A new paradigm for probabilistic neuromorphic programming

P. Michael Furlong[1*] and Chris Eliasmith[1]

[1]Centre for Theoretical Neuroscience, University of Waterloo, 200 University Ave., Waterloo, N2L 3G1, Ontario, Canada.

*Corresponding author(s). E-mail(s): michael.furlong@uwaterloo.ca;
Contributing authors: celiasmith@uwaterloo.ca;

Keywords: probability, Bayesian modelling, vector symbolic architecture, fractional binding, spatial semantic pointers

1 Introduction

Since it was first introduced neuromorphic hardware has held the promise of capturing some of the efficiency of biological neural computation, which is 3-6 orders of magnitude more efficient than engineering solutions. To attempt to capture this efficiency, one key feature of biological neural computation that has been replicated in current neuromorphic hardware is its event-based nature. Neural 'spikes' are used to transmit information both in neuromorphic hardware and many neurobiological systems to minimize long distance communication energy costs.

However, the use of such spiking neural networks (SNNs) brings with it challenges for programming the hardware. While there are a variety of techniques for tackling this challenge (Eliasmith and Anderson, 2003; Sussillo and Abbott, 2009; Denève and Machens, 2016), none have shown convincingly how the broad and powerful class of *probabilistic algorithms* can be efficiently realized in a spiking neural network. In our recent work, we have been exploring the connection between hyper-dimensional computing (HDC), also known as Vector Symbolic Architectures/Algebras (VSAs), and probabilistic computation in order to solve this problem. When we couple this new approach with

our past work on systematically building spiking neural networks using the Neural Engineering Framework (NEF; Eliasmith and Anderson, 2003), a new paradigm for building spiking, probabilistic neural networks arises.

In this brief note, we outline these new methods and describe how they relate HDC to various probabilistic operations. Our particular algebra is based on a VSA known as holographic reduced representations (HRRs; Plate, 1994) but focuses on using it in a continuous manner, giving rise to what are called Spatial Semantic Pointers (SSPs; Komer, 2020; Dumont and Eliasmith, 2020). A more in-depth version of this work can be found in (Furlong and Eliasmith, 2022).

2 A Method for Probabilistic Programming

2.1 Preliminaries

Kernel Density Estimators (KDEs) estimate the probability of a query point x based on the average of its similarity to members of a dataset of n observations $\mathcal{D} = \{x_1, \ldots, x_n\}$. Similarity is measured using kernel functions, $k(\cdot, \cdot)$, which are typically valid density functions. KDEs are defined $f_X(\mathbf{x}) = \frac{1}{nh} \sum_{i=1}^{n} k_h(\mathbf{x}, \mathbf{x}_i)$ for kernel bandwidth $h \in \mathbb{R}^+$.

A problem with KDEs is the memory required to maintain the dataset, $\mathcal{D}$, which can grow without bound, as does the time to compute a query. Rahimi *et al.* (2007) addressed this problem for KDEs and other kernel machines with the introduction of Random Fourier Features (RFFs). RFFs project data into vectors so that the dot product between two vectors approximates a kernel function, *i.e.*, $k(x, y) \approx \phi(x) \cdot \phi(y)$. The data projection is computed $\phi(x) = (\mathrm{e}^{\mathrm{i}\omega_1 x}, \ldots \mathrm{e}^{\mathrm{i}\omega_d x})^T$, where the frequency components ω_i are i.i.d samples from some probability distribution $G(\omega)$. The choice of $G(\omega)$ determines the kernel induced by the dot product.

With RFFs, linear methods can approximate nonlinear kernel methods. Kernels that can be approximated with RFFs of dimensionality $d < n$ improve the memory and time complexity of querying a KDE from linear in the number of samples (n) to linear in the feature representation dimensionality (d).

2.2 Probabilistic Programming with HDC

Here we describe the particular algebra we use and its mapping to probabilistic representations and computations. We use three operators, *bundling*, *binding*, and *unbinding*, to construct latent probabilistic representations, and a fourth, *similarity*, to realize quasi-probabilities, which can later be converted to exact probabilities. These are the same operators as used for the HRR VSA (Plate, 1994), but here with a mapping to continuous representations and an implementation in spiking neurons. Manipulating VSA-represented data using these operators constructs and manipulates representations that induce kernels for structured data, generalizing the method of RFFs.

2.2.1 Operators

Binding, $\circledast$, implemented with circular convolution, is at the core of our approach — in VSAs, binding is used to combine two symbols or state representations together to produce slot-filler pairs, *e.g.*, combining a sensing modality type with a sensor value, or an edge in a graph with the edge's traversal cost. We employ an extension of binding, called *fractional* binding (Komer, 2020; Dumont and Eliasmith, 2020; Plate, 1992), to represent data in a continuous domain, $\mathcal{X} \subseteq \mathbb{R}^m$, into a high-dimensional vector representation (eq. (1)).

$$\phi_{\mathrm{X}}(\mathbf{x}/h) = \mathcal{F}^{-1}\left\{\mathrm{e}^{\mathrm{i}\Theta_X \mathbf{x}/h}\right\} \tag{1}$$

Where $\mathbf{x} \in \mathcal{X}$ and h is a length scale parameter, as in kernel density estimation, and Θ_X are the frequency components of what we refer to as the "axis vector", which can be selected in a number of different ways. Θ_X and h define a "type" representation in the high-dimensional space for the low-dimensional space.

Similarity between two VSA-encoded objects is computed with the vector dot product, $\cdot$. For SSPs similarity has a strict mathematical meaning through the connection to RFFs, the dot product between two SSPs approximates a kernel function, like those used in kernel density estimation (Voelker, 2020; Frady *et al.*, 2021). Importantly for probabilistic modelling, depending on the selection of Θ_X, the dot product between SSPs will induce different kernel (or similarity) functions on the encoded data, expressed:

$$k(\mathbf{x}, \mathbf{x}') \approx \phi_{\mathrm{X}}(\mathbf{x}/h) \cdot \phi_{\mathrm{X}}(\mathbf{x}'/h). \tag{2}$$

Bundling is used in VSA literature to construct vectors that represent sets of objects, where similarity between a vector and a bundle gives a measure of membership in the set. We use bundles of fractionally-bound objects to represent a distribution, and when we compute the similarity between a query point encoded as an SSP with a bundle of SSPs, we get a quantity that approximates the probability of the query point. In math:

$$\hat{f}(\mathbf{x} \mid \mathcal{D}) \approx \phi_{\mathrm{X}}(\mathbf{x}/h) \cdot \frac{1}{nh} \sum_{\mathbf{x}_i \in \mathcal{D}} \phi_{\mathrm{X}}(\mathbf{x}_i/h) \tag{3}$$

where we can replace the normalized sum $\frac{1}{nh}\sum_{\mathbf{x}_i \in \mathcal{D}} \phi_{\mathrm{X}}(\mathbf{x}_i/h)$ with a memory vector that represents the dataset, $M_{\mathrm{X},n}$. This memory can be updated online, allowing for changes in the distribution that reflect the experience of an agent.

Depending on the choice of Θ_X, similarity can take on negative values. Indeed, in the method used by Furlong and Eliasmith (2022), the dot product between SSPs approximates the quasi-kernel sinc function (Voelker, 2020). Consequently, $\hat{f}(\mathbf{x} \mid \mathcal{D})$ is not a probability distribution, but the special-case Fourier Integral Estimtor (FIE; Davis, 1975, 1977). However, probabilities can

be recovered from FIEs through the correction developed by Glad *et al.* (2003):

$$f_X(x) \approx \max\left\{0, \phi_{\mathrm{X}}\left(x/h\right) \cdot M_{X,n} - \xi\right\}. \tag{4}$$

The conversion in eq. (4) unveils a connection between the VSA encoding we use and how individual ReLU neurons can be used to model probability.

Unbinding is the inverse of binding, and is implemented by binding with the pseudoinverse of the argument, $\phi_{\mathrm{X}}(x) \circledast \phi_{\mathrm{Y}}(y) \circledast \phi_{\mathrm{Y}}^{-1}(y) \approx \phi_{\mathrm{X}}(x)$. In cognitive modelling, unbinding can be used to select from bundles a subset where the querying vector matches. We have found that unbinding can be used to condition memory vectors, $M_{\mathrm{XY}} = \sum_{(x_i,y_i)\in\mathcal{D}} \phi_{\mathrm{X}}(x_i) \circledast \phi_{\mathrm{Y}}(y_i)$, selecting only those elements where $y_i \approx y$, that is:

$$f(x \mid y, \mathcal{D}) \propto \phi_{\mathrm{X}} \cdot \left(M_{\mathrm{XY}} \circledast \circledast \phi_{\mathrm{Y}}^{-1}(y)\right). \tag{5}$$

In addition to the above operations, we have also been able to exploit the sparsity of these representations to construct compositional kernels, as binding multiplies kernels and bundling adds them. Exploiting sparsity is a general feature of HDC. Representations of more structured data, like trajectories, graphs, or trees, constructed in VSAs consequently define kernels for those data objects, making this tool for designing neural network a general probabilistic programming framework.

2.2.2 Spiking implementation

While we will not describe the NEF method of implementing these operations in spiking neurons in detail here, this has been done elsewhere at length (Eliasmith and Anderson, 2003; Dumont and Eliasmith, 2020; Furlong and Eliasmith, 2022; Eliasmith, 2013). As well, there is a widely used neural simulator, Nengo (https://nengo.ai) that incorporates the NEF directly. To summarize, the method allows any nonlinear dynamical system defined over any dimensionality of vector space to be embedded in a spiking neural network to a degree of precision determined by the neural resources available. These techniques naturally apply to the operators described above.

3 Discussion and Conclusion

The above connection of HDC and VSAs to probability models show that we can understand them as a probabilistic model of computation that unifies analytical models with neural networks, expanding on other approaches to modelling probability in VSAs (*e.g.*, Frady *et al.*, 2021; Joshi *et al.*, 2017). VSA representations are differentiable allowing for integration with standard machine learning methods. More importantly, descriptions of data under VSAs give practitioners, for free, kernel functions that can be used in probabilistic models, using simple linear operations.

While we have only briefly outlined the approach here, we and others have successfully used it to demonstrate that, for example, Bayesian Optimization can be performed much more efficiently that using standard methods while preserving and generally improving the accuracy of the inference on standard benchmark functions (Furlong *et al.*, 2022). As well, the methods have been used for optimal path planning for drones (unpublished), the unification of biological models of simultaneous localization and mapping with a probabilistic framing of the problem (Dumont *et al.*, under review), as well as the coordination of paths across multiple actors (Furlong *et al.*, 2023). This latter example demonstrates that the technique can be used to combine continuous and discrete probability spaces. While we have focused on the continuous case because it is newer, the methods we have used are well-established for representing symbol-like structures in spiking neurons and performing inference over them (Eliasmith, 2013; Eliasmith *et al.*, 2012).

While we have not discussed specific neuromorphic implementations, NEF models have a long history of being implemented on a variety of neuromorphic hardware (Bekolay *et al.*, 2014), and Nengo has been used with a number of hardware backends, including Loihi (DeWolf *et al.*, 2020) and SpiNNaker (Davies *et al.*, 2013). As a result, all of the critical elements for compiling from a probabilistic program to neuromorphic hardware executables are in place. We look forward to exploring a wide variety of applications using these methods, allowing us to reap the benefits of low power neuromorphic hardware while realizing the effectiveness of probabilistic computation.

References

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

Sussillo, D. and Abbott, L.F.: Generating coherent patterns of activity from chaotic neural networks.: Neuron **63**(4), 544–57 (2009), URL http://dx.doi.org/10.1016/j.neuron.2009.07.018

Denève, S. and Machens, C.K.: Efficient codes and balanced networks: Nature Neuroscience **19**(3), 375–382 (2016), URL http://dx.doi.org/10.1038/nn.4243

Plate, T.A.: Distributed representations and nested compositional structure: University of Toronto, Department of Computer Science (1994)

Komer, B.: Biologically Inspired Spatial Representation: Ph.D. thesis, University of Waterloo (2020)

Dumont, N. and Eliasmith, C.: Accurate representation for spatial cognition using grid cells.: In: CogSci (2020)

Furlong, P.M. and Eliasmith, C.: Fractional binding in vector symbolic architectures as quasi-probability statements: In: Proceedings of the Annual Meeting of the Cognitive Science Society, volume 44 (2022)

Rahimi, A., Recht, B. *et al.*: Random features for large-scale kernel machines.: In: NIPS, volume 3, 5, Citeseer (2007)

Plate, T.A.: Holographic recurrent networks: Advances in neural information processing systems **5** (1992)

Voelker, A.R.: A short letter on the dot product between rotated fourier transforms: arXiv preprint arXiv:2007.13462 (2020)

Frady, E.P., Kleyko, D., Kymn, C.J., Olshausen, B.A. and Sommer, F.T.: Computing on functions using randomized vector representations: arXiv preprint arXiv:2109.03429 (2021)

Davis, K.B.: Mean square error properties of density estimates: The Annals of Statistics 1025–1030 (1975)

Davis, K.B.: Mean integrated square error properties of density estimates: The Annals of Statistics 530–535 (1977)

Glad, I.K., Hjort, N.L. and Ushakov, N.G.: Correction of density estimators that are not densities: Scandinavian Journal of Statistics **30**(2), 415–427 (2003)

Eliasmith, C.: How to build a brain: A neural architecture for biological cognition: Oxford University Press (2013)

Joshi, A., Halseth, J.T. and Kanerva, P.: Language geometry using random indexing: In: Quantum Interaction: 10th International Conference, QI 2016, San Francisco, CA, USA, July 20-22, 2016, Revised Selected Papers 10, 265–274, Springer (2017)

Furlong, P.M., Stewart, T.C. and Eliasmith, C.: Fractional binding in vector symbolic representations for efficient mutual information exploration: In: Proc. ICRA Workshop, Towards Curious Robots, Mod. Approaches Intrinsically-Motivated Intell. Behav., 1–5 (2022)

Dumont, N.S.Y., Furlong, P.M., Orchard, J. and Eliasmith, C.: Exploiting semantic information in a spiking neural slam system: Frontiers in Neuromorphic Engineering (under review)

Furlong, P.M., Dumont, N.S.Y., Antonova, R., Orchard, J. and Eliasmith, C.: Efficient exploration using hyperdimensional bayesian optimization on trajectory spaces (2023), in submission

Eliasmith, C., Stewart, T.C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y. and Rasmussen, D.: A large-scale model of the functioning brain: science **338**(6111), 1202–1205 (2012)

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., Choo, X., Voelker, A. and Eliasmith, C.: Nengo: A Python tool for building large-scale functional brain models: Frontiers in Neuroinformatics **7**(JAN) (2014), URL http://dx.doi.org/10.3389/fninf.2013.00048

DeWolf, T., Jaworski, P. and Eliasmith, C.: Nengo and Low-Power AI Hardware for Robust, Embedded Neurorobotics: Frontiers in Neurorobotics **14** (2020), URL http://dx.doi.org/10.3389/fnbot.2020.568359

Davies, S., Stewart, T., Eliasmith, C. and Furber, S.: Spike-based learning of transfer functions with the SpiNNaker neuromimetic simulator: In: Proceedings of the International Joint Conference on Neural Networks (2013), ISBN 9781467361293, URL http://dx.doi.org/10.1109/IJCNN.2013.6706962