A neural decoding algorithm for continuous position estimation

P. Denning, J.Kim, M. Tonutti (*Mo Monkeys Mo Problems* Team)
Dept. of Bioengineering, Imperial College London, London, UK
email: patrick.denning11@ic.ac.uk, jiyoon.kim11@ic.ac.uk, michele.tonutti11@ic.ac.uk

Abstract— **Spinal cord injuries and other types of trauma can result in complete or partial paralysis for the patient; however, neural function often remains working. Decoding algorithms can be used to obtain and use neural information from the motor cortex in order to assist patients with paralysis, for example to control prosthetic devices. In this paper a fully functional decoder for continuous position estimation is built, making use of a linear regression algorithm and k-nearest neighbours classification. Data had previously been obtained from monkeys performing a centre-out task with targets appearing at 8 different angles. With a set of training data of 98 units and 800 trials, our decoder successfully predicted the intended direction of the movement, given only neural information, with a confidence of 91±1%, and it predicted the continous trajectory of the monkeys arm with a root mean squared error of 18.2 centimetres. Additionally, a variety of methods of classification (such as Support Vector Machines) and decoding parameters are tested and analysed.**

Keywords: neural decoding; linear regression; k-nearest neighbours classification; support vector machine; brain-machine interfaces.

I. INTRODUCTION

Brain-machine interface systems have a multitude of application, including the use of robotic assistive devices by patients with tetra-/paraplegia caused by head trauma, spinal cord injury, stroke or multiple sclerosis [1], provided that cerebral function remains intact. The signal from the motor cortex neurons, or bundles of neurons (units), has been proven to be an efficient and accurate way to control such devices [2][3]. One of the current challenges is to develop continuous decoding algorithm to allow robust and reliable prediction of movement [4]. The most frequently used state-of-the-art algorithms include linear regression [3], population coding [5], and probabilistic methods such as Kalman filters [6] and Bayesian regression [7].

For this project we were given a dataset consisting of the spike trains of 98 neuronal units in the motor cortex of a macaque monkey reaching with his hand for 8 different targets. The data was collected over 182 trials. A neuron spike train was represented as a binary sequence where the neuron is either firing or is not. Also included in the data was the three dimensional trajectory of the monkey's arm. The position data was aligned with the corresponding neuronal data at that time.

The aim of the project was to write an algorithm that could take training data consisting of neuronal spike train information and arm position, and use this to predict the X and Y trajectory of the monkey's arm given only neuronal information as an input. This information could then be used to drive a hypothetical prosthetic device. The project was part of a competition to create an algorithm that gave the lowest root mean square error (RMSE) between the trajectory predicted from the neuron spike train and the actual measured trajectory of the monkey's arm given in the dataset.

II. METHODS

Two functions were written up in MATLAB, one to develop a training algorithm from the data provided, and one to decode incoming neural data and predict the movement based only on the spike trains. The training data consists of the first 100 trials of the data set, while the trial data of the remaining 82 trials. In order to test the algorithm before the final submission, the training data was subsequently divided in two 50-trial subsets, one being used for training and one for testing.

A. Training Function

The training data was loaded and organised in two matrices, one for the neural response and one for the kinematic data, using bins of 15ms, starting from movement onset (t = 300) to 20ms after the end of the movement (i.e. 80 ms before the last data point). A time lag of 80ms was used, meaning that the kinematics at time *t* were related to the neural response at *t-80ms*.

The neural response matrix, **X,** contains the neural spike rate of each of the 98 units (*i),* calculated for each of the 100 trials (*n)* in the 8 different angles (*k)*, for the time going from t = 300ms (movement onset) to the end of the movement (100ms before the end of the spike train). **X** is therefore a matrix with dimensions $(n \cdot b) \times i \times k$. The spike rate was calculated in Hz as

$$s_{(n,b),i,k} = 1000 * \frac{\sum_{t=a}^{a+15} x_{t,n,k}}{15} \quad (1)$$

where $x_t$equals 1 when a spike was recorded, or 0 when the spike was not recorded. a goes from 220ms to $t_{max} - 160$.

The matrix of kinematic data for the x and y velocity in every bin was constructed by calculating the change in x and y position and dividing it by the bin size, as follows

$$vel_x = \frac{x_{upper\_edge} - x_{lower\_edge}}{binsize}$$
$$vel_y = \frac{y_{upper\_edge} - y_{lower\_edge}}{binsize} \quad (2)$$

Y is therefore a $(n \cdot b) \times 2 \times k$ matrix.

In order to establish a transform between neural response and kinematics, and thus to be able to decode and reconstruct incoming neural information, linear filter model was built, using multiple linear regression as follows:

$$\boldsymbol{Y}_{bk} = \beta_0 + \boldsymbol{\beta}_{i1k}\boldsymbol{X}_{ibk} + \boldsymbol{\beta}_{ibk}\boldsymbol{X}_{ibk} = \boldsymbol{\beta_{ibk}}\,\boldsymbol{X}_{ibk} \quad (3)$$

where $\boldsymbol{\beta}$ is a matrix that contains the weighting parameters to be calculated. This was a similar procedure to the one found in Serruya *et al.* [3][4], Hochberg *et al.* [8] and Kim *et al.* [6].
The linear filter was hence constructed by regressing the spike rate matrix onto the kinematics matrix. This was done by adding a column of ones to the neural response matrix, in order to account for β_0, and using the pseudo-inverse function to calculate the parameters $\boldsymbol{\beta_i}$. The closed form solution of the least-square formulation is given by:

$$\boldsymbol{\beta} = (\boldsymbol{X^T X})^{-1}\boldsymbol{X^T Y} \qquad (4)$$

which was calculated using the *pinv* MATLAB function. The matrix of parameters is therefore a $99 \times 2 \times 8$ matrix, with the extra row accounting for β_0. This means that there is a unique set of parameters for both the x and y velocity, and for each of the 8 angles. Furthermore, the covariance of the two matrices was calculated.
Since eight different sets of parameters were calculated, a classification of the incoming spike trains would be needed in order to guess the angle of movement only by using the neural response in the first 320ms – which is the amount of data to which we had access in the first iteration. For this purpose, three different methods were used and compared. The first one consisted in measuring the number of spikes for all units, all trials, and all angles, and then finding the Hamming distance by comparing the incoming spikes with each of the resulting bins. This did not prove as reliable as the other two (see the Results section).
The second method was a Support Vector Machine, a supervised machine learning method widely used for classification purposes [9]. Fig. 1 shows the construction of the SVM feature space matrix for our data. The SVM methods usually provide classification in binary form, while for our purposes we needed multi-class classification. For this reason, classes had to be reduced to a set of multiple binary classification problems to reduce the classification error.
The third method used was *k-nearest neighbours (k-NN), a* non-parametric learning algorithm. This type of classification was shown to have many advantages over other techniques, in terms of accuracy, speed and computational cost [10] [11] [12]. The model was built using the *fitcknn* function of MATLAB, testing different numbers of nearest neighbours and different types of distance metrics (namely Euclidean and cosine) as well as search types (k-d tree and exhaustive). The final parameters were chosen by trial and error in order to minimise the final error, as discussed in the results section. An in depth description of how to build a k-NN model can be found at [10].

B. *Position Estimator Function*

In the *Position Estimator* function, the classification models were used as an input – together with the incoming spike train – to the MATLAB *predict* function. The output was a 98x1 vector containing the calculated preferred direction for each of the 98 units, ranging from 1 to 8, and corresponding to each of the target angles. The final predicted direction for the whole trial was calculated by taking the mode of the output. According to the direction predicted, the corresponding $\boldsymbol{\beta}$ matrix was used for the reconstruction of movement. The reconstruction is defined as

$$\boldsymbol{U} = \boldsymbol{X_{new}\beta} = \boldsymbol{X_{new}}(\boldsymbol{X^T X})^{-1}\boldsymbol{X^T Y} \qquad (5)$$

where $\boldsymbol{X_{new}}$ is the spike rate density, calculated in the same way as in the training function, but further dividing the 20ms of data provided in each iteration in two more 10ms bins. The iterations spanned over the time between 300ms and $t_{max} - 100$. **U** is a vector containing the x velocity and the y velocity for that time bin. Similarly as for the training, a lag of 80ms was assumed.
The obtained velocity was then timed by the bin size (10ms) and the result added to the previously calculated hand position, starting from (0,0) at the first iteration. An error was also calculated using the covariance, assuming that $\varepsilon \sim N(0, \sigma^2)$ [13]. This was done to be used in future error analysis and statistical purposes, but was not included in the final algorithm.

III. RESULTS

In order to test the accuracy of the classification method, the test function provided was modified so that it would output the percentage of correctly classified trials. Table 1 shows the highest classification accuracy reached with the three different methods tested. The classification method that gave the best result was the *k-NN* algorithm. All following results are obtained using that method.
The parameters for the k-NN algorithm were chosen by trial and error. Tables 2 and 3 shows the resulting accuracy using *k-NN* for different parameters in the *fitcknn* function. The final parameters chosen for the model were 10 nearest neighbours, exhaustive search and cosine distance.
The final RMS error of the algorithm was 18.2 cm. With a series of systematic error corrections, this was managed to be decreased to as low as 17.6 cm. Fig. 2 show the output of the code without and with systematic error correction.
Different RMSE values were obtained by changing the parameters. Table 4 and 5 show the RMS error using different lags and bin sizes, with k-NN classification.

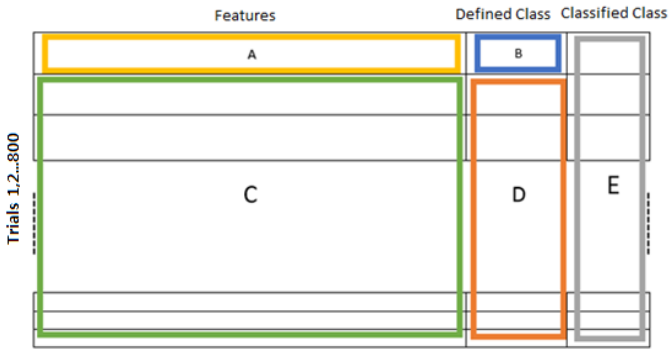

Figure 1. **Visualisation of feature space matrix for the SVM.** "Features" are the eight angles. A is treated as the trial with unknown class. The class of A will be defined under training set. B is the known class of A. C is the training set for SVM to classify whether the trail belongs to classes. The training set is a matrix of trials with known classes, which is the rest of trials apart from A, in order to train up the SVM for unknown classification. D is the grouping variable that divides the training data C into directions. E is the class that is determined by the SVM classifier after every trial is being treated as unknown. If B = E, classification is correct. However, if B ≠ E, the classification is wrong.

TABLE I. CLASSIFICATION ACCURACY FOR DIFFERENT CLASSIFICATION METHODS.

Method	Classification accuracy ±1%
k-nearest neighbours	91
Hamming distance	78
SVM	81

TABLE II. CLASSIFICATION ACCURACY FOR DIFFERENT DISTANCE METRICS, USING K-NN WITH K = 10.

Type of distance metrics	Classification accuracy ±1%
Cosine	91
Euclidean	90
Fiddle Tree	87

TABLE III. CLASSIFICATION ACCURACY FOR K-NN CLASSIFICATION METHOD, USING COSINE DISTANCE METRIC AND EXHAUSTIVE SEARCH.

Number of neighbours (k)	Classification accuracy ±1%
5	89
8	88
10	91
15	86

TABLE IV. RMSE FOR DIFFERENT BIN SIZES, USING AN 80MS LAG.

Bin size (ms)	RMSE (cm)
10	18.4
15	18.2
20	18.8
30	21.1

TABLE V. RMSE FOR DIFFERENT LAG TIMES, USING 15MS BINS.

Lag (ms)	RMSE (cm)
0	21.2
20	19.1
40	18.9
60	19.3
80	18.2
100	20.8

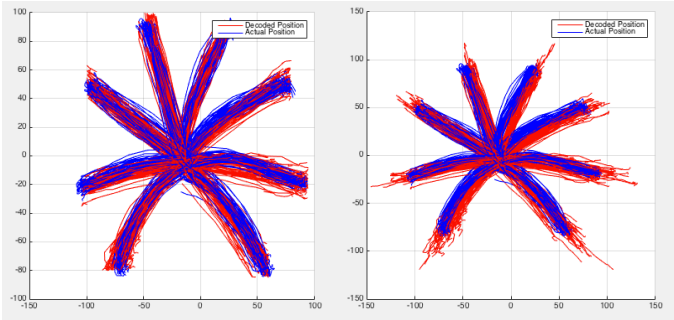

Figure 2 – ***(a) Output of the test function with systematic error correction. (b) Output of the test function without error correction.***

It is possible to notice the accuracy of the classification algorithm from the absence of predicted trajectories in between the eight angles. The systematic error correction shifted the prediction in order to match more closely the curved trajectory of the actual position and to avoid the overshooting, which was likely due to an overestimation of the expected velocity. RMSE in (a) = 18.2 cm. RMSE in (b) = 17.6 cm.

IV. DISCUSSION

Classification

The classification results using SVM compare well with other attempts using the same technique, although best results (>80%) are obtained when used in combination with Artificial Neural Networks and Bayesian classification [14][15]. Although this type of comparison is only meaningful is applied to similar set and type of data, it is safe to consider a classification of >80% as a satisfactory result [16][2]. *K-NN* yielded a much higher correct classification rate, which agrees with what has been found in a variety of cases, especially in spike sorting [10][11][12]. Our result of >90% is therefore very satisfactory.

There are still a number of parameters that could be varied and different metrics to be tested – such as Mahalnobis distance, which in some cases gave correct classification of >98% [17] – but due to the limited time available this was not possible.

Lag

In previous works, the optimal lag times chosen vary greatly. Wu *et al.* [18] obtained best results using a 2-bin lag (140ms), while Kim *et al.* found that no lag at all yielded the best results [6][12]. Both teams used Kalman filtering for their decoding algorithm. Interestingly, Yu *et al.* claim that, while for the majority of units in their experiment kinematics trail neural response, for some units the opposite is true, meaning that the lag would be negative [7]. These observations confirm our assumption that the optimal lag for a given data set should be found by trial and error.

Bin Size

The same applies to the size of the bins in which the training data was divided. For instance, Yu *et al.* used 10ms bins [7]; Serruya *et al.* and Hochberg *et al.* 50ms [4][8]; Kim *et al.* either 50 or 100ms [6][12]. While our algorithm initially used 20ms bins, since the trial data provided was divided in such way, we found that the optimal subdivision was dividing the training data in 15ms bins, and each new trial data iteration in two further 10ms bins.

RMSE Analysis

The lowest RMSE error obtainable with our algorithm was 18.2 cm without systematic error correction, and 17.6 with basic error correction, was higher than the average result of the competition, and much higher than for more accurate algorithms such as in Wu *et al.* [18], who obtained an MSE of 6.28 cm^2 (meaning a RMSE of 2.50 cm), or Serruya *et al.* [4], with an RMSE of about 2.8cm for a comparable amount of data points.

The main reason for this is likely the fact that the decoder was based on simple linear regression, with little error correction and no probabilistic prediction included. Recursive Bayesian decoders [7] and Kalman filters [12] have been proven to provide more accurate prediction of the movements. A complete evaluation of these methods is found in [19]. This is because in those methods the prediction is not only based on the spike rate, but also on the probability of a future spike rate (and hence position) happening given the firing history and the current position. This allows for an *a priori* prediction and an *a posteriori* update on the calculated hand position. Such techniques were not implemented in our algorithm because of their complexity and the time restraints on the project.

Furthermore, given the nature of the task, it would have been possible to use further spatial information in order to improve the accuracy of the prediction. For example, knowing the exact position of the eight targets could have been used to avoid the overshooting seen in Fig. 2, while the fact that the movements were all in the same x-y space could have allowed for the use of relative position (with respect to (0,0)) rather than velocity as kinematic data. Some studies suggest that the best results are given by calculating both, as well as the acceleration [8][9]. The latter, in fact, was shown to provide crucial information in order to reach the most accurate results by Wu *et al.* [18]. Our decision of not restraining the prediction to the absolute positions in the experimental space, however, allows for the development of a more flexible algorithm, which could work in any 3D space without the need for extensive calibration or dependence on fixed targets. Acceleration should be included in future development.

Another aspect to consider is that there was no selection of neurons to use for the decoding; instead, all 98 units were used independently of the predicted direction. Using only the most responsive neurons for a specific angle might possibly lead to more accurate results.

A complete evaluation of the algorithm would also require a substantial amount of further analysis, including computational cost calculation and statistical validation (t-test, Monte Carlo methods, bias-varianc, etc. – as suggested by [19].)

V. Conclusions

The decoder built by our team provides a relatively accurate continuous estimation of the position of the monkey's hand for trials in which only neural information was available. The direction was successfully decoded from the spike rate before movement onset through a k-NN algorithm, with an accuracy of >90%. The final RMS error was recorded to be 18.2cm, which is higher than the class average for this project, and higher than what previous studies have managed to achieve through other methods that were not implemented in our decoder.

VI. Attributions

P. Denning performed initial literature review and background research, and explored the use of k-NN as a classification method. J. Kim developed the code for the SVM algorithm and worked on the classification using Henning distance. M. Tonutti developed the linear regression algorithm for training, the k-NN model and the position estimator function.

VII. Acknowledgments

A particular acknowledgment and whole-hearted thanks to Dr. Simon Schultz, Marie Tolkienh, Tomaso Muzzu and Romy Lorenz for the great support throughout the term.

VIII. References

[1] *Paralysis*. (2014). Retrieved on 30th March 2015 from NHS: http://www.nhs.uk/conditions/Paralysis/Pages/Introduction.aspx

[2] Chapin JK., Moxon KA, Markowitz RS, Nicolelis MA. (1999) "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex". Nat Neurosci. Jul;2(7):664-70.

[3] Serruya MD, Hatsopoulos N, Paninski L, Fellows MR, Donoghue JP. (2002) "Instant neural control of a movement signal". Nature. Mar 14;416(6877):141-2.

[4] Serruya MD, Hatsopoulos N, Fellows M, Paninski L, Donoghue J.. (2003). "Robustness of neuroprosthetic decoding algorithms." *Biol Cybern. 2003 Mar;88(3):219-28.*

[5] Georgopolus AP, Schwartz A, Kettner RE. (1986). Neuronal Population Coding of Movement Direction. *Science*.

[6] Kim SP, Simeral JD, Hochberg LR, Donoghue JP, Black MJ. (2008). "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia." *Journal of Neural Engineering* .

[7] Yu BM, Kemere C, Santhanam G, Afshar A, Ryu SI, Meng TH, Sahani M, Shenoy KV. (2007) "Mixture of trajectory models for neural decoding of goal-directed movements." J Neurophysiol 97: 3763–3780, 2007. First published February 28, 2007; doi:10.1152/jn.00482.2006.

[8] Hochberg LR, Serruya MD, Friehs GM, Mukand JA, Saleh M, Caplan AH, Branner A, Chen D, Penn RD, Donoghue JP. (2006) "Neuronal ensemble control of prosthetic devices by a human with tetraplegia." Nature. 2006 Jul 13;442(7099):164-71.

[9] Tong S, Koller D. (2001) "Support Vector Machine Active Learning with Applications to Text Classification". Journal of Machine Learning Research, pp. 45-66

[10] Jain, A.K. (1991) "A k-nearest neighbor artificial neural network classifier" Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference, Volume II: 515-520.

[11] Seetha M, Sunitha KVN, Malini Devi G. (2012) "Performance Assessment of Neural Network and K-Nearest Neighbour Classification with Random Subwindows" International Journal of Machine Learning and Computing, Vol. 2, No. 6, December 2012

[12] Kim EK1, Wu JT, Tamura S, Close R, Taketan H, Kawai H, Inoue M, Ono K. (1993) "Comparison of neural network and k-NN classification methods in medical image and voice recognitions." Med J Osaka Univ. 1993 Sep;41-42(1-4):11-6.

[13] Wallisch P, Lusignan M, Benayoun M, Baker TI, Dickey AS and Nicholas G. (2009) "Matlab for Neuroscientists". Second edition. Elsevier Inc.

[14] Osareh A, Mirmehdi M, Thomas B, Markham R. "Comparative Exudate Classification using Support Vector Machines and Neural Networks". University of Bristol Publications.

[15] Samanta B, Al-Balushi KR, Al-Araimi SA. (2003) "Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection". Volume 16, Issues 7‑8, October‑December 2003, Pages 657‑665

[16] Byvatov E, Fechner U, Sadowski J, Schneider G. (2003) "Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification". J. Chem. Inf. Comput. Sci. 2003, 43, 1882-1889.

[17] Weinberger KQ, Blitzer J, Saul LK. "Distance Metric Learning for Large Margin Nearest Neighbor Classification." University of Pennsylvania Pubblications.

[18] Wu W, Black MJ, Gao Y, Bienenstock E, Serruya M, Shaikhouni A, Donoghue JP. "Neural Decoding of Cursor Motion using a Kalman Filter" Brown University Pubblications.

[19] Chen Z. "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond" Manuscript.

[20] Ofner P, and Muller-Putz GR, ¨Decoding of velocities and positions of 3D arm movement from EEG." Manuscript.