A Recurrent Dynamic Model for Efficient Bayesian Optimization*

P. Michael Furlong
Centre for Theoretical Neuroscience)
University of Waterloo
Waterloo, Canada
michael.furlong@uwaterloo.ca

Nicole Sandra-Yaffa Dumont
Centre for Theoretical Neuroscience
University of Waterloo
Waterloo, Canada
nicole.dumont@uwaterloo.ca

Jeff Orchard
Center for Theoretical Neuroscience
University of Waterloo
Waterloo, Canada
jorchard@uwaterloo.ca

***Abstract*—Bayesian optimization is an important black-box optimization method used in active learning. An implementation of the algorithm using vector embeddings from Vector Symbolic Architectures was proposed as an efficient, neuromorphic approach to solving these implementation problems. However, a clear path to neural implementation has not been explicated. In this paper, we explore an implementation of this algorithm expressed as recurrent dynamics that can be easily translated to neural populations, and present an implementation within the Lava programming framework for Intel's neuromorphic computers. We compare the performance of the algorithm using different resolution representations of real-valued data, and demonstrate that the ability to find optima is preserved. This work provides a path forward to the implementation of Bayesian optimization on low-power neuromorphic computers, permitting the deployment of active learning techniques in low-power, edge computing applications.**

***Index Terms*—Bayesian optimization, vector symbolic algebras, fractional power encoding, recurrent dynamics**

I. Introduction

Active learning is important for agents that are continually learning about the systems with which they interact. Bayesian optimization (BO) is a widely used black-box optimization procedure used for sample efficient active learning in the presence of noisy observations. In BO, the next sample to observe is selected by optimizing an information-theoretic *acquisition function*. Optimization problems are well matched to neuromorphic hardware [1], and, given the utility of BO, translating Bayesian optimization to neuromorphic computing warrants investigation.

Gaussian process regression (GPR)-based methods for BO are well established, with popular choices of acquisition function being based either on the Upper Confidence Bound algorithm [2] or on Mutual Information [3]–[5]. In these methods Gaussian processes provide the posterior distribution over observations necessary to compute the information theoretic quantities that lie at the heart of their respective acquisition functions. Unfortunately, these algorithms have sample selection memory and time complexity that grows in the square (t^2) and the cube (t^3), respectively, of the number of samples collected, t. The unbounded complexity growth is not compatible with applications with bounded resources, as may be found in mobile robotics, aerospace, or edge computing applications.

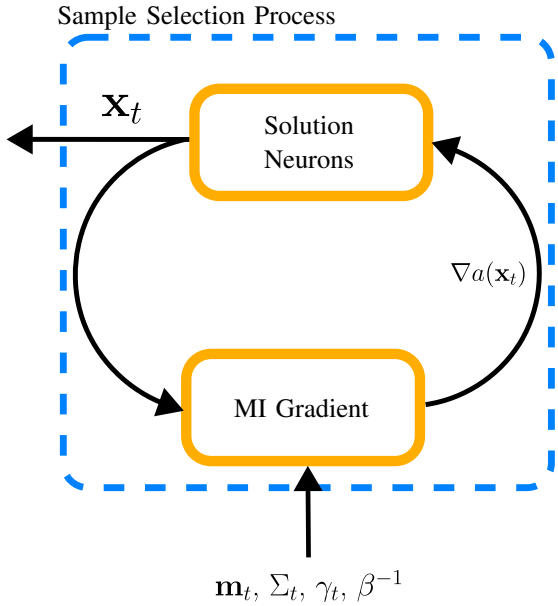

Fig. 1: The overall network is implemented using two connected Lava processes. Parameters of the Bayesian linear regression (BLR) acquisition function are learned outside of the process and are updated in response to observations collected by the network. At any given time, the solution neurons are representing the current estimate of the next sample to take, which eventually converges to the true solution.

Furlong *et al.* proposed a constant complexity BO algorithm that operated in the space of representations defined by a Vector Symbolic Algebra (VSA)[1] [6]–[9]. The efficiency was gained by exploiting the connection between VSA representation and reproducing kernel Hilbert spaces [10], [11], in order to invert the kernel trick [12] and approximate the distribution over observations. While this algorithm proposes BO in a space of neurally-plausible representations, a neurally-plausible implementation of the algorithm was not proposed

This work was funded in part by an Intel Neuromorphic Research Community Grant

[1]More commonly known as Vector Symbolic Architectures

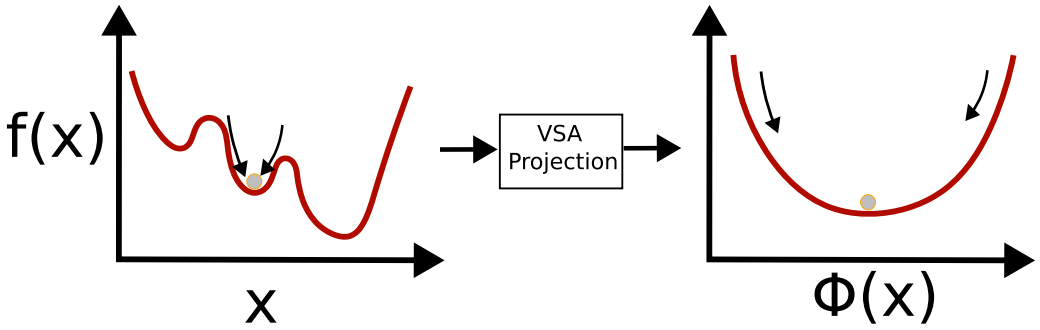

Fig. 2: VSAs projects data into a high-dimensional vector space. Reformulating the mutual information objective function in the high-dimensional vector space turns the problem into a convex optimization.

at that time.

In this paper, we propose an implementation of Furlong *et al.*'s algorithm as a recurrent dynamical system, providing a path forward to neural implementation. Further, we use the Hexagonal Spatial Semantic Pointer (Hex-SSP) encoding to represent data, which further ties the implementation to grid cells [13], [14]. We also provide an implementation of the optimization algorithm written in Lava, a programming framework for the Intel Loihi neuromorphic processors [15], [16]. We test the performance of this implementation using different numerical precision.

The contributions of this work include:

1) Modelling Bayesian optimization as recurrent dynamics;
2) Implement Bayesian optimization in the Hexagonal SSP representation space;
3) The implementation of the algorithm using the Lava [15] framework;
4) The comparison of algorithm performance as a function of numerical precision; and
5) Comparison of an approximation of Bayesian optimization that simplifies the learning rule and demands on neuromorphic implementations.

In the rest of the document we provide a brief introduction to the representations and algorithm used in this paper (Section II), then we provide the derivation of the recurrent dynamic implementation of the optimization algorithm and the process-based implementation in the Lava framework (Section III), and then present results on standard optimization target functions (Section IV), and conclude in Section V.

II. Background

Below (Section II-A), we provide a brief overview of how the Holographic Reduced Representation (HRR) VSA represents and encodes data, through the representation of atomic components (atomic symbols and vectors) and the operations that are used to construct representations of data structures. Those interested in greater detail are referred to the work of [9], [11], [17], [18]. We also briefly describe the BO formulation presented in Furlong *et al.* [19], which is the optimization algorithm we use in this document (Section II-B).

A. VSA Data Encoding

VSAs are a family of algebras that can be used to implement cognitive models that can be implemented by neural networks [6]–[9], [18]. VSAs operate on high-dimensional vectors using a limited set of algebraic operations, in this work we use the HRRs algebra of [8]. We prefer this algebra because it is dimensionality preserving.

In this document, we use representations of continuous data designed to model the activity of grid cells in the medial entorhinal cortex [14], representations we refer to as Hexagonal Spatial Semantic Pointers (SSPs) [13], [14], or Hex-SSPs. Data are encoded using Hex-SSPs by taking an input vector, $\mathbf{x} \in \mathbb{R}^m$, and projecting it into a d-dimensional vector space:

$$\phi(\mathbf{x}) = \mathcal{F}^{-1}\left\{\mathrm{e}^{iA\mathbf{x}}\right\},$$

where $A \in \mathbb{R}^{d\times m}$ is an encoding matrix, which is constrained to have conjugate symmetry, and $\mathcal{F}^{-1}$ is the inverse discrete Fourier transform. The matrix A is designed such that the dot product between encoded values approximates a kernel that is a sum of sinc kernels aligned along an m-simplex (see [14] for details). This differs from the approach of [19], which used a randomly generated A matrix, but the optimization should be agnostic to the encoding scheme, as will be seen below.

One can also use VSAs to represent discrete and categorical objects, and can further compose these atomic representations to create representations of various data structures. More detail on how to construct such representations are available in references [8], [9], [17], [20]. Performing BO over more complex spaces is described in forthcoming work [21]. Because the dot product in this space induces a positive definite kernel, we can use it in the kernel trick for approximating Gaussian processes, and hence in Bayesian optimization, which we describe below.

B. BO Formulation

The BO formulation we use follows Furlong *et al.*'s implementation of Contal *et al.*'s mutual information Bayesian optimization [5]. The objective is to find the sampling location, x^*, that maximizes a function $f(\cdot)$. The function domain is

sampled to provide a set of candidate sampling locations, $\mathcal{X}$. The algorithm computes an acquisition function, given by

$$a_t(\mathbf{x}) = \mu_t(\mathbf{x}) + \sqrt{\gamma_t + \sigma_t^2(\mathbf{x})} - \sqrt{\gamma_t}, \tag{1}$$

where $\mu_t(\mathbf{x})$ is the current estimate of $f(\mathbf{x})$, $\sigma_t^2(x)$ is the predicted variance of $\mu_t(\mathbf{x})$, and γ_t accumulates the predicted variance of previously observed locations. The highest-scoring candidate sample location is selected for follow-up observations, which are used to update the algorithm that predicts $\mu_t(\mathbf{x})$ and $\sigma_t^2(\mathbf{x})$.

The mutual information algorithm of Contal *et al.* [22] used Gaussian process regression to estimate a Gaussian distribution over observations $y = f(\mathbf{x})$, parameterized by a mean and variance. In our algorithm, the mean and variance, $\mu_t(x)$ and $\sigma_t^2(x)$ respectively, are provided by a BLR over the SSP representation of the points in $\mathcal{X}$.

Inverting the kernel trick, Furlong *et al.* approximates the components of the acquisition function using BLR. The online update rules for the BLR parameters are defined, per Bishop [23, §3.3] as:

$$\Sigma_t^{-1} = \Sigma_{t-1}^{-1} + \beta\phi(\mathbf{x}_t)^T\phi(\mathbf{x}_t) \tag{2}$$

$$\mathbf{m}_t = \Sigma_t\Sigma_{t-1}^{-1}\mathbf{m}_{t-1} + \Sigma_t\beta\phi(\mathbf{x}_t)y_t. \tag{3}$$

The predicted mean and variance were computed using

$$\mu_t(\mathbf{x}) = \mathbf{m}_{t-1}^T\phi(\mathbf{x}) \tag{4}$$

$$\sigma_t^2(\mathbf{x}) = \frac{1}{\beta} + \phi(\mathbf{x})^T\Sigma_{t-1}\phi(\mathbf{x}). \tag{5}$$

Ultimately, the objective is to maximize the acquisition function, $a_t(\mathbf{x})$. Fortunately, when expressed in the vector space of Hex-SSPs, this function is convex (see Fig. 2 for an illustration), and can be readily optimized using gradient methods. We describe this gradient method below.

III. Method

Encoding gradient-based optimization into recurrent neural dynamics is a well-established practice. To translate this into the Lava programming framework of the Loihi2 processor, we break the optimization routine down into independent processes, which mirror the model predictive control optimization described in. Below we start by deriving recurrent dynamics that must be implemented by the neural implementation (Section III-A). Next we sketch the processes used to implement the optimization and that we have implemented in Lava (Section III-B). Finally, we demonstrate the performance of the Lava implementation and how performance changes as the precision of the numerical representations are changed.

A. Gradient Derivation

The acquisition function we are trying to optimize is given in (1). To compute the gradient, we take the gradient of $a(\mathbf{x})$ with respect to the vector $\phi(\mathbf{x})$,

$$\nabla_\phi a(\mathbf{x}) = \mathbf{m} + \frac{1}{\sqrt{\phi(\mathbf{x})^T\Sigma\phi(\mathbf{x}) + \beta^{-1} + \gamma_t}}\Sigma\phi(\mathbf{x})\,. \tag{6}$$

Then, for a certain step size η, the dynamics that maximize the objective function are given by:

$$\phi(\mathbf{x})_{t'} = \phi(\mathbf{x})_t + \eta\nabla_\phi a(\mathbf{x})\,. \tag{7}$$

To implement these methods in neurons, we would use a population of neurons to represent the current solution, $\mathbf{x}_t$, and then a second population to transform the current solution state into the gradient step for the next iteration of the function. Over time the solution neurons converge to the optimal solution, and the state can be decoded downstream.

B. Lava Implementation

The vectors that are passed between the two populations (current solution state, current gradient) are the interface between these populations. Hence, we can abstract away the neural implementation as processes that are communicating vectors recurrently. In Lava, we implement this method using two processes:

1) The Solution neurons - these neurons hold an encoding of the solution, $\phi(\mathbf{x})_t$. The process is functionally defined in Algorithm 1.
2) The MI Gradient process - this population computes the gradient as a function of the state of the solution neurons. The process is functionally specified in Algorithm 2.

The system defined by these two processes can be run for a fixed number of steps, or until convergence. The architecture of the overall network is given in Fig. 1. Note that the MI gradient method relies on the parameters learned by the Bayesian Linear Regression.

Algorithm 1 Solution Neurons Behaviour Specification

0: **procedure** SolutionNeurons(**Input:** $\nabla a(\mathbf{x})_t$, η, **Output:** $\mathbf{x}_t$)
0: $\quad\nabla a(\mathbf{x})_t \leftarrow \text{receive}(\mathbf{input})$
0: $\quad\mathbf{x}_t \leftarrow \mathbf{x}_t + \eta\nabla a(\mathbf{x})_t\mathrm{d}t$
0: $\quad\text{send}(\mathbf{x}_t)$
0: **end procedure**=0

The process receives input from the process that computes the gradient and integrates that update into the internal state, $\mathbf{x}_t$, with a direct connection and a recurrent connection, as described in Algorithm 1. To implement the gradient dynamics we define the process described in Algorithm 2. The full code for this implementation is available at **removed for review**.

IV. Results

We tested the algorithm on the optimization target functions used in [19]. We compare against an implementation of Contal *et al.'s* algorithm [5], where $\mu_t(x)$ and $\sigma_t^2(x)$ are provided by a Gaussian Process regression using a Matérn kernel, operating on the raw input vectors, $x \in \mathcal{X}$. GP-Matern was implemented using the GPy library [24] and uses 64-bit numerical precision.

Both algorithms are initialized with ten observations that are used to optimize the kernel length scale hyperparameter.

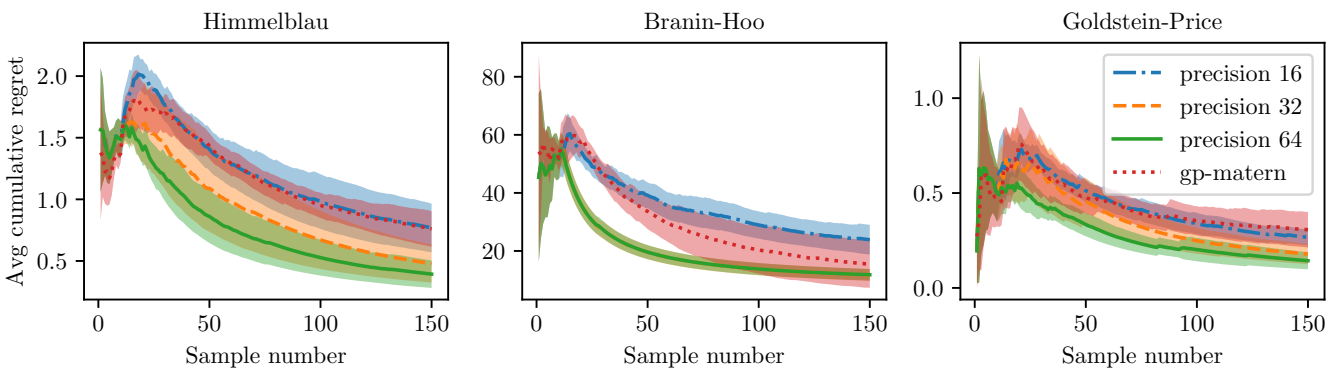

Fig. 3: The Average cumulative regret for the Lava implementation of the neural dynamics version of the Spatial Semantic Pointer Bayesian Optimization (SSP-BO) algorithm. In all cases, we see improvements in performance as sample number increases, but performance tends to decrease with lower-precision floating point representations. Additionally, our approach implemented with 32- or 64-bit precision outperforms the non-neural GP-Matern baseline algorithm, which uses 64-bit precision. Test run for $N = 10$ trials, shaded regions represent a 95% confidence interval.

Algorithm 2 MI Gradient Process Specification

procedure SOLUTIONNEURONS(**Input:** $\mathbf{x}_t$ **Parameters:** $\mathbf{m}_t, \Sigma_t, \beta^{-1}, \gamma_t$, **Output:** $\nabla a(\mathbf{x})_t$)
  $x_t \leftarrow \text{receive}(\textbf{input})$
  $s_t \leftarrow \mathbf{x}_t^T \Sigma_t \mathbf{x}_t$
  $\nabla a(\mathbf{x})_t \leftarrow -\left(\mathbf{m}_t + \frac{1}{\sqrt{s_t+\gamma_t+\beta^{-1}}}\Sigma_t \mathbf{x}_t\right)$
  $\text{send}(\nabla a(\mathbf{x})_t)$
end procedure=0

For SSP-BO, candidate sample locations are transformed into SSPs, $\phi(x)$. In this work, we use the same length scale parameter, h, for all vector elements, although that is not necessary. Initialization points were selected by randomly shuffling the candidate locations and using the first 10 points. For both algorithms, the hyperparameters were only optimized on the initial 10 samples, and not modified afterwards.

The average regret performance is displayed in Fig. 3. From these results we draw three observations. First, we were successfully able to implement the optimization algorithm using the recurrent dynamics. Second, the algorithm is able to continually optimize the value of the selected actions despite the precision. Thirdly, the lower the precision, the slower the rate of reduction in regret for the optimization algorithm. Notably, our approach implemented with 32- or 64-bit precision outperforms the non-neural GP-Matérn baseline algorithm, which uses 64-bit precision. Additionally, the 16-bit precision implementation is only marginally worse than the GP-Matérn baseline.

By inspection, the variability of the algorithm performance does not appear to be substantially affected by the numerical precision. We also observe that in the Himmeleblau and Goldstein-Price the regret performance of the algorithm increases with the decrease in precision. In the case of the Branin-Hoo algorithm the 64- and 32-bit floating point representations have identical performance.

V. Discussion and Conclusion

We have shown that the proposed Bayesian optimization method successfully translates to implementation in recurrent dynamics. This proof of concept of a recurrent dynamics implementation of Bayesian optimiziation in the VSA space provides a pathway to implementing the BO algorithm on neuromorphic hardware. We were able to implement this algorithm simply due to the convexification of the acquisition function, which was enabled by optimizing in the VSA space. We have also shown that while the optimization procedure is sensitive to the precision of the implementation, the optimization procedure is still able to continually improve results.

The dynamics described in this system are agnostic of the encoding scheme, which suggests that this optimization scheme may well extend to other data formats encoded using VSA methods. We have released a Lava implementation of the recurrent dynamics which we will integrate into the Lava optimization package.

However, we observe that the VSA encoding that we use in this implementation relies on dense representations. In order to make better use of the resources available on the Loihi 2, this algorithm will best benefit from integration with a sparse fractional binding technique, akin to the method proposed by Frady *et al.* [25]. We are also investigating methods for approximating the Bayesian linear regression, in order to reduce the complexity of the learning rule, and hence the circuitry required to implement the algorithm.

References

[1] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic com-

puting with loihi: A survey of results and outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.
[2] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022, 2010.
[3] D. Y. Little and F. T. Sommer, "Learning in embodied action-perception loops through exploration," *arXiv preprint arXiv:1112.1125*, 2011.
[4] D. Y.-J. Little and F. T. Sommer, "Learning and exploration in action-perception loops," *Frontiers in neural circuits*, vol. 7, p. 37, 2013.
[5] E. Contal, V. Perchet, and N. Vayatis, "Gaussian process optimization with mutual information," in *International Conference on Machine Learning*, pp. 253–261, PMLR, 2014.
[6] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.
[7] P. Smolensky, G. Legendre, and Y. Miyata, "Integrating connectionist and symbolic computation for the theory of language," Tech. Rep. CU-CS-628-92, University of Colorado, Boulder, 1992.
[8] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural networks*, vol. 6, no. 3, pp. 623–641, 1995.
[9] C. Eliasmith, *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.
[10] A. R. Voelker, "A short letter on the dot product between rotated Fourier transforms," *arXiv preprint arXiv:2007.13462*, 2020.
[11] E. P. Frady, D. Kleyko, C. J. Kymn, B. A. Olshausen, and F. T. Sommer, "Computing on functions using randomized vector representations," *arXiv preprint arXiv:2109.03429*, 2021.
[12] A. Rahimi, B. Recht, *et al.*, "Random features for large-scale kernel machines.," *NIPS*, vol. 3, no. 4, p. 5, 2007.
[13] B. Komer, T. C. Stewart, A. Voelker, and C. Eliasmith, "A neural representation of continuous space using fractional binding.," in *CogSci*, pp. 2038–2043, 2019.
[14] N. S.-Y. Dumont and C. Eliasmith, "Accurate representation for spatial cognition using grid cells," in *42nd Annual Meeting of the Cognitive Science Society. Toronto, ON: Cognitive Science Society*, pp. 2367–2373, 2020.
[15] Intel's Neuromorphic Computing LabINCL, *Lava Software Framework*. Intel, 2021.
[16] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient neuromorphic signal processing with loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 254–259, IEEE, 2021.
[17] A. R. Voelker, P. Blouw, X. Choo, N. S.-Y. Dumont, T. C. Stewart, and C. Eliasmith, "Simulating and predicting dynamical systems with spatial semantic pointers," *Neural Computation*, vol. 33, no. 8, pp. 2033–2067, 2021.
[18] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.
[19] P. M. Furlong, T. C. Stewart, and C. Eliasmith, "Fractional binding in vector symbolic representations for efficient mutual information exploration," in *ICRA Workshop: Towards Curious Robots: Modern Approaches for Intrinsically-Motivated Intelligent Behavior. 1ś5*, 2022.
[20] D. Kleyko, M. Davies, E. P. Frady, P. Kanerva, S. J. Kent, B. A. Olshausen, E. Osipov, J. M. Rabaey, D. A. Rachkovskij, A. Rahimi, *et al.*, "Vector symbolic architectures as a computing framework for nanoscale hardware," *arXiv preprint arXiv:2106.05268*, 2021.
[21] P. M. Furlong, N. S.-Y. Dumont, R. Antanova, J. Orchard, and C. Eliasmith, "Efficient exploration using neuromorphic Bayesian optimization on trajectory spaces," *in submission*, 2023.
[22] J. Conklin and C. Eliasmith, "A controlled attractor network model of path integration in the rat," *Journal of computational neuroscience*, vol. 18, no. 2, pp. 183–203, 2005.
[23] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[24] GPy, "GPy: A Gaussian process framework in Python." http://github.com/SheffieldML/GPy, since 2012.
[25] E. P. Frady, D. Kleyko, and F. T. Sommer, "Variable binding for sparse distributed representations: Theory and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.