Locality-preserving encodings (LPE):

Representing Continuous Values and Functions

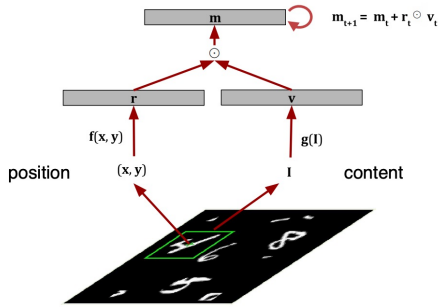
How can we extend HD Computing/VSA to non-symbolic computation?

Neuroscience 299 (Module 8)
October 20, 2021

Chris Kymn

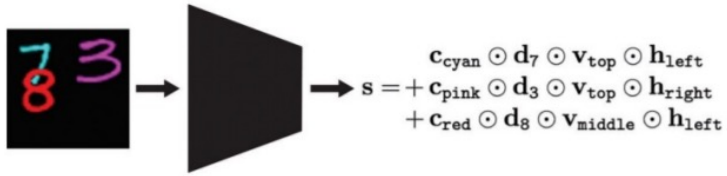
We've glimpsed at these ideas before…

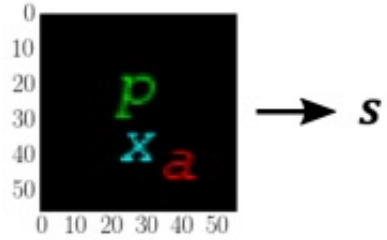
Visual working memory as a superposition of 'what' and 'where' bindings
(Eric Weiss, Ph.D. thesis)



2.1 STORING SEQUENCES BY TRAJECTORY-ASSOCIATION

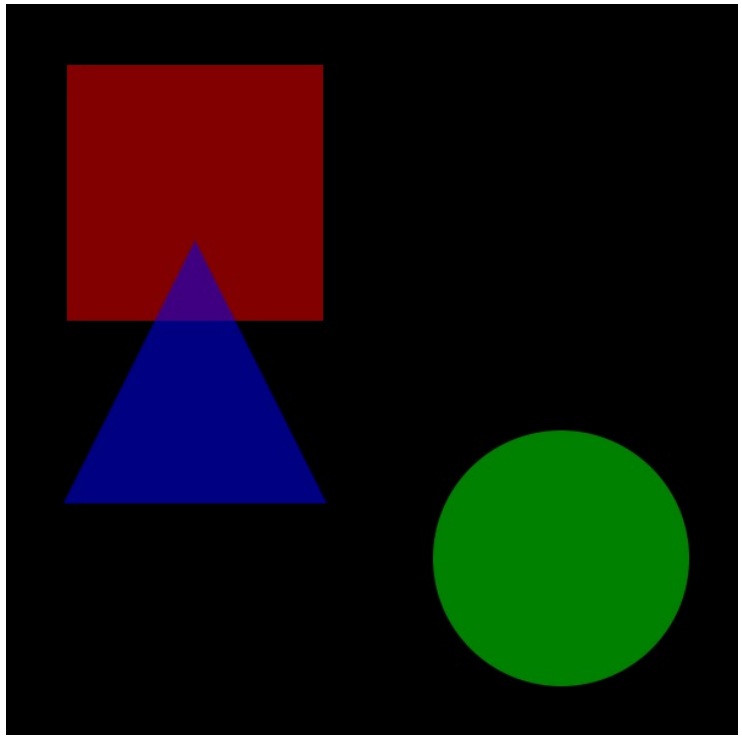
Elements of the sequence and loci (points) on the trajectory are all represented by n-dimensional vectors. The loci are derived from a single vector $\mathbf{k}$ – they are its successive convolutive powers: $\mathbf{k}^0$, $\mathbf{k}^1$, $\mathbf{k}^2$, etc. The convolutive power is defined in the obvious way: $\mathbf{k}^0$ is the identity vector and $\mathbf{k}^{i+1} = \mathbf{k}^i \circledast \mathbf{k}$.





$$\mathbf{s} = \sum_{x,y,c} \mathbf{R}_c \odot \mathbf{h}^x \odot \mathbf{v}^y \cdot Im(x, y, c)$$

Generalizing from Vectors to Symbols



VSA Formulation

$$I = \mathbf{c}_{red} \odot \mathbf{s}_{square} \odot \mathbf{v}_{top} \odot \mathbf{h}_{left} + \mathbf{c}_{blue} \odot \mathbf{s}_{triangle} \odot \mathbf{v}_{middle} \odot \mathbf{h}_{left} + \mathbf{c}_{green} \odot \mathbf{s}_{circle} \odot \mathbf{v}_{bottom} \odot \mathbf{h}_{right}$$

VFA Formulation

$$I = \mathbf{c}^{630} \odot \mathbf{s}_{square} \odot \mathbf{v}^{54.7} \odot \mathbf{h}^{51} + \mathbf{c}^{465} \odot \mathbf{s}_{triangle} \odot \mathbf{v}^{198} \odot \mathbf{h}^{48.24} + \mathbf{c}^{532} \odot \mathbf{s}_{circle} \odot \mathbf{v}^{352.1} \odot \mathbf{h}^{344}$$

LPE: Problem formulation

- Input: $\mathbf{x} \in \mathbb{R}^d$, $d \geq 1$
- Output: $\mathbf{z}(\mathbf{x}) \in V^n$, $n >> d$
- Goal: find a mapping x -> $\mathbf{z}(\mathbf{x})$ that is a Locality Preserving Encoding (LPE)
  - High similarity between $\mathbf{x}_1$ **&** $\mathbf{x}_2$ results in high similarity between $\mathbf{z}(\mathbf{x}_1)$ & $\mathbf{z}(\mathbf{x}_2)$
  - Low similarity between $\mathbf{x}_1$ **&** $\mathbf{x}_2$ results in high similarity between $\mathbf{z}(\mathbf{x}_1)$ & $\mathbf{z}(\mathbf{x}_2)$

Some methods we will cover today

- Older methods for designing LPEs
  - Thermometer code
  - Float/sliding code
  - Scatter code
- Kernel LPEs and Vector Function Architectures (VFAs)
- Designing kernels in VFAs

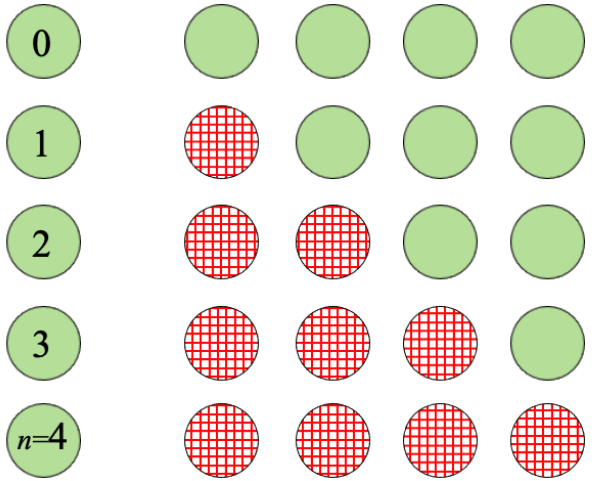
https://arxiv.org/abs/2109.03429

LPE for scalars: Thermometer code

- Encodes levels by the number of "activated" units
- Quantize *x* into *n* discrete levels *x* -> *s*, *s* ∈ [0, *n*]
- **z**(0) consists of all 0s.
- For other levels, the components of **z**(*s*) are determined by:

$$z_i(s) = \begin{cases} +1, & i \leq s \\ 0, & \text{otherwise} \end{cases}$$

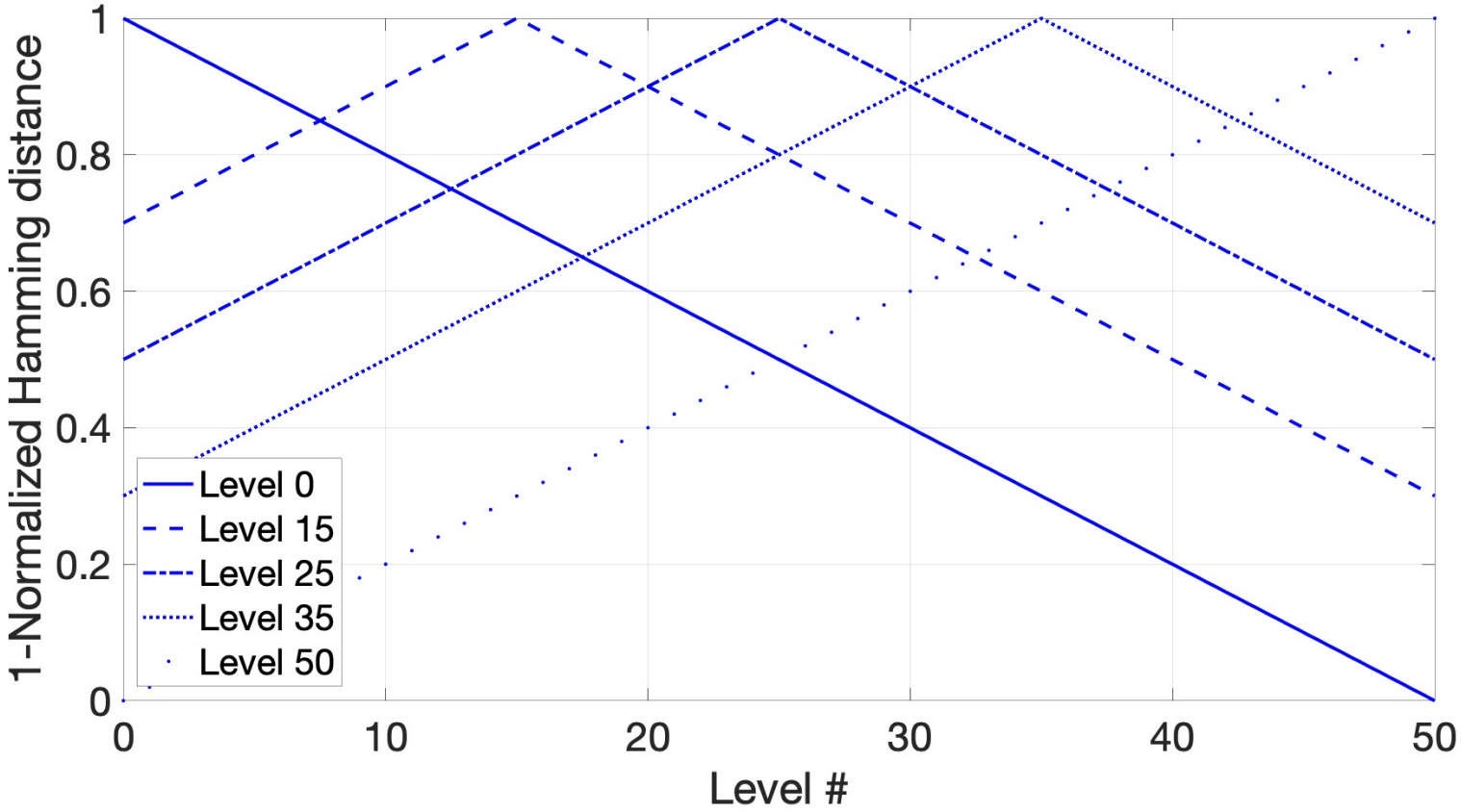
- Thermometer code can only represent *n*+1 levels



P.A. Penz, "The closeness code: An integer to binary vector transformation suitable for neural network algorithms," IEEE First Annual International Conference on Neural Networks (ICNN), pp. 515–522, 1987.

Thermometer code: Example

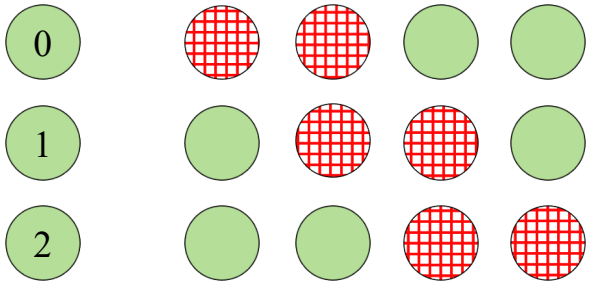
- n=50



LPE for scalars: Float/Sliding code

- Encodes levels by the number of "activated" units
  - The number of activated units in a code is fixed to *w*
  - Activated units are consecutive –> float
- Quantize *x* into *n* discrete levels *x* -> *s*, *s* ∈ [0, *n*-*w*]
  - Float/Sliding code can only represent *n*-*w*+1 levels
- For **z**(0): first *w* units are 1s; the rest are 0s.
- For other levels, the components of **z**(*s*) are determined by:

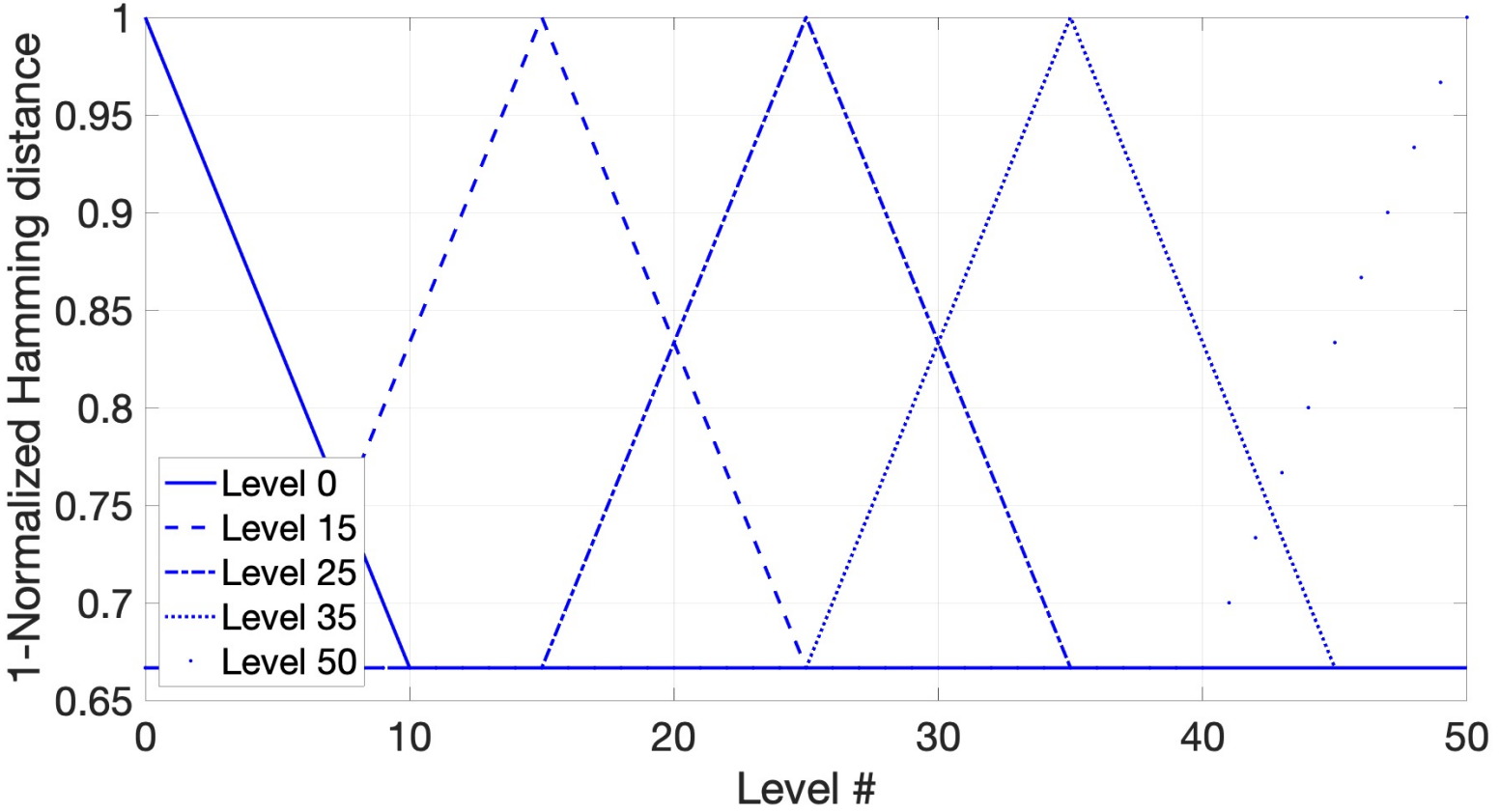
$$z_i(s) = \begin{cases} +1, & s \le i < s + w \\ 0, & \text{otherwise} \end{cases}$$



A.D. Goltsev, "An assembly neural network for texture segmentation," Neural Networks, vol. 4, no. 9 pp. 643–653, 1996.

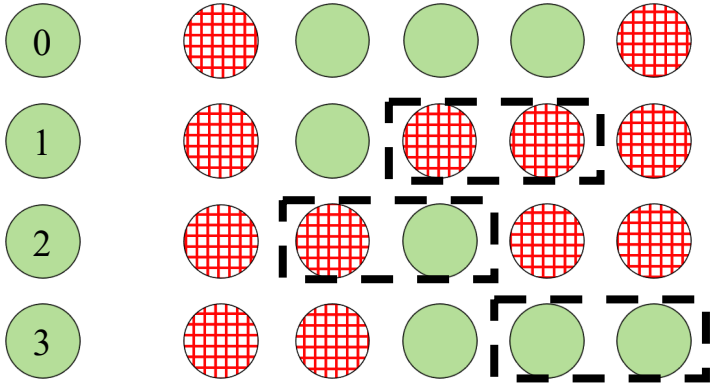
Float/Sliding: Example

- n=60; w=10



LPE for scalars: Scatter code

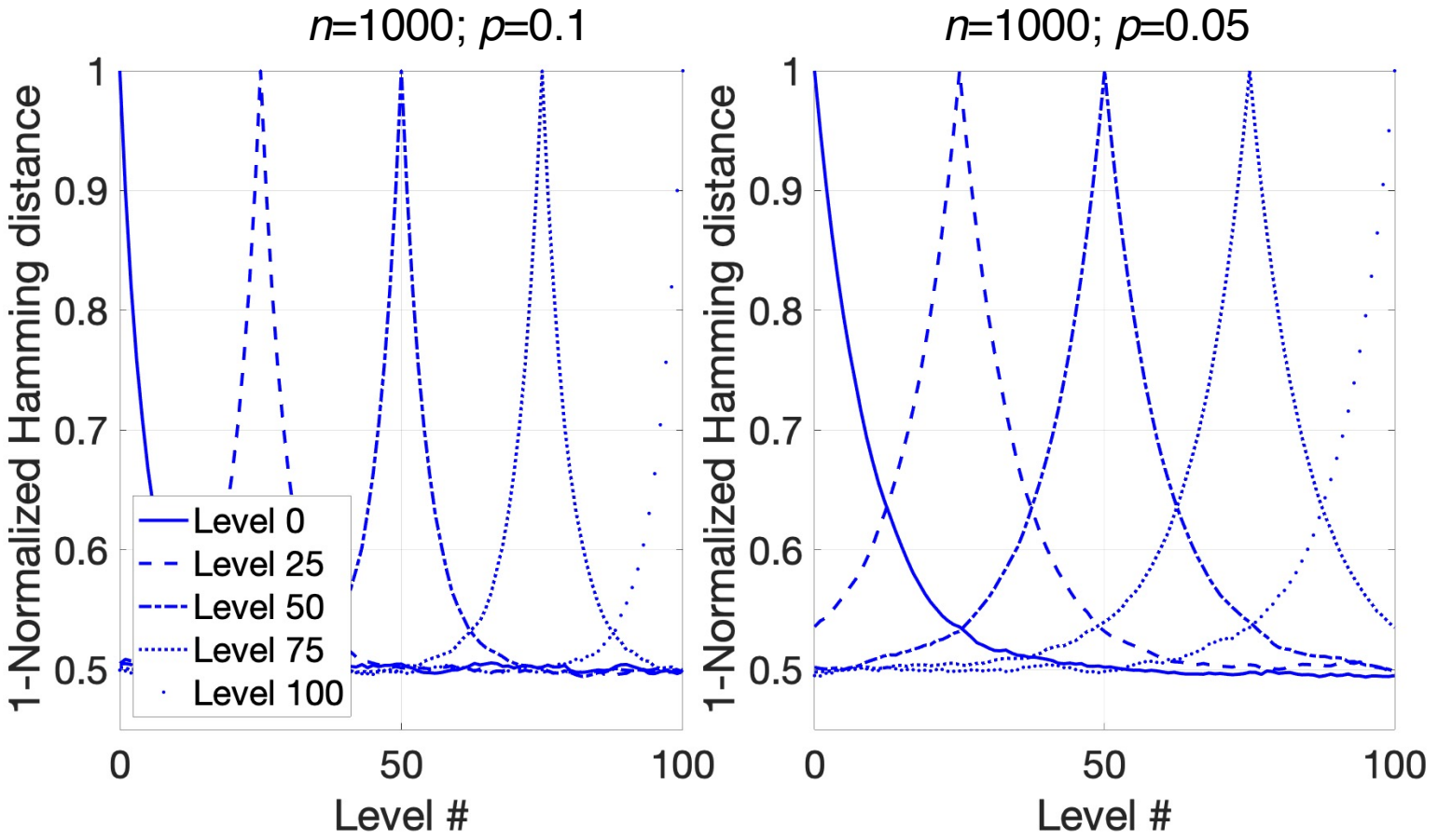
- Encodes levels by randomly swapping some of the units
- Similarity decays nonlinearly
- Real-valued input *x is* quantized into levels
  (note: no strict limitation on the number of levels)
- **z**(0) is a random dense binary vector
- Each subsequent code is obtained from the previous one by randomly swapping its components with some probability *p:*



$$z_i(s) = \begin{cases} z_i(s-1) + 1 \mod 2, & r_i \leq p \\ z_i(s-1), & \text{otherwise} \end{cases}$$

D. Smith and P. Stanford, "A random walk in Hamming space," International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 465–470, 1990.

Scatter code: Examples



Kernel LPE definition (formal)

Definition 1: A randomizing LPE encoding function $f : r \in \mathbb{R} \to \mathbf{z}(r) \in \mathbb{C}^n$ is a *kernel-LPE (KLPE)*, that is, it induces a similarity kernel, if the following requirement holds:

> In the limit for large dimensionality n, the *inner product* between point representations defines a translation-invariant similarity kernel:
>
> $$\mathbf{z}(r_1)^\top \overline{\mathbf{z}(r_2)} \overset{\text{large } n}{\longrightarrow} K(r_1 - r_2), \tag{9}$$
>
> with a kernel function $K(d)$, which is real-valued, assumes its maximum at $d = 0$ and gradually reaches zero for large $|d|$.Under quite general conditions, the convergence in (9), should be fast:
>
> $$\left|\mathbf{z}(r_1)^\top \overline{\mathbf{z}(r_2)} - K(r_1 - r_2)\right| \leq \epsilon \text{ for } n \propto O(g(m, \epsilon)) \tag{10}$$
>
> i.e., the required dimensionality of vectors is not very large (Rahimi and Recht, 2007; Thomas et al., 2020). We will discuss the required dimensionality of vectors for different concrete KLPE methods

Definition 2: A KLPE *is compatible with a VSA binding operation* $\circ$, if the addition of two values of the encoded variable can be represented by binding the individual representations of the values:

$$\mathbf{z}(r_1 + r_2) = \mathbf{z}(r_1) \circ \mathbf{z}(r_2). \tag{11}$$

Kernel LPE definition (informal)

- Inner product between point representations defines a translation invariant similarity kernel

$$\mathbf{z}(r_1)^\top \overline{\mathbf{z}(r_2)} \overset{\text{large } n}{\longrightarrow} K(r_1 - r_2)$$

- Compatible with VSA binding if binding adds encoded values:

$$\mathbf{z}(r_1 + r_2) = \mathbf{z}(r_1) \circ \mathbf{z}(r_2)$$

Example of Kernel LPE: Fractional binding

- Fractional binding is inspired by self-binding **z** *i* times with itself:

$$\mathbf{z}(x) = \underbrace{\mathbf{z} \odot \cdots \odot \mathbf{z}}_{\text{x times}} = \mathbf{z}^x$$

- With Fourier HRR:

$$\mathbf{z}(x) = [e^{i\phi_1 x}, e^{i\phi_2 x}, ..., e^{i\phi_n x}] \; ; \; \phi_j \sim U[-\pi, \pi)$$

- Observe that:
  - For FHRR, **z**(x) is well defined for any x ∈ $\mathbb{R}$
  - $\mathbf{z}(x_1) \odot \mathbf{z}(x_2) = \mathbf{z}(x_1 + x_2)$

T.A. Plate, "Holographic Recurrent Networks," Advances in Neural Information Processing Systems (NIPS), pp. 34-41, 1992.
T.A. Plate, "Distributed Representations and Nested Compositional Structure," University of Toronto, PhD Thesis, 1994.

Different Kernel LPEs

FPE with Hadamard product binding (Frady):

$$\mathbf{z}^{hp}(r) := (\mathbf{z})^r$$

FPE with Circular convolution binding (Plate, Eliasmith):

$$\mathbf{z}^{cc}(r) := \mathbf{z}^{(\circledast r)} \quad = \quad \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{z})^r\right) = F^{-1}\left(F\mathbf{z}\right)^r$$

Block-local circular convolution (Frady et al. IEEE TNNLS 2021):

$$\mathbf{z}^{lcc}(r)_{(\text{block } i)} := \mathbf{z}^{(*_B\, r)}_{(\text{block } i)} = F^{-1}\left(F\mathbf{z}_{(\text{block } i)}\right)^r$$

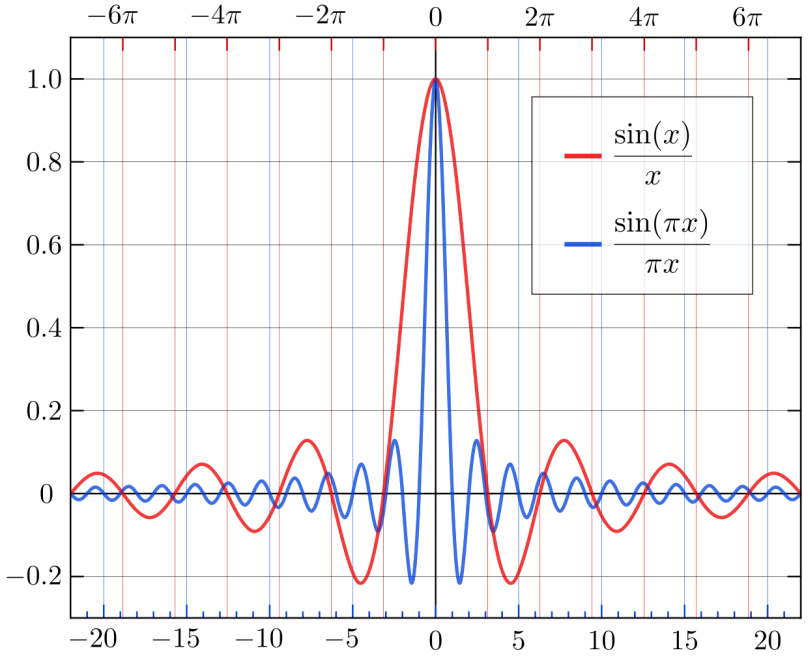
Resulting representations:

=> ”phasor”

=> ”real-valued”

=> ”sparse”

FPEs with uniformly sampled base vectors have a universal kernel



https://en.wikipedia.org/wiki/Sinc_function

Definition of Vector Function Architecture (VFA)

Definition 3: The combination of a VSA with a KLPE that is compatible with the VSA binding operation induces an RKHS of functions with inner product $\langle f, g \rangle := \int_{-\infty}^{\infty} f(r)g(r)dr$ that we will call a *Vector Function Architecture (VFA)*:

a) A function of the form:

$$f(r) = \sum_k \alpha_k K(r - r_k) \tag{12}$$

is represented via (10), $f(r) = \mathbf{y}_f^\top \overline{\mathbf{z}(r)}$, by the vector:

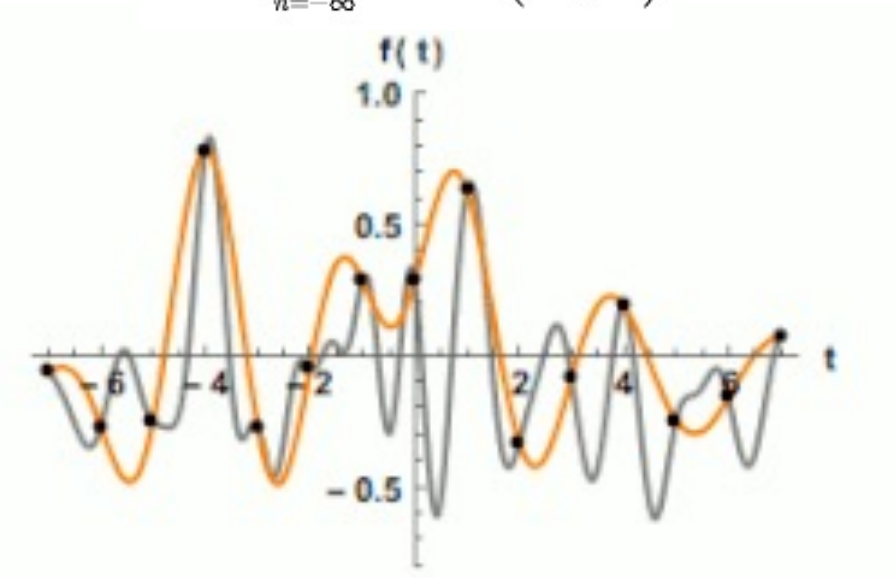
$$\mathbf{y}_f \quad = \quad \sum_k \alpha_k \mathbf{z}(r_k) \tag{13}$$

Approximating functions with sinc

Function Representation in VFA

$$f(r) = \sum_k \alpha_k K(r - r_k)$$

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT}{T}\right)$$



https://en.wikipedia.org/wiki/Whittaker%E2%80%93Shannon_interpolation_formula

Manipulating functions in VFA

- Point-wise readout of a function

$$f(s) = \langle f, K_s \rangle = \mathbf{y}_f^\top \overline{\mathbf{z}(s)}$$

- Point-wise addition

$$\mathbf{y}_{f+g} = \mathbf{y}_f + \mathbf{y}_g$$

- Function shifting

$$f(x) \to g(x) = f(x + r)$$

$$\mathbf{y}_g = \mathbf{y}_f \circ \mathbf{z}(r)$$

- Function convolution

$$\mathbf{y}_{f*g} = \mathbf{y}_f \circ \mathbf{y}_g$$

- Overall similarity between functions

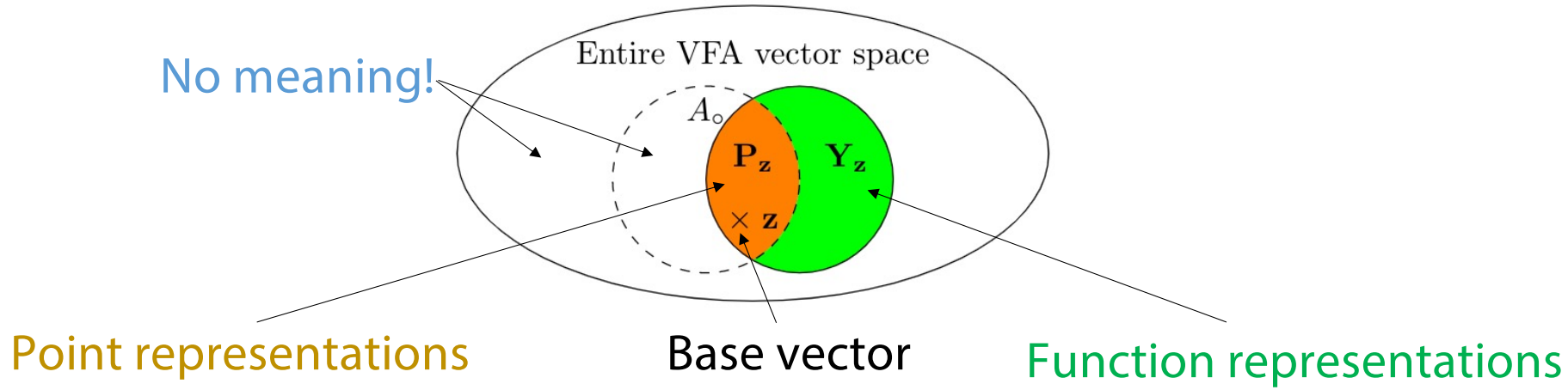
$$\langle f, g \rangle = \mathbf{y}_f^\top \overline{\mathbf{y}_g}$$

Vector space in FPE-VFA

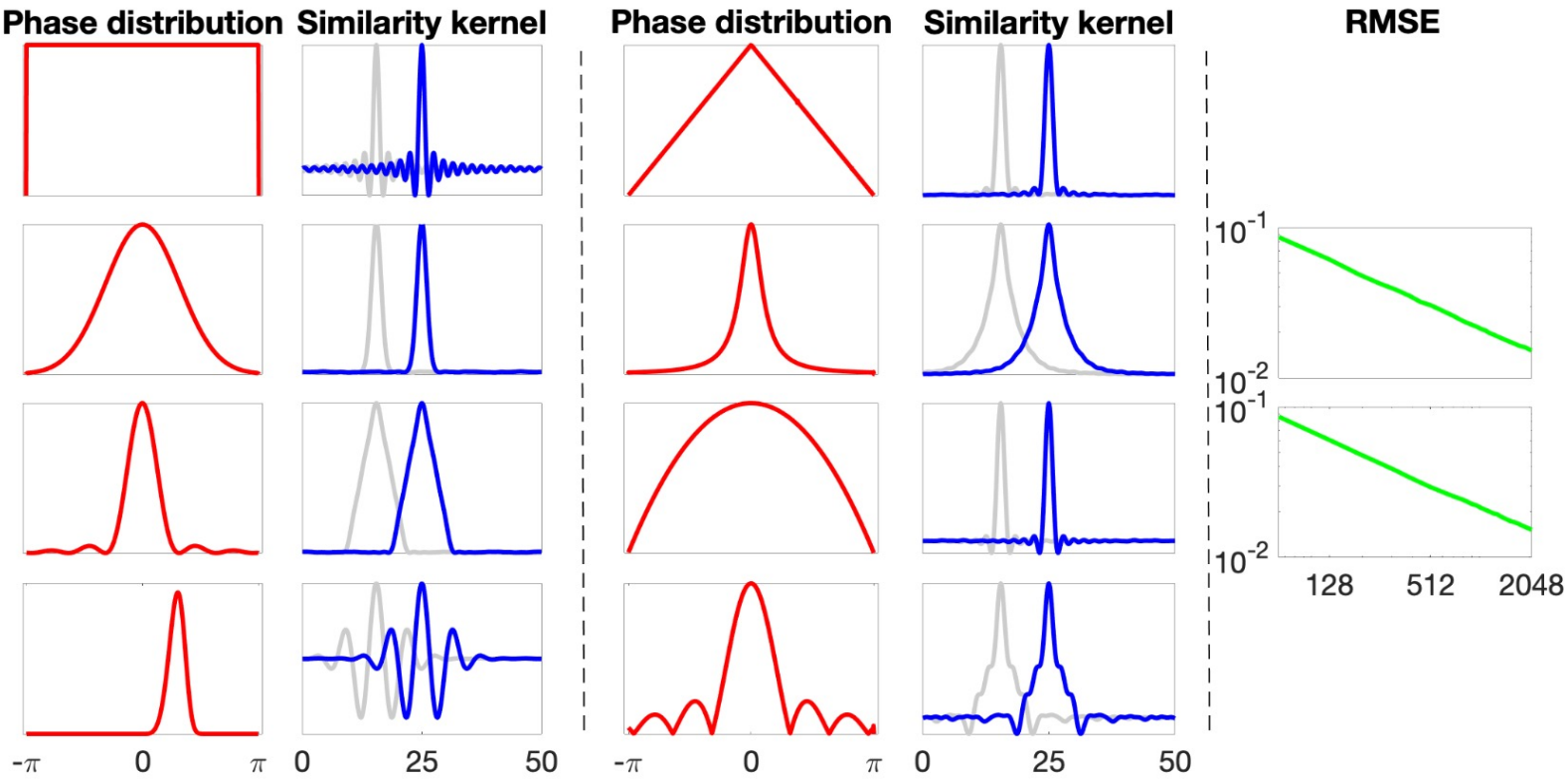
Let us denote the set of norm-preserving vectors with respect to the binding operation $\circ$ with:

$$A_\circ := \{\mathbf{z} : ||\mathbf{v} \circ \mathbf{z}||^2 = ||\mathbf{v}||^2 \; \forall \; \mathbf{v}\} \tag{23}$$

Note that with the base vector chosen in $A_\circ$, all points r are mapped by (22) to vectors also within $A_\circ$.



Phase distribution of the base vector determines similarity kernel

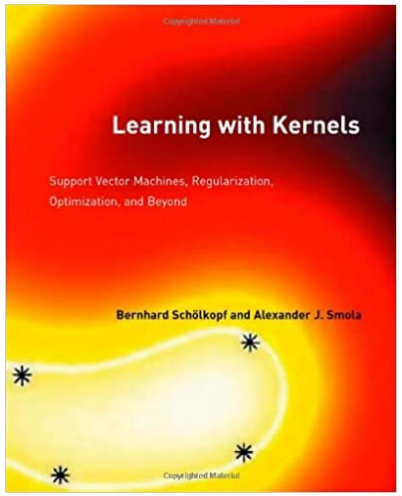


Random Features for Large-Scale Kernel Machines

Ali Rahimi and Ben Recht

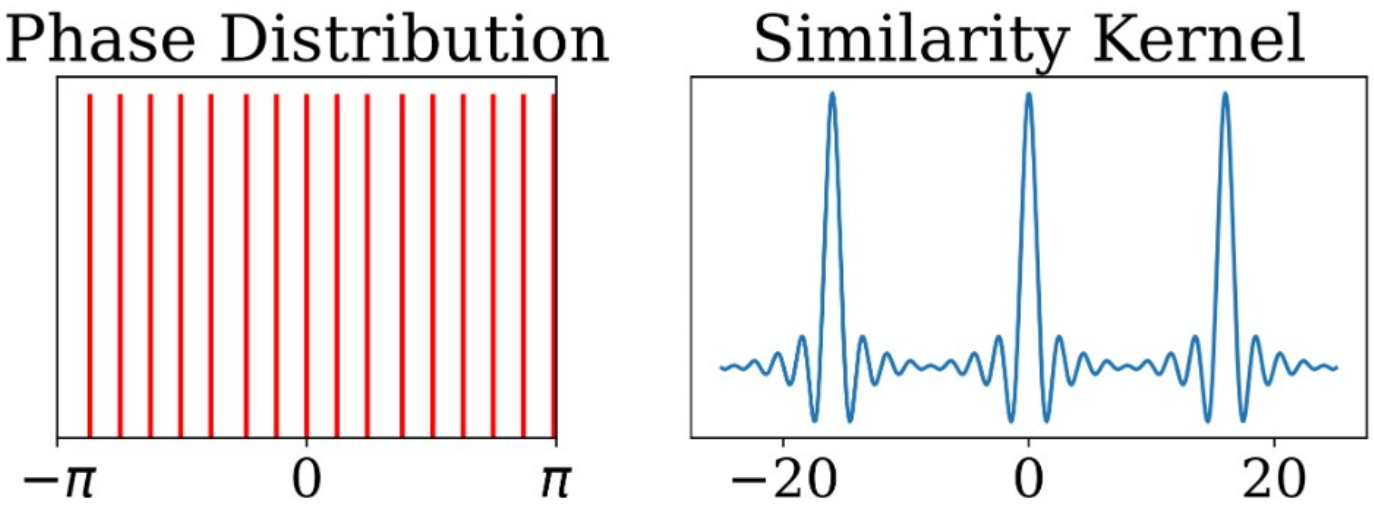
Algorithm 1 Random Fourier Features.

Require: A positive definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$.
Ensure: A randomized feature map $\mathbf{z}(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^D$ so that $\mathbf{z}(\mathbf{x})'\mathbf{z}(\mathbf{y}) \approx k(\mathbf{x} - \mathbf{y})$.
Compute the Fourier transform p of the kernel k: $p(\omega) = \frac{1}{2\pi} \int e^{-j\omega'\delta} k(\delta)\, d\Delta$.
Draw D iid samples $\omega_1, \cdots, \omega_D \in \mathcal{R}^d$ from p and D iid samples $b_1, \ldots, b_D \in \mathcal{R}$ from the uniform distribution on $[0, 2\pi]$.
Let $\mathbf{z}(\mathbf{x}) \equiv \sqrt{\frac{2}{D}} \left[\cos(\omega_1'\mathbf{x}+b_1) \cdots \cos(\omega_D'\mathbf{x}+b_D) \right]'$.



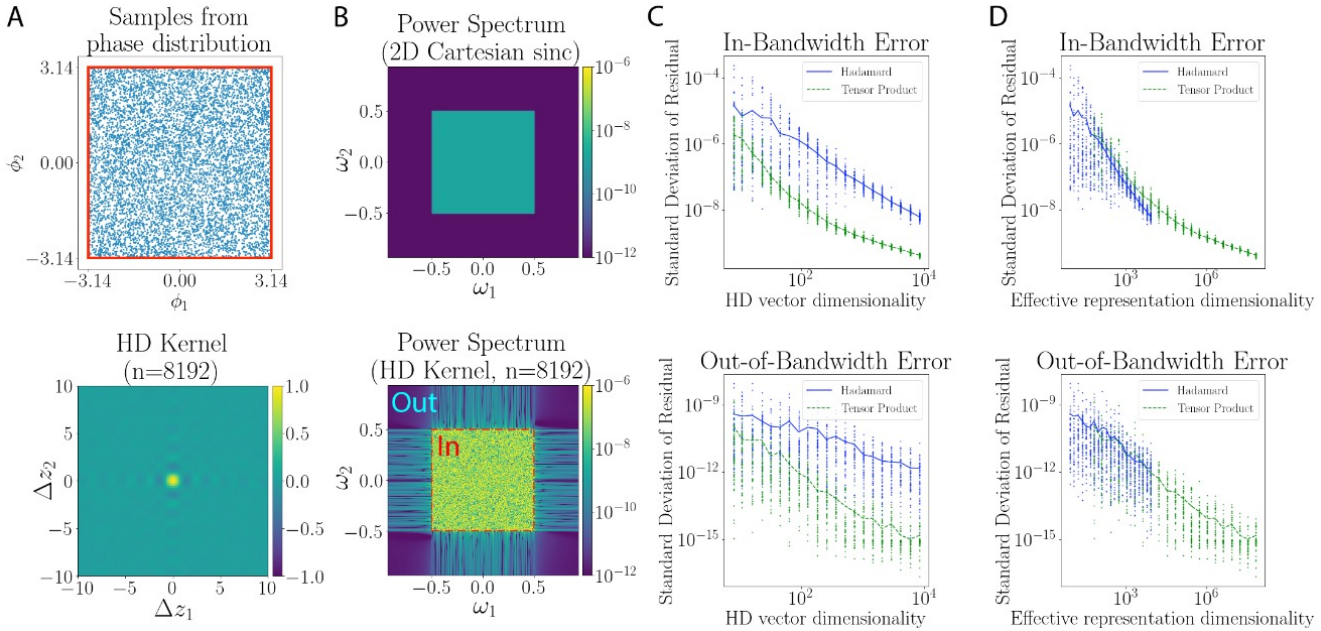
Theorem 1 (Bochner [13]). *A continuous kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ on $\mathcal{R}^d$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.*

Discrete phase distributions produce periodic kernels

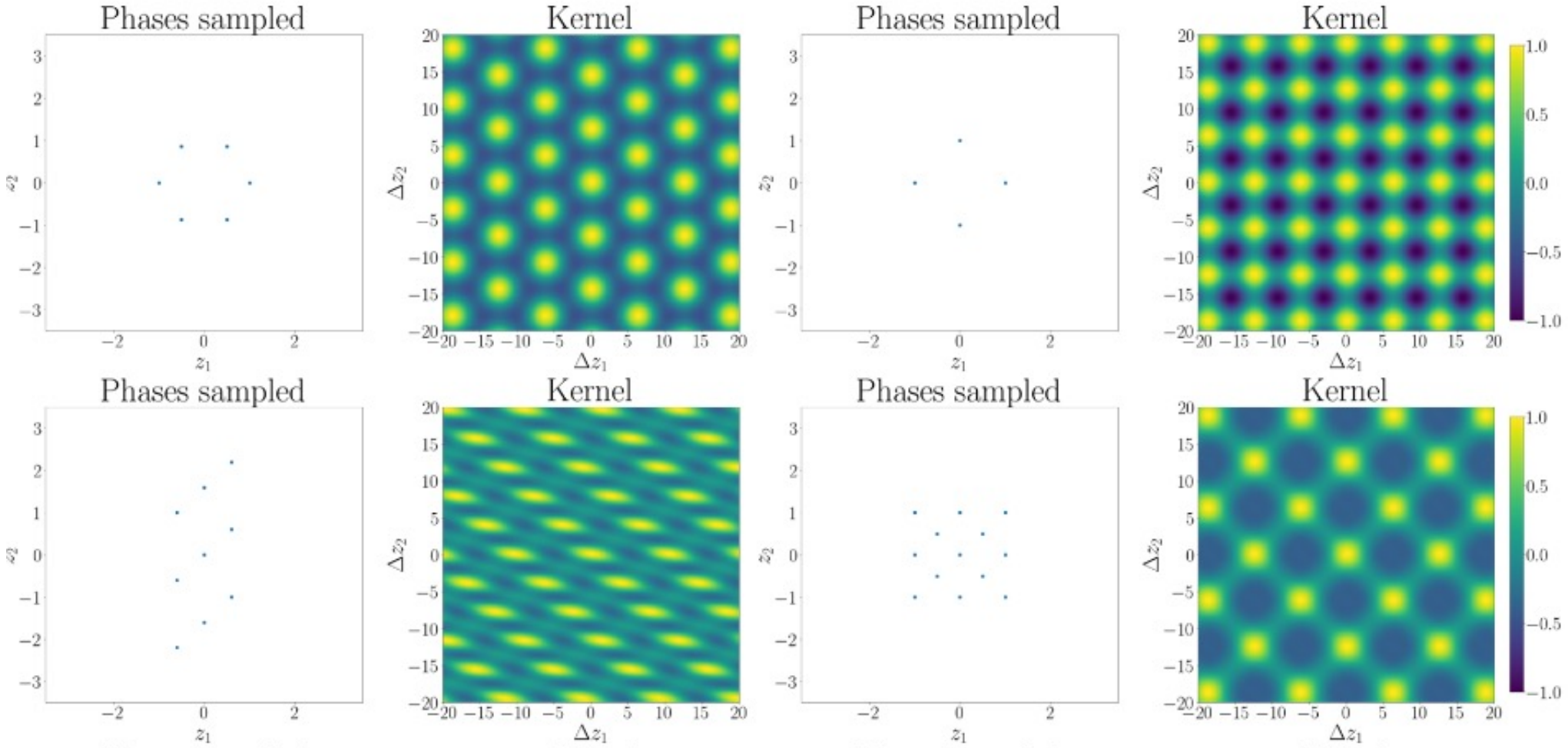


Binding FPEs to produce multi-dimensional kernels

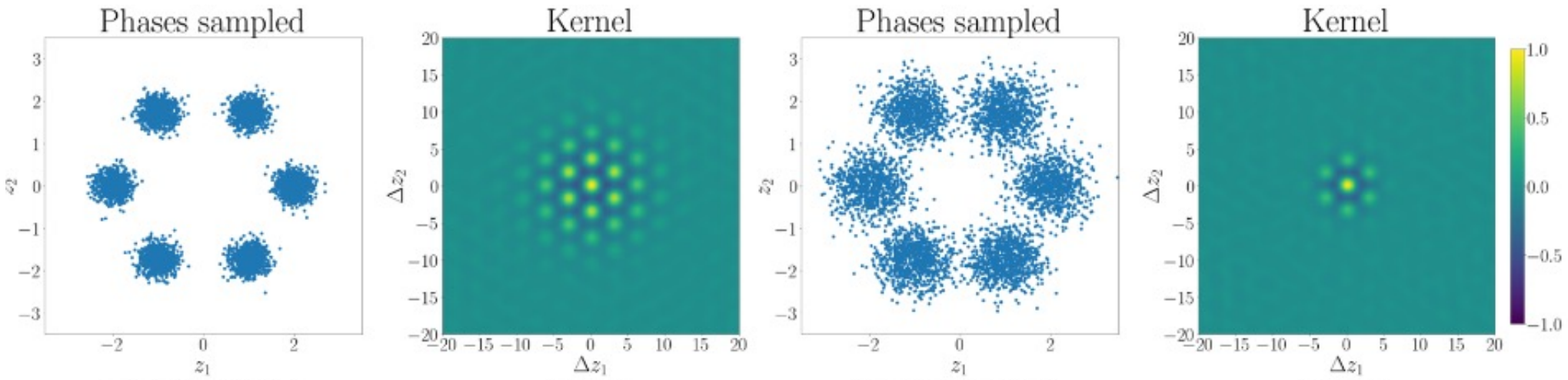
$$\mathbf{z}(\mathbf{r}) = \mathbf{z}_1(r_1) \circ \mathbf{z}_2(r_2) \circ \ldots \circ \mathbf{z}_m(r_m)$$



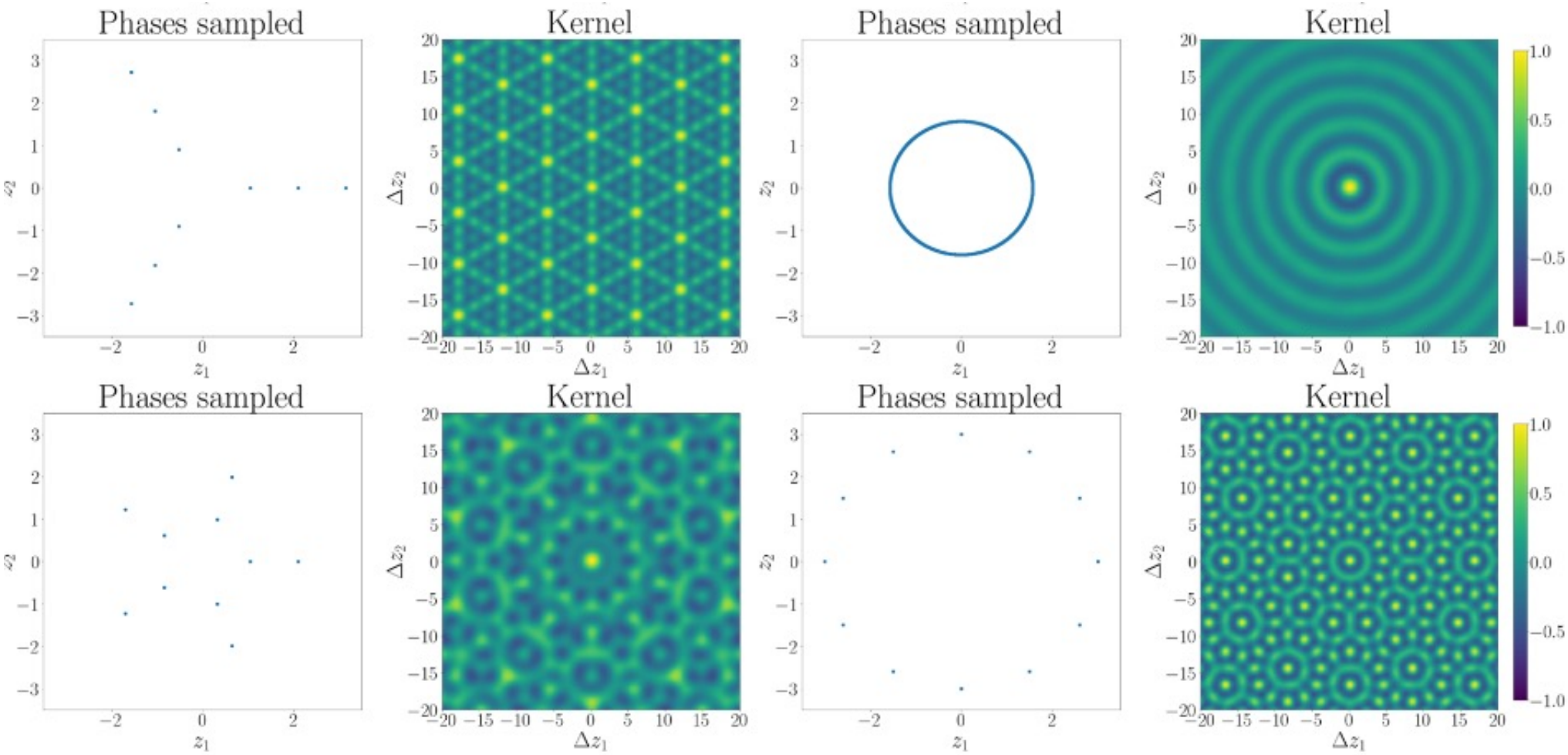
Periodic 2D kernels (Lattices)



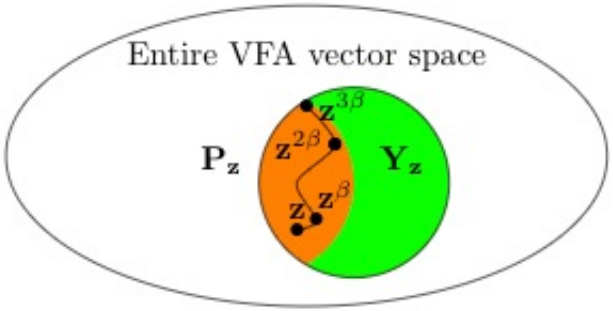
Periodic 2D kernels (finite coherence)



Periodic 2D kernels



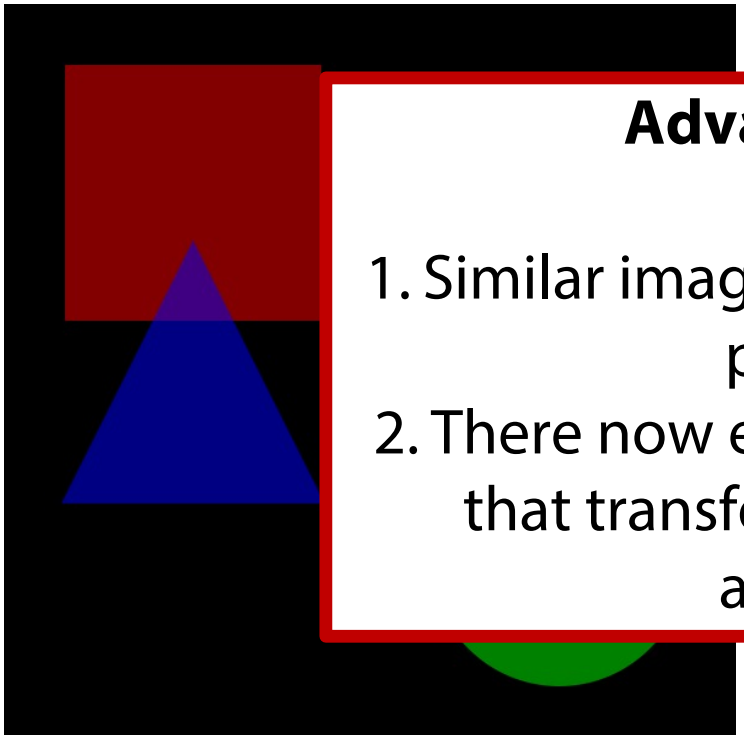
Decoding (parsing) VFA vectors



Two-step method of decoding:

- Find closest pair of anchor points (spaced with $0 << \beta < 1$) stored in associative memory
- Determine exact position on path by optimizing $c(s) = [\mathbf{z}^s]^\top \overline{\mathbf{x}} = \sum_j e^{\imath(s-r)\phi_j}$

Generalizing from Vectors to Symbols



VSA Formulation

$$\odot \mathbf{v}_{top} \odot \mathbf{h}_{left}$$

$$e \odot \mathbf{v}_{middle} \odot \mathbf{h}_{left}$$

$$\odot \mathbf{v}_{bottom} \odot \mathbf{h}_{right}$$

mulation

$$\odot \mathbf{v}^{54.7} \odot \mathbf{h}^{51}$$

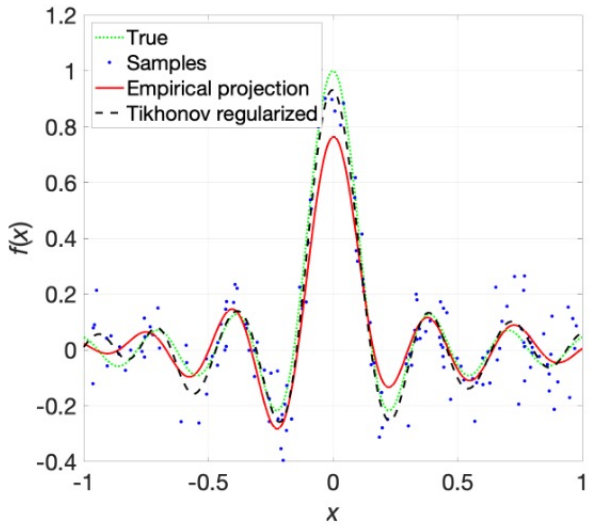
$$\odot \mathbf{v}^{198} \odot \mathbf{h}^{48.24}$$

$$+ \mathbf{c}^{532} \odot \mathbf{s}_{circle} \odot \mathbf{v}^{352.1} \odot \mathbf{h}^{344}$$

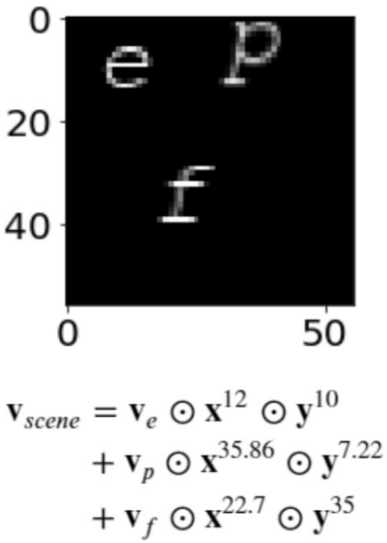
Advantages:

1. Similar images have high inner product
2. There now exists an operation that transform one scene to another!

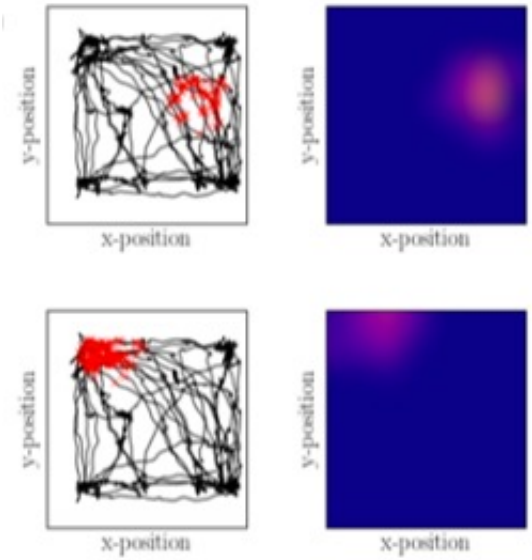
Some applications



Machine learning



Compositional image encodings



Modeling in neuroscience