Long-Short Term Memory Neural Decoder for Cortical Brain-Machine Interface

N. Armengol Urpí[1], P. H. Tseng[2]

[1] Department of Information and Communication Technologies and Department of Experimental and Health Sciences, Universitat Pompeu Fabra, Barcelona, Spain, nuriaarmengol95@gmail.com

[2] Center for Neuroengineering, Duke University, Durham, United States of America, pohe.tseng@duke.edu

Summary

Brain-machine interfaces (BMIs) translate neural activity into control commands for assistive and prosthetic devices. In this project, Long-short term memory (LSTM), a recurrent neural network architecture, was applied to BMI control and compared to several decoding algorithms. Offline decoding was tested on neuronal population recordings in motor cortex of a monkey performing a center-out task using a joystick. Hyper-parameter optimization was performed using grid search so as to obtain network parameters that maximized decoder performance. The optimized LSTM learned quickly, and significantly outperformed the decoder Wiener filter. In the online implementation, LSTM decoded activity of approximately 150 motor cortical neurons and enabled a monkey to perform center-out reaching movements to eight peripheral targets. LSTM online performance exceeded that of the Unscented Kalman Filter, as judged by the number of correctly performed trials, decoded trajectory straightness and trajectory length to target as measures of accuracy. These results suggest that LSTM can replace many of current decoders in scientific and clinical BMI implementations.

1. Motivation

In the past decade, brain machine interfaces (BMIs) have increased popularity when holding promise to restore sensitivity and mobility to people who suffer from sensory or motor deficits, by connecting their brain structures to assisted devices and thus, offering them greater interaction with the world. BMIs are devices that transform extracellular activity from single or a population of neurons in the motor areas of cerebral cortex and convert them into commands to control external actuators, such as limb prostheses, wheelchairs or computer cursors. Recently, there has been significant work and progress in the field of motor BMIs, resulting in proficient control in non-human primates and humans. They record the neural activity from motor cortical areas, use a mathematical algorithm termed the "decoder" to convert this activity into control commands for an external device and provide visual feedback of the generated movement to the subject. The most critical component to ensure high performance of a BMI system, based on stable, reliable control signal, is the decoder, whose performance, although huge advances have been done, still remain unsatisfactory. Designing a decoder requires the choice of a model structure that describes the relationship between the control signal and kinematic states, which is used to derive the decoder that estimates the kinematics from neural activity. In most of BMIs, the encoding model assumes that the spike counts are a linear function of the intended kinematics, resulting in decoders such as the Wiener and the Kalman filter. Due to its ability to capture aspects of the dynamics in the kinematic data, Kalman filter generally performs better than Wiener. However, the linearity of Kalman filter limits its power whenever the relation between input data and output is non-linear. For this reason, an n-th order Unscented Kalman Filter (UKF) [1] which extends the observation model to a quadratic function and which augments the movement state variables with a history of *n-1* recent states was developed to improve BMI performance. Yet, the non-linearity of UKF is parametric. So as to skip the need of specifying a parametric form but still accounting for non-linear relationships among states and neural data, recently, neural networks which have been claimed to be able to infer this non-linearity without the need of a parametric form have also been developed for real-time BMIs [2].

In the current thesis, a novel decoding algorithm using a type of recurrent neural networks called Long Short-Term Memory (LSTM) is used. Recurrent neural networks (RNNs) are extremely powerful neural network models that are useful for learning nonlinear relationships between input and/or output that are sequences with complex temporal dependencies. Unlike standard feedforward neural networks, RNN are architectures with cyclic connections among adjacent time-steps, that allow them to retain information about data for long periods of time, and thus they are usually able to learn relationships between input and output that occur within long time windows. One of the main challenges when using RNN is the difficulty in training due to vanishing and exploding gradient problems that occur when training them via backpropagation for long time-steps, which at the end prevent the network to learn long-time dependencies in data [3]. LSTM is a type of RNN designed to overcome this problem while showing superior ability to model temporal sequences with long range dependencies than conventional RNNs.

Standard RNNs have already been used in the problem of BMI [2], but to this day, there has not been any reported decoder algorithm which uses LSTM. Hence, a study in the feasibility of LSTM for BMI was performed in the current project. It was hypothesized that given its recurrent property, the decoder would not require a predefined parametric form to model the relation between the neural data and kinematics. The decoder would model the dynamics of the internal brain by using self-connected artificial neurons to infer the relation between the activity of real neurons with their own past and the delayed motor commands they cause. More importantly, in contrast to Wiener and Kalman filter, the temporal relationship is learned by the algorithm during the training so that it is not required to specify the size of the temporal window in both online/offline predictions.

The decoder was tested offline in two different datasets containing neuronal population recordings in a monkey performing a center-out task using a hand-held joystick to control a cursor on a screen towards a target. LSTM decoder was also tested online to decode activity of approximately 150 motor cortical neurons so as to enable a monkey to perform center-out reaching movements to eight peripheral targets in real-time.

The results demonstrated not only its successful operation but also that it significantly improved offline and online performance when compared to Wiener and UKF algorithms, suggesting that LSTM can replace many of current decoders in scientific and clinical BMI implementations.

2. Methodology

The LSTM network was self-implemented in Python language version 3.5, using the open source software library in Python Tensorflow. The architecture used was based on the one originally described by Graves and Schmidhuber [4] commonly referred as *vanilla* LSTM. Vanilla LSTM incorporates forget gates and peephole connections, two variations proposed by [5] and [6] which were not implemented in the original LSTM architecture [7]. The vector formulas for a vanilla LSTM layer forward pass are given below and a detailed schematic can be seen in Figure 1.

$$\boldsymbol{g}^{(t)} = \sigma\left(\boldsymbol{x}^{(t)}W^{gx} + \boldsymbol{h}^{(t-1)}W^{gh} + \boldsymbol{b_g} + \boldsymbol{s}^{(t-1)}W^{p_g s}\right)$$

$$\boldsymbol{i}^{(t)} = tanh\left(\boldsymbol{x}^{(t)}W^{ix} + \boldsymbol{h}^{(t-1)}W^{ih} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{x}^{(t)}W^{fx} + \boldsymbol{h}^{(t-1)}W^{fh} + \boldsymbol{b}_f + \boldsymbol{s}^{(t-1)}W^{p_f s}\right)$$

$$\boldsymbol{s}^{(t)} = \boldsymbol{f}^{(t)} * \boldsymbol{s}^{(t-1)} + \boldsymbol{g}^{(t)} * \boldsymbol{i}^{(t)}$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{x}^{(t)}W^{ox} + \boldsymbol{h}^{(t-1)}W^{oh} + \boldsymbol{b}_o + \boldsymbol{s}^{(t)}W^{p_o s}\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)}\phi\left(\boldsymbol{s}^{(t)}\right)$$

where $\boldsymbol{x}$ is the input of the LSTM at every time step t containing the binned spike count of real monkey neurons, $\boldsymbol{h}$ is the output of the LSTM cell, $\boldsymbol{i}$ is the input node, $\boldsymbol{s}$ is the internal state and $\boldsymbol{f}, \boldsymbol{g}$ and $\boldsymbol{o}$ are the forget, input and output gates respectively. All $\boldsymbol{b}$ are bias terms. Where no explicit mathematical symbol is used between matrices, matrix product is computed while $*$ stands for Hadamard product. Sigmoid function (σ), hyperbolic tangent function (*tanh)* and ϕ are point-wise non-linear activation functions. ϕ can be *tanh,* ReLu or sigmoid function but in the current implementation was omitted due to lack of evidence it was needed.

All W are matrices of weights representing the strength of the edges; the ones with subscript p are the peephole connections allowing output gate to modulate the flow of information depending on current internal state. LSTM architecture implemented consisted on only one memory cell per block and only one hidden layer since the performance of the decoder decreased when adding more layers or more memory cells per block.

Finally, since dealing with a regression problem, given the LSTM output at time t $\boldsymbol{h}^{(t)}$, there was need to map the output layer *nx1* dimension to a *Sx1* dimension, where *n* is the number of artificial neurons in the network and *S* is the dimension of the predicted output $\overline{\boldsymbol{y}}^{(t)}$: $\overline{\boldsymbol{y}}^{(t)} = \boldsymbol{h}^{(t)}W^{hy} + b_y$;

W^{gx}, W^{ix} and W^{fx}, W^{gh}, W^{ih} and W^{fh} were initialized from a truncated normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.08.$ μ and σ values were set after trial and error method. $W^{hy}, W^{p_g s}, W^{p_f s}$ and $W^{p_o s}$ were initialized to zero. All bias vectors were initialized to zero except from $\boldsymbol{b}_f$, whose elements were all initialized to 2. By setting it to such a large number, the forget gate, with its sigmoid function, is consequently initialized to a value that is close to 1, preventing the network to forget at initial time-steps, enabling gradient flow and thus allowing LSTM to learn to solve problems with long-range dependencies [8].

The previous summarize the Python implemented LSTM forward pass, which leads to a total of 17 trainable parameters.

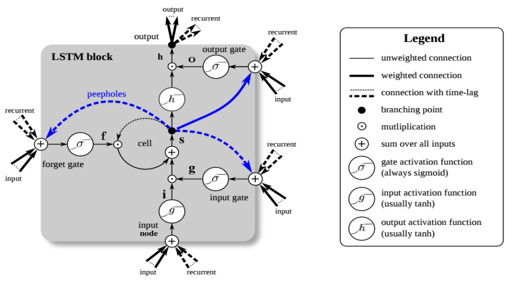

Figure 1*: Vanilla LSTM schematic with one hidden layer and one memory cell or LSTM block. Edited from Greff. K et al.*[9]

LSTM network was trained by minibatch gradient descent method using the Adaptive Moment Estimation Optimizer [10]. The error function used was Huber Loss described by Huber in 1964 [11], less sensitive to outliers in data than the squared error loss. At every training epoch the training procedure was done by Truncated Backpropagation Through Time algorithm in which only a fixed *n* (*n*=number of unrollings) number of previous time-steps was used for propagating the error. After every training epoch, the cell state $\boldsymbol{s}$ and the output of the LSTM $\boldsymbol{h}$ were reset to zero.

In addition, by way of a precaution, to address possible exploding gradient problems, gradient clipping technique [12] was implemented: gradients for all parameters were shrank by a global ratio when its global norm exceeded a clipping norm which was experimentally set to 1.25.

Finally, dropout and L2-regularization techniques were used to prevent overfitting.

2.1. Offline experimental methodology

LSTM decoder performance was tested offline in a center-out task which aimed to decode position of a cursor displayed on a screen controlled by a monkey using a hand-held joystick, from its brain neural data. The data was recorded from an experiment conducted in Nicolelis Laboratory (Duke University, North Carolina, USA) in an adult rhesus macaque which was chronically implanted in both hemispheres with microwire arrays in multiple cortical areas. Spiking activities in the primary motor cortex (M1) and primary somatosensory cortex in both hemispheres were recorded and sorted using template matching algorithms within commercially available software (Plexon Inc., Dallas, TX). Z-scored neural spike count of 50ms time bin were used as input for the network. 2 session datasets with a duration of approximately 1hour and the other 30 minutes were used to evaluate the performance of the decoder. The first session contained neural data of 635 neurons and the second 406.

Optimization of the hyperparameters describing the architecture of the network to achieve the best decoder performance as judged by mean-squared Euclidean distance statistic (MSE) was done by grid search. The *Duke Shared Cluster Resource* (DSCR) was used to optimize 5 different parameters: number of unrollings for every training epoch, number of neurons in the hidden layer, L2-regularization parameter, dropout probability and number of outputs to decode (only cursor position or cursor position and velocity).

2.2. Online experimental methodology

LSTM decoder was tested in a real-time experiment in which Monkey C was implementing an 8-peripheral target center-out task. Monkey C, an adult rhesus macaque, was chronically implanted with microwire arrays in multiple cortical areas. Spiking activities in the left M1 and premotor dorsal cortex (PMd) were recorded and sorted using the same template matching algorithms. All animal procedures were performed in accordance with the National Research Councils Guide for the Care and Use of Laboratory Animals and were approved by the Duke University Institutional Animal Care and Use Committee. As an average, Monkey C had a total of 150 neurons recorded, number which could slightly vary from day to day after sorting procedure.

So as to train LSTM decoder, neural data and cursor position were recorded while Monkey C was performing center-out task using the joystick to move a virtual cursor to specified targets located along the periphery and the center of a circle. The subject had to achieve the target within specific time and hold at the target for 1000ms to successfully complete the trial and get a liquid reward. Manual control was done approximately for 4 minutes so as to obtain an average of 2500 slices of neural spike count of 50 or 100ms bin size training data. An analog 3-axis potentiometer joystick (CH-400R-P3, Hangzhou Chuang Hong Electric Co.) was used to capture hand motion data. Only the *x* and *y* axes were used in these experiments. After training, cursor changed to be brain-controlled although monkey was still allowed to move the joystick. To assess LSTM decoder performance, a 10th order UKF decoder recently implemented in Nicolelis Laboratory [13] was alternately used with LSTM, switching after approximately 5 minute experiment.

3. Results

The performance of the decoder in the offline center-out task computed as an average across the 2 session datasets and for the 5-fold crossvalidation sets was evaluated in terms of Pearson correlation and obtained an r_x=0.9001±0.0219 and r_y =0.8891±0.0409 (μ±SEM), being r_i the Pearson correlation for position coordinate *i* and SEM the standard error of the mean (see Figure 2). L2-norm Wiener decoder, already implemented in Nicolelis Laboratory to brain-control a wheelchair by decoding rotational and translational wheelchair velocities from monkey brain data [14] was used to compare performance among the 2 decoders. LSTM substantially outperformed Wiener. A Wilcoxon signed-rank test comparing paired differences between the two decoder performances for every k-fold crossvalidation and every session rejected the null hypothesis at the 0.0039 and 0.002 significance level for covariate *x* and *y* respectively.

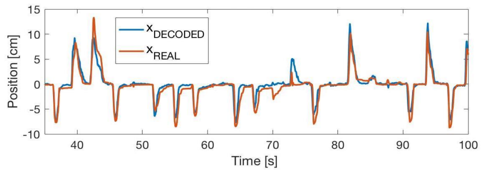

Figure 2: *Offline decoded x cursor position on screen during manual control. It shows part of one of the 5-fold crossvalidation results in Session 9, (r_x=0.9411) for the whole crossvalidation set*

In the online experiment, LSTM decoder was trained to generate monkey arm kinematics data from multi-unit arrays implanted in PMd/M1 in a center-out 8 reach task. LSTM decoder performance was compared to joystick-control and UKF decoder control by means of four statistics: average number of successful trials, trajectory length to target, trajectory deviation from a straight line to target and target acquire time. The number of successful trials for all 3 conditions (manual, LSTM and UKF) was computed as an average among 2 session experiment days considering the total number of trials done with every specific condition. The success rate across all trials was 100%, 92.17% and 88.59% for motor control, LSTM control and UKF control respectively. LSTM also outperforms UKF when it comes to both trajectory deviation and trajectory length to target. LSTM cursor moved smoothly and straight to the target, while UKF took longer paths to target and suffered from high jittering (see Figure 3), which prevented the cursor from staying inside the target for 1000ms as soon as the target size was reduced. Although the shortest trials were reported for LSTM control, LSTM target acquire time was higher than for UKF, probably due to a few really long duration trials which increased the median of the statistic. More experiments need to be done.

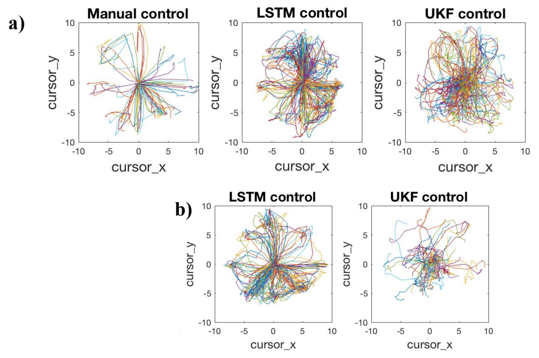

Figure 3: *Comparison of cursor trajectories during center-out task when using manual control, LSTM and UKF brain control for a) Session 1 and b) 2*

A 1000-permutation test was computed so as to test whether the statistics were significantly different among the two tested decoders. For both sessions, LSTM deviation from a straight line was significantly lower than UKF (α=0.01). Trajectory length was lower than UKF for both sessions but the difference was only significant for Session 2 (see Figure 4). Taken together the LSTM online performance exceeded that of the 10th order UKF, as judged by the number of correctly performed trials, trajectory straightness and trajectory length as measures of accuracy.

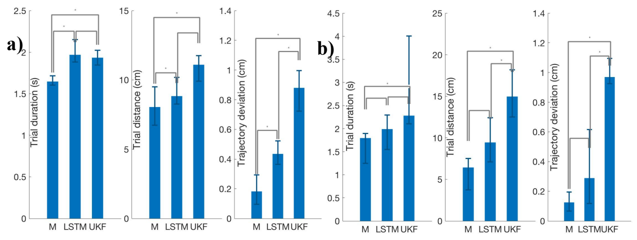

Figure 4: *Comparison of trajectory deviation, trajectory length and time to target, between manual control (M), UKF and LSTM brain control during a) Session 1 and b) Session 2. '*' symbol represents there is a statistical difference in the two groups (α=0.01)*

4. Conclusion

In this study, we aimed to test whether LSTM neural network could be used as an algorithm decoder for BMI approaches. One of the huge advantages is that it does not need of a predefined parametric form to model the relation between the neural data and kinematics but it is able to infer through training the relation between the activity of real neurons with their own past and the delayed motor command they cause. Due to its two main properties: network dynamics provided by its recurrent connections and nonlinear and distributed computation, it seems to fulfill all criteria for an excellent BMI decoder. The results proved that LSTM offline decoding of position of a cursor on a screen during a monkey center-out task outperformed one of the most used decoders, the L2-norm Wiener decoder algorithm. Most importantly, LSTM decoder was able to decode position of a cursor in a real-time experiment, outperforming the current state-of-the-art 10th order UKF, as measured by trajectory deviation from straight line and trajectory length, improvements which were both statistically significant. In addition, success trial rate was also increased by 3%.

There are some key next steps to be done as part of further expanding the capabilities of LSTM as a BMI decoder. Nevertheless, taken together, the results obtained suggest that the implemented LSTM decoder is a powerful tool for BMI decoder offline, and more importantly, online applications, which could be further used in the near future for other tasks, such as wheelchair navigation.

References

[1] Z. Li, J. E. O'Doherty, T. L. Hanson, M. A. Lebedev, C. S. Henriquez, and M. A. Nicolelis, "Unscented Kalman filter for brain-machine interfaces," PLoS.One., vol. 4, no. 1932–6203 (Electronic), p. e6243, 2009.

[2] D. Sussillo et al., "A recurrent neural network for closed-loop intracortical brain–machine interface decoders," J. Neural Eng., vol. 9, no. 2, p. 26027, 2012.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," IEEE Trans. Neural Networks, vol. 5, no. 2, pp. 157–166, 1994.

[4] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in Proceedings of the International Joint Conference on Neural Networks, 2005, vol. 4, pp. 2047–2052.

[5] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," Neural Comput., vol. 12, no. 10, pp. 2451–2471, 2000.

[6] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, 2000, pp. 189–194 vol.3.

[7] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1–32, 1997.

[8] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," Iclr, no. 2013, pp. 1–8, 2014.

[9] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," IEEE Transactions on Neural Networks and Learnin Systems Systems, 2016.

[10] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," Int. Conf. Learn. Represent. 2015, pp. 1–15, 2015.

[11] P. J. Huber, "Robust Estimation of a Location Parameter," Ann. Math. Stat., vol. 35, no. 1, pp. 73–101, 1964.

[12] R. Pacanu, T. Mikolov, and Y. Bengio, "On the Difficulties of Training Recurrent Neural Networks," Icml, no. 2, pp. 1–9, 2013.

[13] S. Li, J. Li, and Z. Li, "An improved unscented kalman filter based decoder for cortical brain-machine interfaces," Front. Neurosci., vol. 10, no. DEC, 2016.

[14] S. Rajangam et al., "Wireless Cortical Brain-Machine Interface for Whole-Body Navigation in Primates," Sci. Rep., vol. 6, no. 1, p. 22170, 2016.