# Decoding Motion: Linear Models for Neural Signal Processing

ECE 189: Advanced Honors Seminar (with Professor Jonathan Kao)

Rathul Anand
*University of California, Los Angeles*
*rathul@g.ucla.edu*

***Abstract*—This report explores the design and implementation of a brain-machine interface decoder to predict the motor kinematics, specifically reaching, from neural signals recorded in the primary and premotor cortices of a monkey. Using a dataset of 506 center-out and center-in reaching trials, we feature engineer and build a linear decoder trained to predict velocities of the monkey's hand from binned neural spike activity. This report compares different preprocessing pipelines, features, and regularization techniques. Our optimized decoder achieves an MSE of 995, improving significantly from a baseline MSE of 3492, for a real-time brain-machine interface.**

## I. Introduction

Brain-machine interfaces (BMI), specifically read-out BMIs, enable us to read neural spike data from a small region of the brain into an external device. At the core of these BMIs lies a neural decoder, transforming these noisy and intertwined signals into meaningful features that we can leverage for kinematic predictions like hand trajectories.

This project focuses on decoding motor trajectories from neural signals recorded during center-out and center-in reaching tasks from 192 neurons in a monkey's primary motor cortex and premotor cortex. Using an optimal linear decoder, we evaluate the effectiveness of various preprocessing methods to extract meaningful features, including various low-pass filters, band-pass filters, high-pass filters, derivatives of spike data, population coupling, and several combinations of these.

## II. Dataset and Preprocessing

Our dataset comes from monkey experiments performed in Professor Krishna V. Shenoy's lab, consisting of 506 trials of neural recordings from a single monkey. These trials are contiguous, and each trial $i$ includes spike activity from 192 neurons (96 from each cortex) recorded as sparse binary matrices at a 1 ms resolution $Y$ of dimension $N \times T_i$ where $N$ is the number of features (neurons) and $T_i$ is the number of milliseconds the trial took, kinematic data of cursor position in 3D space, target information for each center-out reach, and other metadata. All trials were successful.

We first binned neural activity with a 25 ms bin width $Y_{\text{bin}}$, with features calculated as either the sum of the spikes counts in each bin. We set aside 20% of the data for testing, and train on the remaining 80%. From this binned neural data, we decode $X_{\text{bin}}$, the velocity of the training data $v(t)$ at each bin $t$, where $v(t)$ is the first order Euler approximation of the velocity over the bin.

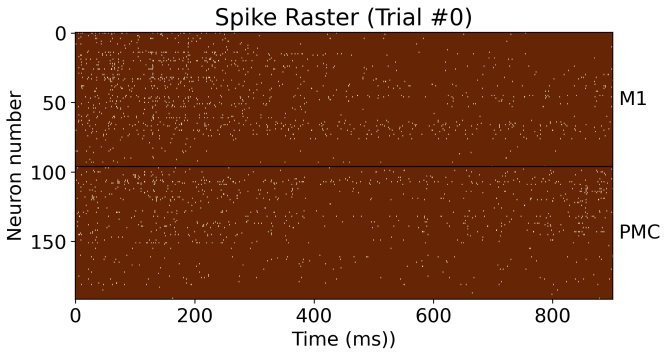


Fig. 1. Example spike raster from the first trial.

## III. Decoder Design

Given features $Y_{\text{bin}}$ with $N$ rows and kinematic data $X$ to decode, we start by appending a row of 1's to $Y_{\text{bin}}$ to get the input to our decoder, $Y$. This essentially adds a bias term so we can learn an affine decoder. Then, to solve for the decoder matrix $L$ such that $\hat{x}^{(i)} = Ly^{(i)}$ that minimizes the mean squared error (MSE) between all predicted $\hat{x}^{(i)}$ and ground truth $x^{(i)}$, $\sum_i \|\hat{x}^{(i)} - x^{(i)}\|_2^2$, we can solve:

$$L = XY^\dagger = XY^T(YY^T)^{-1} \tag{1}$$

We also empirically noticed that, likely given the high noise of neural data and limited training data, the decoder was significantly overfitting on the training data. To combat this, we introduce a strong $\ell_2$ regularization to give the model preference towards choosing an $L$ of smaller magnitude, corresponding to a simpler solution. Our loss function now becomes

$$\mathcal{L}(L) = \sum_i \|\hat{x}^{(i)} - x^{(i)}\|_2^2 + \lambda \|L\|_2^2 \tag{2}$$

where $\lambda$ is our $\ell_2$ regularization hyperparameter. Then, we can solve for the optimal $L$ as:

$$L = XY^T(YY^T + \lambda I)^{-1} \tag{3}$$

## IV. Feature Engineering, Experiments, and Results

### A. Baseline

As a baseline, we binned neural spike data by aggregating into non-overlapping bins of 25 ms. We compute the number of spikes in every bin to represent the neural activity during

each interval. This sufficiently small size transforms sparse high-resolution spike data into a more computationally manageable form capturing more global trends, while maintaining temporal patterns. For each feature, we'll show the true trajectories plotted alongside the decoded trajectories.

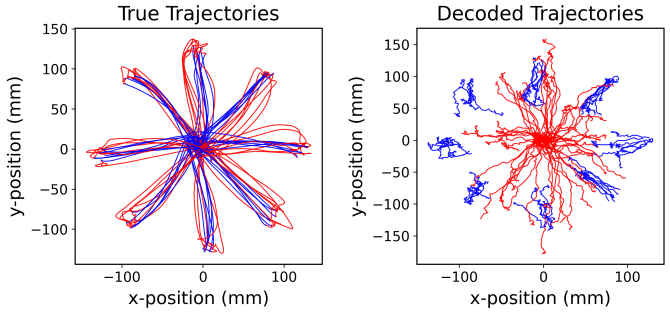


Fig. 2. Baseline linear classifier with $\lambda = 0$.

We see the decoder reasonable decodes center-out reaches (red), but unsuccessful with center-back reaches (blue). This is likely because the two different tasks represent two different modes in the neural signal data distribution, which are difficult for a linear classifier to learn and generalize. Linear classifiers also lack robustness against noise and fail to incorporate more informative higher-order temporal patterns. $\ell_2$ regularization also does not increase our accuracy.

TABLE I
BASELINE LINEAR CLASSIFIER WITH VARYING $\ell_2$ REGULARIZATION.

| $\lambda$ | Test MSE |
|---|---|
| 0 | 3492.56 |
| $1 \times 10^{-5}$ | 3492.59 |
| $1 \times 10^{-4}$ | 3492.87 |
| $1 \times 10^{-3}$ | 3495.72 |

### B. Low-Pass Filtering

Next, we implemented a low-pass filter to smooth neural signals with a causal Gaussian low pass filter, defined by the following kernel parameterized by standard deviation $\sigma$:

$$G(t) = \exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{4}$$

truncated at $4\sigma$ and defined for positive $t$. We also computed an exponential moving average with a causal exponential decay kernel parameterized by decay $\alpha$:

$$E(t) = \alpha(1-\alpha)^t \tag{5}$$

and finally, a sinc kernel:

$$\text{sinc}(t) = \frac{\text{sinc}(\pi t)}{\pi t} \tag{6}$$

also truncated with a cutoff and window size we tune. Since neural signals are inherently noisy but motor trajectories are smooth, we hope that low-pass filtering can supress some of this high-frequency noise and isolate the more informative trends in lower frequencies. We achieve fast filtering by transforming the signal to the frequency domain with an FFT.

We see that the Gaussian filter and EMA filter are most performant, and the sinc filter falls farthest behind. The strongest results are summarized below. We also tuned the $\ell_2$ regularization for the Gaussian filter with $\sigma = 100$.

The performance of the best decoder is visualized.

TABLE II
PERFORMANCE OF LOW-PASS FILTERS.

| Method | Parameters | Performance |
|---|---|---|
| Gaussian | $\sigma = 50$, truncate $= 4.0$ | 1877.21 |
| Gaussian | $\sigma = 80$, truncate $= 4.0$ | 1659.50 |
| Gaussian | $\sigma = 100$, truncate $= 4.0$ | **1621.53** |
| Gaussian | $\sigma = 120$, truncate $= 4.0$ | 1632.64 |
| EMA | $\alpha = 0.005$ | 2487.83 |
| EMA | $\alpha = 0.01$ | 1755.05 |
| EMA | $\alpha = 0.02$ | 1780.73 |

TABLE III
TUNING $\ell_2$ REGULARIZATION OF GAUSSIAN WITH $\sigma = 100$.

| $\lambda$ | Test MSE |
|---|---|
| 0 | 3492.56 |
| $1 \times 10^{-3}$ | 1563.24 |
| $1 \times 5 \cdot 10^{-3}$ | 1455.30 |
| $1 \times 10^{-2}$ | **1411.64** |
| $1 \times 5 \cdot 10^{-2}$ | 1455.40 |
| $1 \times 10^{-1}$ | 1597.92 |

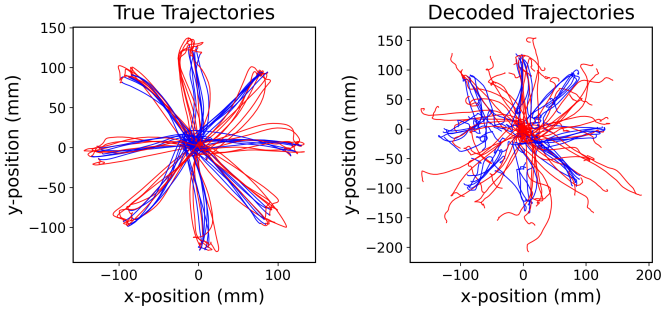


Fig. 3. Low pass Gaussian filter with $\sigma = 100$, $\lambda = 10^{-2}$.

### C. Intermediate and Higher Frequency Features

We also experiment with concatenating additional features to provide the model with more information and enhance its capacity for nonlinear decision-making. Particularly, we add high-pass filtering and band-pass filtering data.

We calculate our high-pass data by taking $\text{HP}_t = \text{Raw}_t - \text{LP}_t$, and our band-pass data by taking $\text{BP}_t = \text{LP}_{\sigma_{\text{high}}} - \text{LP}_{\sigma_{\text{low}}}$. Our baseline consists of stacked bin-summed and low-pass filtered data. We then extend this by stacking combinations of high-pass and band-pass filtered data with varying parameters. The most effective combinations are summarized. We also visualize the best model.

TABLE IV
PERFORMANCE OF VARYING FILTERING METHODS.

| Bin-Sum | LP $\sigma$ | BP $\sigma_{\text{low}}, \sigma_{\text{high}}$ | HP $\sigma$ | MSE Test |
|---|---|---|---|---|
| ✓ | N/A | N/A | N/A | 1621.53 |
| ✓ | N/A | N/A | 200 | 1726.85 |
| ✓ | 100 | N/A | 200 | 1513.53 |
| ✓ | 100 | 100, 200 | 200 | 1751.76 |
| ✓ | N/A | N/A | 10000 | 15185.17 |
| ✓ | 100 | N/A | 10000 | 1519.30 |
| ✓ | 100 | 100, 10000 | 10000 | **1453.11** |

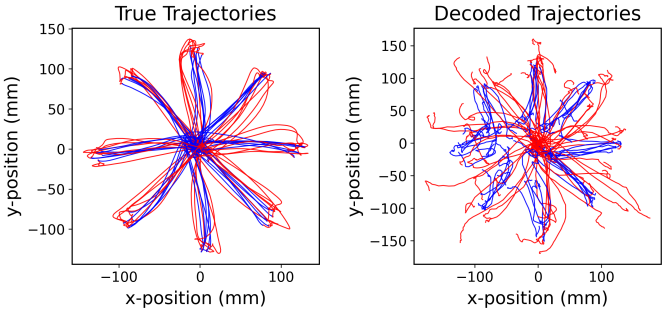


Fig. 4. Bin-summed + low-pass + band-pass + high-pass data.

### D. Derivative of the Neural Data

We also compute derivatives of binned neural signals using forward differences, with padding for the first derivative along each dimension. These derivatives are incorporated into our decoding pipeline as they may capture changes in neural activity that correlate more strongly with motor intent, as opposed to relying solely on absolute firing rates. This approach introduces an additional nonlinear transformation, revealing features that were previously not decodable in our linear model.

In our experiments, we observe that derivatives alone perform poorly, likely due to the removal of spiking intensity. However, combining derivatives with raw binned features significantly improves performance over binned features. We visualized the decoded trajectories from this combination, which shows smoother predictions.

TABLE V
PERFORMANCE OF DERIVATIVE DATA.

| Bin-Sum | Derivative | MSE Test |
|---|---|---|
| ✓ | ✗ | 3492.56 |
| ✗ | ✓ | 9203.17 |
| ✓ | ✓ | **2511.91** |

### E. Combining Existing Features

Based on existing best-performing features, we can perform a grid search on feature combinations and hyperparameters to make the most performant decoder with the features we have. We find that the best of these approaches combines the binned sums and derivatives with a Gaussian low-pass filter of

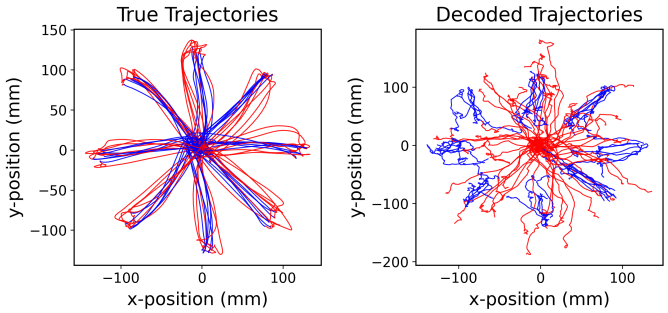


Fig. 5. Bin-summed + derivative data.

$\sigma = 100$, and $\lambda = 10^{-2}$ $\ell_2$ regularization, which we visualize.

TABLE VI
PERFORMANCE COMPARISON OF DIFFERENT CONFIGURATIONS.

| Bin | Derivative | LP | HP | BP | $\ell_2$ regularization ($\lambda$) | MSE Test |
|---|---|---|---|---|---|---|
| ✓ | ✓ | N/A | N/A | N/A | 0 | 2511.91 |
| ✓ | ✓ | N/A | N/A | N/A | 0.001 | 2516.68 |
| ✓ | ✓ | N/A | N/A | N/A | 0.01 | 2562.65 |
| ✓ | ✓ | ✓ | N/A | N/A | 0 | 1503.97 |
| ✓ | ✓ | ✓ | N/A | N/A | 0.001 | 1460.60 |
| ✓ | ✓ | ✓ | N/A | N/A | 0.01 | **1378.02** |
| ✓ | ✓ | ✓ | ✓ | N/A | 0 | 3542.45 |
| ✓ | ✓ | ✓ | ✓ | N/A | 0.001 | 1375.32 |
| ✓ | ✓ | ✓ | ✓ | N/A | 0.01 | 1406.94 |
| ✓ | ✓ | ✓ | ✓ | ✓ | 0 | 2484.45 |
| ✓ | ✓ | ✓ | ✓ | ✓ | 0.001 | 1488.88 |
| ✓ | ✓ | ✓ | ✓ | ✓ | 0.01 | 1433.26 |
| N/A | ✓ | ✓ | ✓ | ✓ | 0 | 1640.94 |
| N/A | ✓ | ✓ | ✓ | ✓ | 0.001 | 1527.70 |
| N/A | ✓ | ✓ | ✓ | ✓ | 0.01 | 1455.72 |

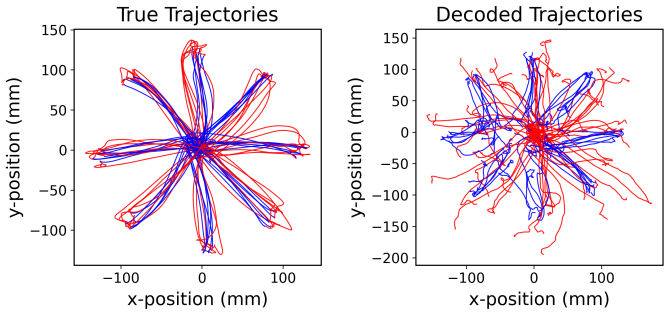


Fig. 6. Bin-summed + derivative + low-pass filter data.

### F. Population Coupling

Population coupling measures how each neuron's activity relates to the population's average as a correlation coefficient. This captures whether a neuron fires more or less when the population as a whole becomes more active. We introduce this in an effort to introduce more population-wide features, rather than just temporal features. However, we find that the introduction of this feature, even with $\ell_2$ regularization, performs worse on the test set. We were able to achieve

a minimum MSE of 1443.76 when combining population coupling with the above best decoder.

### G. Square Roots

We also try taking the square root of the binned sums instead of the sums alone. We can also perform this operation on the derivatives. This adds an extra nonlinearity to the data, which can increase the capacity of our decoder and enable it to be more expressive. When combining square root bins and derivatives to our above best decoder, we improve our MSE loss to **1352.28**. Given the increase in feature size, this is likely still despite more overfitting.

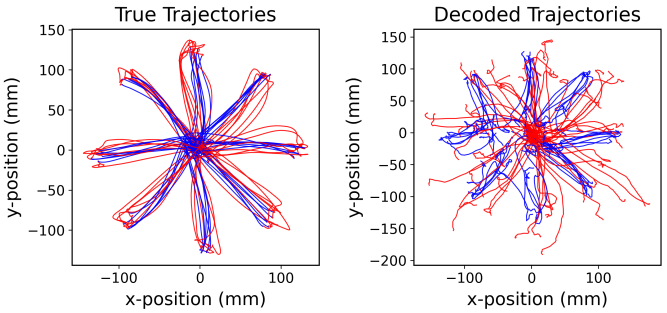


Fig. 7. Bin-summed with(out) sqrt + derivative with(out) sqrt + low-pass data.

### H. Normalization

We also try normalizing neural features per-feature. Per-neuron normalization has no effect as each normalization is linear, and a different linear transformation is applied to each neuron's data negating the effects of the normalization. We also find that normalization as a whole actually harms performance. This is likely because it removes information regarding the relative overall strength and activity of each neuron. Minimum MSE loss increases to 1453.33.

### I. Historical Features

Combining our insights from above, we combine binned data and derivative data both raw and with a square root, high-pass filters with $\sigma = 100$, low-pass Gaussian filters with $\sigma = 100$, and low-pass EMA with $\alpha = 0.05$). Given these, one major improvement we can make is encoding historical information into each timestep. We hypothesize that historical neuron activity is highly informative in forecasting future activity, beyond just the fixed 25 ms bin. We also hypothesize that more recent information is more important, and data further in the past is less informative.

To this end, we incorporated historical features into the setup described above. In particular, we process data by first stacking the above features, then computing an exponential moving average with $\alpha = \frac{2}{w+1}$ for window sizes $w = 2, 4, 8, 16$. This also greatly improves performance, although with higher computational cost. This setup is still causal and can be used for real-time decoding, as we only rely on past signal values. We display the results of tuning the $\ell_2$ norm on this setup (note that we find a higher optimal $\lambda$ as we have more features which demand more aggressive regularization), and visualize the best performing model.

TABLE VII
TUNING $\ell_2$ NORM ON HISTORICAL MODEL.

| $\lambda$ | MSE Loss |
|---|---|
| 0.001 | 2904.19 |
| 0.005 | 1733.47 |
| 0.01 | 1351.25 |
| 0.05 | **995.64** |
| 0.1 | 1035.16 |
| 0.5 | 1525.19 |

## V. Discussion

This study highlights the critical role of feature engineering in neural decoding, revealing the importance of nonlinear temporal and contextual features for predictive performance. Techniques like low-pass filtering and derivatives enhanced performance significantly. These methods demonstrate that respecting the temporal structure, noise characteristics, and nonlinearity of neural signals is crucial for effective decoding.

## VI. Future Directions

To further improve decoding performance, future work could explore the characteristics of the neural data distribution we uncovered from the weaknesses of linear decoders. Separating center-out and center-back reaches via a dual decoder is one such promising direction to handle these two modes of the overall distribution.

Additionally, nonlinear decoders like RNNs, LSTMs inherently capture long-range temporal dependencies aligning with our results. State-space models, Kalman filtering, and potentially WaveNet-style architectures may also improve accuracy with more data. Deep learning methods show potential for their ability to learn the best features from the raw data through hierarchical representations and cross-neuron information sharing, which we found to be crucial but challenging in our manual feature engineering.

Finally, dimensionality reduction techniques like PCA could identify key neural features while disentangling the highly intertwined networks that the neurons fire through. They can also improve computational complexity, making such decoders more practical. These steps, combined with the feature engineering insights in this report, can pave the way towards practical and accurate real-time applications for BMIs.