Decoders for Neural Prosthetics

Vivek Athalye, Ritesh Kolte, Vighnesh Rege

Introduction

The field of neural prosthetics involving motor control centers on the problem of decoding motor cortex neural data into a predicted trajectory of arm movement. Neuroscience presents the perspective that neurons can be interpreted as noisy sensors, roughly Poisson distributed, of a smooth underlying process of time-varying firing rates, and these firing rates of motor cortex neurons encode motor control information. For our project, we have been given data from Professor Krishna Shenoy's Neural Prosthetics Group that contains neural data and hand position data of a macaque for several trials and their state of the art neural-decoded trajectories based on monkey arm-reach trials.

Information about the experimental setup

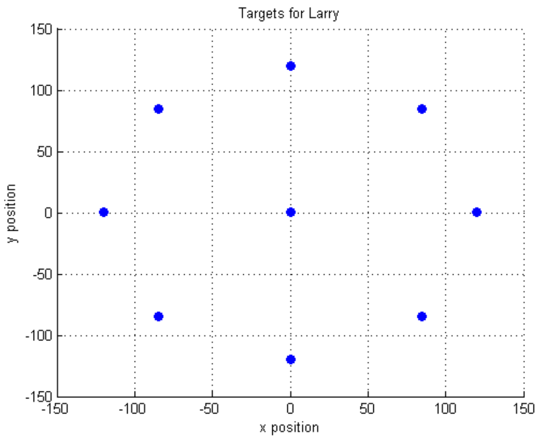
In each trial, the macaque ("Larry") is shown a dot on the computer screen which it is trained to point its finger at. The finger is fitted with a bead which enables accurate measurements of the hand position. There are 9 possible locations for the dots, 8 of them equidistantly placed on the circumference of a circle and the 9th dot in the center of that circle. Though the arm moves in 3 dimensions, the movement is mainly in the 2-dimensional plane (near the screen). Hence, we will consider only the X and Y coordinates of the hand position for our analysis. An electrode array is implanted in Larry's motor cortex which records the spiking of 96 neurons. The firing rate of each neuron is calculated by summing the spikes it generates in a 50 ms ('bin width') window. The time between the start of successive windows is 17 ms (this value chosen since the hand position updates are obtained at 60Hz, i.e. every 16.66 ms).

Naïve Bayes Modeling

For a quick first step in our project, we decided to use the Naïve Bayes classification method, generalized to the case when both the input and the output can take on multiple values. We mapped our problem to the Naïve Bayes framework for two different label interpretations:

1) Labels of feature vectors representing predicted velocity in Cartesian coordinates at the next time instant (i.e. a 2 dimensional label)
2) Labels of feature vectors representing direction of the predicted velocity at the next time instant (a 1-dimensional label)

Our feature vector is a 98 dimensional vector, whose first 96 elements are the firing rates corresponding to the 96 neurons being studied and the last 2 are the current position in 2 dimensions (X, Y). Neuron's spiking is modeled as a Poisson process, and thus estimating the mean firing rate amounts to summing the number of spikes over a bin width period of time. As a first pass, we used a bin width of 50ms, yielding a maximum case spike count of 20 spikes within a bin width period. We decided to discretize the spike frequencies and X, Y position into 10 values so that all the components of the binned 98 dimensional vector take on 10 values each. The range of X, Y values that the monkey's hand reached was not very large (<30cm), so 10 bins for each axis was not a coarse quantization. Due to the large number of labels in some cases, we applied Laplace smoothing to all the parameters of the Naive Bayes model: $p(x_j = l \mid y = k)$ and $p(y) = k$.



For the first approach, the label vector is a 2-dimensional vector, whose components are the velocities in the X and Y directions. Note that if each velocity direction takes on k labels, the 2-dimensional label can take on k^2 values. For the second approach, the label denotes one of the possible 8 directions in which Larry's aim should ideally move.

While testing in the first approach, we classified a prediction erroneous if the predicted velocity in either the X or the Y direction was more than a bin away from the true velocity bin in that direction. In the second approach, additionally we also calculated another error metric, which allows the predicted velocity to lie only in the actual bin.

Approach 1	
Bins	If (more than one bin away) {Error}
8	2.62 %
12	8.83 %
15	16.28 %
18	21.40 %
25	30.92 %

Approach 2 (8 bins)	
If (more than one bin away) {Error}	If (not the true bin) {Error}
36.29 %	59.69 %

Linear Regression

Many neuroscience labs use a linear regression model as a base decoder to map neural signals to motor actions [1, 2]. In our formulation, a control signal consisting of the current time step's neural vector and the cursor's current position was used to decode the current time step's velocity.

$$x_t = \begin{bmatrix} neural_t \\ position_t \\ 1 \end{bmatrix} \qquad velocity_t = \theta^T x_t \qquad position_{t+1} = position_t + velocity_t$$

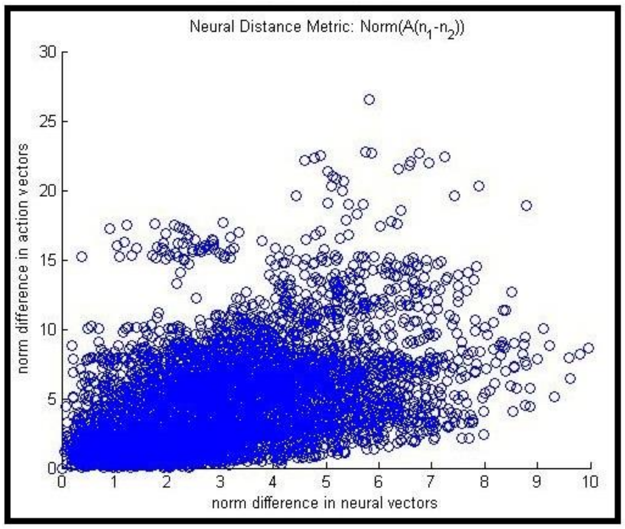
We formulated our problem of learning theta in the standard Least Squares form, where represents our training control vectors (concatenation of neural and position), represents our parameter matrix, and represents the training velocities resulting from control vectors. We used the result of this decoding as a baseline for further improvements. $\min_\theta \|X^T \theta^T - V^T\|$

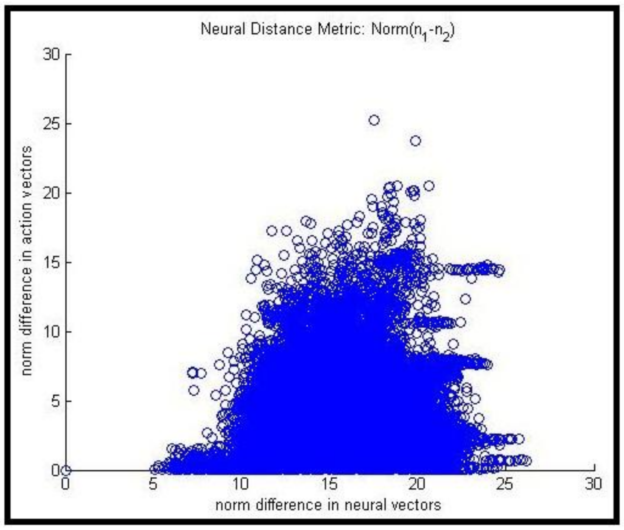
From a neuroscience perspective, in many labs correlation is used as a neural-decode error metric because it demonstrates that a relationship exists between neural-decoded trajectories and actual hand trajectory. In many labs, a decoder's success is then visualized by plotting an offline decoded trajectory versus the hand's actual trajectory.
We adopt an engineering perspective in which neural decoding is a prosthetic application enabling patients to control cursors to targets. From this perspective, our neural decoder will be run in online trials during which a patient will observe where the cursor moves and provide feedback to control the cursor. Thus, we visualize our decoder by observing at each time step the difference between the direction the hand and the decoder took. We also use an error metric based on application: to measure accuracy, we bin continuous directions into angle intervals, and calculate % error based on how often our continuous decoder's predicted direction is not within the appropriate bin. We also measure standard deviation of our angle prediction error in order to have a feel of our angle error interval. This error perspective guided our design of neural decoder improvements.

From an engineering perspective, the neural prosthetic's aim is to allow a person continuous control of a cursor to reach a target. Thus, we attempted to use neural data to extract auxiliary motor features that could assist in doing neural decode towards a target.
1) Decode distance to the final target at each time step. Motivation: We could use this decoded parameter to enforce a smooth velocity distribution for the cursor over time. Result: 199.3% error and extremely large standard deviation. Our linear regression model was unable to capture this feature.
2) Decode direction angle to the final target as opposed to the direction of hand movement. Motivation: If this feature is encoded in neural data along with information about the hand's trajectory, we can directly decode to target, or we can regularize our neural decode so that if it errors from hand path, it does so more frequently towards the target. If these two are independent signals, then using both could lead to a more robust neural decoding. Result: This signal could not reliably be decoded using linear regression—we found a very large standard deviation of error.

Locally Weighted Linear Regression (LWLR)

Interested in the idea that there is nonlinearity in the way neural data encodes motor features, we thought we might improve performance by implementing Locally Weighted linear regression, thus finding locally linear relationships between neural data and actions. In order to implement locally weighted linear regression (LWLR), we need to assign weights to the feature vectors 'close' to the test feature vector under consideration. The challenge in adapting LWLR to our problem lies in selecting this measure of 'closeness'. The Euclidean distance is not a natural distance metric



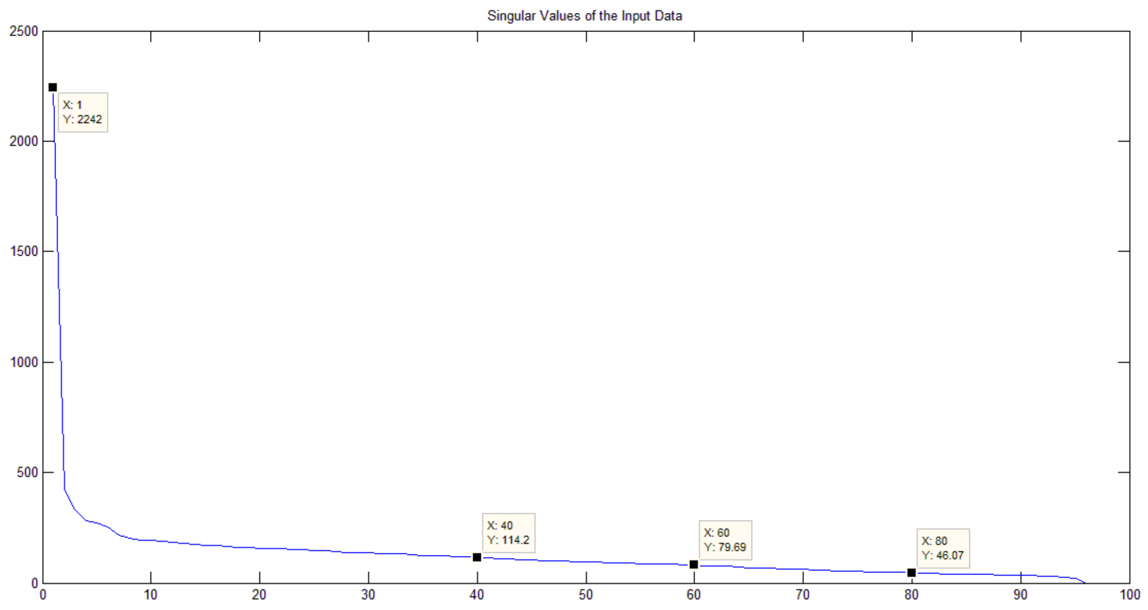


for the neural vectors. Intuitively, when we take the Euclidean norm of the difference between two neural vectors, we end up mixing their components in a haphazard manner. A good way to decide whether two neural vectors are close would be to look at the actions they cause. So if we were to take the Euclidean distance between the velocity vectors induced by the neural impulses, we would get a better measure of their closeness. In order to verify whether this intuition works, we use the graphical analysis shown below. For the data points under consideration, we picked 2 at a time and evaluated the norm of the difference between the corresponding velocities as the y-coordinate and the 'distance' between the neural vectors pertaining to the chosen metric as the x-coordinate. A good distance metric would ensure similar actions for 'similar' neural vectors. And hence, a necessary condition is cluttering of points around the origin.

The distance metric in the left figure uses the matrix A obtained via linear regression and turns out to be much better than normal Euclidean distance metric. We also tried an iterative procedure where we use A in the distance metric to sort vectors for LWLR so as to obtain $\Theta^{(1)}$. We then update $A = \Theta^{(1)}$ and use it to sort and so on. This procedure however did not show much improvement after the first step (and the error seemed to compound with every step for certain test points).

Dimensionality Reduction

It has been observed by many researchers in the neuroscience community that a small dimensional subspace of the neural data is sufficient to characterize the hand movement. We did a quick PCA on the neural data used for training to verify this and the plot of the singular values confirmed this claim. (Note: Other techniques such as factor analysis, Gaussian-process factor analysis have been used to good effect for dimensionality reduction in the problem of decoding neural data.) Reduction of dimensionality also has the benefit of noise reduction, by retaining only those dimensions which have maximum 'information' about motor features and discarding the signal and importantly noise in the other dimensions. Since the neural data is extremely noisy (variance of Poisson distributions is equal to the mean), reducing dimensions is considered to be an essential step in the task of predicting motor actions.



Regularization

By plotting our decode results, we observed the influence of neural noise on straight line path decodes. Between sequences of correct predictions, predictions would sporadically be far off. In order to counteract this effect, we optimized a new objective in solving for our parameter matrix that penalizes a large derivative in velocity. This lead to the following optimization formulization known as Tikhonov regularization:

$$\min_{\theta} \|X^T\theta^T - V^T\|_2 + \gamma\|\Delta(X^T\theta^T)\|_2$$

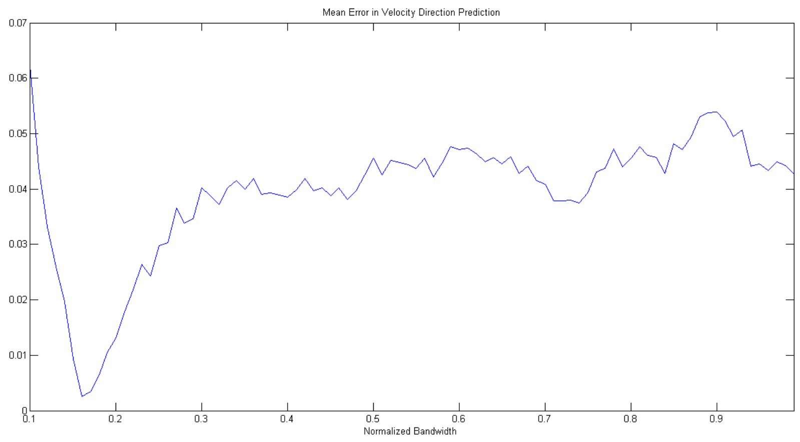
We were able to vary the gamma parameter in order to change our penalty on velocity deviation. This formulation revealed a tradeoff between the cursor's robustness and maneuverability (ability to change direction quickly). We hoped to try different penalty functions (equating to different norms in the regularization term) to allow a certain level of velocity change while penalizing sudden large velocity changes past a threshold.
However, our trials did not work as well as expected. Rather than reducing rapid changes in our decoding, the magnitude of our decodes became smaller with increased γ. The reason is as follows: The method has the appropriate probabilistic interpretation as stochastic-robust approximation when there is noise in X (our neural data). $X^T\theta^T$ will have more variation for "larger" θ, and by Jensen's inequality, variation in $X^T\theta^T$ will increase the average value of $\|X^T\theta^T - V^T\|$. This equation balances a tradeoff between making $\|X^T\theta^T - V^T\|$ small with the desire for a small θ to minimize variation in $\|\Delta X^T\theta^T\|$. While this felt intuitively correct, we observed that regularization only served to shrink our parameter matrix values and our decode magnitudes.
We attempted to counter this problem by transforming decode variables from v_x, v_y to $\theta, \|v\|$. If we could regularize the variation in predicted scalar θ, we would penalize rapidly changing direction rather than changing magnitude, which we were less concerned with. However, the nonlinear transformation of variables decreased our success in using linear regression to decode hand movement direction, and thus this method did not have success.

Filtering

The idea of constraining decodes for smoother hand movements as contrasted to the sometimes-jagged prediction from linear regression inspired our approach of smoothing the linear regression prediction with a discrete-time low-pass filter. A cursory description of the idea we implemented was to learn the filter coefficients so as to minimize the error term.

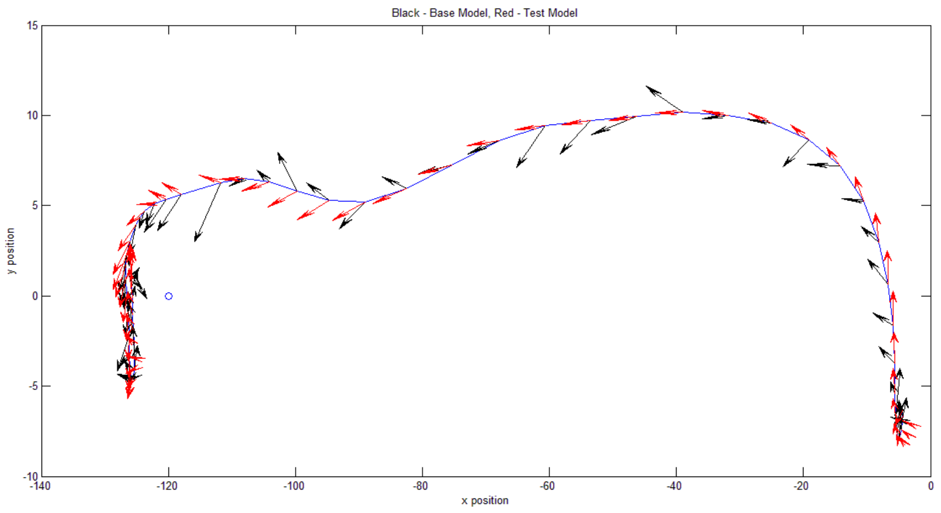
We selected a low-pass Butterworth IIR low-pass filter for our purpose. For similar frequency response characteristics, an IIR filter has a much lower order as compared to a FIR filter. Among the IIR filter designs, the Butterworth filter has a monotonic frequency response. The other typically used filter designs (Chebyshev I, Chebyshev II, Elliptic) have ripples in their frequency response. In our case, where the frequency denotes how fast a hand can move, it made sense that the frequency response not have ripples. If it did have ripples, it would mean that in some interval of the frequency axis, we would penalize lower frequencies more than higher frequencies (i.e. we would be implicitly saying that moving the hand at some speed is difficult than moving it at a higher speed).

For the Butterworth low-pass filter parameterization, specifying the corner frequency specifies the complete filter. Hence, the task before us was to choose a corner frequency. This was done by varying the corner frequency from 0 to 1 over the normalized frequency range and selecting that frequency which minimized the mean over the training data of the angle between the actual velocity and the velocity predicted by linear regression.



The adjoining figure shows the mean error in angle for different values of the corner frequency (for bin width = 50 ms, K = 40). We see that a minimum is achieved at a normalized frequency of 0.15-0.16 which corresponds to 4.5-5 Hz (the frequency at which trials are obtained (sampling frequency) of our system is 60 Hz.)

The following figure shows the unfiltered linear regression prediction in black and the filtered linear regression prediction in red, superimposed on the actual trajectory in blue. We see that though the filtered prediction takes a while to follow up when the hand makes a sharp movement, for most of the trajectory, it matches better with the actual trajectory. An inherent disadvantage of the filtering is that if the hand is about to stop, the filtered prediction will take infinite time to stop (never exactly stopping). A FIR filter will reduce this time due to the finite length of its impulse response, however, at the cost of an increase in the filter order to obtain the same mean deviation angle.



Results

Each cell in the following tables contains 3 rows:
1) Percentage of trials where the prediction was more than a bin away
2) Mean angle (in degrees) between the predicted direction and the true direction
3) Standard deviation (in degrees) of the angle between the predicted direction and the true direction

	Bin width = 50ms	Bin width = 100 ms	Bin width = 150ms	Bin width = 200ms
Linear Regression	31.12 % 1.9658 72.4528	25.61 % 0.7276 66.2790	24.05 % 2.3924 63.9629	22.55 % 3.2960 62.5602

Linear Regression (Regularization with γ=0.3,0.8,0.2)	30.34 % -0.9121 72.47	24.2 % 0.56 64.87	23.56 % -1.60 63.46	22.4 % -3.24 62.3
Locally Weighted Linear Regression	22.5 % -10.1174 71.04	22.5 % -9.0926 63.96	23 % -5.6892 64.1	21.5 % -7.2271 65.04
(Unfiltered) Linear Regression (K =10)	34.27 % -0.9440 76.1305	27.73 % -2.9654 69.1662	23.01 % -1.8998 63.5175	21.22 % 3.2981 60.0536
Filtered Linear Regression (K = 10)	27.51 % -0.9209 68.6491	25.40 % -1.1847 66.8405	21.94 % 0.7033 62.5088	23.25 % 2.6395 62.3827
Filtered Linear Regression (K = 40)	27.85 % 1.5482 69.8131	27.11 % 1.5079 69.6530	21.97 % 1.2803 61.8588	22.05 % 4.2140 61.8217

Conclusions

Our project centered on used a linear mapping between neural space and motor action space, and an engineering approach via regularization, filtering, and training sample weighting to produce a smoother, more accurate neural cursor application. Some extensions of the decoder models we tried are a kernelized regression and discrete direction decoding via a cascade of SVMs in which we proceed from broad to specific direction classification. In our work on LWLR, we see an opportunity for manifold learning to provide a non-Euclidean measure of distance between neural vectors.s

Taking a broad view of the problem, the challenge is two-fold: to learn the mathematical structure by which the brain encodes and develops control signals for the hand, and to separate noise from signal in neural channels. If we were to proceed doing work in this area, we would not attempt further hacks for manipulating the linear model. It would be important to explicitly model noise, and devise techniques for learning neural encoding.

In the future, a neural network model that learns neural features and maps them to motor kinematics could be used to learn mathematical structure. While noise modeling in the brain is an active area of research, a ready to implement model that battles noise is the Robust Kalman Filter which utilizes real time convex optimization techniques to incorporate an added sparse error term.

References

(Linear Model used in offline decoding)
[1] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. L. Nicolelis. Learning to control a brain-machine interface for reaching and grasping by primates. PLoS, Biology, 1(2):001–016, 2003.
[2] J. Wessberg, C. Stambaugh, J. Kralik, Laubach M. Beck, P., J. Chapin, J. Kim, S. Biggs, M. Srinivasan, and M. Nicolelis. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature, 408:361–365, 2000.
[3] Carlos E. Vargas-Irwin, Gregory Shakhnarovich, Payman Yadollahpour, John M. K. Mislow, Michael J. Black, and John P. Donoghue. Decoding Complete Reach and Grasp Actions from Local Primary Motor Cortex Populations J. Neurosci. 30: 9659-9669; doi:10.1523/JNEUROSCI.5443-09.2010
(Parameters represented in tuning functions of neurons)
[4] D. Moran and A. Schwartz. Motor cortical representation of speed and direction during reaching. Jrnl. of Neurophysiology, 82(5):2676–2692, 1999.