Recurrently compressing a rolling time window using spiking neurons

Aaron R. Voelker and Chris Eliasmith

Centre for Theoretical Neuroscience, University of Waterloo
{arvoelke, celiasmith}@uwaterloo.ca

ABSTRACT: We present a dynamical system that optimally compresses a rolling time window of input into a low-dimensional state-space. The state of this system represents the history of its input by a linear combination of time-invariant polynomial basis functions. We use the Neural Engineering Framework (NEF) to compile this system onto a recurrent spiking neural network. The activity of this "delay network" naturally approximates a Taylor series expansion, over time, centered about every point within the window. We show that this permits the computation of arbitrary nonlinear computations across the history of an input stimulus, while outperforming Echo State Networks in accuracy and training/testing time, without explicitly simulating any training examples.

ADDITIONAL DETAIL: A continuous-time rolling window of length $\theta > 0$, denoted $[u(t-\theta') : 0 \le \theta' \le \theta]$, may be *temporally compressed* into a low-dimensional state $\mathbf{x}(t) \in \mathbb{R}^d$, by the following linear time-invariant dynamical system that approximately delays an input signal by θ seconds (Voelker & Eliasmith, 2017):

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t), \quad A = \begin{pmatrix} -\nu_0 & -\nu_0 & \cdots & -\nu_0 \\ \nu_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \nu_{d-1} & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \nu_0 & 0 & \cdots & 0 \end{pmatrix}^\top, \tag{1}$$

where $\nu_i := \frac{(d+i)(d-i)}{i+1}\theta^{-1}$, for $i = 0 \dots d-1$. This compression is derived by applying Padé approximants to $e^{-\theta s}$ about the complex point $s = 0$, which yields an optimal low-degree polynomial expansion of $\mathcal{L}\{u\}(s)$ about its zeroth frequency. The state $\mathbf{x}(t)$ represents the rolling window via a linear combination of time-invariant polynomial basis functions:

$$\boxed{u(t-\theta') \approx \sum_{i=0}^{d-1} \mathcal{P}_{i,d}\left(\frac{\theta'}{\theta}\right) x_{d-1-i}(t), \quad 0 \le \theta' \le \theta,} \tag{2}$$

which we have solved in closed-form (Voelker & Eliasmith, 2017, eq. 14):

$$\mathcal{P}_{i,d}(r) = \binom{d}{i}^{-1} \sum_{j=0}^{i} \binom{d}{j} \binom{2d-1-j}{i-j} (-r)^{i-j}. \tag{3}$$

A set of polynomials $\{\mathcal{P}_{i,d}\}$ are shown in Figure 1. This defines a linear map from the d-dimensional state, $\mathbf{x}(t)$, to the infinite-dimensional rolling window, $[u(t-\theta') : 0 \le \theta' \le \theta]$. In other words, (2) is the optimal pseudoinverse of the linear compression defined by (1). Moreover, each point along the window, along with its Taylor series expansion up to order $d-1$, as well as any other linear operation across the rolling window (e.g., Laplace transform), may be expressed using (2) and (3) as a known dot-product with $\mathbf{x}(t)$.

To implement this system as a recurrent neural network, we use the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003) to map (1) onto the dynamics of a lowpass synapse with time-constant τ:

$$A^H = \tau A + I, \quad B^H = \tau B, \quad H(s) = (\tau s + 1)^{-1}. \tag{4}$$

The resulting "delay network" (DN) is depicted in Figure 2. Generalizations to arbitrary linear synapses, including those modelling pure feedback delays, may be found in Voelker & Eliasmith (2017). The state $\mathbf{x}(t)$ is *encoded* by a heterogeneous population of neurons, which allows arbitrary nonlinear functions across the window, $y(t) = f(\mathbf{x}(t))$, to be *decoded* from its activity (Eliasmith & Anderson, 2003).

We use Nengo 2.5 (Bekolay et al., 2014) and nengolib 0.4 to construct each network. To demonstrate our ability to compute nonlinear window functions, we consider the function $y(t) = u(t-\theta)\,u(t)$, visualized

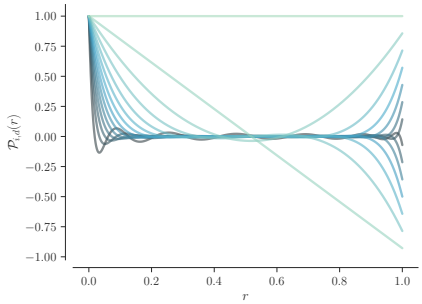

Figure 1: Polynomial basis functions $\mathcal{P}_{i,d}$ from (3) with $d = 14$. Functions are sorted from $i = 0 \ldots d-1$ by darkness. The state of the delay network, $\mathbf{x}(t)$, represents the rolling time window of input history by a linear combination of these functions (see (2)).

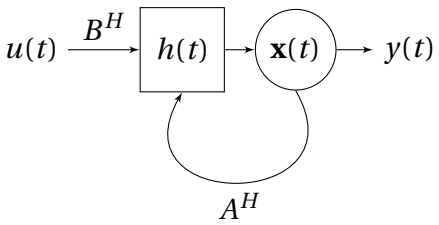

Figure 2: Delay network (DN) architecture. The synapse model $h(t)$ is driven by $A^H\mathbf{x}(t) + B^H u(t)$ to yield the state $\mathbf{x}(t)$. This state is nonlinearly encoded by a heterogeneous population of neurons and subsequently decoded to estimate the desired $y(t)$.

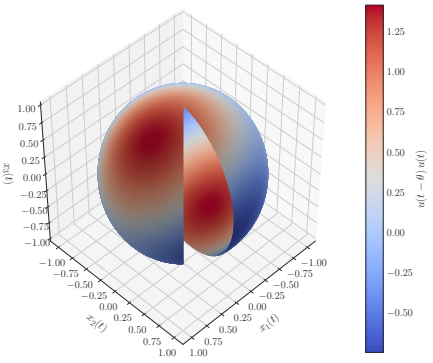

Figure 3: Visualizing the autocorrelation function $y(t) = u(t-\theta)\,u(t)$ with $d = 3$. Each point along the surface of the unit-sphere is $\mathbf{x}(t)$, and is colored according to its corresponding value of $y(t)$ obtained from (2). A slice of the shell is cut away for visualization.

in Figure 3. When integrated over time, this is the autocorrelation of u with lag θ, which has numerous applications in signal processing (e.g., detecting repeating events). We fix $\theta = 0.1$ s across all experiments. To compute this function accurately, we sample a proportion of the encoders from the diagonal combinations of $\mathcal{P}_{i,d}(0)$ and $\mathcal{P}_{i,d}(1)$. The inputs $u(t)$ are drawn from white-noise, band-limited with a cut-off frequency of 30 Hz. Nengo allows us to consider populations of either spiking LIF neurons or non-spiking sigmoid neurons, without any additional changes to the model specification. To optimize for the decoders, we map d-dimensional evaluation points onto desired target outputs using (2), and apply Nengo's regularized least squares solver, which avoids the need to explicitly simulate the network on any input signals.

To compare with Reservoir Computing, we ported a Python implementation of the Echo State Network (ESN), from Lukoševičius & Jaeger (2009), to Nengo, and implemented it analogously to the DN. We used Hyperopt (Bergstra et al., 2015) to explore the space of model hyperparameters (e.g., d, τ, input gain, recurrent gain, L2-regularization) across 100 iterations containing 10 trials each. Each network consisted of 1,000 neurons, simulated with a time-step of 1 ms. Each trial used a training signal of length 10,200, a testing signal of length 2,200, and the first 200 outputs were discarded. We then cross-validated the best set of hyperparameters (in terms of mean normalized RMSE across all test signals) using another 25 trials. Complete source code may be found at `https://github.com/arvoelke/cosyne2018`.

We obtained a mean normalized RMSE of 0.059 for the sigmoid DN, and 0.518 for the spiking DN, compared to 0.843 for the ESN. Reducing the input frequency to 15 Hz improves the ESN's accuracy to be on par with the non-spiking DN, and thus we attribute this difference to the inherent difficulty of autocorrelating a high-frequency signal (relative to θ) using random feedback weights, as opposed to using optimally derived weights as in the DN. In addition, trials took on average 5.10 s for the sigmoid DN, 6.44 s for the spiking DN, and 17.7 s for the ESN. This difference is a consequence of not simulating the DN for training, and from using factorized weight matrices (i.e., encoders and decoders) to simulate the DN.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: A Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, *7*(48).

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: A Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, *8*(1), 014008.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Voelker, A. R., & Eliasmith, C. (2017). Improving spiking dynamical networks: Accurate delays, higher-order synapses, and time cells. *Neural Computation (In Press)*.