# Predicting 3-Dimensional Arm Trajectories from the Activity of Cortical Neurons for Use in Neural Prosthetics

Cynthia Chestek – CS 229 Midterm Project Review 11-17-06

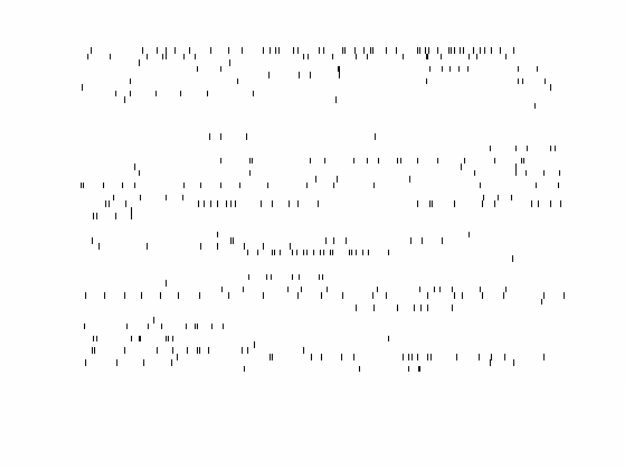
## *Introduction*

Neural prosthetics is a relatively new field that involves recording from neural activity in the cortex and decoding a useful signal that can be used to control an external device. There has been considerable recent interest in using such a system to provide brain controlled robotic limbs to amputees. However, due to the increased safety concerns and cost to patients undergoing neurosurgery, the viability of brain controlled robotic limbs depends on the ability to control many more degrees of freedom than currently available devices, and produce nearly arbitrary movement. While sufficiently articulated robotic limbs can be created, it still an open question whether sufficient control information can be obtained using the output of a few dozen neurons. Significant prior work has been done with animals selecting fixed targets by controlling a cursor on a screen with neural activity. One approach involves decoding the desired end point directly. This approach has resulted in the highest rate of information transfer reported to date, 6.5 bps. [1] However, controlling a robotic limb would require the second approach, which involves decoding trajectories rather than end point. Several groups have taken this approach, including recent work in humans, which utilized a Kalman filter for closed loop control of a cursor on a computer screen. This group and others have also demonstrated the ability for humans and monkeys to control robotic limbs with limited degrees of freedom in similar closed loop systems. [2-3,6]

However, a practical clinical system for a brain controlled robotic limb could not involve training on every possible reaching location the patient might require. Therefore, this class project evaluates the ability of several of these decoding algorithms to generalize to targets not seen in the training data set. This will involve driving a two-segment arm model in Matlab with neural data obtained from rhesus macaques making hand reaches to one of eight commanded targets. Several methods were used including linear regression with different position encoding schemes, a Kalman filter using the neural firing rates as a noisy measure of the desired position, and Naïve Bayes encoding discrete movement steps. Several groups still use linear regression to reconstruct end point trajectories due to its ease of implementation and fairly high performance [5,6]. However, it does not constrain the endpoint to realizable arm movements. A Kalman filter does constrain the movements to reasonable trajectories, and represents the current state of the art in human FDA trials for brain computer interfaces [7]. Finally, Naïve Bayes is similar to direct endpoint systems that have the highest reported performance, though extending this into trajectory decoding requires discretizing all aspects of the movement. These different approaches were first compared according to overall ability to generate correct trajectories in this experimental paradigm, and then evaluated based on how well the learned models generalized to targets left out of the training data.

## *Methods*

The neural data used for this project has been previously analyzed in [4]. It was obtained from a single rhesus macaque monkey, with a 96-electrode array implanted in pre-motor cortex. Each electrode can record the activity of ~0-4 neural units. The animal performed an eight-target center out reach task, while this activity was recorded for a total of 2.5 hours or 1072 reaches. Neural data was sorted using the Sahani spike sorting algorithm, which using automated clustering in principle component space, based on the spike waveforms [8,9]. Figure 1a. shows an example of the activity of individual neurons for a single reach, which is the raw input to this learning algorithm. Using the rate-coding assumption that relevant information is contained in a unit's average firing rate, the number of spikes was summed over into 100 ms bins, to provide an estimate of firing rate across time. The input matrix also includes 10 copies of each neuron's firing rate with 100 ms lags up to 1 sec in the past, similar to [5].



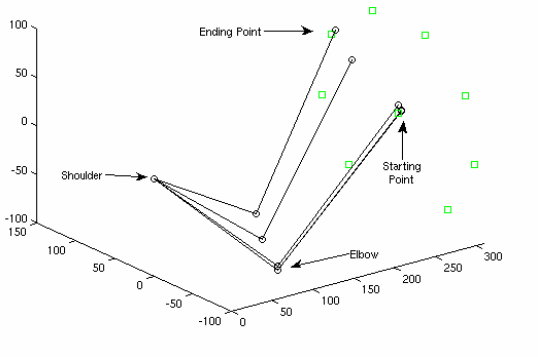



**Figure 1a. Neural Data for one Reach**

**Figure 1b. Example Reach Trajectory**

For this data set, only the (x,y,z) position of the hand was recorded. Therefore, a few assumptions were necessary to extrapolate kinematics of the whole arm. From observation of animals performing similar reaches, the shoulder position seems to remain fixed and the elbow position remains as low as possible, presumably to minimize energy consumption. Figure 1b shows an example of a reach in three dimensions, which is the desired output of the learning algorithm.

For the first method, linear regression was performed with limb position specified by either xyz position of the hand, joint angles, or change in xyz position from the previous position. Neural firing rates are causally linked to muscle force through a significantly non-linear relationship. Therefore, it's difficult to predict which features of movement will result in the best linear estimation, so a few different features of the motion were evaluated.

$$Y_{handpos} = X_{neurfiring} B_{model} \quad (1) \qquad B = (X^T X)^{-1} X^T Y \quad (2)$$

The Kalman filter used a linear estimation of hand position and velocity in the x,y,z directions from firing rates as the measurement, similar to linear regression, as in [7]. Velocity was estimated from distance traveled between sequential timesteps. The state equations for the filter are given in equations (3) and (4). The measurement at time i is the firing rates of all the neurons, X(i). This measurement is assumed to be generated from a linear relationship with the kinematic variables Y(i) as well as a noise term q(i), which is assumed to be Gaussian. The system also includes a term to describe how the state propagates in time with a added random walk element. These 4 matrices, H, q, A, and w, were assumed to be constant in time. Full derivations of the closed forms of these solutions can be found in [7]. Both the measurement noise q(i) and movement distribution w(i) was observed to be reasonably Gaussian in the training data set.

$$X(i) = HY(i) + q(i) \quad (3) \qquad X(i+1) = AX(i) + w(i) \quad (4)$$

Using a generative approach with Naïve Bayes (NB), movement was encoded as a step of a given length at a given angle each 100 ms timestep. Several aspects of movement were discretized. First, a model was created to differentiate moving vs stationary. The x position of the hand was categorized as either touching or not touching the screen. Movement in the y,z plane used discrete steps in one of 32 discretized angles. Step size was simply equal to the average distance traveled during 100 ms of active motion. Firing rate was modeled as $Xi|yj \sim Poisson(\lambda ij)$ for each neuron and each category (moving vs not moving, touching vs touching, and 32 discrete angles). Since $E(Xi|yj) = \lambda ij$, estimates were determined as in (5). To attempt to better generalize to angles not seen in the training set. To accomplish

this, smoothness of lambda was enforced across the target angles y using linear interpolation in one experiment. Note that p(y) was also enforced to be a uniform distribution, so that NB determined the most likely target using P(x|yj) alone, as shown in (6).

$$\lambda_{nj} = \frac{\sum_i 1(y^{(i)} = j) \cdot x_n^{(i)}}{\sum_i 1(y^{(i)} = j)} \quad (5) \qquad \max_j p(x \mid y_j) = \sum_n \log\left(\frac{\lambda_{nj}^{x_n}}{x!} e^{-\lambda_{nj}}\right) \quad (6)$$

## *Results*

*Overall Performance*

Linear regression successfully recreated from neural data three-dimensional trajectories highly correlated with the original trajectories. Figure 2 shows an example of a successful predicted reach. The training set used to generate the model included the first 896 reaches in the data set, while the test set included 129 subsequent reaches. Correlation coefficients between the predicted and actual xyz coordinates are shown in Table 1, column 1. The fourth row gives the percentage of time in which the hand came closest to the correct target, and the bottom row shows what percentage of the time the hand stopped within 3.5 cm of the target (in 3 dimensions). To try and determine whether the neural activity would be more closely tuned to alternate kinematic parameters, such as joint angle and change in position (rather than the absolute position) regression was performed representing the output in those formats. The results are shown in columns 2 and 3. Angle encoding performed just as well as hand position encoding, whereas decoding change in hand position led to cumulative errors that dramatically reduced performance. Finally, the Kalman filter had very similar performance to the linear regression, as shown in the fourth column

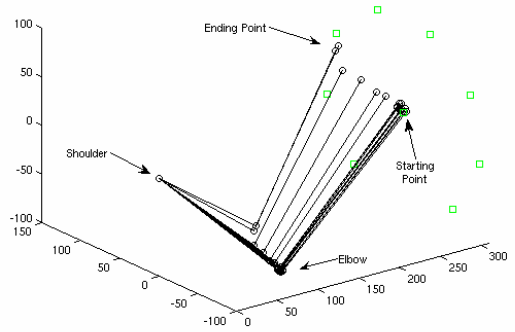



**Figure 2. Linear Regression Predicted Reach**

**Table 1.**

| | Hand xyz | Joint Angle | Δxyz | Kalman Filter |
|---|---|---|---|---|
| ρx | 0.43 | 0.43 | 0.16 (p<0.001) | 0.05 (not sig) |
| ρy | 0.82 | 0.82 | 0.59 | 0.80 |
| ρz | 0.73 | 0.73 | 0.46 | 0.70 |
| %correct | 81.4% | 83% | 59% | 80% |
| %hit | 69.8% | 69.8 | 90% | 68% |

Naïve Bayes was also applied to this problem, as described in the methods section. Model parameters λij were trained on the first 896 reaches in the data set and tested on 129 subsequent reaches. The performance is shown in Column 1 of Table 2. NB outperformed linear regression in both correlation with the actual reach position and error rate in hitting the target. Figure 3 shows an example of a successful reach using NB. Note that the trajectory looks reasonable despite being discretized.

All together, these measurements together establish a baseline of these algorithms' performance on this particular dataset.

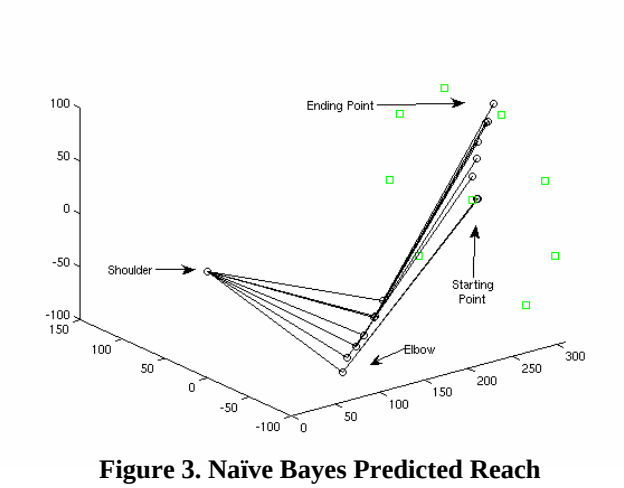



**Figure 3. Naïve Bayes Predicted Reach**

**Table 2**

| | Naïve Bayes |
|---|---|
| ρx | 0.22 |
| ρy | 0.93 |
| ρz | 0.90 |
| %correct | 87.6% |
| %hit | 87.6% |

*Generalization*

All three algorithms were then trained with (using absolute position) with a dataset of 938 reaches missing 1 of 8 of the targets, and tested on 134 reaches to that target. Performance is shown in Table 3, while Figure 3 shows examples of resulting trajectories. Naïve Bayes predictably generalized the worst. It consistently picked neighboring targets, as shown in Figure 4 (right), and only rarely hit the correct target when guesses were oscillating between the two neighbors. Linear regression fared somewhat better. Correlation coefficients remained fairly high, and it hit the target 51% of the time, compared to 69% previously. The Kalman filter had the best generalization performance at a 60% hit rate. This makes sense, since linear regression often moved roughly in the correct direction with unreasonable abrupt jumps from place to place. The Kalman filter always produced reasonable trajectories that were also roughly the correct direction.

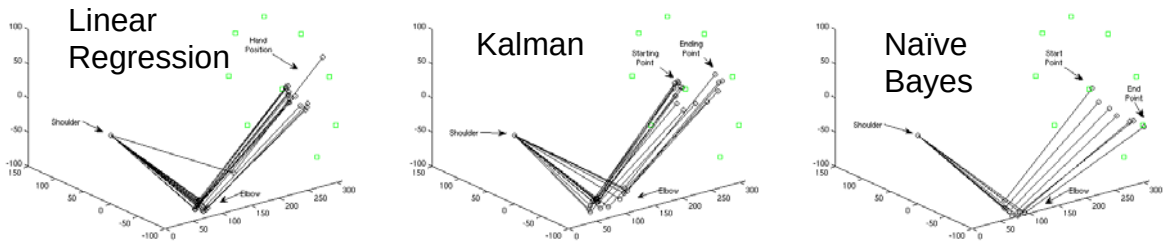



**Figure 4. Reaches to Novel Targets**

**Table 3.**

| | Linear Regression | Kalman Filter | Naïve Bayes |
|---|---|---|---|
| ρx | 0.43 | 0.002 (notsig) | 0.22 |
| ρy | 0.76 | 0.77 | 0.78 |
| ρz | 0.57 | 0.54 | 0.30 |
| %correct | 75% | 76% | 29% |
| %hit | 51% | 60% | 21% |

The generalization of Naïve Bayes could be somewhat improved by finding the λij for each angle and smoothing them according to a cosine fit. Figure 5 shows an example of the smoothed λij for all 32 possible angles in which the algorithm can take a step. This increased the generalization performance to 50% for the missing target as shown in Table 4. However, general performance on the other targets was substantially reduced to 62%.

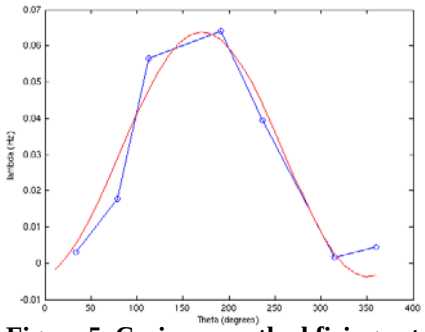



**Figure 5. Cosine smoothed firing rates**

**Table 4**

| | Novel Target | Other Targets |
|---|---|---|
| ρx | 0.22 | 0.22 |
| ρy | 0.91 | 0.90 |
| ρz | 0.12 ($p<0.001$) | 0.72 |
| %correct | 50% | 62% |
| %hit | 50% | 62% |

### *Discussion*

All three models had good performance with a complete training set that included all 8 targets. Naïve Bayes performed the best, taking advantage of the limited number of possibilities. Conversely, Naïve Bayes had the worse performance on novel targets, almost exclusively going to the targets near the correct target. The Kalman filter had the best performance on novel targets, presumably because it made the strongest modeling assumptions about the system. Not only did it include a linear estimate of position and velocity from firing rates, but it also included a model for how the hand is capable of moving.

In the future, generalization will be a growing issue in neural prosthetic design. Current systems controlling a 2-D cursor take approximately 0.5 hour of model training for 2 hours of use, so it would be difficult to use a more inclusive training set. [3] Transitioning to motion prosthetics, not only will systems have to reach to novel locations, but users may also require control of speed and posture. It will not be straightforward to infer average neural firing rate across all of these variables, even if the signal did not have a significant noise component. Therefore, given such limited input, learning algorithms will likely have to make very strong and correct model assumptions about how the arm tends to move. However, with such a system, it may be possible to achieve nearly arbitrary movement in a brain-controlled arm prosthetic.

### *References*

I would like to acknowledge Gopal Santhanam, Byron Yu, and Afsheen Afshar for allowing me to use neural data they had collected for another research program. Also, Prof. Krishna Shenoy provided guidance regarding the appropriate algorithms and the open questions in the field of neural prosthetics.

[1] Santhanam, G., Ryu, S. I., Yu, B. M., and K. V. Shenoy. “A high performance brain computer interface.” *Nature*. 442:195-198.
[2] Taylor, D. M., Tillery, S. I. H., and A. B. Schwartz. “Information conveyed through brain control: Cursor versus robot.
[3] Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D. and J. P. Donoghue. “Neuronal ensemble control of prosthetic devices by a human with tetraplegia.” *Nature*. 442:164-171.
[4] Yu, B. M., Ryu, S. I., Churchland, M. M., and K. V. Shenoy. “Improving neural system performance by combining plan and peri-movement activity.” *Proc. IEEE EMBS*. San Francisco, CA Sept 1-5, 2004. p4516-4519.
[5] Carmena, J. M., Lebedev, M. A., Henriquez, C. S., and M. A. L. Nicolelis. “Stable ensemble performance with single neuron variability during reaching movements in primates.” *J. Neurosi*. 25(46):10712-10716.
[6] Lebedev, M. A., Carmena, J. M., O’Doherty, J. E., Zacksenhouse, M., Henriquez, C. S., Principe, J. C., and M. A. L. Nicolelis. “Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface.” *J. Neurosci*. 25(19):4681-4693.
[7] Wu, W., Gao, Y, Bienenstock, E., Donoghue, J. P. and M. J. Black. “Bayesian population decoding of a motor cortical activity using a Kalman filter.” Neural Computation 18:80-118, 2006.
[8] M. Sahani, “Latent variable models for Neural Data Analysis.” Computational and Neural Systems Program: California Institute of Technology, 2004.
[9] Santhanam, G., Sahani, M., Ryu, S. I., and K. V. Shenoy. “An extensible infrastructure for fully automated spike sorting during online experiments.” *Proc 26th IEEE EMBS*, San Francisco, CA. 4380-4384, 2004.