n)|_M - |\mathbf{e}(n)|_W, \quad (26)$$

where Ω_k indicates the kth sample window and K is a window size (40 samples in the experiment). Note that $|\mathbf{e}(n)|_M$ and $|\mathbf{e}(n)|_W$ are aligned synchronously in time, indicating errors for the same hand position. If we assume that $|\mathbf{e}(n)|_M - |\mathbf{e}(n)|_W$ in (26) realizes an independently and identically distributed (i.i.d.) density function of the difference of $|\mathbf{e}|$ between models, then $\Delta(k)$ approximately follows a Gaussian distribution due to the CLT no matter which model is compared. Hence, we can apply the t-test to the Gaussian random variable Δ that is realized by $\{\Delta(k)\}$ for $k = 1, \ldots, N_k$, where N_k is a number of windows ($N_k = 75$ in experiments). The hypotheses for the one-tail t-test then become

$$H_0: \quad E[\Delta] \geqslant 0, \qquad H_A: \quad E[\Delta] < 0. \quad (27)$$

Given the significance level of α, if the null hypothesis is rejected we can claim with the confidence level of $(1-\alpha)$ that the compared model produces less error on average than the Wiener filter.

Table 3. The t-test results for the performance differences between the Wiener filter and other models.

	Food reaching		Target hitting	
Significance level	0.01	0.05	0.01	0.05
NLMS	1	1	0[a]	0
Gamma	1	1	1	1
Subspace	1	1	1	1
Weight decay	1	1	1	1
Kalman	0	0	0	1
TDNN	0	0	0	0
NMCLM (FIR)	1	1	0	0
NMCLM (gamma)	1	1	0	0

[a] The test result of 0 indicates the acceptance of null hypothesis, while 1 indicates the rejection of null hypothesis.

The test results are presented in table 3. For the food reaching task, the null hypothesis is rejected for all models except TDNN. Note that TDNN shows higher mean SER during rest, but with a relatively large variance. For the target hitting task, however, only four models including the

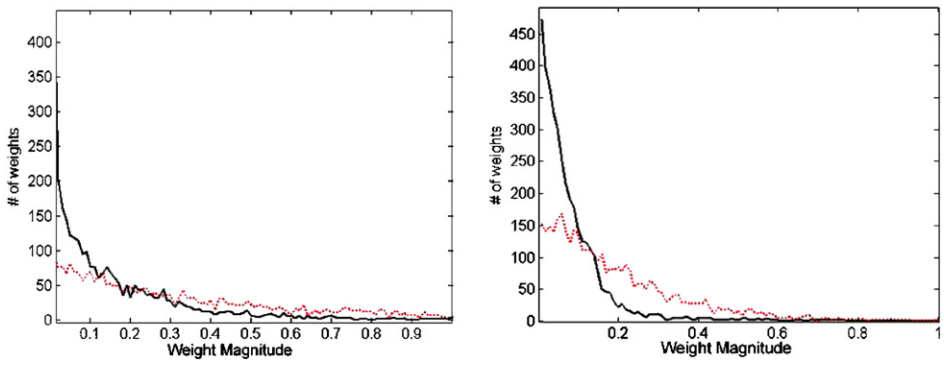

Figure 13. The histogram of the magnitudes of weights over all coordinates of hand positions, trained by weight decay (solid line) and the NMLS (dotted line) for two tasks: (a) food reaching and (b) target hitting.

gamma filter, the subspace Wiener filter, the linear model with weight decay and the Kalman filter, are shown to yield statistically different distribution of $E[|\mathbf{e}|]$. These results are fairly consistent with performance measures in tables 1 and 2.

6.3. Additional performance evaluations

To investigate further the difference between the subspace of PCA and PLS, the first three projection vectors are estimated by setting $\lambda = 0$ or 1 in (13) as presented in figure 12. Note that PLS yields separate vectors corresponding to each hand position coordinate since it utilizes the desired response, while PCA needs only one projection for all coordinates. In the food reaching task, the projection vectors of PCA have large weights on the neurons that fire frequently. For instance, the neurons indexed as 42, 57 and 93 are empirically known to have the largest firing counts. Since the neural firing data are sparse, PCA attempts to build a subspace with frequently firing neurons. On the other hand, the weights in PLS projection have larger values on different neurons which do not fire very frequently such as the neurons indexed as 7 and 23. From the sensitivity analysis studied in [33], these neurons are known to significantly contribute to input–output mapping of BMIs. Therefore, PLS is able to utilize the information from important neurons that do not even fire very frequently by exploiting the joint space. It also explains why the MSE is better in smaller subspace dimensions. For the target hitting task, we can also observe that more neurons are involved in the projection vectors in PLS than PCA. The neurons with larger weights in the PCA projection, again, are observed to fire more frequently. It is interesting to observe that for the target hitting task the subspace dimension obtained from the cross-validation is of the same order as the number of neurons obtained in the neuron dropping analysis performed in [33]. In fact, the number of important neurons for which the correlation coefficient between model outputs and desired hand trajectories is maximized is 35, which is close to the subspace dimension of 44.

In order to demonstrate the effect of weight decay, the histogram of the weight magnitude including all directions of the hand positions is depicted in figure 13. Note that the number of weights that have smaller magnitudes increases with weight decay. For instance, the number of weights that are close to zero is approximately 345 for weight decay versus 75 for NLMS in figure 13(a) and 460 for weight decay versus 150 for NLMS in figure 13(b). It shows that more weights are pruned by weight decay, thus the degree of freedom of the model reduces. Hence, the reduced degree of freedom can help generalization as examined by measuring performance in the test data.

6.4. Comparison with the Kalman filter and population vector algorithm

In tables 1–3, we also present comparisons with the Kalman filer that is a linear model based on a different modeling approach. We can see that the Kalman filter has the smallest number of free parameters among all the models, while its performance is not the best for the food reaching task, but approaches the best result for the target hitting task. We were expecting that the ability to modify the Kalman gain during test to accommodate time-varying fluctuations would result in a better performance, but probably the errors incurred in estimating the large covariance matrix linking the states with the input are the culprit for the timid overall performance. Also, the apparent nonlinearity shown in the kinematics of the food reaching task (e.g. discrimination between movement and rest) might prevent the Kalman filter, in which temporal priors of kinematics are linearly modeled, from estimating kinematics more accurately. However, this bottleneck can be broken to some extent by extending the states to include a discrete state which discriminates between the active and inactive periods of movements. In fact, a recent work of Wood *et al* has reported the increase of decoding performance by inferring this discrete state as well as hand kinematics through a Bayesian model [34].

The last model to be compared is the decoding model proposed by Georgopoluous and called the population vector algorithm (PVA) [35–37]. The PVA approach is built upon experimental evidence that individual neurons in an ensemble modulate their firing rates for movement directions. In this

algorithm, the preferred directions of individual neurons are independently determined, and subsequently weighted by their instantaneous firing rates. The sum of the weighted preferred directions defines the population vector which estimates the current movement direction. This PVA model is given by [36]

$$\mathbf{P}(n) = \sum_{i=1}^{M} \frac{x_i(n) - \bar{x}_i}{C_i} \cdot \frac{\mathbf{B}_i}{C_i}, \tag{28}$$

where the population vector $\mathbf{P}(n)$ is a weighted sum of preferred direction vector $\mathbf{B}_i$ and the weight is determined by each cell's firing activity; $x_i(n)$ is the firing rate of cell i at time instance k, $\bar{x}_i$ is a mean firing rate and C_i is a normalization factor to scale the maximum firing rate to 1. This model determines the movement direction from neural activity, but the reconstruction of hand trajectory also requires the estimation of the speed. Georgopoluous *et al* [35] directly used the magnitude of the population vector, $|\mathbf{P}(n)|$, for estimating the instantaneous speed, and Moran and Schwartz [36] extended the PVA model to include the speed of the hand. Then, the trajectory of the hand position was reconstructed by a time series of population vectors that were connected tip to tail as time increased.

The PVA model extended by Moran and Schwartz was applied to our 2D target reaching data. The evaluation in test data demonstrated that the performance of PVA was far below the Wiener filter (e.g. the correlation coefficient of the PVA was on average 60–70% of the Wiener). These results are similar to those communicated by Wu *et al* [27]. As discussed by these authors, this poor performance of the PVA in the target reaching data is mainly due to the fact that the error can propagate through the temporal integration of the population vectors and the sampled neuronal population may not encode the movement direction uniformly. Schwartz *et al* also indicated in their analysis of decoding algorithms the weakness of the PVA compared to the Wiener filter in terms of the fact that the PVA does not take full advantage of the temporal characteristics of neural firing activity [37]. In fact, the direction-based PVA model may not fit as well as other models for the reconstruction of complicated continuous trajectory including the data presented in this paper. Yet, one must note that the performance comparison presented here can be less significant since the PVA model may not be suitably optimized to our data.

However, in the closed-loop BMIs where an individual can utilize the sensory feedback to control neural prostheses, the PVA can be modified to yield more accurate reconstruction. Recently, Taylor *et al* [1] proposed the adaptive method for the population vector algorithm in the closed-loop system by adjusting the population vector parameters according to task performance. The related studies have shown that the adaptation of the parameters during brain control significantly increased the target acquisition performance [38]. These studies may indicate that there are still a lot of opportunities for the PVA to suit well for more complicated BMI applications in the closed-loop environment.

7. Discussion

Inspired by the performance of Wiener filter algorithms in estimation of movement parameters from the activity of large (∼100 cells) neuronal ensembles, we conducted an extensive comparison study of MIMO filters in BMI design. As test data, we used two datasets, each collected in different monkey experiment, one in a New World monkey reaching for food and one in an Old World monkey hitting a visual target. Although in certain comparisons different models had very similar performance quality, we anticipate that with the development of BMI field and especially with the increase in the number of simultaneously recorded neurons, some of these ideas will find important applications. In the present datasets, all the MIMO filters including the standard Wiener filter performed very well in spite of the large number of degrees of freedom (over 3000 parameters) and the absence of regularization. The major reason for such high performance is the excellent quality of the neuronal recordings and the detection and sorting algorithms. Multielectrode arrays were strategically implanted in cortical areas known to be associated with arm and hand movements. In addition, special care was taken to keep experimental conditions controlled and restricted to specific task requirements. It still remains to be seen how the linear models scale up as the range of motor performances and experimental conditions becomes more complex.

The number of parameters of the linear model was decreased using two different approaches for pruning in time and in the space of the electrodes. In the time dimension, we used gamma delay operators instead of ideal delays to decrease the number of coefficients while spanning the same memory depth (although with a coarser resolution). The gamma model produces statistically better models when compared to the Wiener filter.

Pruning in electrode space is achieved using two different strategies: selecting important channels and using regularization methods to control complexity. The selection of channels with PCA (input neuron information) does not perform well; however, a combination of PCA and PLS that chooses neurons based on their importance in the joint (input and desired signals) space is able to statistically outperform the conventional Wiener filter in both tasks. Likewise, the weight decay regularization also statistically outperforms the Wiener filter. However, the regularization parameter must be appropriately selected in cross-validation, otherwise the performance is very brittle. Therefore, we conclude that the tools of regularization theory are an asset for optimal modeling in BMIs, but the improvements are smaller than expected in spite of being statistically significant. Perhaps, regularization based on different norms may improve performance further. Note that the number of neurons (e.g. 30–40) that were optimally found in subspace projection is relative to the total number of neurons (e.g. 192). Consequently, it may be possible to obtain a larger subset of optimal neurons if the total number of neurons increases. We also expect higher improvements using the subspace projection methods for larger datasets (more than

200 neurons; Carmena J M, Lebedev M A and Nicolelis M A L, unpublished observations) and anticipate that these techniques will be important in the foreseeable future when the number of simultaneously recorded neurons surpasses 1000.

The nonlinear model showed better performance for one dataset, but not for the other. Nonlinear models significantly outperformed the linear counterparts for the food reaching task, mostly due to their ability to follow better the non-movement (hand at rest) portions of the desired response. This is due to their ability to 'shut-off' parts of the network by virtue of nonlinearity. However, in the target hitting task where the hand is almost always moving, the performance was very similar, being statistically indistinguishable from the Wiener filter. Given the complexity of brain networks and no *a priori* reason for them to have linear properties, this was unexpected and may reflect the fact that it is harder to train nonlinear models to the same specification of the linear ones. Or simply that due to the large input space of BMIs finding a linear projection space of reduced dimension (2D or 3D) is sufficient when performance is the only metric. In addition, one would expect a better performance in a nonlinear model when it matches in some ways the performance of the real brain network, otherwise it would falter. Linear models on the other hand already incorporate well-known properties of cortical neurons, such as directional tuning (typically described by a cosine function), sensitive to position, velocity and force.

We speculate that some nonlinear topologies may have practical advantages when BMIs are implemented in real-time digital signal processors. The work reported in [39] shows that when memory constraints and clock cycles are taken into consideration a recursive MLP (RMLP) requires a smaller computation bandwidth and resources than the FIR filter trained with NLMS. However, training of the RMLP is still more complex than the NLMS algorithm, so further work to find nonlinear topologies that train faster should be sought. In terms of the regularization techniques, the gamma model and the weight decay can easily be implemented in DSPs, but the subspace Wiener filters require a substantial increase in computation. Therefore, further work to simplify these algorithms should also be pursued. In terms of deployment, a BMI with 100 channels to predict 2D or 3D hand trajectories based on the regularized NLMS filter can be implemented in real time in a small Texas Instruments C33 WiFi board recently developed by our group.

A comment regarding prediction performance of these algorithms in terms of correlation coefficient (CC) is in order. The CC of all these algorithms is capped at 0.8 for the food reaching tasks and 0.7 for the target hitting tasks. It is important to investigate if this limit is related to missing data (only a tiny percentage of the motor cortex neurons are probed) or if it is the intrinsic spatio-temporal nonstationarity of the data that is not properly captured by this class of models that learn based on stationarity assumptions. Another important issue that should not be forgotten in the design of better BMIs is how to effectively include neurophysiology knowledge both in the filter topologies and in the cost functions.

Once the models are fitted to the data, there is a wealth of information that can be gained from analyzing the filter coefficients. This is an area of research that is only now starting and can provide a new window to understand population coding during behavior.

Acknowledgments

This work is supported by DARPA project # N66001-02-C-8022. JMC is supported by the Christopher Reeve Paralysis Foundation.

References

[1] Taylor D M, Helms Tillery S I and Schwartz A B 2002 *Science* **296** 1829–32
[2] Chapin J K, Markowitz M A, Moxon R S and Nicolelis M A L 1999 *Nat. Neurosci.* **2** 664–70
[3] Wessberg J, Stambaugh C R, Kralik J D, Beck P D, Laubach M, Chapin J K, Kim J, Biggs S J, Srinivasan M A and Nicolelis M A L 2000 *Nature* **408** 361–5
[4] Serruya M D, Hatsopoulos N G, Paninski L, Fellows M R and Donoghue J P 2002 *Nature* **416** 141–2
[5] Kennedy P R, Bakay R A E, Moore M M, Adams K and Goldwaithe J 2000 *IEEE Trans. Rehabil. Eng.* **8** 198–202
[6] Taylor D M, Helms Tillery S I and Schwartz A B 2003 *IEEE Trans. Neural Syst. Rehabil. Eng.* **11** 195–9
[7] Carmena J M, Lebedev M A, Crist R E, O'Doherty J E, Santucci D M, Dimitrov D F, Patil P G, Henriquez C S and Nicolelis M A L 2003 *PLoS Biol.* **1** 193–208
[8] Wu W, Black M J, Gao Y, Bienenstock E, Serruya M D, Shaikhouni A and Donoghue J P 2003 *Neural Decoding of Cursor Motion Using a Kalman Filter (Adv. NIPS* vol 15*)* ed S Becker, S Thrun and K Obermayer (Cambridge, MA: MIT Press) pp 133–40
[9] Gage G J, Ludwig K A, Otto K J, Ionides E L and Kipke D R 2005 *J. Neural Eng.* **2** 52–63
[10] Musallam S, Corneil B D, Greger B, Scherberger H and Andersen R A 2004 *Science* **305** 258–62
[11] Nicolelis M A L, Dimitrov D, Carmena J M, Crist R E, Lehew G, Kralik J D and Wise S P 2003 *Proc. Natl Acad. Sci. USA* **100** 11041–6
[12] Haykin S 1996 *Adaptive Filter Theory* (Upper Saddle River, NJ: Prentice-Hall) pp 194–437
[13] Haykin S 1996 *Neural Networks: A Comprehensive Foundation* (New York: Macmillan) pp 156–438
[14] de Jong S 1993 *Chemometr. Intell. Lab. Syst.* **18** 251–63
[15] Principe J C, de Vries B and de Oliveira P G 1993 *IEEE Trans. Signal Process.* **41** 649–56
[16] Kim S P, Sanchez J C, Erdogmus D, Rao Y N, Wessberg J, Principe J C and Nicolelis M A L 2003 *Neural Netw.* **16** 865–71
[17] Lewicki M 1998 *Network* **9** R53–78
[18] Wood F, Black M J, Vargas-Irwin C, Fellows M and Donoghue J P 2004 *IEEE Trans. Biomed. Eng.* **51** 912–8
[19] Gentle J E 1998 *Numerical Linear Algebra for Applications in Statistics* (Berlin: Springer) pp 93–5
[20] Hoerl A E and Kennard R W 1970 *Technometrics* **12** 55–67
[21] Stone M and Brooks R J 1990 *J. R. Stat. Soc.* B **52** 237–69
[22] Kim S P, Rao Y N, Erdogmus D and Principe J C 2003 *Proc. IEEE Int. Conf. on Acoustic Speech and Signal Processing (Hong Kong, April 2003)* vol 6 pp 312–14
[23] Bishop C M 1995 *Neural Networks for Pattern Recognition* (Oxford: Oxford University Press) pp 371–9
[24] Neal R 1996 *Bayesian Learning for Neural Networks* (Cambridge: Cambridge University Press) pp 29–50

[25] Larsen J, Svarer C, Andersen L N and Hansen L K 1998 Adaptive regularization in neural network modeling *Neural Networks: Tricks of the Trade (Lecture Notes in Computer Science* vol 1524*)* ed G B Orr and K R Muller (Berlin: Springer) pp 113–32
[26] Geisser S 1975 *J. Am. Stat. Assoc.* **50** 320–8
[27] Wu W, Gao Y, Bienenstock E, Donoghue J P and Black M J 2005 *Neural Comput.* **18** 80–118
[28] Gao Y, Black M J, Bienenstock E, Shoham S and Donoghue J P 2002 *Probabilistic Inference of Arm Motion From Neural Activity in Motor Cortex (Adv. NIPS* vol 14*)* ed T G Dietterich, S Becker and Z Ghahramani (Cambridge, MA: MIT Press) pp 221–8
[29] Fancourt C L and Principe J C 1997 *Proc. IEEE Int. Conf. on Acoustic Speech and Signal Processing (Atlanta, GA, May 1996)* vol 4 pp 3325–8
[30] Jacobs R A, Jordan M I, Nowlan S J and Hinton G E 1991 *Neural Comput.* **3** 79–87
[31] Cover T and Thomas J 1991 *Elements of Information Theory* (New York: Wiley) pp 12–49
[32] Sanchez J C, Erdogmus D, Rao Y N, Principe J C, Nicolelis M A L and Wessberg J 2003 *Proc. IEEE EMBS Neural Engineering Conference (Capri Island, Italy, March 2003)* pp 59–62
[33] Sanchez J C, Carmena J M, Lebedev M A, Nicolelis M A L, Harris J G and Principe J C 2004 *IEEE Trans. Biomed. Eng.* **51** 943–53
[34] Wood F, Prabhat, Donoghue J P and Black M J 2005 *Proc. IEEE (Shanghai, China, September 2005)* pp 1544–7
[35] Georgopoluous A P, Kettner R E and Schwartz A B 1988 *J. Neurosci.* **8** 2928–37
[36] Moran D W and Schwartz A B 1999 *J. Neurophysiol.* **82** 2676–92
[37] Schwartz A B, Taylor D M and Helms Tillery S I 2001 *Curr. Opin. Neurobiol.* **11** 701–7
[38] Helms Tillery S I, Taylor D M and Schwartz A B 2003 *Rev. Neurosci.* **14** 107–19
[39] Sanchez J C 2004 From cortical neural spike trains to behavior: modeling and analysis *PhD Dissertation* University of Florida, Gainesville, FL