Recursive Bayesian Filter Based Neural Decoding Algorithm Estimating 2D Monkey Hand Trajectories

Amira El Fekih, Gauri Gupta, Iani Gayo and Joanna-Svilena Haralampieva
(The Spike Girls Team)
Department of Bioengineering, Imperial College London

Abstract -- **A neural decoding algorithm was designed to potentially steer a brain-machine interface controlled prosthetic device in a 2D environment. Its target angle prediction accuracy lies at 96% with a total run time of 4.941 seconds and an RSME of 35mm over 100 trials.**

I. Introduction

The aim of the group project was to implement a neural decoding algorithm estimating precise hand trajectories from recorded neural data of a monkey as it reaches for eight different targets on a fronto-parallel screen. The data, provided by the laboratory of Prof. Krishna Shenoy at Stanford University [1], consists of time-discretized spike trains of 98 neural units recorded over 100 trials for each of the reaching angles. Additionally, the simultaneous 3D hand trajectory data for each of those trials is given.

The continuous estimation algorithm accepts a set of training data (consisting of hand trajectories and neural data for all angles) for training the model in a supervised manner, and a set of testing data (consisting of neural data only) for model performance evaluation purposes as input. By the help of the trained model, the neural decoding algorithm classifies the specific target angle and causally estimates the monkey's x- and y- hand positions over time from the spike trains of the testing data. Hereby, consecutive 20ms increments of recorded data are being fed into the model.

The implemented algorithm draws upon concepts from general recursive Bayesian filtering applied to recorded firing rates, neural tuning curves and population vector analysis as well as nearest neighbor (NN) classifications for estimating specific target angles and predicting 2D hand trajectories.

The performance of the estimator is finally evaluated using the root mean squared error (RMSE) of the predicted 2D hand trajectories compared to the monkey's actual arm movements. Additionally, the total run time is captured.

II. Methods - Training

The 100 trials provided were split into training and testing with a 80:20 ratio. The aim of training is to learn four model parameters. These are then fed into the testing function.

A. Average Spike Rate

The average spike rate is how much each neural unit fires on average (over 80 trials) and is computed from the first 300ms of spike train data. During this time, the monkey's hand position remains stationary. However, it is planning its action and the spike pattern observed differs for every angle. This will be used to classify which angle the monkey is most probable to reach for given the first 300ms of test data.

B. Baseline Firing Rate

A 3D matrix of the average firing rate of each neural unit during movement across time for every trial and every angle is computed using the spike data from 300ms onwards. The mean, which is the baseline firing rate, and the standard deviation of this matrix computed across all trials and all angles will be used to standardize the firing rate values in the test data. Assuming the first 300ms were the planning stage before movement onset, they were ignored when calculating the baseline firing rate as the planning stage could have a different resting state than the movement stage.

C. Probability of Heading in a Certain Angle Direction Given the Observation of a Spike, P(A|S)

$$P(A|S) = \frac{(P(S|A) * P(A))}{P(S)} \quad (1)$$

The tuning curve [2] matrix is obtained by first normalizing the mean firing rates. The mean firing rates are computed by averaging the firing rates across the trials. Normalizing feature scales the mean firing rates such that the values are between zero to one for each angle. Secondly, the values in the matrix are converted to probabilities by dividing each element in the matrix by the respective sum of mean firing rates of each neural unit. From the tuning curve matrix, the probabilities of observing a spike for each neural unit given a known angle, P(S|A) in Equation -, can be extracted. Applying Bayes' theorem (1) to the tuning curve matrix allows the derivation of the training matrix, P(A|S), which is the probability of moving in one of the eight directions within a 20ms interval given spiking in the test data. Hereby, P(A) is the probability of observing one of the eight angles so is simply 1/8 due to the experiment setup. P(S) is a normalisation factor, that can be calculated from the summation of $P(S|A) * P(A)$ across all angles.

D. Displacement Matrix

The displacement matrix is the average Euclidean distance between successive x and y coordinates of hand positions every 20ms computed for each angle. This will help determine the magnitude of movement for each 20ms of test data.

III. Methods - Testing

The position estimator function consists of three main steps:

A. Nearest Neighbor (NN) Classification

The average spike rate is found from the first 300ms of the test data, and then fed to a NN classifier. This function calculates the two-norm of the difference between the average spike rate, with the eight average spike rates for every angle obtained from the training data. The target angle yielding the smallest norm is then assigned to this test set. This angle is then used to determine which magnitude vector from the displacement matrix is used for population vector analysis.

B. A Variation of Population Vector Analysis

The firing rate (FR) for every 20ms interval is found from 300ms onwards from the test data. Naturally in their resting state, some neurons may fire more than others. Therefore, standardisation (2) is performed to account for the natural firing rate of the neuron, and obtains a value that indicates how the firing rate differs relative to its usual baseline activity.

$$Standardised\ FR = \frac{FR - FR_{baseline}}{\sigma_{FR_{baseline}}} \quad (2)$$

A variation of population vector analysis [3] is performed using the standardized firing rates of every neuron. The training matrix, corresponding to P(A|S), is weighted by the standardised firing rates of every neuron (Standardised FR). The result is then summed over all neurons to obtain an 8x1 vector, which indicates the contribution of every angle as $Angle\_weight_i$ (3) towards the direction of movement within the 20ms interval.

$$Angle\_weight_i = \sum_{n=1}^{98} (P(A_i|S)_n * Standardised\ FR) \quad (3)$$

The final direction (4) is then found by multiplying each angle weight with the corresponding unit vector for that angle, where u_i corresponds to the unit direction for each angle. The obtained direction is then converted to a unit direction vector.

$$Direction(x,y) = \sum_{i}^{8} Angle\_weight_i * u_i \quad (4)$$

Now that the direction of movement is known, the magnitude of the displacement within this 20ms interval must be determined. For this, the mean of the first 10 values of the magnitude vector is taken as the scaling factor for every unit direction vector.

C. Update Training Matrix Using Bayes' Recursion Formula

Lastly, the training matrix P(A|S) is updated using the general Bayesian recursion formula (5), under the Markovian assumption that the angle A_t depends only on A_{t-1}[4].

$$P(A_t|S_1 \dots S_t) = \alpha * P(S_t|A) \sum_{A_{t-1}}^{8} P(A_t|A_{t-1})\, P(A_{t-1}|(S_1 \dots S_{t-1})) \quad (5)$$

$$\alpha = \frac{1}{\sum_{Angle=1}^{8} (P(S_t|A) \sum_{A_{t-1}}^{8} P(A_t|A_{t-1})\, P(A_{t-1}|(S_1 \dots S_{t-1}))}$$

$P(A_t|(S_1 \dots S_t))$: New posterior, i.e. updated training matrix
$P(S_t|A)$: Tuning Curve (Likelihood)
$P(A_t|A_{t-1})$: Transitional probabilities
$P(A_{t-1}|(S_1 \dots S_{t-1}))$: Previous posterior, ie previous training matrix
α : Normalisation factor incorporating probabilities of all angles

As well as the firing rate at the current 20ms interval, the firing rates from previous intervals can be used to indicate the probability of heading towards a certain direction, allowing for more accurate predictions of the monkey's trajectory.

The Bayesian recursion formula includes a transitional probability [4] matrix, that states the probability of moving to another angle, given the angle currently at. For this, the assumption was made that the probabilities follow a gaussian-like distribution, where the likelihood of moving to different angles decreases, as the difference between current angle to the new angle increases. For this, a normal distribution function is implemented that takes in standard deviation as a parameter. Finally, the new training matrix is found using the previous posterior (current training matrix) and the tuning curve found from training.

IV. Results

The results for the estimated hand position trajectories of two different trials are plotted in Fig. 1 and Fig. 2.

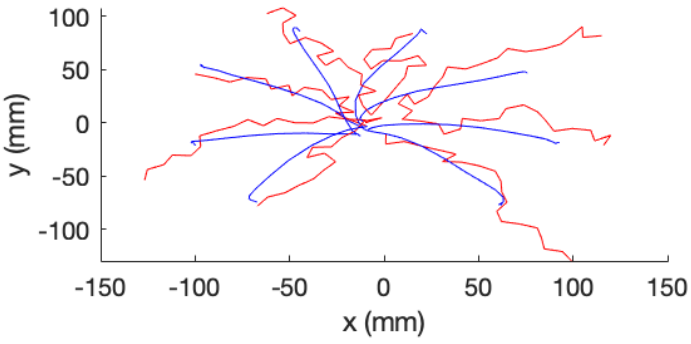

Figure 1: Actual (blue) and estimated(red) hand trajectories for single test trial, 8 different angles

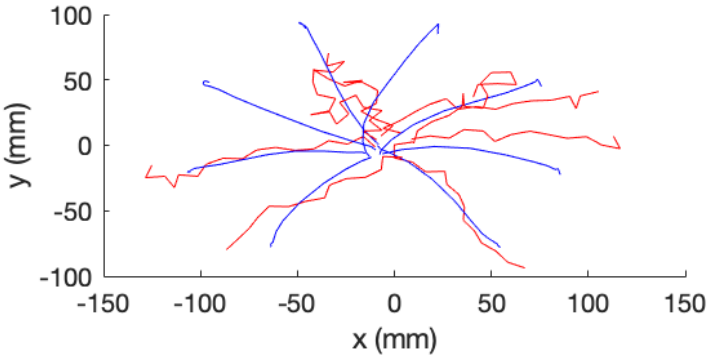

Figure 2: An example of a trial with less accurate estimated hand trajectories

The resulting RSME and algorithm run times are summarizes in Tab. 1. A confusion matrix for the accuracy of the NN classification is shown in Fig. 3.

	80 Train, 20 Test	**Theoretical: 100 Train, 80 Test**
RSME	35.342	N/A
Training time	0.213 s	0.267 s
Testing time	4.728 s	18.914 s
Total time	4.941 s	19.181 s

Table 1: RSME and computation time for 80:20 training:testing ratio, and inferred values for external100:80 training:testing ratio

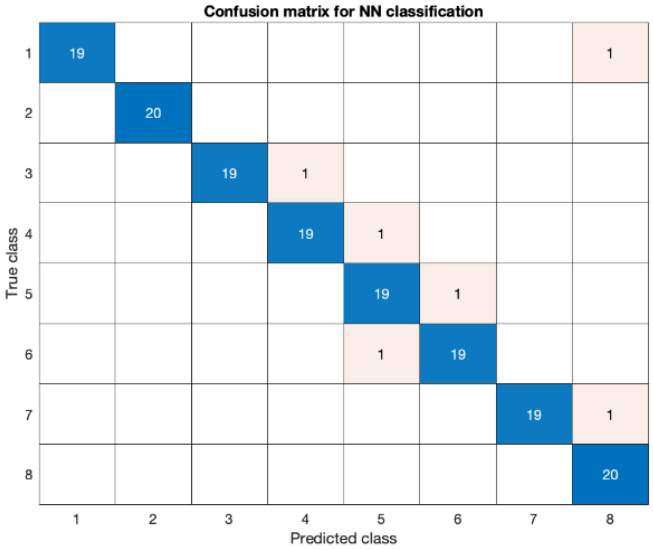

Figure 3: Confusion matrix which shows number of correctly predicted angles from NN classification

V. Discussion and Alternative Approaches

Overall, the implemented algorithm is relatively fast at approximately 4.9 seconds for an 80:20 training to testing data partition ratio (Tab. 1). Comparing trajectories from different trials, it was observed that except for a few cases where the algorithm might have misclassified, the predicted paths seem to lead in the correct direction (Fig. 1 and 2). In addition, the NN target angle classifier is highly accurate with a success rate of approximately 96% (Fig. 3).

The model extrapolates the direction of movement for a 20ms step for the test data, from the training matrix consisting of probabilities that are computed over the entire duration of the spike train of the training data. Thus, it rests on the rough assumption that in the training data, when going in a certain target direction, the monkey is mostly moving in that direction every step of the way (almost in a straight line from the centre to the final destination). This is a reasonable approximation seeing the shape of the actual paths in the training data.

However, with an RMSE of approximately 35 mm, the predicted path still seems to be erratic. The following discussion addresses the different components of the algorithm and how they can be improved.

A. *Bayesian Filtering Parameters*

The transitional probabilities of the Bayesian filter were modelled in a Gaussian-like distribution centered on the current angle with the most likely angles being the ones closest to it. This is to encourage less chaotic paths. The standard deviation was adjusted through trial and error. It can be optimized to yield more accurate results. Another way of making the estimated trajectory more consistent would be to bias all unit directions towards the classified target angle.

B. *Tuning Curve*

When constructing the tuning curve, values are averaged over time, thus valuable temporal information is lost. This is due to the fact the method is not specific for a time period in the spike train when, for ex., for a neural unit, P(S|A) could be consistently higher at a certain time than at another. More specifically, there could be repeating temporal patterns of firing. These patterns would have to be shorter than 20ms to be useful in this case. However, the data is very sparse, making 20ms patterns unlikely, which is why the decision was made to average over time and deal with spiking rates for the entire time length.

Another limitation of the tuning curve is that it does not take advantage of the prior knowledge of the target direction. In addition, the spatial resolution is restricted by the use of only 8 discrete angles to cover the entire 360 range.

To deal with these three problems, one method would be to segment the spike trains and the hand position data from the training data into 20ms intervals. Then, given the target direction, the directions of the 20ms hand position vectors could be categorized into sub-directions covering the 60 degrees around the target direction. Depending on the desired precision, there could be more or less vectors covering the 60 degree range. Subsequently, the 20ms interval spiking rates for all 98 neural units are grouped according to which of the sub-direction

vector it was classified as. Finally, all the 20ms spiking segments that correspond to a particular sub-direction vector are averaged to build a tuning curve for that sub-direction vector given the target direction (Fig. 4). In the testing phase, given the pre-computed target direction, the appropriate set of sub-direction tuning curves and thus training matrices they translate into (Bayes' theorem) can be selected. The problem with this method is that the training step might become very slow.

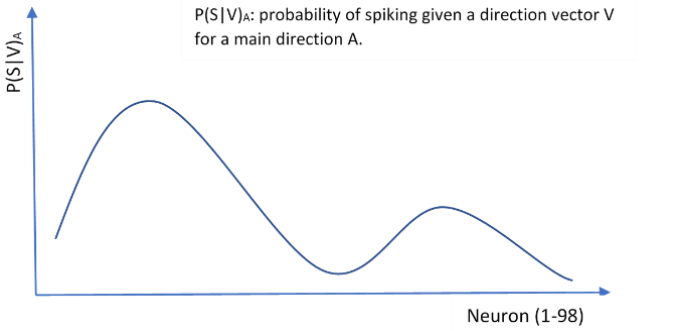

Figure 4: Representation of tuning curves for alternative method

C. Magnitude of Unit Vectors

The RMSE value was extremely high when the unit vectors were scaled by the magnitude vector for the corresponding 20ms interval. However, when the unit vectors were multiplied by an average of the magnitude vector for the first ten 20ms the RMSE value was at its lowest.

The first reason for this is that an individual value from the displacement matrix does not take into account the high standard deviation for the speed of movement between different trials within the same 20ms interval.

In addition, the standard deviation of our trajectory points is too high, meaning that if one incorrectly determined unit vector was scaled significantly more than the others, it is very likely to skew the entire trajectory towards an undesirable path. Hence it is preferable to deal with a total average movement speed across trials and time.

It was found that averaging for the first ten values gave the correct constant magnitude such that the scaled unit vectors' lengths add up to the correct length of the trajectory based on the amount of 20ms intervals received in total.

One compromise between both methods would have been to use a moving average filter on the displacement matrix. This method seemed plausible, but in practice when carried out whilst experimenting with the code - it was found that though the results were better than scaling by individual 20ms intervals, it was still not on par with the fixed averaging method.

D. Alternative and Additional Machine Learning Methods

A completely different, but potentially effective approach would have been to implement a 20-state hidden Markov model for every 20 ms snippet where every observation y_i is a single value from the spike train (1ms). However, this is a potentially a computationally intensive method.

With more time, spatial filtering like contrast enhancement and dimensionality reduction techniques (principal or independent component analysis) would have been used to achieve less redundancy and more significant spiking data.

ATTRIBUTIONS

All group members contributed equally to writing the report as well as implementing the MATLAB codes submitted.

REFERENCES

[1] https://shenoy.people.stanford.edu/overview (Acessed: 20th April 2020)

[2] C.Clopath, Neural Decoding Competition Rulebook, BMI Spring 2020, p. 4

[3] http://www.math.pitt.edu/~bdoiron/assets/dayan_abbott.pdf (Acessed: 10th March 2020)

[4] C. Clopath, Brain-Machine Interfaces - Lecture 3, "Bayesian Filtering", Imperial College London, London, 30th January 2020