Motor Cortical Decoding Using an Autoregressive Moving Average Model

Jessica Fisher
Michael Black, advisor

March 3, 2006

Abstract

We develop an Autoregressive Moving Average (ARMA) model for decoding hand motion from neural firing data and provide a simple method for estimating the parameters of the model. Results show that this method produces comparable reconstructions of hand position to the previous Kalman filter and linear regression methods. The ARMA model combines the best properties of both these methods, producing reconstructed hand trajectories that are smooth and accurate. This simple technique is computationally efficient making it appropriate for real-time prosthetic control tasks.

1 Introduction

The advancement of computer technology has led to the emergence of many new interdisciplinary fields. In particular, computer science has recently been applied to a variety of problems in medicine. Among these is the problem of restoring mobility to amputees or people who have been paralyzed. One approach to this is the creation of neural motor prostheses. Essentially, one would like to place an implant in the patient's brain, and by decoding signals detected from this implant, control an artificial limb or functional electrical stimulation of the patient's existing limb. This problem is interdisciplinary in nature, as it involves medicine, engineering, neuroscience, and computer science, as well as many other fields.

One of the primary problems in the development of practical neural motor prostheses is the formulation of accurate, efficient methods for decoding neural signals. The electrical activity of the brain is recorded in terms of discrete action potentials ("spikes"). The number of spikes a neuron produces in a particular time interval is counted, and this number is called the "firing rate." Here we focus on the problem of reconstructing the trajectory of a primate hand given the extracellularly recorded firing rates of a population of neurons in the arm area of the primary motor cortex (M1).

Various machine learning techniques, mathematical models, and decoding algorithms have been applied to this problem [1–7]. The simplest and most common of these is the linear regression method which represents hand position at a given time instant as a linear combination of population firing rates over some preceding time interval [2, 3, 6]. While simple and relatively accurate, the resulting reconstructions require post hoc smoothing to be practical in a neural prosthesis [8].

Alternatively, Bayesian decoding methods have been used, including the Kalman filter [4] and particle filter [5, 7]. In contrast to the linear regression method, Bayesian methods include an explicit temporal smoothness term that models the prior probability of hand motion. The Kalman filter is particularly appropriate for prosthetic applications given its accuracy and efficiency (for both training and decoding) [8]. Unlike the linear regression method which uses a large history of firing data to reconstruct hand motion at every time instant, conditional independence assumptions in the standard Kalman filter restrict it to using only the current firing rates of the neurons. While the hand trajectories decoded with the Kalman filter do not require post hoc smoothing, they still lack the smoothness of natural hand motion [8].

Here we develop a simple Autoregressive Moving Average (ARMA) model for the neural decoding task that combines the linear regression method with the smoothness term of the Kalman filter. By using more data at each time instant than the Kalman filter, accuracy is improved, and by adding a spatial smoothing term to the linear regression method, smooth trajectories are obtained without post hoc smoothing.

An ARMA model was suggested by [1] in the context of decoding a center-out reaching task. They extended the method to model a non-linear relationship between firing rates and hand motions using Support Vector Regression. Here we apply the simpler ARMA model to a more complex task involving arbitrary 2D hand motions. We show that a very simple algorithm suffices to estimate the parameters of the ARMA process, and that the resulting decoding method results in reconstructions that are highly correlated to the true hand trajectory. We explore the choice of parameters and provide a quantitative comparison between the ARMA process, linear regression, and the Kalman filter, expanding on the results reported in [9]. We found that the simple ARMA process provides smooth reconstructions that are typically more accurate than linear regression and sometimes more accurate than the Kalman filter.

2 Methods

2.1 Recording

Our experiments here use two sets of previously recorded and reported data [10] in which a Bionic Technologies LLC (BTL) 100-electrode silicon array [11] was implanted in the arm area of the primary motor cortex of a *macaca mulatta* monkey. The monkey was trained to move a two-joint planar manipulandum to control a feedback cursor on a computer screen. The position of the manipulandum and the neural activity were recorded simultaneously, and the neural activity was summed into 70-ms bins for data set A, and 40-ms bins for data set B. The task the monkey performed was a "pinball" task [4] in which he moved the cursor to hit a target on the screen, and when he succeeded, a new target appeared. Neural signals were detected on 42 electrodes for data set A and 50 electrodes for data set B, and a simple thresholding operation was used to detect action potentials. The spikes on each electrode were treated as one (potentially) multi-unit channel of data. In [12] it was found that multi-unit data provided decoding accuracy on a par with the best single unit data obtained by human spike sorting. The data sets were divided into separate training and testing sets using held-out data, in which the testing set ranged from approximately 8 to 15 percent of the total length of the data set. For data set B, the entire data set encompassed 200 seconds of recording; data set A encompassed 370 seconds of recording.

2.2 Linear Regression

The linear regression method is the most common method used for motor cortical decoding and assumes that the current hand state (position, velocity, and acceleration) can be represented as a linear combination of the current firing rates of a population of neurons. Least-squares regression is performed to determine the coefficients (the "filter") for this linear combination based on training data, and the filter is then used on testing data to decode the state at each time instant [2, 6]. The linear relationship is described by

$$\mathbf{X} = \mathbf{ZF} \tag{1}$$

where $\mathbf{X}$ is a state matrix containing the (x_t, y_t) hand positions at time instants $t = 1 \ldots T$, $\mathbf{Z}$ is a matrix of firing rates of the $N = 42$ or 50 multi-units over the same time period (for data sets A and B, respectively), and $\mathbf{F}$ is the linear filter matrix relating hand positions and firing activity. In particular,

$$\mathbf{X} = \begin{pmatrix} x_T & y_T \\ x_{T-1} & y_{T-1} \\ \vdots & \vdots \\ x_1 & y_1 \end{pmatrix},$$

$$\mathbf{Z} = \begin{pmatrix} z_T^1 & \ldots & z_{T-k}^1 & z_T^2 & \ldots & z_{T-k}^n & 1 \\ z_{T-1}^1 & \ldots & z_{T-k-1}^1 & z_{T-1}^2 & \ldots & z_{T-k-1}^n & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z_k^1 & \ldots & z_1^1 & z_k^2 & \ldots & z_1^n & 1 \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} fx_1 & fy_1 \\ fx_2 & fy_2 \\ \vdots & \vdots \\ fx_{kN} & fy_{kN} \\ f_1 & f_2 \end{pmatrix}$$

where x_t represents the hand x position at time t (analogous for y), z_t^i represents the firing rate of neuron i at time t, fx_p represents the p^{th} filter coefficient for x (same for y), and the f_p's represent constant offset terms. Note that k is a constant representing a time window of neural firing rates to be considered. Also, the column of ones in $\mathbf{X}$ provides a constant bias term. This model can easily be expanded to include velocity and acceleration.

We solve for $\mathbf{F}$ by minimizing the squared error $\|\mathbf{X} - \mathbf{ZF}\|_2^2$ which gives the solution for the filter matrix $\mathbf{F}$ as

$$\mathbf{F} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{X} = \mathbf{Z}^+\mathbf{X}$$

where $\mathbf{Z}^+$ is the pseudo-inverse of $\mathbf{Z}$. The hand position $\mathbf{x}_t = (x_t, y_t)$ at a particular time instant can be reconstructed as

$$\mathbf{x}_t = \mathbf{z}_{t-k}^{t}\mathbf{F}$$

where $\mathbf{z}_{t-k}^{t} \in \Re^{1\times Nk}$ is a vector representing a k time bin history of neural firing preceding time instant t (a row of the $\mathbf{Z}$ matrix above).

2.3 Autoregressive Moving Average (ARMA) Model

The linear filter models how hand position is related to neural activity over some time interval, but it does not explicitly model anything about how hands move. The motion of the hand is constrained by physics and the properties of the body, and therefore evolves smoothly over time. Consequently we add an additional term to the linear regression method to model this smoothness assumption; that is, the current state is represented as a linear combination of a history of firing rates and the preceding hand states, given by

$$\mathbf{x_t} = \mathbf{A}\mathbf{x}_{t-m}^{t-1} + \mathbf{F}\mathbf{z}_{t-k}^{t} \tag{2}$$

where $\mathbf{A} \in \Re^{D\times Dm}$ (D is the dimensionality of the state vector $\mathbf{x}$) and $\mathbf{F} \in \Re^{D\times Nk}$, and m and k are parameters determining how many previous time steps (of hand state and neural firing respectively) to include in the calculation. $\mathbf{x}_{t-m}^{t-1} \in \Re^{Dm\times 1}$ is a column vector containing the concatenation of the states from times $t-m$ to $t-1$, and $\mathbf{z}_{t-k}^{t} \in \Re^{Nk\times 1}$ is a column vector containing the concatenation of the firing rates for all neurons from times $t-k$ to t. We call this an Autoregressive Moving Average (ARMA)(m, k) model [13].

The parameters to be estimated in this model are $\mathbf{A}$ and $\mathbf{F}$. We alternate learning these (beginning with $\mathbf{F}$) using the same least squares minimization as linear regression:

$$\mathbf{F} = (\mathbf{Z}^T)^+(\mathbf{X}_2 - \mathbf{A}\mathbf{X}_1) \tag{3}$$

$$\mathbf{A} = (\mathbf{X}_1^T)^+(\mathbf{X}_2 - \mathbf{F}\mathbf{Z}) \tag{4}$$

where $\mathbf{X}_2 \in \Re^{D\times T-1}$ is a matrix of states from times 2 to T (T being the total number of time steps), $\mathbf{Z} \in \Re^{Nk\times T-1}$ is a firing rate matrix where each column is of the form $\mathbf{z}_{t-k}^{t}$ as described above, and $\mathbf{X}_1 \in \Re^{Dm\times T-1}$ is a matrix in which each column is of the form $\mathbf{x}_{t-m}^{t-1}$ as described above.

Initially, we set $\mathbf{A}$ to a matrix of zeros and learn $\mathbf{F}$ (note that this is simply the standard linear regression method). To determine when convergence has occurred, we use the mean squared error of the training data [4]. Convergence has occurred when the difference between the previous iteration's error and the current error is less than some parameter ϵ. In our experiments, we varied ϵ between 0.001 and 1000.

2.4 Kalman Filter

The Kalman filter has been proposed for decoding motor cortical data [4]. Like the above methods it assumes a linear relationship between neural firing and hand kinematics (though formulated as a generative model) and like the ARMA model assumes the hand motion evolves linearly, as follows:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w} \tag{5}$$

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{q} \tag{6}$$

where $\mathbf{w} \sim N(0, \mathbf{W})$ and $\mathbf{q} \sim N(0, \mathbf{Q})$. A recursive Bayesian method is used to predict $\mathbf{x}_t$ given $\mathbf{z}_t$. While the reader is referred to [4] for details, it is worth noting that the Kalman filter reconstructs hand kinematics as

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}(\mathbf{A}\mathbf{x}_{t-1}))$$

were $\mathbf{K}_t$ is the Kalman gain matrix. Note that the first term is the same as in the ARMA model but is restricted to using only the previous time instant due to a first order Markov assumption used to derive the filter. Note also that the second term is a linear function of the difference between the firing rates and the predicted firing rates, $\mathbf{H}\mathbf{A}\mathbf{x}_{t-1}$. In contrast, the ARMA model uses a linear function of the firing rates themselves.

2.5 Lag

The introduction of lag has been shown to improve the results of the Kalman filter [4]. To implement a lag, we shift the data so that $\mathbf{x}_t$ corresponds to $\mathbf{z}_{t-\ell}$ for some lag ℓ, rather than $\mathbf{z}_t$. We ran experiments both with no lag ($\ell = 0$), and with a constant lag of 2 time bins (140ms for data set A, 80ms for data set B), which has been shown to provide good results for the Kalman filter [4].

Method	Data Set A CC X	Data Set A CC Y	Data Set B CC X	Data Set B CC Y
Linear Regression	0.6647 ± 0.0197	0.8131 ± 0.0239	0.7395 ± 0.0026	0.7988 ± 0.0082
Kalman Filter	0.7086 ± 0.0214	0.8646 ± 0.0053	0.9146 ± 0.00109	0.9070 ± 0.0014
ARMA Model	0.6895 ± 0.0345	0.8762 ± 0.0095	0.8027 ± 0.0064	0.7772 ± 0.0133

Table 1: Comparison of mean and variance of correlation coefficients for X and Y position among different decoding methods, both data sets.

2.6 Statistical Analysis

Two different types of statistics were calculated with respect to the results of each decoding method: the mean squared error and the correlation coefficients. The mean squared error is defined as

$$MSE_d = \frac{\sum_{i=1}^{T}(x_d^i - \hat{x}_d^i)^2}{T} \tag{7}$$

where x_d^i is dimension d of the true state at time i, and $\hat{x}_d^i$ is dimension d of the estimated state at time i. The correlation coefficient is defined as follows:

$$CC_d = \frac{\sum_t (x_d^t - \bar{x}_d)(\hat{x}_d^t - \bar{\hat{x}}_d)}{\sqrt{\sum_t (x_d^t - \bar{x}_d)^2 \sum_t (\hat{x}_d^t - \bar{\hat{x}}_d)^2}} \tag{8}$$

where $\bar{x_d}$ represents the mean of dimension d of the state, and $\bar{\hat{x}}_d$ is that of the estimated state.

In order to compare the methods, for each data set, each method was run ten times using different held out data each time. The mean and variance of the resulting 10 values for the mean squared error and correlation coefficients were then calculated and compared for each method.

3 Results

Table 1 shows a comparison of the correlation coefficients for each decoding method on both data sets. These correlation coefficients and the mean squared error results are shown graphically in figure 5. For linear regression and ARMA, a time history of five time bins was used. Note that this may not be the optimal number of time bins for each set of data. In all cases, position, velocity, and acceleration were included in the state vector. The values shown are the mean values obtained from 10 trials using different held

ϵ	Training CC	Testing CC
0.1	1.9994	1.4021
1	1.9994	1.4216
10	1.9989	1.4428
100	1.9938	1.4230
1000	1.9405	1.3421

Table 2: Effect of varying the convergence parameter ϵ on data set B. Correlation coefficients reported are the sum of X and Y coefficients.

Method	Lag	MSE X	MSE Y	CC X	CC Y
Linear Regression	0	5.394	1.857	0.769	0.901
(13 history bins)	2	5.398	1.861	0.769	0.901
Kalman Filter	0	5.788	1.903	0.726	0.902
	2	4.281	1.806	0.804	0.914
ARMA Model	0	3.388	1.433	0.820	0.923
(7 history bins)	2	3.364	1.507	0.825	0.926

Table 3: Effects of constant lag of 2 time bins on the three models, data set A

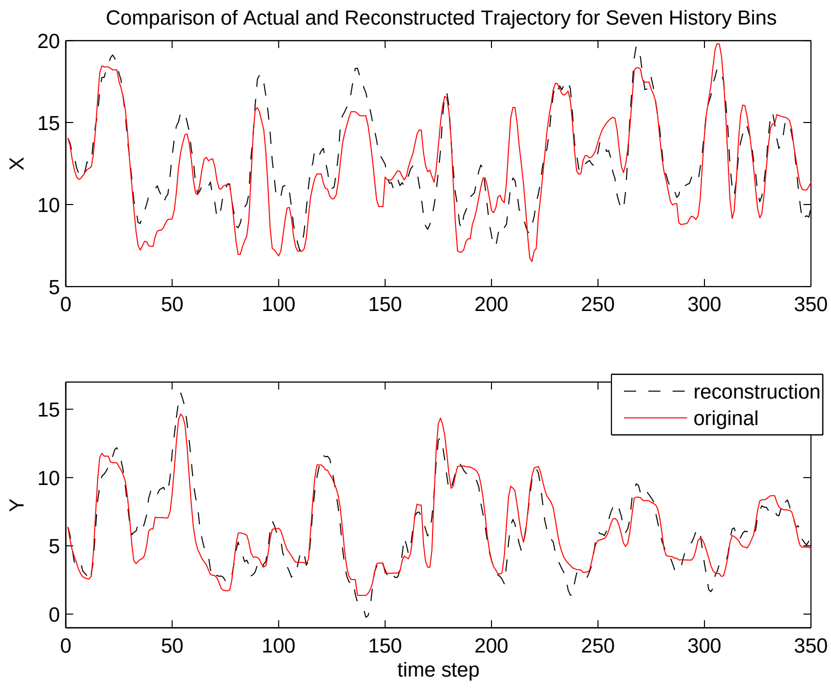


Figure 1: Comparison of actual and reconstructed trajectory using the ARMA(1, 7) model on data set A. The solid line shows the actual trajectory and the dashed line shows the reconstructed trajectory. Seven bins of history in the firing rate were used, and the state was a six-dimensional vector containing position, velocity, and acceleration in x and y. The convergence parameter ϵ was 0.001.

out data each time, and the corresponding variance. A constant lag of 2 time bins was used for linear regression and the Kalman filter.

For data set A, in the Y dimension the ARMA model was best (had the highest correlation coefficient), followed by the Kalman filter, and finally linear regression. However, in the X dimension, there was no significant difference between the ARMA model and the other two methods, although the Kalman filter was significantly better than linear regression. On data set B, the Kalman filter performed better than the other methods in both the X and Y dimensions. In the X dimension, the ARMA model performed better than linear regression, but in the Y dimension linear regression performed better than the ARMA model. There were no significant differences in the mean squared error.

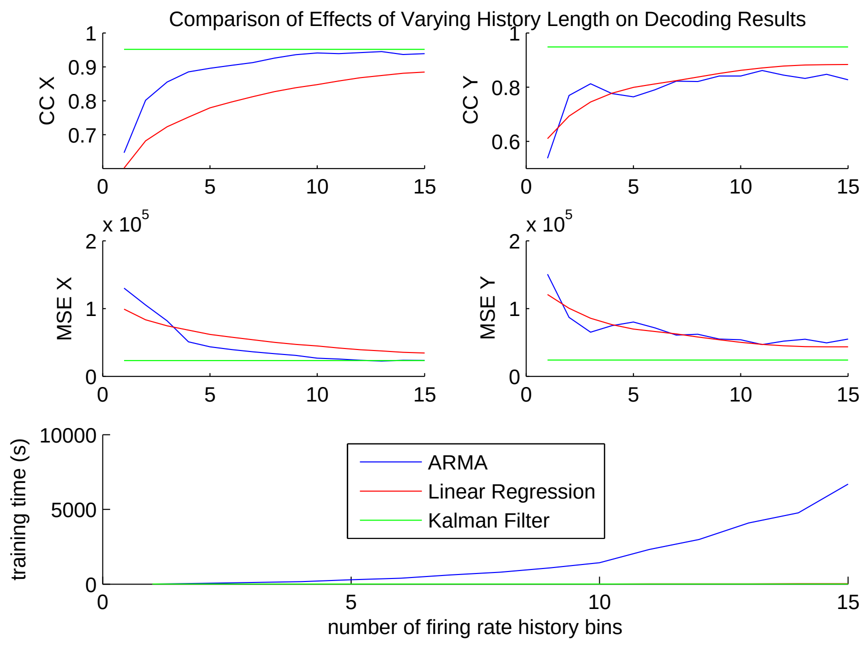


Figure 2: Comparison of the effects of history length on decoding results for all three methods, data set B. 184s of training data and 16s of test data were used. For ARMA, $\epsilon = 10$. The top two plots show correlation coefficients in the X and Y dimensions, the middle two show mean squared error in the same dimensions, and the bottom plot shows training time. Note that the Kalman filter can only use a single history bin and thus is a horizontal line in all plots.

Fig. 1 shows a comparison of the actual and reconstructed trajectories for the ARMA method on data set A. Fig. 4(a) shows the mean squared error with respect to the number of history bins included in the firing rate, varied from one to twenty, and Fig. 4(b) shows the correlation coefficients in the same case. The lowest mean squared error and highest correlation coefficients appeared at seven bins of firing rate history, in this case.

Figure 2 shows the same information for data set B. In this case, the number of history bins was varied from 1 to 15 for both the ARMA model and linear regression, and these are compared to the Kalman filter. The top two plots show the results for correlation coeffecients, the middle two show the mean squared error, and the bottom plot shows the training time versus the number of history bins. Both the ARMA method and linear regression

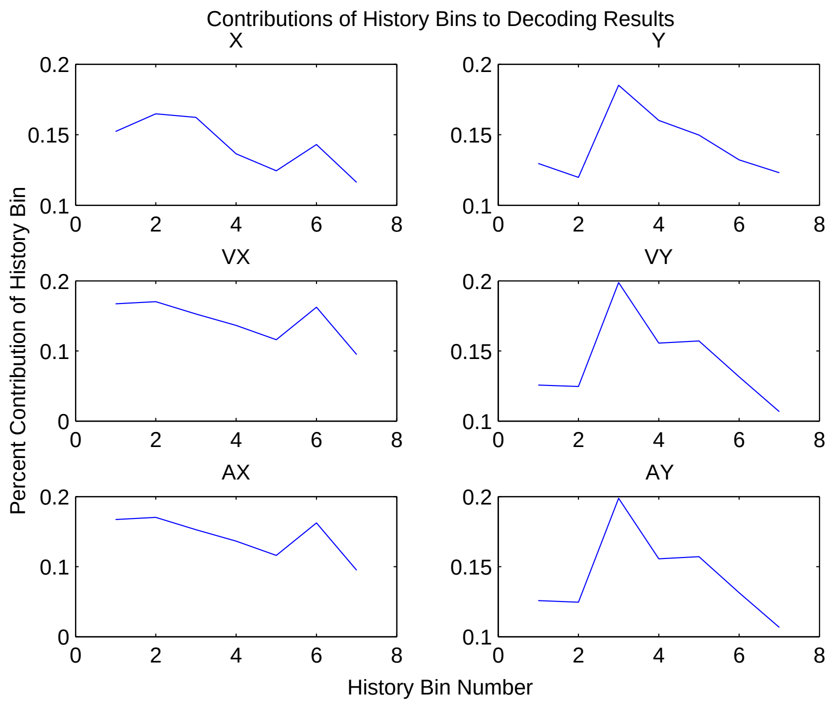
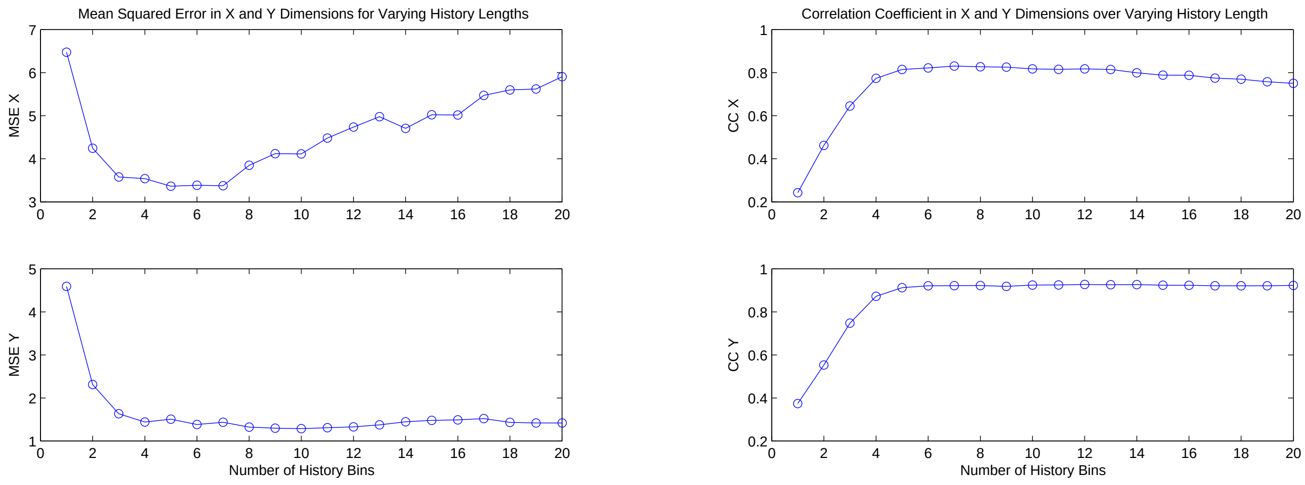


Figure 3: Percent contribution of each history bin to the ARMA decoding result for all six hand state parameters (X and Y positions, velocities, and accelerations). 7 history bins were used with $\epsilon = 10$ on data set B.

approach the Kalman filter in both correlation coeffecients and mean squared error, and in the X dimension, they in fact produce a lower mean squared error than the Kalman filter. However, the time plot indicates the high cost of a large number of time bins to the ARMA model.

Figure 3 shows the contribution of each bin of history to the reconstructed value for the ARMA model. The filter analyzed here was generated on data set B using ARMA(1,7) with $\epsilon = 10$. It appears that in general for the X-related parameters (position, velocity, and acceleration) the first two bins are the most important and the contributions of bins 3 to 7 decline, with an anomalous higher contribution from bin 6. In the Y dimensions, the third bin appears to have the highest contribution, with a declining trend in bins 4 through 7.

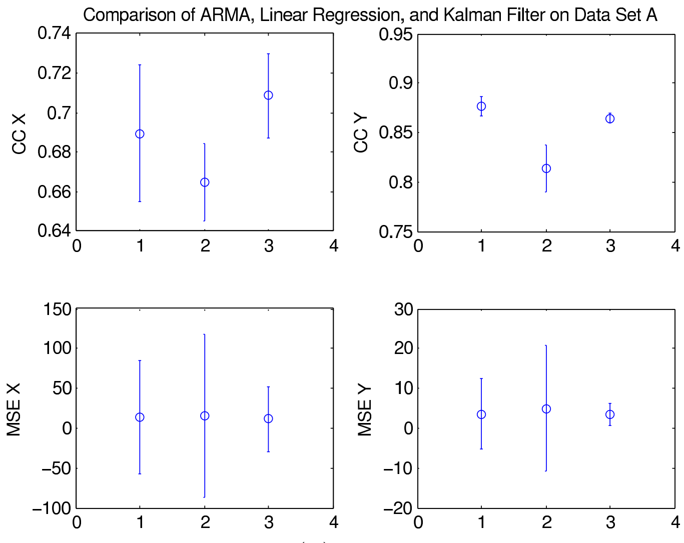
When history was added in the state term (i.e., ARMA(m, k) where $m > 1$), the error increased and the correlation coefficient decreased. We believe



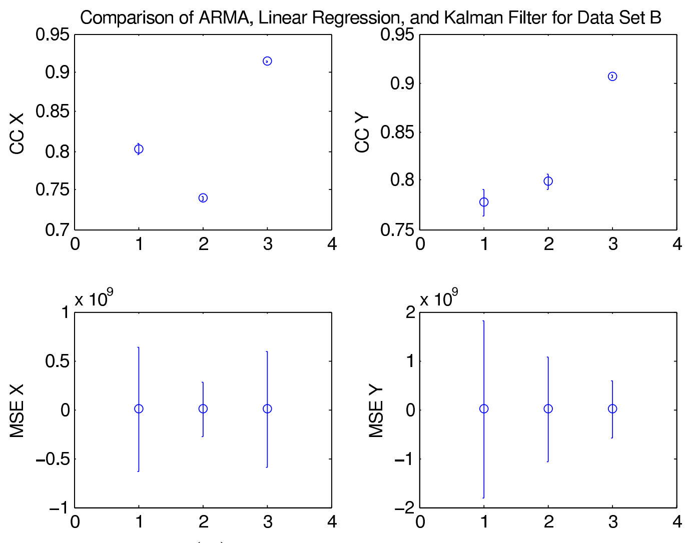
(a) Mean Squared Error

(b) Correlation Coefficient

Figure 4: Data Set A. The number of history bins in the firing rate term was varied from one to twenty, and the mean squared error and correlation coefficients between the reconstruction and the true trajectory were calculated in x and y. In both cases, the top graph shows the x dimension and the bottom the y dimension. (a) shows the mean squared error, and (b) shows the correlation coefficients.



(a) Data Set A



(b) Data Set B

Figure 5: Results of statistical analysis comparing the three methods. Each method was run 10 times. The points on the plots represent the mean result of these runs, and the error bars represent the variance. For each plot, the points represent ARMA, linear regression, and the Kalman filter from left to right. The top two plots show the correlation coefficient results and the bottom two show the mean squared error results. (a) and (b) show the results for data sets A and B, respectively.

this to be due to over-fitting, since the error on the training data was very small, while the error on the testing data increased dramatically. A constraint on the norm of $\mathbf{A}$ might eliminate this problem. Overfitting was also a problem on data set B, even without using a state history larger than one bin. Table 3 shows the correlation coeffecients on the training and test data for values ϵ from 0.01 to 1000. The correlation coeffecient on the training data is surprizingly high, and larger ϵ values seem to non-intuitively produce better results in testing.

We compared the mean squared error and correlation coefficients for the ARMA model with and without lag, and found that they were effectively the same. Table 3 shows the effect of lag on ARMA and the other models on data set A. The lag seems to make little difference in the linear regression method as well as the ARMA process, although it greatly improves the performance of the Kalman filter. The Kalman filter has been shown to be further improved by implementing non-uniform lag (so that each neuron has its own lag) [4], but we do not consider that model here.

Additionally, we investigated the state vector. In particular, we tried decoding with ARMA using only the position, as opposed to the position, velocity, and acceleration. This provided mixed results; the correlation coefficient was lower without the extra terms (0.823 in x and 0.923 in y), but the mean squared error was slightly less (3.313 in x and 1.457 in y), suggesting that the extra data may not be necessary.

Finally, we examined the training time required for each method. Training the Kalman filter is insignificant, while the linear filter and the ARMA process both are highly dependent on the number of history bins used. The ARMA process is also dependent on the choice of the convergence parameter ϵ, in that the smaller ϵ is, the longer it takes to converge. We found for data set A that for 13 history bins, the linear filter took approximately 20 seconds to train, while for seven history bins and $\epsilon = 0.001$, the ARMA process took approximately six minutes. Setting ϵ equal to 0.01 cuts the training time in half, while still providing good results (MSE 3.865 in x and 1.469 in y, CC 0.796 in x and 0.921 in y). Although this training is significantly longer than that of the other methods, it is still feasible, especially since decoding incoming data is essentially instantaneous once training is complete (0.1ms to decode a single time instant, as opposed to 0.7ms for the Kalman filter). However, for data set B, training took much longer, as shown in Figure 2. Since increasing ϵ provided better results in this case (due to overfitting), the training time can be easily reduced using higher ϵ.

4 Discussion

We found that the ARMA model provided comparable neural decoding results (mean squared error and correlation coefficients) to the linear regression method and the Kalman filter. In some cases, the ARMA model provided better results than either method, but statistical analysis indicates that this is not always the case. The ARMA method is, however, typically more accurate than linear regression in terms of correlation coefficients. These results suggest that the advantage of the ARMA model over the linear filter is that the model includes information about the previous state as well as neural firing. The Kalman filter also uses information about the motion of the hand and the firing rate, but in common usage it is constrained by a first-order Markov assumption, so that neural data from only a single time instant is considered at each time step. The statistical analysis suggests that the Kalman filter is likely the most reliably good option to use. However, the simplicity of the ARMA method makes it attractive as an option for neuroscientists.

The optimization method used to estimate parameters for the ARMA process is simple and fast for small histories and larger values of the convergence parameter ϵ. The training time increases as the number of history bins increases and as the convergence parameter is reduced. However, once the model is trained, decoding can be performed on-line.

A restriction of the ARMA process is its linearity. By changing the linear relationship with the firing rate to a nonlinear function or a kernel (as in [1]), better results may yet be achieved. Additionally, the Kalman filter with non-uniform lag has been shown to provide results similar to or better than those of the ARMA process [4] but the ARMA method is much simpler to understand and use.

A more sophisticated model of Support Vector Regression has been developed for motor cortical data, but on a simpler center-out task [1]. Adapting this SVM model to the sequential random tracking task used here may provide even better results. Finally, these methods should be explored with more datasets and in on-line experiments.

Acknowledgment

This work was supported by NIH-NINDS R01 NS 50967-01 as part of the NSF/NIH Collaborative Research in Computational Neuroscience Program, and by Brown University. The authors thank M. Serruya, M. Fellows, and J. Donoghue for providing the data for our research, F. Wood and C. Jenkins for many useful discussions, A. Shaikhouni for first suggesting an ARMA process, and A. Anagnostopoulos for helpful hints.

References

[1] L. Shpigelman, K. Crammer, R. Paz, E. Vaadia, and Y. Singer, "A temporal kernel-based model for tracking hand-movements from neural activities," *Advances in NIPS*, vol. 17, 2004, to appear.

[2] M. Serruya, N. Hatsopoulos, M. Fellows, L. Paninski, and J. Donoghue, "Robustness of neuroprosthetic decoding algorithms," *Biological Cybernetics*, vol. 88, pp. 219–228, Feb. 2003.

[3] J. Wessberg and M. Nicolelis, "Optimizing a linear algorithm for real-time robotic control using chronic cortical ensemble recordings in monkeys," *J Cog Neuro*, vol. 16, no. 6, pp. 1022–1035, 2004.

[4] W. Wu, M. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. Donoghue, "Neural decoding of cursor motion using a kalman filter," *Advances in NIPS*, vol. 15, pp. 133–140, 2003.

[5] A. Brockwell, A. Rojas, and R. Kass, "Recursive bayesian decoding of motor cortical signals by particle filtering," *J Neurophysiol*, vol. 91, pp. 1899 – 1907, Apr 2004.

[6] J. Carmena, M. Lebedev, R. Crist, J. O'Doherty, D. Santucci, D. Dimitrov, P. Patil, C. Henriquez, and M. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *Public Library of Science Biology*, vol. 1, no. 2, pp. 193–208, 2003.

[7] Y. Gao, M. Black, E. Bienenstock, S. Shoham, and J. Donoghue, "Probabilistic inference of arm motion from neural activity in motor cortex," *Advances in NIPS*, vol. 14, pp. 221–228, 2002.

[8] W. Wu, A. Shaikhouni, J. Donoghue, and M. Black, "Closed-loop neural control of cursor motion using a kalman filter," *Proc. IEEE EMBS*, pp. 4126–4129, Sep 2004.

[9] J. Fisher and M. Black, "Motor cortical decoding using an autoregressive moving average model," *Proc. IEEE Engineering in Medicine and Biology Society*, pp. 1469–1472, Sept. 2005.

[10] M. Serruya, N. Hatsopoulos, L. Paninski, M. Fellows, and J. Donoghue, "Brain-machine interface: Instant neural control of a movement signal," *Nature*, vol. 408, pp. 361–365, 2000.

[11] E. Maynard, C. Nordhausen, and R. Normann, "The Utah intracortical electrode array: A recording structure for potential brain-computer interfaces," *Electroencephalography and Clinical Neurophysiology*, vol. 102, pp. 228–239, 1997.

[12] F. Wood, M. Fellows, J. Donoghue, and M. Black, "Automatic spike sorting for neural decoding," *Proc. IEEE EMBS*, pp. 4009–4012, Sep 2004.

[13] J. Hamilton, *Time Series Analysis.* Princeton, New Jersey: Princeton University Press, 1994.