Neural decoding for Prosthetic Control: Insights from Brain-Machine Interface Research

A. Berthon, T. Bru, D. d'Haultfoeuille, Y. Wang

Dept. of Bioengineering, Imperial College London, London, UK

Abstract :

The focus of most present-day Brain-Machine Interfaces research and development is the replacement or restoration of lost natural outputs, like motor commands. This paper presents a typical approach to designing a neural decoder for controlling a hypothetical prosthetic device using intracortical recordings from a macaque performing 3D target-reaching movements. Using a K-Nearest Neighbors (KNN) algorithm to predict the target angle and spike density based on Peri-Stimulus Time Histograms (PSTH), we were able to predicting the trajectory of the hand in real-time. Our results demonstrate the efficiency of this approach based on neural activity patterns.

Index Terms: Brain Machine Interface, Neural Decoder

1 INTRODUCTION

The brain has a remarkable ability to learn, adapt, predict, and generate motor commands to execute various tasks. Understanding the relationship between the neural activities in the motor cortex and the resultant movements could be essential to the development of medical devices for patients with physical disabilities. A Brain-Machine Interface (BMI) translates the Central Nervous System (CNS) activities into artificial outputs, altering its interactions with both the internal and external environments. In particular, it employs decoding algorithms that analyse brain activities and convert them into commands effective in driving the devices, such as a robotic arm.

In this paper, we will describe and explain the design of a neural decoder to control a hypothetical prosthetic device. It uses intracortical recordings from a monkey performing various 3D target-reaching movements in order to predict the physical trajectory of its hand during each reaching process. To achieve real-time precision of the prosthesis, both accuracy and efficiency must be prioritised for the selected decoding algorithms. The performance of the algorithms implemented will be discussed, as well as their potential improvements.

2 BACKGROUNDS

2.1 Experiment

The data we used for the decoder comes from the experiment conducted in the laboratory of Prof. Krishna Shenoy at Stanford University. Hundreds of electrodes were implanted into the premotor cortex of a macaque. They recorded the neural activity, also known as the spikes, while the monkey made its movements. The monkey reached for targets on a fronto-parallel screen with its hand position being tracked simultaneously.
During the tasks, the monkey was instructed to reach for 8 different target points, each corresponding to a distinct angle ($30\pi/180$, $70\pi/180$, $110\pi/180$, $150\pi/180$, $190\pi/180$, $230\pi/180$, $310\pi/180$, $350\pi/180$). Each angle was reached for 182 times.

2.2 Data description

The monkey had to perform a 3D movement with its arm. However, the decoder predicts a 2D trajectory, as most of the arm movements were in the plane direction along the horizontal axis, with minimal vertical movements (perpendicular to the plane) only.

Although 182 trials have been recorded for each angle, we only had access to 100 of them, with the remaining 82 trials reserved by the research team for the performance assessment. Thus, we have exploited in total 8*100 = 800 trials of data.
The dataset contains time-discrete spike-trains from 98 neural units, with 30% from well-isolated single-neuron and the rest from multi-neurons units, as well as the hand positions for each trial (x, y, z). Each spike train is made up of a sequence of 0 and 1, where 1 represents a firing and 0 for no activity.

It captures each millisecond of neural activity and arm movement, with recordings spanning from 300 milliseconds before to 100 milliseconds after each movement, providing a comprehensive temporal window of the monkey's neural activity and corresponding hand positions.

3 METHODS

Since our aim was to predict the trajectory of the monkey's movements with maximum accuracy, we have divided this goal into 2 main sub-tasks. First, we needed to determine which angle is more likely to be reached by the monkey based on its neural activity, i.e the neurons firing rate at each time bin. Then, given the estimated angle we had to develop a model predicting the trajectory of the monkey's hand following this direction.

3.1 Data pre-processing

In order to evaluate the performance of our classifiers later on, we divided the data into a training data set containing the information of the first 78 trials, and a test data set for the remaining 22 trials. We optimised this ratio to balance the model accuracy with computational efficiency.

3.1.1 Model parameters For this part, we needed to extract essential parameters from the training set that would be used later in the decoder.

As neural recordings often contain noises and variabilities caused by external factors, we averaged the spiking train across all trials to cancel out these unwanted effects. Thus, for each angle i and spikes contained in a 20ms-interval, we computed the mean of the position and the velocity across all trials and within the duration of the trials.

Subsequently, we calculated the mean firing rate as a spike density for a specific neuron i and an angle j based on Peri-Stimulus-Time Histograms (PSTH). We obtained 8x98 PSTHs in total for the training data. The formula used for the PSTH is given as follows ([2]):

$$\text{PSTH}_{j,i} = \frac{1}{\Delta t \cdot M} \cdot n_{j,i}(t, t + \Delta t)$$

- j: index of the angle.
- i: index of the neuron $\in [1; 98]$
- t: the time bin size, $t = 20ms$.
- M: number of trials in the training dataset, so $M = 78$.
- $n_{j,i}(t, t+\Delta t)$: number of spikes summed over all the training trials for the angle j and the neuron i between the time points t and $t + \Delta t$.

One problem with the dataset was that not all recordings were of the same length. This posed dimension compatibility issues later during the implementations of the K-Nearest Neighbors (KNN) and the trajectory prediction algorithms. To amend this and standardise the data formats, we added to the parameters the value of the minimum length ($size_{min}$) of the recordings for each of the angles.

3.1.2 Test dataset As for the training dataset, we calculated the mean firing rate of the test data set using PSTH over the first 320ms.

3.2 Classification for target angle prediction

The first task required a good classification model to associate specific neuron firing patterns with the corresponding targeted angle. The KNN algorithm has proven its robustness, while being relatively simple and intuitive to implement. Given our data preprocessing steps where we averaged the firing rates for each of the 8 angles across all training trials, we set K, the number of nearest neighbors, to 1. This setting was necessary because each angle was represented by a single averaged data point in our model. The purpose of our KNN algorithm was to determine which of the eight angle's spike train was the most similar to a new spike train taken from the test data set. To do this, we calculated the Euclidian distances between the PSTH value of the new data with the PSTH value of each angle (all computed over the first 320ms of recording). We selected the angle for which the distance is minimal. The operation of the Kalman filter we have implemented is described on the schematic figure 1.

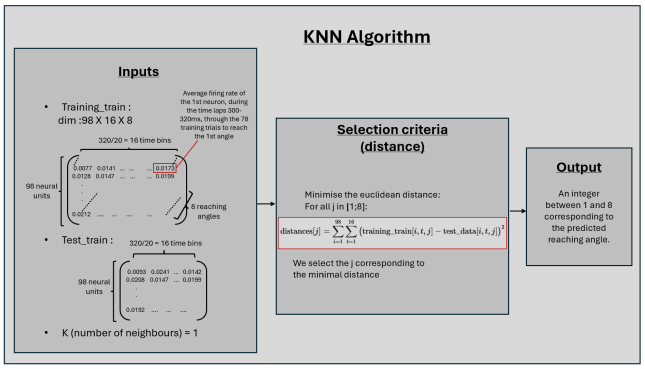

Figure 1. Diagram of operation and variables implemented in the KNN algorithm.

The KNN method is advantageous as it does not make any assumption about the underlying data distribution, and it is capable of capturing complex and non-linear relationships without explicitly modeling the distribution.

3.3 Trajectory prediction

The second task involved the prediction of the actual trajectory.

The first approach we tried was also the one we kept as it yielded the best results. After predicting the angle, we used the mean position $\bar{x}$ and $\bar{y}$ associated to this angle - defined as one of the model parameters during the pre-processing step.
The main advantage of the "*mean position*" approach lied with the computation time and its robustness - For one, there was no additional calculation to make. On the other hand, there were few or no deviations leading to unstable predictions. However, this method does not rely on the spike trains of the new test data, and therefore it can not be generalised for other experiments that might include more possible target angles or a continuous trajectory for example.

An alternative solution to the trajectory prediction was to use a Kalman filter, as its efficiency has been acknowledged in many BMI applications for neural decoding ([3]). However, the results presented low accuracy, with a Root Mean Squared Error (RMSE) around 40 cm. Thus, we have abandoned this method, even though the unpleasing results could be due to errors occurred during the implementation stage (see 5).

4 RESULTS

To examine the performance of our algorithms, we calculated the RMSE between the test hand's vector positions x_i and y_i and the estimated hand positions $\widehat{x_i}$ and $\widehat{y_i}$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \widehat{x_i})^2 + (y_i - \widehat{y_i})^2}$$

given that N is equal to the number of position predictions.

We obtained a RMSE value of 15.370 cm. The running process lasted for 31.757 sec. The results of learning with 176 (22*8)

reaching movements in the eight directions are shown in Figure 2.

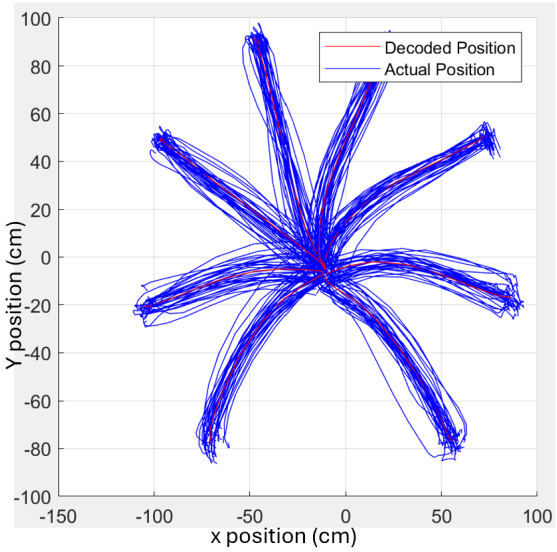

Figure 2. Simulation of reaching movements in eights directions. Movements were performed from central to peripheral targets.

To investigate the causes of error in our model, we looked at the accuracy of our KNN to see whether we should focus our improvement efforts on the first or the second part. We determined that the accuracy of our KNN was around 96% (depending on the training trials/test trials split). We also obtained the following confusion matrix (Figure 3):

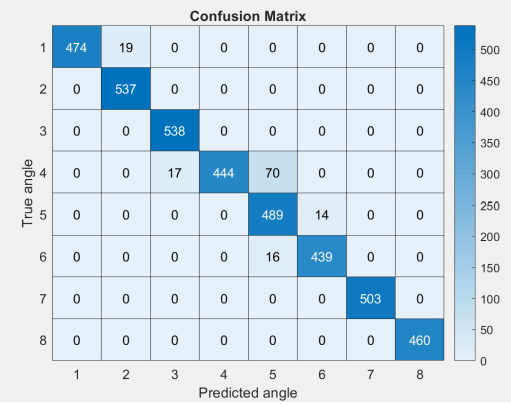

Figure 3. Confusion matrix of the KNN algorithm used to predict the movement angle, demonstrating little inaccuracies.

This matrix shows little inaccuracies. The predicted angle is most of the time accurate, or close to the actual angle. The overall error level is then null or relatively small.

5 DISCUSSION

The results of our final model placed us in the 11^{th} position of the competition, given that the winning team obtained a RMSE of 6.966 cm and a running time of 31.461 seconds.
Our BMI has yielded satisfactory results, although there are still many ways of improving that have not been implemented due to the time constraint and the varied levels of programming experience within our group. We will discuss in the following subsection the alternative solutions that have been considered.

5.1 Selection of neurons

A selection of appropriate neurons could have been employed during the pre-processing stage of the data set, as some neurons could exhibit similar rates of firing, indicating potentially some redundancy or erroneous clustering during spike sorting. On top of that, certain neurons have shown minimal firing activity, while some others could be affected by external noises. Thus, the model was partially trained on irrelevant, redundant, or useless datasets, which lowered its final accuracy. We could have then adopted a manual selection of relevant neurons, discarding the redundant, noisy ones and retaining only tuned neurons. This step would help optimise both the execution and the learning processes. Finally, the selection process could have been performed with techniques such as Principal Component Analysis (PCA) ([4]) or ANOVA, something we have tried to implement before but with no success.

5.2 Angle prediction

The KNN algorithm only took the first 320 milliseconds of neural data as the input, as it is the time period we have access to at the start (i.e. the first time our estimation function is called). For the sake of simplicity, and given our final results were already quite satisfactory, we have decided not to change it. To perfect the algorithm, we could have expanded our approach by using all available data up to the current time t in the test dataset, which progressively increases with each function call. With this more extensive dataset, we could potentially improve motion prediction accuracy further, especially in cases where the initial few milliseconds of data are insufficient for accurate determination of the movement angle.

5.3 Trajectory prediction

5.3.1 Time restriction Our trajectory prediction model was restricted by the constraint we gave to ensure data uniformity. Indeed, we only considered data falling within the time limit imposed, i.e the minimum recording length over all trials for each angle. Thus, when the test trial was longer than this time limit, for each time point over the limit, the predicted position was fixed at the last averaged position of the training trials (at $t = min_{size}$). We therefore did not take into account changes of the position over time. One idea for improvement would be to continue evolving this position for longer test recordings, for example with interpolation.

5.3.2 Kalman filter As mentioned above, a classical Kalman filter for the trajectory decoder should have been a suitable choice, as it has shown great efficiency in our competitors' models.

5.3.3 Neural network We also deployed a neural network for angle prediction that achieved an acceptable high accuracy range between 90% to 94%. This model, structured as a multi-layer perceptron with a hidden layer of 25 units and ReLU activation, effectively captured complex patterns from the neural firing data. However, the K-Nearest Neighbors model has not only exceeded

the neural network's prediction accuracy but also acquired significantly better running-time efficiency.

6 CONCLUSION

We have presented an approach for designing a neural decoder intended to control a theoretical prosthetic device. Using the K-Nearest Neighbors algorithm to identify the intended angle of movement based on a monkey's neural activities, as well as the mean position for trajectory prediction, our model can learn swiftly from limited training data and provide real-time estimations of the monkey's hand position with good accuracy.

However, in real-world scenarios, individuals with impairments might aim to reach for targets from more than certain confined angles. This introduces added complexity as it requires appropriate adaptation of the classification process for both the initial angle-predicting task and the trajectory decoder.

Acknowledgements

This research have been carried out for a competition within the module of Brain-Machine Interface, instructed by Prof. Periklis Pantazis and Prof. Claudia Clopath.

Attributions

We did our best to share and divide the workload, especially in terms of the different steps of the approach. Nevertheless, due to the significant variance in programming competence and background, the development has been carried out mostly by a minority.

The completion of the different parts has been divided up as follows:

- Alexia: Data pre-processing and KNN algorithm, trajectory estimation (using the mean position) and report redaction.
- Thomas: Implementation of a neural network for angle and trajectory predictions, optimisation of KNN (tried to use neurons combinations) and report redaction.
- Diane: Trajectory estimation using the Kalman Filter, running time optimization and report redaction.
- Yitong: Attempted implementation of a neural network for angle prediction, slight error reduction for the Kalman Filter and report polishing.

REFERENCES

[1] Rao RPN, "Brain-Computer Interfacing: An Introduction", *Cambridge University Press*, p. 39-98, 2013.

[2] Wulfram Gerstner and Werner M. Kistler, "Spiking Neuron Models - Single Neurons, Populations, Plasticity", *Cambridge University Press*, 2002.

[3] Wu, Wei & Black, Michael & Gao, *et al.*, "Neural Decoding of Cursor Motion Using a Kalman Filter", *Brown University*, p. 117 - 124, 2002.

[4] Duncan Gillies, "DOC493: Intelligent Data Analysis and Probabilistic Inference", *Lecture 15*, 2011.