A Coarse-Grain Algorithm for Decoding Arm Trajectories from Primate Motor Cortex Activity

In completion of BE9-MBMI Brain-Machine Interfaces (2018-2019) at Imperial College London

Shafa Balaram*, Alex Dack†, Adel Haddad‡ and Aishwarya Pattar§
Email: *sb3215@ic.ac.uk, †ad1114@ic.ac.uk, ‡afh115@ic.ac.uk, §ajp15@ic.ac.uk

***Abstract*—The development of algorithms for human centric brain-machine interfaces are of major importance for future advanced medical technologies. A novel biologically inspired coarse-grain algorithm for decoding arm movements from motor neuron firing patterns has been theorised. This work goes on to implement the algorithm on monkey neural data and benchmarks the algorithm against other standard techniques. Despite the simplicity of the algorithm, its strong performance demonstrates the efficacy of the biologically inspired paradigm. The work is then concluded with opportunities for further development of the algorithm.**

***Index Terms*—BMI, k-nearest neighbour, NLMS, LSTM**

I. Introduction

As part of the course Brain-Machine Interfaces, small teams of students were required to develop an algorithm to decode monkey brain data that would execute within five minutes. The aim of the competition was to take neural spiking data recorded from the motor cortex, and apply signal processing techniques to produce a temporal prediction of arm position. This report details the approach, inspiration and efficacy of the algorithm developed by the authors.

Given the application context, the guiding principle for the design of the algorithm was to mimic biological insight with simple computational techniques. The rationale for this was that treating the system as a black box function makes the algorithm prone to overfitting and non-physical outputs.

It is known that primates plan their arm movements in a general direction, which is detectable as a firing pattern in the pre-motor cortex [1], then the movement is executed by firing in the motor cortex [2]. Hence, it was decided that the neural recordings before the onset of the movement could be exploited to extract useful information.

We decided to mimic the system by designing our algorithm to be a coarse-grain approach. That is, the algorithm performs a classification on the pre-movement data which returns an approximate angle from a small set of possible angles. Then, according to the output of the classification, the pre-trained weights of a regressive model are selected and the temporal arm position is returned as the output.

Despite the application to monkeys it is not hard to imagine that such tasks help to lay the groundwork for the future development of algorithms for human centric brain-machine interfaces. Such work has the potential for application in invasive BMI devices for recreating movement for patients with motor impairments.

II. Approaching the Problem

A. Description of Data

Primate neural data was provided by the laboratory of Prof. Krishna Shenoy at Stanford University. The data structure consists of `N` trials for each of the `8` reaching angles. Therefore, there are `8N` experiments in total. Each experiment consists of 1kHz sampled spike trains from `98` neurons and the corresponding hand position trajectory in the (x, y, z) coordinates given in mm; this data was taken from 300ms before the onset of movement to 100ms after movement completion.

Provided the task only required predictions in the (x, y) coordinates, the z-axis was investigated and found to have low variation, so it was possible to assume that the effect of this axis was negligible.

B. Pre-movement Classification

Existing work [1] demonstrates that there is neural electrical activity before the onset of movement, correlating to movement intention. In light of this, we wanted to be able to identify the intended movement.

As the data set provides `8` reaching angles, this task was formulated as a classification problem. There are many machine learning approaches that are suitable for this task, however in our experimentation the k-nearest centroid on an extracted feature space was our preferred approach.

C. Hand Trajectory Prediction

As the hand position trajectory is a continuous output that varies temporally, it was then necessary to apply a regression algorithm. In line with our coarse-grain approach, the pre-movement classification was used to select the model that had been trained on the corresponding reaching angle.

For the trajectory prediction we used an approach that combined the neuron population vector encoding with an NLMS adaptive filter. A population vector weights the input by each neuron's preferred direction. The NLMS allows for the weights to adjust according to the expected position.

This model was selected as it is efficient to implement, grounded in biological evidence [2] and allows for the addition of layers of complexity.

D. Mathematical Theory behind the NLMS Algorithm [3]

The Least Mean Square (LMS) algorithm is an adaptive filter which can be used to predict one-step ahead. It can be applied to non-stationary signals, $y(n)$, at each time-step n and is parameterized by its filter length M.

The prediction $\hat{y}(n)$ is the scalar product between the signals $\mathbf{w}(n)$ and $\mathbf{y}(n)$:

$$\hat{y}(n) = \mathbf{w}^T(n)\mathbf{y}(n) = \sum_{m=1}^{M} w(m,n)y(n-m) \tag{1}$$

Where $\mathbf{w}(n) \in \mathbb{R}^{M\times 1}$ is the filter weights and $\mathbf{y}(n) \in \mathbb{R}^{M\times 1}$ is the M previous values of the input teaching signal.

The weights $\mathbf{w}(n)$ are recursively updated by the minimisation of the instantaneous cost function:

$$J = \frac{1}{2}e^2(n)$$

Where $e(n)$ is the deviation of the prediction $\hat{y}(n)$ from the teaching signal $y(n)$:

$$e(n) = y(n) - \hat{y}(n) \tag{2}$$

The minimisation is implemented using the gradient method, with the weights initialised to zero:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}}J(n) \tag{3}$$

Where μ is the constant step-size. An expression can be derived for $\nabla_{\mathbf{w}}J(n)$ using the chain rule:

$$\begin{aligned}\nabla_{\mathbf{w}}J(n) &= \frac{\partial J(n)}{\partial e(n)}\frac{\partial e(n)}{\partial y(n)}\frac{\partial y(n)}{\partial \mathbf{w}(n)} \\ &= -e(n)\mathbf{y}(n)\end{aligned} \tag{4}$$

Hence, the weight update equation becomes:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{y}(n) \tag{5}$$

Eq.(5) shows that the performance of the LMS filter is dependent on the magnitude of the learning rate. While a larger step-size has been shown to increase the speed of convergence of the weights, the trade-off is the increased error between the steady-state prediction and the teaching signal [4]. Hence, an NLMS filter is implemented with an adaptive step-size $\mu(n)$ which is obtained by normalising the fixed step-size μ_0 by the power of the filter input signal $\mathbf{y}(n)$:

$$\mu(n) = \frac{\mu_0}{||\mathbf{y}(n)||_2^2 + \epsilon} \tag{6}$$

Where ϵ is a small regularisation constant added to ensure that the step-size remains bounded for small filter input values.

III. Methods

A. Training

The whole data set was used to train the model to identify the reach angle and trajectory of the monkey's arm.

Firstly, classification of the feature space was implemented to characterise each reaching angle. The following methodology was implemented:

a) For every reaching angle in each trial, and for all of the 98 neurons, the mean and standard deviation of spike rate was calculated for the first 300ms. This time frame was chosen as it is the period before the onset of movement. The 2D matrix containing this data was of size `8N × (98 × 2)`.
b) Then, for each of the 98 neuron across all experiments, the reaching angle averaged mean spike rate and standard deviation was calculated. The values were outputted by the classification function as `meanXTrain`, a row vector, and subtracted from the 2D matrix in step a) to normalise the data.
c) Using the normalised data, for each angle, every neuron's mean and standard deviation spike rate was averaged across all trials. This resulted in each of the 98 neurons having an average mean and average standard deviation for each reaching angle, which defines the angle's centroid. Each angle's centroid was returned by the classification function as a 2D matrix of size `8 × (98 × 2)`

The returned `meanXTrain` and centroids were then used in the Testing phase to identify the monkey's intended angle of movement.

To determine the weights to predict the arm movement trajectory, the neuronal activity post movement (i.e. after 300ms) was analysed:

a) For each neuron across all experiments, the mean and standard deviation of the spike rate was calculated with a sliding 20ms window from after the first 300ms up to 100ms before each experiment ended.
b) These values were then averaged across all experiments to obtain an average mean and average standard deviation of spike rate for each neuron for a given reach angle. The data was of size `8 × 98`.
c) All neurons were then ordered in descending value of standard deviation and then for each neuron, the preferred direction was identified as the angle with the highest standard deviation. In cases where multiple reach angles shared standard deviations within 10% of the maximum, the average of these reach angles was set as the preferred angle.
d) For each reach angle the following properties were found: the average (x, y) trajectory, the maximum change in (x, y) per 20ms time step, and the furthest position the monkey reached from its start point.

The NLMS algorithm was then implemented to train each neuron's spike rate weights for each of the reach angles. In equations (1), (2), (3): $y(n)$ is the average trajectory and $\hat{y}(n)$ is the predicted trajectory calculated using the mean neural activity in the previous 20ms period. The variables for NLMS were initialised as $\mu = 0.5$ and $\epsilon = 0.02$. The weight vectors for each angle were then used during the Testing phase.

B. Testing

The k-nearest centroid method, where $k = 1$, was used with the first 300ms of testing data to classify the angle of the reaching direction. The identified reach angle was used to select reach angle specific weights, obtained from training, to predict the change of the arm movement based on the mean spike rate of the last 20ms of testing data.

The arm trajectory was incrementally determined with each additional 20ms sample of data provided. Mean neuronal activity was calculated from the latest 20ms of data and multiplied by the corresponding weights to obtain the position. If at any time the predicted change in arm movement was greater than the reach angle specific maximum found in training, the change was capped to this maximum. This process continued until one of the following stopping criteria was met:

- The trajectory had reached the furthest point the monkey can reach for the given angle. The trajectory was then capped at this value because it is impossible for the monkey to reach further than its arm length, lim_{arm}.
- If all the weights have been used and no more remain for further prediction, the trajectory remains at its last point irrespective of if more data is provided. This is because prediction cannot be continued.
- When no more data samples are provided to the algorithm.

C. *Optimisation*

The testing scheme employed involved first provided 320ms of data, then after every step an additional 20ms was provided such that the algorithms received 320ms then 340ms, etc.

Due to the execution time constraint, the code was optimised to the testing scheme to prevent repeated calculations during run time. This involved removing any repeated classifications and predicting trajectories using only the most recent 40ms of data.

To reduce dimensionality, thereby minimising computations, it was attempted to select neurons with spike rate standard deviation greater than 85% of the maximum standard deviation for both training and testing, as opposed to using all neurons as seen in other previous work [2].

To make the highly discrete predictions appear smoother, an exponential filter was implemented. This was coupled with an (x, y) change limit, lim_{xyStep}, to cap all predictions to at most 20% more than the maximum (x, y) change from the training set.

D. *Other Methods*

Recurrent neural networks (RNNs) have been used successfully in sequence-to-sequence prediction applications owing to their internal memory. Hence, the authors have opted for a special type of RNN architecture that can capture long-term time dependencies without being afflicted with vanishing gradients, the long short-term memory (LSTM) [5], for the task of predicting the hand position of the monkey given its neural spike signals. The *MATLAB Deep Learning Toolbox* was used [6] to implement an LSTM architecture which outputs a sequence of (x, y) locations of the monkey's hand given the input features derived from the raw neural spike signals recorded. Neural data post movement of length 325ms was extracted across all the `8N` experiments. The cumulative neural spike rate is calculated before computing the mean and standard deviation across the different recording units. Hence, the features and hand position datasets are of size `8N × (98 × 2) × 325` and `8N × 2 × 325`. The datasets are then shuffled randomly before being split into a ratio of 80% and 20% for training and validation respectively.

The LSTM architecture and hyperparameters for training on a CPU were as follows:

a) It has a sequence input layer of the same size as the features (`98 × 2`), 100 hidden units and a fully-connected layer to predict the (x, y) location using regression.
b) An Adam optimiser was used with an adaptive learning rate, initially at 0.01 and halving every 50 epochs. The gradient was also clipped at a threshold of 1.
c) A mini-batch of size 160 is forward and back-propagated through time for each of the 300 epochs.

IV. Results

Algorithm Type	**RMSE$_{98}$**	**RMSE$_{35}$**	**Testing Time** (98/35)
NLMS	21.0470	29.6934	132.81s/60.71s
NLMS + lim_{arm}	16.8454	19.3159	142.96s/58.71s
NLMS +lim_{xyStep}	21.0463	28.3750	152.44s/59.81s
NLMS + lim_{arm} + lim_{xyStep}	16.8473	19.2937	129.35s/56.81s
LSTM	8.45	11.0	≈ 30 mins / ≈ 20 mins

TABLE I: Algorithm RMSE comparison
lim_{arm}: Arm length constraint, lim_{xyStep}: (x, y) change constraint

A. *NLMS*

There were 35 neurons that had a spike rate standard deviation above 85% of the set's maximum standard deviation. As can be seen in Table I, the RMSE is better (lower) when using the entire neuron set.

Training time was on average 25.44s, for both 98 and the 35 neurons. Testing time was on average 143.21s or 58.41s for using 95 or 35 neurons respectively.

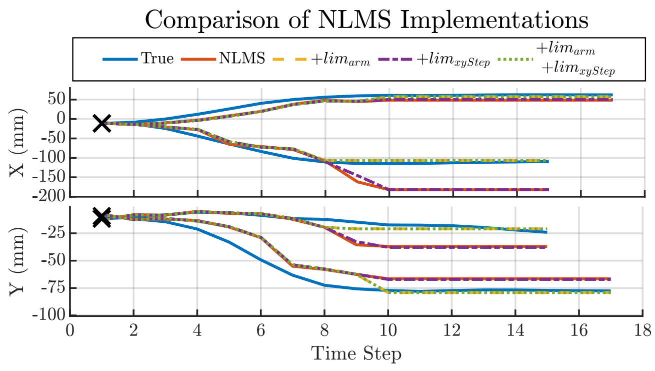

Fig. 1: NLMS with different constraints, 98 neurons, 2 angles

B. *Other Methods*

The LSTM achieved RMSE's of 3.74 in the training dataset and 8.45 in the testing dataset. When using the selected 35 neurons, the RMSEs increased to 6.79 and 11.0 respectively.

V. Discussion

A. Classification Step

Being the first step of the algorithm, the performance of the classification is vital as the regression is dependent on its output. The classification used achieved a respectable 96% categorical accuracy.

The use of the mean and standard deviation of individual neurons as a feature space produced a substantial improvement compared to using the data without feature extraction, or just the mean by itself. This might be a result of the physical importance of the Fano factor, the ratio of variance to mean. Despite the computational cost of extracting the features, the comparison step only requires finding the minimum Euclidean distance between the point to be classified and the 8 centroids that correspond to the 8 reaching angles. This has a good computational complexity for the performance that is being achieved.

There are several disadvantages of using a pre-movement classification step. Firstly, the regression is dependent on the output so if there is a misclassification this error will be magnified by the regression model. Moreover, due to the structure of the experiment containing only 8 reaching angles, the algorithm is designed for this very specific situation. If the monkey's hand moves in a direction between two discrete consecutive reaching angles, the algorithm is likely to underperform. Thus, it can be hypothesised that the algorithm has been overfitted to the current scenario rather than the data.

To overcome the misclassification of the reaching angle, other machine learning techniques could be used: a multi-layered perceptron with a softmax output would improve the categorical accuracy without requiring a feature extraction step, improving time complexity but falls back on space complexity and is prone to overfitting; replacing the Euclidean distance metric with an alternative metric computation or a learnt distance metric would improve classification accuracy and be very similar in time complexity.

An approach that overcomes the discussed limitations from the limited number of discrete reaching angles, would be to remove the classification step and use a regression step for the entire signal.

B. Regression Step

The final NLMS-based prediction algorithm has a RMSE of 16.85. This is not a state-of-the-art result, however, its simplicity made it reasonable to implement in the time-frame with the added benefit of having an intuitive result, than that of a Machine Learning black box. Furthermore, the output produced physically realistic predictions and extreme events could be prevented, especially when implementing naturalistic constraints to the output.

The results in Table I, show the effect of lim_{xyStep} is less significant than the effect of lim_{arm}. However, its influence becomes more significant when NLMS results were poor, such as when using fewer neurons.

There are many disadvantages to this approach, which leaves opportunities for improvement. Firstly, the output is not smooth and appears to jump in discrete steps, which is a result of the discrete nature of the NLMS. Secondly, the NLMS is an adaptive filter which means that it ideally would receive the true value back in real-time to update its future predictions. This is highlighted in Fig. 1, where there are only enough weights for 9 prediction steps, as a result of training the algorithm on the duration of the shortest trial. This explains the improved performance on the lim_{arm} method, as it sets the (x, y) position to the extended arm length if the predictor runs out of weights, or when the prediction suggests an arm extension outside the extended arm length. This all suggests that this type of algorithm might be slightly inappropriate for this exact task. However, in other applications of BMI, somatosensory feedback has been used to help reinforce learning, hence this algorithm could be useful in future applications.

As the algorithm is modular and it has been identified that the regression step is the area that requires the most improvement, the LSTM approach was investigated. Moreover, the LSTM acted as a baseline to compare our approach as it is a state-of-the-art technique that accounts for non-linearities that are likely present in this biological system.

The LSTM allows for regression of a prediction and is much more generalisable, it learns directly from the dataset and does not account for the biological relevance of the data. In addition to the initial classification step, the hyperparameters could be tuned and with increased computational power such as a GPU, an optimum performance could be achieved in a shorter training time. Coupling the initial classification step with the LSTM would could possibly improve performance. However, the LSTM's improvement comes at the cost of substantially longer training time and the use of existing toolboxes, both of which were outside the competition guidelines.

VI. Conclusion

In conclusion, a coarse-grain algorithm that uses a k-nearest centroid classification step to select appropriate weights for a population vector predictor, trained using an NLMS adaptive filter, has been implemented and reviewed. Despite the efficacy of the classification and the competitive overall performance, a number of limitations and future developments for the algorithm have been highlighted.

Contributions

Report Writing *†‡§, Code *†‡§, Literature Review §.

References

[1] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor cortex and the recognition of motor actions," *Cognitive Brain Research*, vol. 3, no. 2, pp. 131 – 141, 1996. Mental representations of motor acts.

[2] A. Georgopoulos, J. Kalaska, R. Caminiti, and J. Massey, "On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex," *Journal of Neuroscience*, vol. 2, no. 11, pp. 1527–1537, 1982.

[3] D. Mandic, *Adaptive Signal Processing & Machine Intelligence lecture notes*. 2019.

[4] S. Haykin, *Adaptive Filter Theory*. Pearson Education India, 5th ed., 2014.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov 1997.

[6] "MATLAB Deep Learning Tutorial." https://uk.mathworks.com/help/deeplearning/examples/time-series-forecasting-using-deep-learning.html. [Online; last accessed Apr. 2019].