Fractional Binding in Vector Symbolic Architectures as Quasi-Probability Statements

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

Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, ON, Canada

Abstract

Distributed vector representations are a key bridging point between connectionist and symbolic representations of cognition. It is unclear how uncertainty should be modelled in systems using such representations. One may place vector-valued distributions over vector representations, although that may assign non-zero probabilities to vector symbols that cannot occur. In this paper we discuss how bundles of symbols in Vector Symbolic Architectures (VSAs) can be understood as defining an object that has a relationship to a probability distribution, and how statements in VSAs can be understood as being analogous to probabilistic statements. We sketch novel designs for networks that compute entropy and mutual information of VSA-represented distributions. In this paper we restrict ourselves to operators proposed for Holographic Reduced Representations, and representing real-valued data. However, we suggest that the methods presented in this paper should translate to any VSA where the dot product between fractionally bound symbols induces a valid kernel.

Keywords: Vector Symbolic Architecture; Fractional Binding; kernel estimation

Introduction

Vector Symbolic Architectures (VSAs) bridge connectionist and symbolic representations of cognition, but it is unclear how probability should be modelled using VSAs. We suggest that VSAs already model probability and we show that if the similarity function between fractionally bound quantities induces a (quasi-)kernel function, then VSA statements on bundles of vector symbols are analogous to probabilistic statements. While we restrict ourselves to the operators proposed for Holographic Reduced Representations (HRRs; Plate, 2003), and in particular their use in the Semantic Pointer Architecture (SPA; Eliasmith, 2013), we infer that the presented results translate to VSAs where similarity induces a meaningful kernel or quasi-kernel functions of the encoded data points, and where similarity distributes over bundling.

Prior approaches use populations of spiking neurons to predict vector representations of probability functions (e.g, Ma et al., 2008; Sharma et al., 2017; Sharma, 2018). Here individual neurons represent bins around sample points in the domain of a probability distribution and their activity corresponds to the probability mass in that bin. Eliasmith (2013, §7.4) illustrates how the cortical-basal ganglia-thalamus loop can effect Bayesian inference (as has Bogacz (2015)) and further, how to update distributions using HRR operations. In this paper, we go further and provide explicit VSA operations for computing operations on probability distributions – how to marginalize a distribution, how to compute entropy, and how to compute the mutual information between two random variables.

Another approach to representing probability in spiking neurons is to treat the spiking activity as samples from a distribution with a one-to-one mapping between neurons and random variables (Buesing et al., 2011; Pecevski et al., 2011). By integrating neural activity one can extract samples from the probability distribution encoded in the network, akin to a classification neural network with a softmax output layer.

It is implicit in the methods discussed in this paper that we can construct neural networks where the activity of a neuron corresponds to a bin around one value or a multi-modal distribution over multiple values. Recognizing the connection between VSA statements and probability-like statements provides a great deal of flexibility when designing networks to compute probabilistic inference, and also to interpret cognitive models in a quasi-probabilistic manner.

First we review concepts that are necessary to build the connection between VSAs and kernel density estimators. Next we draw the analogies between VSA operations and probability statements. The remainder of the document discusses information theoretic quantities that may be calculated by neural networks, including novel circuits for computing entropy and mutual information using HRRs, and discusses implications of using the SPA to model probability.

Preliminaries

In this section we review the concepts that make the connection between VSAs and kernel density estimators. First, we briefly discuss Kernel Density Estimators (KDEs) and how RFFs have been used to improve the memory and time complexity of these systems. Next, we briefly review VSA operations, grounded in the use case of the SPA.

Kernel Density Estimators and Random Fourier Features

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 $P(X = x) = \frac{1}{nh}\sum_{i=1}^{n} k_h(x, x_i)$ 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 $\phi(x) = (e^{i\omega_1 x}, \dots e^{i\omega_d x})^T$, 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[1], and as $d \to \infty$ the kernel approximation is exact.

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). We can see the memory benefits by applying the kernel approximation to the definition of a KDE:

$$P(X = x) = \frac{1}{nh} \sum_{i=1}^{n} \phi\left(\frac{x}{h}\right) \cdot \phi\left(\frac{x_i}{h}\right)$$

Because the dot product distributes over the summation, we can rewrite it as:

$$P(X = x) = \phi\left(\frac{x}{h}\right) \cdot \frac{1}{nh} \sum_{i=1}^{n} \phi\left(\frac{x_i}{h}\right)$$

For a fixed dataset, the term $\frac{1}{nh}\sum_{i=1}^{n} \phi\left(\frac{x_i}{h}\right)$ is a vector, making the complexity of querying the KDE $O(d)$, instead of $O(n)$. The EXPoSE algorithm (Schneider, Ertel, & Ramos, 2016; Schneider, 2017), for example, uses RFFs for fast anomaly detection in large datastreams in finite memory. Fourier features have also been applied to Gaussian Process Regression (Rahimi et al., 2007; Mutnỳ & Krause, 2019). As we will discuss next, there is a connection between RFFs and the generation of random vectors used in fractional binding in VSAs.

Vector Symbolic Architectures

VSAs represent symbols as vectors, and provide operators for acting on those symbol vectors. The symbols can represent discrete concepts, like integers or other atomic symbols, real-valued quantities, and even data structures (Eliasmith, 2013; Voelker et al., 2021; Kleyko et al., 2021). We focus on a special type of vector symbol called a Spatial Semantic Pointer (SSP; Komer, 2020). Algorithm 1 is one possible algorithm for generating new vector symbols, called every time a new vector symbol is needed. There are two choices that can be made in this procedure: the distribution used for generating the frequency components, $\theta_{i,j}$; and the dimensionality of the vector, d. Like with RFFs, different generating functions for the frequency components induces different kernel functions (Frady et al., 2021), and can provide improvements in the efficiency of the representation, (Komer, 2020; Dumont & Eliasmith, 2020), but we will use a uniform distribution over the frequency components. This leaves the choice of dimensionality as a choice constrained by application specific needs.

Algorithm 1 An algorithm for generating vector symbols to encode m-dimensional data. $\mathcal{U}(\cdot,\cdot)$ is the uniform distribution and $\mathcal{F}^{-1}$ indicates the inverse Fourier transform.

function GENERATE_VECTOR(d, m)
  $\Theta_{\mathrm{X}} = (\theta_{X,j,k}) \in \mathbb{R}^{d \times m}\, s.t.\, \theta_{X,j,k} \sim \mathcal{U}(-\pi, \pi)$
  $\mathrm{X} \leftarrow \mathcal{F}^{-1}\left\{e^{i\Theta_{\mathrm{X}}}\right\}$
  return X

Vectors generated by Algorithm 1 are both unit vectors and *unitary* vectors, meaning that the magnitude of all frequency components is 1. This property has two benefits: repeated convolution preserves the vector's magnitude; and the dot product is preserved up to scale between the Fourier and time domains. That is, for two vector symbols, $\mathrm{X}_1, \mathrm{X}_2$, it is the case that $w\mathrm{X}_1 \cdot \mathrm{X}_2 = \mathcal{F}\{\mathrm{X}_1\} \cdot \mathcal{F}\{\mathrm{X}_2\}$ for some $w \in \mathbb{R}$ (Voelker, 2020).

With symbols defined we now turn to SPA operations. The four operators used in this document are *similarity*, *bundling*, *binding*, and *unbinding*. *Similarity*, $\cdot$, compares two vector symbols. In the SPA *similarity* is the vector dot product. For two vectors, $x, y \in \mathbb{R}^d$, when $x = y$, $x \cdot y$ should be 1, and when $x \neq y$, $x \cdot y \approx 0$.

Bundling, also called *collecting* or *superposition*, denoted $+$, combines two vectors into a new vector that maintains some similarity with its constituent elements, *i.e.*, $x \cdot (x+y)$ and $y \cdot (x+y)$ should both be relatively large. In the SPA bundling is vector addition. Bundles of vectors can be understood as a set of the constituent symbols. Similarity distributes over bundling, $x \cdot (y+z) = x \cdot y + x \cdot z$, meaning the sum of similarity between a vector and all elements of a bundle can be computed with one operation.

Binding, denoted $\otimes$, combines two vector symbols into a new symbol that is dissimilar to either of the constituent components, *i.e.* $x \cdot (x \otimes y) \approx 0$ and $y \cdot (x \otimes y) \approx 0$. We implement binding with circular convolution[2], denoted $\circledast$. We will use $\otimes$ as notation except where we exploit properties of circular convolution. Binding is the basis for representing numbers. For integers, one generates a vector symbol, X, referred to as an *axis vector*, and binds it with itself an integer number of times (Komer, 2020; Dumont & Eliasmith, 2020). This is written $\mathrm{X}^n = \bigotimes_{i=1}^{n} \mathrm{X}$, where $n \in \mathbb{Z}$. Since SPA binding is circular convolution, and convolution in the time domain is multiplication in the Fourier domain, we can write $\mathrm{X}^n = \mathcal{F}^{-1}\left\{e^{i\Theta_{\mathrm{X}} n}\right\}$. Binding can be applied to real-valued numbers, called *fractional binding*, which we will use in this paper extensively (Plate, 1995; Komer, 2020).

[1] Depending on the desired kernel, there are more accurate encodings, see Sutherland and Schneider (2015)

[2] The SPA admits other binding operators, *e.g.* the Vector-derived transformation binding of Gosmann and Eliasmith (2019).

Unbinding, denoted $\oslash$, undoes, approximately, the binding operation. Given vector symbols, $x, y \in \mathbb{R}^d$, and $z = x \otimes y$, then $x \cdot (z \oslash y) \approx 1$ and $y \cdot (z \oslash x) \approx 1$. Unbinding in the SPA can be implemented by binding with an "inverted" vector. The pseudo-inverse of a vector $x = (x_1, \ldots, x_d)^T$ is $x^{-1} = (x_1, x_d, x_{d-1}, \ldots, x_2)^T$. Thus we write unbinding as either $y \approx z \oslash x$ or $y \approx z \otimes x^{-1}$.

With these operations, VSAs can produce models of cognition that map onto populations of neurons, as well as construct complex data structures and programs for neuromorphic hardware (Eliasmith, 2013; Kleyko et al., 2021). Readers interested in other VSAs operators are referred to (Gosmann & Eliasmith, 2019; Neubert et al., 2019; Schlegel et al., 2020). In the following section we show how to interpret VSA statements as density estimators.

Analogies to Probability Operations

In this section we show how operations in the SPA relate to probability statements. Section 7.4 of (Eliasmith, 2013) outlines one (strict) relationship between operations on vector symbols and probability distributions. We provide a different interpretation of how to convert similarities into probabilities. We assume a fixed dataset, $\mathcal{D} = \{x_1, \ldots, x_n | x_i \in \mathbb{R}^m\}$ of n samples of m-dimensional data. We discuss how the VSA operations discussed above imply probability statements.

Binding Encodes Data

Fractional binding projects data from some domain $\mathcal{X} \subseteq \mathbb{R}^m$, into the vector representations of the SPA. Fractional binding is mathematically equivalent to the inverse Fourier transform of data encoded with RFFs. As discussed in (Voelker, 2020), if our semantic pointers are unitary, the dot product is preserved, up to scale. As with RFFs, the frequency component distribution determines the kernel induced under similarity (Sutherland & Schneider, 2015; Frady et al., 2021). In this paper we use the uniform distribution over $[-\pi, \pi]$ to generate frequency components. We use a length scale parameter, h, so when we write $\mathrm{X}^{x/h}$ we mean $\mathcal{F}^{-1}\left\{e^{i\Theta x/h}\right\}$, for $x \in \mathbb{R}^m$.

In theory, data axes can have different generating distributions for their axis vectors, as long as the generated vectors remain unitary. However, when computing similarity it is important to encode all data with the same axis vectors and length scale parameter(s), h. It is beyond the scope of this paper, but these elements may form a concept of a data type for VSAs. We assume that for each data set the axis vectors will be randomly generated once and we will denote data encoded with those vectors and a particular length scale as $\mathrm{X}^{x/h}$.

Similarity Computes Probability

The fundamental analogy we are drawing is between computing probability with KDEs and measuring the similarity of a query point with a bundle of fractionally bound vector symbols. We define our estimator as $\hat{f}(x|\mathcal{D}) = \mathrm{X}^{x/h} \cdot \frac{1}{nh} \sum_{x_i \in \mathcal{D}} \mathrm{X}^{x_i/h}$. For any domain space $x \in X \subseteq \mathbb{R}^m$, we will denote the normalized sum as $M_{X,n} = \frac{1}{nh} \sum_{x_i \in \mathcal{D}} \mathrm{X}^{x_i/h}$. If we wish to highlight a subdivision of the elements in the vector representation x we may denote the sum $M_{X \times Y,n}$.

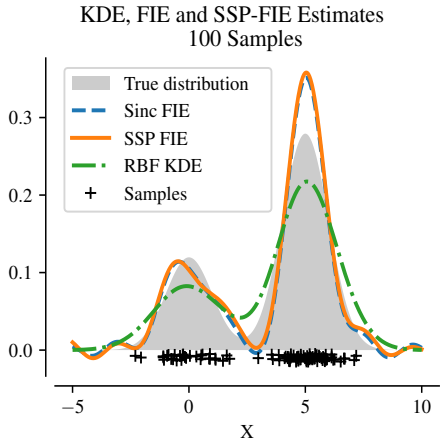

Figure 1: Fourier Integral Estimator, its approximation using 2048-d SSPs, and a KDE using a Gaussian kernel for 100 samples drawn from a GMM, indicated by the shaded region.

Using Algorithm 1 to generate axis vectors, the dot product between two SSPs induces the normalized sinc function (Voelker, 2020), which is a quasi-kernel, as it takes on negative values. Consequently, $\hat{f}(x|\mathcal{D})$, is not a KDE, but is the special-case Fourier Integral Estimator (FIE) (Davis, 1975, 1977). An optimal length scale, h, exists for the FIE (Glad et al., 2007; Chacón et al., 2007), and can be estimated by solving the equation $\|\varphi_n(1/h)\| = \frac{1}{\sqrt{n+1}}$ on $1/h \in [0, \sqrt{n}]$, where $\varphi_n(t)$ is the empirical characteristic function (Glad et al., 2007), or by cross validation. Fig. 1 shows the performance of a true FIE, and its approximation using a 2048-dimensional SSP representation.

While the FIE is not a probability estimator, it can be converted to one. Two techniques for doing so are due to Glad *et al.* (Glad et al., 2007, 2003) and Agarwal et al. (2016). Glad et al. (2003) developed corrections for different classes of quasi-kernels. The particular correction for the FIE is:

$$f_X(x) \approx \max\left\{0, \hat{f}_{\mathrm{FIE}}(x|\mathcal{D}) - \xi\right\}$$

$\xi \in \mathbb{R}$ is selected so $\int_{-\infty}^{\infty} \max\left\{0, \hat{f}_{\mathrm{FIE}}(x|\mathcal{D}) - \xi\right\} dx = 1$. If we use our VSA implementation of the FIE, we can rewrite the conversion as:

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

By inspection, this conversion is equivalent to a ReLU neuron with a bias, $b = -\xi$, and either $W = M_{X,n}$ or $W = \mathrm{X}^{x/h}$ are the synaptic weights. Letting $W = M_{X,n}$ frames populations of neurons as estimating the probability of a query point, $\mathrm{X}^{x/h}$, under different distributions, assuming a different $M_{X,n}$ for each neuron in the population. In this interpretation populations of neurons can be understood as collections of (inexact) density estimators, imprecision compounded by the

differences between ReLU and biological neurons' transfer functions. A population of neurons could be used as a boosted version of a KDE. In the same way, each neuron could be understood to represent a distribution conditioned on a random variable. On the other hand, if $W = \mathrm{X}^{x/h}$ then a population of neurons could be sampling the domain of the probability distribution represented by $M_{X,n}$, similar to the technique used in (Sharma et al., 2017). In the case where the memory, $M_{X,n}$, contains only one encoded point, this network would be an explicit probability code (Ma et al., 2008).

An alternative to the ReLU-style conversion is developed by Agarwal et al. (2016). Here the FIE output is squared, $f(x) \approx \|\phi_X(x;h) \cdot M'_{X,n}\|^2$, using a modified version of our standard memory term $M'_{X,n} = \sum_{x_i \in \mathcal{D}} c_i \mathrm{X}^{x_i/h}$. The modification requires solving for a set of weighting parameters, c_i. This technique is used by (Frady et al., 2021), to construct KDEs using a VSA. We observe that conversion of Agarwal et al. (2016) at least superficially resembles Born's rule for converting the quantum wave function into a probability (Born, 1926), hinting at a deeper connection between models of cognition based on quantum theory (Pothos & Busemeyer, 2013), as suggested by Stewart and Eliasmith (2013).

Both conversions have parameters that must be solved for and we do not comment on which method is preferable. Agarwal et al. (2016) provides an efficient method for solving for the weighting parameters, but it does require evaluating the Gram matrix. Solving for Glad *et al.*'s ξ requires computing the integral of a non-linear function of a VSA estimator. Regardless of the chosen conversion, the analogies to probability operations laid out in this paper hold. In this paper the symbol $\overset{C}{\approx}$ indicates that operations on vector symbols are approximating a density, using one of the above conversions, C, and indicate converted values by $f_X(x) \approx C[\mathrm{X}^{x/h} \cdot M_{X,n}]$.

Bundling Updates Beliefs

Updating a belief with observations is vector addition. If we have a memory unit, $M_{n-1} = \frac{1}{(n-1)h} \sum_{x_i \in \mathcal{D}_{n-1}} \mathrm{X}^{x_i/h}$, then updating the memory, and the distribution, is simply updating the running mean M_n. To ensure the KDE stays normalized by the number of samples we should write the update as:

$$M_{X,n} = \frac{1}{nh} \mathrm{X}^{x_n/h} + \frac{n-1}{n} M_{X,n-1}$$

If $M_{X,n}$, is represented by a population of neurons there is a concern of saturating the activity of the population. If the running average is computed exactly, then the length of $M_{X,n}$ should stay at 1, but computing the exact average requires an unbounded representation for n.

There is an implementation concern of whether or not to store the memory as a normalized or unnormalized sum. In either case, if the desire is to operate with normalized quasi-probabilities then there is a need to keep track of the length scale, h, and the number of data points in the kernel estimator. However, if an unnormalized $M_{X,n}$ is being represented by a population of neurons, there is the risk of saturating the activity of the neurons. It has been suggested that the saturation could act as a form of normalization (Eliasmith, 2013, §7.4).

Unbinding is Analogous to Conditioning

There are three ways to understand the unbinding operator acting on fractionally bound representations. First, $\mathrm{X}^{x/h} \otimes \mathrm{Y}^{y/h} \oslash \mathrm{Y}^{y'/h}$ can be viewed as shifting the representation y' units along the Y-axis. Committing to all query points being evaluated at $y = 0$ uncovers two other interpretations of unbinding. The second interpretation of unbinding is currying the evaluation of a joint probability distribution, *i.e.* $g(X) = f(X, Y = y) \overset{C}{\approx} \mathrm{X}^{x/h} \otimes Y^0 \cdot \sum_{x_i, y_i \in \mathcal{D}} \mathrm{X}^{x_i/h} \otimes Y^{\frac{y_i - y}{h}}$. Finally, recognizing that $f(X|Y = y) = \frac{1}{\eta} f(X, Y = y)$, then the unbinding operator can be understood as an unnormalized conditioned distribution. Normalizing the conditional distribution will require either memory or time, as we show below.

Encoding a 2-dimensional distribution with observations $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ in the usual way, $M_{XY,n} = \frac{1}{nh} \sum_{(x_i, y_i) \in \mathcal{D}} \mathrm{X}^{x_i/h} \circledast \mathrm{Y}^{y_i/h}$. To condition $M_{XY,n}$ on an observation, $Y = y$, we unbind the value y from the sum, $M_{XY,n}$, giving us:

$$M_{XY,n} \oslash \mathrm{Y}^{y/h} = \sum_{(x_i, y_i) \in \mathcal{D}} \mathrm{X}^{x_i/h} \otimes \mathrm{Y}^{\frac{y_i - y}{h}}$$

Taking the dot product between a query point $\mathrm{X}^{x/h} \otimes \mathrm{Y}^{y'/h}$ and the unbound memory, $M_{X|Y=y,n} = M_{XY,n} \oslash \mathrm{Y}^{y}$, then the result should be:

$$\begin{aligned} &\mathrm{X}^{x/h} \otimes \mathrm{Y}^{y'/h} \cdot M_{X|Y=y,n} \\ &= \mathrm{X}^{x/h} \otimes \mathrm{Y}^{y'/h} \cdot \sum_{(x_i, y_i) \in \mathcal{D}} \mathrm{X}^{x_i/h} \otimes \mathrm{Y}^{(y_i - y)/h} \end{aligned}$$

Setting $y' = 0 \implies \mathrm{X}^{x/h} \otimes \mathrm{Y}^0 = \mathrm{X}^{x/h}$, meaning $\mathrm{X}^{x/h} \cdot \left(M_{XY,n} \oslash \mathrm{Y}^{y/h}\right)$ is a valid similarity. The result is analogous to the joint probability of the query point $X = x$ with a fixed $Y = y$. It can be converted to a kernel-smoothed estimate of $\sum_{x_i \in \mathcal{D}} \mathrm{X}^{x_i/h}$ near $Y = y$ by a location-dependent normalizing term (Wand & Jones, 1995). This requires keeping a separate memory for the conditioning variables, as seen below. However, because the similarities can be negative, we may estimate $\pm f(x|y)$. Thus we convert before normalizing, to effect a conditional kernel density estimator (Rosenblatt, 1969)

$$f(X|Y = y) \approx \frac{C[\mathrm{X}^{x/h} \cdot M_{X|Y=y,n}]}{C[\mathrm{Y}^{y/h} \cdot M_{Y,n}]}$$

where $M_{Y,n} = \sum_{x_i, y_i \in \mathcal{D}} \mathrm{Y}^{y_i/h}$. This adds a burden of maintaining memory for every set of conditioning variables. Alternatively, one could normalize by the sum of all possible values of x, re-writing it as:

$$f(X|Y = y) \approx \frac{1}{\eta} C[\mathrm{X}^{x/h} \cdot M_{X|Y=y,n}]$$

where $\eta = \int_{\mathcal{X}} C[\mathrm{X}^{x/h} \cdot M_{X|Y=y,n}] dx$. This approach requires the time and mechanisms to compute η. To demonstrate the

performance of the conditioned distribution we compare a 2D Gaussian distribution with the analytical solution to conditioning, and the quasi-distribution induced by unbinding.

We created a bundle of $n = 5000$ observations sampled from the distribution $(x\ y)^T \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 1 \end{smallmatrix}\right]\right)$. We computed the solution to $P(Y|X = x)$ for the conditioning values $x \in \{-2,-1,0,1,2\}$, shown in the top row of Fig. 2, and computed $M_{Y|X=x,n}$. The SSP-FIE estimated distribution is given in the bottom row of Fig. 2. As with the 1D distribution, we used a 2047-d SSP representation. The mean of the distribution was computed using Born's rule and a normalizing constant, $\eta \approx \int_{-\infty}^{\infty} \|\mathrm{X}^{x/h} \cdot M_{X|Y,n}\|^2 dx$. This requires recomputing the normalizing constant, as does the approach of Glad *et al.*. In either case, we assert that $M_{Y|X,n}$ contains information that can approximate the conditional distribution.

Other Operations

To further expand on the above relationships we show how some standard probability operations can be implemented using bundles of fractionally bound vector symbols. We explain how to compute marginal distributions, entropy, and mutual information, and sketch how to sample from this distribution representation.

Marginalization produces a distribution over a subset of variables $U \subset V$ from a distribution over the variables V. This is conducted by integrating over the marginalized variables. In math: $f_X(x) = \int_{\mathcal{Y}} f_{XY}(x,y)dy$. Using our analogy we can re-write the marginalization process as

$$f_X(x) \overset{C}{\approx} \int_{\mathcal{Y}} \mathrm{X}^{x/h} \otimes \mathrm{Y}^{y/h} \cdot \left(\sum_{(x_i,y_i)\in\mathcal{D}} \mathrm{X}^{x_i/h} \otimes \mathrm{Y}^{y_i/h} \right) dy$$

and since binding and the dot product distribute over addition:

$$f_X(x) \overset{C}{\approx} \left(\mathrm{X}^{x/h} \otimes \int_{\mathcal{Y}} \mathrm{Y}^{y/h} dy \right) \cdot \left(\sum_{(x_i,y_i)\in\mathcal{D}} \mathrm{X}^{x_i/h} \otimes \mathrm{Y}^{y_i/h} \right)$$

Note that $\int_{\mathcal{Y}} \mathrm{Y}^{y/h} dy$ is another vector, and can be approximated by sampling the space Y, or it can be computed directly if the range of integration is finite. Denote $\Phi_Y = \int_{\mathcal{Y}} \mathrm{Y}^{y/h} dy$, then marginalization becomes $f_X(x) \overset{C}{\approx} \left(\mathrm{X}^{x/h} \otimes \Phi_Y\right) \cdot M_{XY,n}$. Assuming that $\mathrm{X}^{x/h}, \Phi_Y$, and $M_{XY,n}$ are column vectors, noting that convolution is commutative, and that circular convolution can be written as a matrix-vector product between one argument and the circulant matrix, $\mathrm{Circ}(\cdot)$, of the other argument, we can make the following simplification:

$$\left(\mathrm{X}^{x/h} \otimes \Phi_Y\right) \cdot M_{XY,n} = \left(\mathrm{Circ}(\Phi_Y)\mathrm{X}^{x/h}\right)^T M_{XY,n} \quad (1)$$

$$= \mathrm{X}^{x/h} \cdot \left(\mathrm{Circ}(\Phi_Y)^T M_{XY,n}\right) \quad (2)$$

So there is a linear map that marginalizes $M_{XY,n}$. The SSP estimator and the true marginalized distribution of the multivariate distribution over $(x\ y)^T$ is shown in Fig. 3.

Sampling from a KDE is fairly easy: randomly select a point in the dataset, then sample from the kernel function centred at that point. Compressing data points into a vector prevents direct sampling, further, the similarity-induced kernel is not a true kernel function. Rejection sampling could work for sampling from $M_{X,n}$, but it requires logic for the acceptance/rejection of generated samples and a mechanism for generating the initial random samples. Research into synaptic sampling (Elliott & Eliasmith, 2009; Buesing et al., 2011; Kappel et al., 2015) and basal ganglia models (Stewart et al., 2010) may provide some benefit.

Entropy ($\hat{H}_t$), a non-linear function of probability, can be estimated online by sampling observations $\mathrm{X}^{x/h} \sim G(M_{X,n})$, and updating a running average of their negative log probability, $\hat{H}_t = \frac{1}{t}\left(-\log\left(C[\mathrm{X}^{x/h} \cdot M_{X,n}]\right)\right) + \frac{t-1}{t}\hat{H}_{t-1}$. Representing an unbounded number of observations, t, in a neural network is challenging. Alternatives to an exact running average include low-pass filtering and computing entropy over a fixed window of samples.

Entropy can also be estimated in one additional time step, at the cost of memory. The single time step entropy computation relies on recognizing that in the Glad's conversion, $f(x) = \max\{0, \mathrm{X}^{x/h} \cdot M_{X,n} - \xi\}$, either $\mathrm{X}^{x/h}$ or $M_{X,n}$ can be the synaptic weights of a neuron. For sample points, $x_s \in X_S$, one can construct a neural network with a weight matrix, W_S:

$$W_S = \begin{bmatrix} | & | & & | \\ \mathrm{X}^{x_1/h} & \mathrm{X}^{x_2/h} & \dots & \mathrm{X}^{x_S/h} \\ | & | & & | \end{bmatrix}^T$$

and output $a(M_{X,n}) = max\{0, W_S \cdot M_{X,n} - \xi\}$. Then $a(M_{X,n})$ is a vector of the probabilities of the samples X_S. Approximate entropy can be computed $\hat{H} = \frac{1}{S}\sum_{i=1}^{S} -\log(a_i(M_{X,n}))$. The negative log can be computed by a single layer neural network, trained using *transformation* principle of the Neural Engineering Framework (Eliasmith & Anderson, 2003).

Mutual Information is a useful tool in a number of applications, including action selection for information gain (e.g., Loredo, 2003; Krause et al., 2008; Arora et al., 2019). Shannon Mutual Information is defined $\mathbb{E}\left[\log\left(\frac{f_{XY}(X,Y)}{f_X(X)f_Y(Y)}\right)\right]$. As with entropy, mutual information can be computed by time-averaged sampling, or by sampling the domain(s).

If X is a space of actions and Y is a space of observations, mutual information can be used to guide action selection by optimizing the objective $x = \arg\max_{x\in\mathcal{X}} \mathbb{E}\left[\log\left(\frac{f_{XY}(X,Y)}{f_X(X)f_Y(Y)}\right)\right]$. Rewriting the objective $x = \arg\max \mathbb{E}\left[\log\left(\hat{f}(Y|X=x)\right)\right] - \mathbb{E}\left[\log\left(\hat{f}(Y)\right)\right]$ shows we need two estimates. From our previous results we know we can rewrite $\hat{f}(Y|X=x) = C[W_Y \cdot M_{XY,n} \oslash \mathrm{X}^{x/h}]$, $\hat{f}(Y) = C[W_Y \cdot \mathrm{Circ}(\Phi_X)^T M_{XY,n}]$. W_Y is the weights of a neural network sampling the domain $\mathcal{Y}$, as for entropy. It should be possible to construct a neural network that outputs the expected information gain of an action, $\mathrm{X}^{x/h}$, for $M_{XY,n}$. Closing the loop should be possible using neural optimization techniques like the LCA optimization algorithm (Rozell et al., 2008; Shapero et al., 2014).

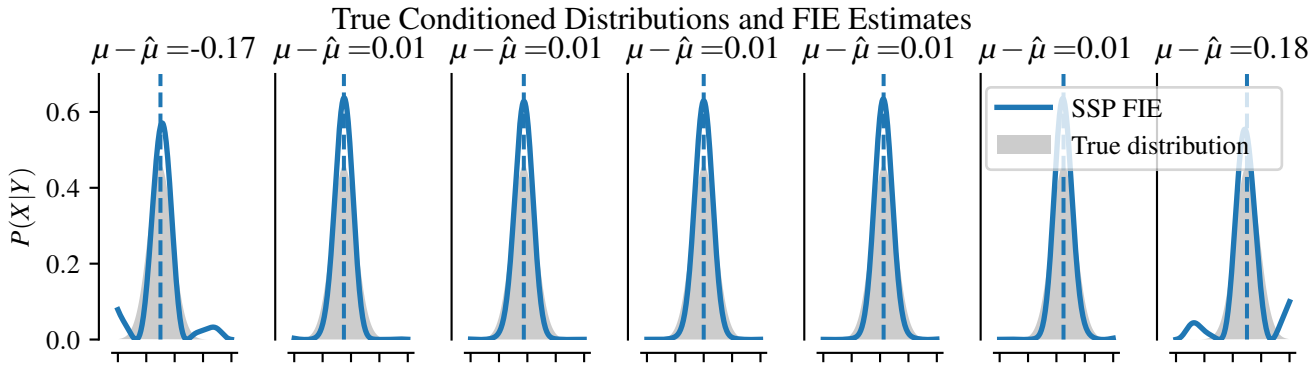

Figure 2: Conditioned distributions and their estimate from one sampling of 5000 observations. There is error between the true, μ, and estimated means, $\hat{\mu}$, for each conditioned distribution, however, we see the estimated mean in all cases shifts in the correct direction.

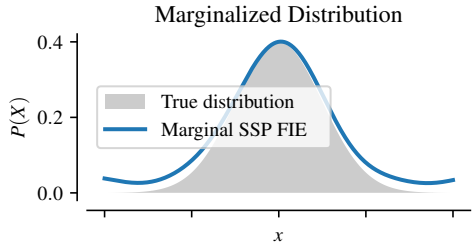

Figure 3: The multivariate distribution marginalized over Y and the SSP-FIE estimator using 5000 samples.

Discussion

There are two benefits of this framing of bundles of SSPs. First, it lets us interpret SSP representations as probabilistic statements. Second, it frames VSAs as a language for programming neuromorphic hardware to compute probabilities, possibly in constant time. However, there are some considerations for this approach: The accuracy of kernel approximations induced by similarity may be further limited by hardware, e.g., maximum firing rates, bit precision, and so on.

As well, *a priori* selection of the length scale, h, may not be suitable for online operations. It is not immediately obvious how to adjust h online, or how to implement that optimization as a learning rule for a neural network. Similarly, the FIE to KDE conversion parameters, Glad *et al.*'s bias term, ξ, and the Born rule normalizing constant, must be modified for new observations.

Working with non-normalized probabilities is possible, but may saturate neural populations. Avoiding saturation requires some form of forgetting, but what kind of forgetting is best for an application, and what this implies for the probability estimates, remains to be determined. A fixed discount factor in the range $]0, 1[$, instead of computing the running average, would produce exponential decay, inducing something like a recency bias. Conversely, saturation could provide normalization, as suggested by Eliasmith (2013).

We do not consider the neurological plausibility of interpreting SSP-probability statements as models of biology. If one neuron can represent a probability distribution, why are there so many neurons? It could be that neural populations represent boosted estimates of the true density estimate, or that more than one density is being represented, or that they are compensating for the discrepancies between the transfer functions of biological neurons and the ReLU function used in the Glad-style transformation.

Furthermore, algorithms for optimizing hyperparameters, and their implemented in neuromorphic hardware remain to be determined. As well, the choice of particular kernels have implications for biology (see Dumont & Eliasmith, 2020). In this paper we restrict ourselves to one quasi-kernel, but it should be possible to implement products of kernels, through binding, and addition of kernels through the concatenation of hypervectors. The search for network architectures that produce desirable kernels and probability statements is an open challenge that could benefit from existing research in neural architecture search. Further, we would like to explore modelling the probabilities of mixed integer and real-valued data, as well as more complex structured data, like those discussed in (Eliasmith, 2013; Voelker et al., 2021; Frady et al., 2021).

Conclusion

In this paper we have illustrated a connection between representations in the SPA and operations on probability distributions. Specifically, we have sketched novel methods for conditioning distributions and computing entropy and mutual information using HRRs. VSAs have garnered interest for combining symbolic and connectionist models of cognition, and more recently as a programming framework for neuromorphic computers (e.g., Mundy, 2017). The results in this paper show that VSAs can act like a *probabilistic* programming language, and can build probabilistic models of cognition. Open questions remain about best choices for implementation, and the limitations of hardware, but the connection between kernel methods and VSAs allows us to bring probabilistic models to cognitive modelling and neuromorphic computing.

Acknowledgments

The authors would like to thank Nicole Sandra-Yaffa Dumont, Drs. Jeff Orchard, Bryan Tripp, and Terry Stewart for discussions that helped improve this paper. This work was supported by CFI and OIT infrastructure funding as well as the Canada Research Chairs program, NSERC Discovery grant 261453, NUCC NRC File A-0028850, and AFOSR grant FA9550-17-1-0026.

References

Agarwal, R., Chen, Z., & Sarma, S. V. (2016). A novel nonparametric maximum likelihood estimator for probability density functions. *IEEE transactions on pattern analysis and machine intelligence*, *39*(7), 1294–1308.

Arora, A., Furlong, P. M., Fitch, R., Sukkarieh, S., & Fong, T. (2019). Multi-modal active perception for information gathering in science missions. *Autonomous Robots*, *43*(7), 1827–1853.

Bogacz, R. (2015). Optimal decision making in the cortico-basal-ganglia circuit. In *An introduction to model-based cognitive neuroscience* (pp. 291–302). Springer.

Born, M. (1926). Quantenmechanik der stoßvorgänge. *Zeitschrift für Physik*, *38*(11), 803–827.

Buesing, L., Bill, J., Nessler, B., & Maass, W. (2011). Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology*, *7*(11), e1002211.

Chacón, J., Montanero, J., Nogales, A., & Pérez, P. (2007). On the existence and limit behavior of the optimal bandwidth for kernel density estimation. *Statistica Sinica*, *17*(1), 289–300.

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

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

Dumont, N., & Eliasmith, C. (2020). Accurate representation for spatial cognition using grid cells. In *Cogsci.*

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

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

Elliott, L., & Eliasmith, C. (2009). Mcmc with spiking neurons. In *Nips workshop on bayesian inference in the brain.*

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

Glad, I. K., Hjort, N. L., & Ushakov, N. (2007). Density estimation using the sinc kernel. *Preprint Statistics*, *2*, 2007.

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

Gosmann, J., & Eliasmith, C. (2019, 05). Vector-derived transformation binding: An improved binding operation for deep symbol-like processing in neural networks. *Neural Computation*, *31*(5), 849-869. doi: 10.1162/neco_a_01179

Kappel, D., Habenschuss, S., Legenstein, R., & Maass, W. (2015). Synaptic sampling: a bayesian approach to neural network plasticity and rewiring. *Advances in neural information processing systems*, *28*, 370–378.

Kleyko, D., Davies, M., Frady, E. P., Kanerva, P., Kent, S. J., Olshausen, B. A., ... et al. (2021). Vector symbolic architectures as a computing framework for nanoscale hardware. *arXiv preprint arXiv:2106.05268*.

Komer, B. (2020). *Biologically inspired spatial representation* (Unpublished doctoral dissertation). University of Waterloo.

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, *9*(2).

Loredo, T. (2003). Bayesian adaptive exploration in a nutshell. *Statistical Problems in Particle Physics, Astrophysics, and Cosmology*, *1*, 162.

Ma, W. J., Beck, J. M., & Pouget, A. (2008). Spiking networks for bayesian inference and choice. *Current opinion in neurobiology*, *18*(2), 217–222.

Mundy, A. (2017). *Real time spaun on spinnaker functional brain simulation on a massively-parallel computer architecture*. The University of Manchester (United Kingdom).

Mutnỳ, M., & Krause, A. (2019). Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems 31*, 9005–9016.

Neubert, P., Schubert, S., & Protzel, P. (2019). An introduction to hyperdimensional computing for robotics. *KI-Künstliche Intelligenz*, *33*(4), 319–330.

Pecevski, D., Buesing, L., & Maass, W. (2011). Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS computational biology*, *7*(12), e1002294.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural networks*, *6*(3), 623–641.

Plate, T. A. (2003). Holographic reduced representation: Distributed representation for cognitive structures.

Pothos, E. M., & Busemeyer, J. R. (2013). Can quantum probability provide a new direction for cognitive modeling? *Behavioral and brain sciences*, *36*(3), 255–274.

Rahimi, A., Recht, B., et al. (2007). Random features for large-scale kernel machines. In *Nips* (Vol. 3, p. 5).

Rosenblatt, M. (1969). Conditional probability density and regression estimators. *Multivariate analysis II*, *25*, 31.

Rozell, C. J., Johnson, D. H., Baraniuk, R. G., & Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, *20*(10), 2526–2563.

Schlegel, K., Neubert, P., & Protzel, P. (2020). A comparison of vector symbolic architectures. *arXiv preprint arXiv:2001.11797*.

Schneider, M. (2017). *Expected similarity estimation for large-scale anomaly detection* (Unpublished doctoral dissertation). Universität Ulm.

Schneider, M., Ertel, W., & Ramos, F. (2016). Expected similarity estimation for large-scale batch and streaming anomaly detection. *Machine Learning*, *105*(3), 305–333.

Shapero, S., Zhu, M., Hasler, J., & Rozell, C. (2014). Optimal sparse approximation with integrate and fire neurons. *International journal of neural systems*, *24*(05), 1440001.

Sharma, S. (2018). *Neural plausibility of bayesian inference* (Unpublished master's thesis). University of Waterloo.

Sharma, S., Voelker, A., & Eliasmith, C. (2017). A spiking neural bayesian model of life span inference. In *Cogsci.*

Stewart, T. C., Choo, X., Eliasmith, C., et al. (2010). Dynamic behaviour of a spiking model of action selection in the basal ganglia. In *Proceedings of the 10th international conference on cognitive modeling* (pp. 235–40).

Stewart, T. C., & Eliasmith, C. (2013). Realistic neurons can compute the operations needed by quantum probability theory and other vector symbolic architectures. *Behavioral and Brain Sciences*, *36*(3), 307.

Sutherland, D. J., & Schneider, J. (2015). On the error of random fourier features. *arXiv preprint arXiv:1506.02785*.

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

Voelker, A. R., Blouw, P., Choo, X., Dumont, N. S.-Y., Stewart, T. C., & Eliasmith, C. (2021). Simulating and predicting dynamical systems with spatial semantic pointers. *Neural Computation*, *33*(8), 2033–2067.

Wand, M. P., & Jones, M. (1995). *Kernel smoothing* (1st ed. ed.). London ;: Chapman Hall.