An Extension to Basis-Hypervectors for Learning from Circular Data in Hyperdimensional Computing

Igor Nunes, Mike Heddes, Tony Givargis and Alexandru Nicolau
Department of Computer Science
UC Irvine, Irvine, USA
{igord, mheddes, givargis, nicolau}@uci.edu

2023 60th ACM/IEEE Design Automation Conference (DAC) | 979-8-3503-2348-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAC56929.2023.10247736

***Abstract*—*Hyperdimensional Computing* (HDC) is a computation framework based on random vector spaces, particularly useful for machine learning in resource-constrained environments. The encoding of information to the hyperspace is the most important stage in HDC. At its heart are *basis-hypervectors*, responsible for representing atomic information. We present a detailed study on basis-hypervectors, leading to broad contributions to HDC: 1) an improvement for level-hypervectors, used to encode real numbers; 2) a method to learn from circular data, an important type of information never before addressed in HDC. Results indicate that these contributions lead to considerably more accurate models for classification and regression.**

***Index Terms*—hyperdimensional computing, basis-hypervectors, circular data, machine learning, information theory**

I. Introduction

For some time now, machine learning has largely dictated not just academic research and industrial applications, but aspects of modern society in general. Such aspects range from the widespread recommendation systems, which influence the way we consume products, news and entertainment, to social networks that impact the way we behave and relate. Given such broad importance, the demand for devices capable of learning has spread to resource constrained platforms such as embedded systems and Internet of Things (IoT) devices [1]. Such scenarios present obstacles to established methods, designed primarily to operate on powerful servers.

The most notable class of such methods is Deep Learning, which has achieved superhuman performance on certain tasks [2]. Although initially inspired by the remarkably efficient animal brain, Deep Learning owes much of its high energy and computational cost to neurally implausible design choices [3]–[5]. The dilemma is that these choices, especially error back-propagation and large depth, are also key drivers of its success [6]. Given this, the search for alternatives has received significant attention from the scientific community [7], [8].

One emerging alternative is *Hyperdimensional Computing* (HDC) [9]. Like Deep Learning, HDC is also inspired by neuroscience, but the central observation is that large circuits are fundamental to the brain's computation. Therefore, information in HDC is represented using *hypervectors*, typically 10,000-bit words where each bit is *independently and identically distributed* (i.i.d). This i.i.d relationship, also known as holographic information representation, provides inherent robustness since each bit carries exactly the same amount of information. Furthermore, the arithmetic in HDC, as detailed in Section II, is generally dimension-independent, which opens up the opportunity for massive parallelism, providing the efficiency sought in HDC.

HDC has already proven to be useful in several applications, including both learning problems, such as classification [10] and regression [11], and classical problems, such as consistent hashing [12]. Regardless of the application, the most fundamental step in HDC is mapping objects in the input space to the hyperspace, a process called *encoding*. Encoding functions have been proposed for several different types of data, such as time series [13], text [14], images [15] and graphs [16]. All these encodings have one thing in common: they start by encoding simple information (e.g. pixel values, vertices and edges, letters, amplitudes of a signal), which are then combined to represent something complex. In this work we study *basis-hypervectors*, also called *seed-hypervectors*, the encoding of these atomic pieces of information.

Basis-hypervectors are a cornerstone of HDC and directly effect the accuracy of learned HDC models as we show in Section VI. We start with a study of the two existing types of basis-hypervectors, *random* and *level-hypervectors*, used respectively to represent uncorrelated and linearly-correlated data. Inspired by information theory, our first contribution is an improved method to create level-hypervectors.

Based on the improved level-hypervectors, our main contribution is a basis-hypervector set for *circular data*. Circular data are derived from the measurement of directions, usually expressed as an angle from a fixed reference direction. In addition, it is common to convert time measurements, such as the hours of a day, to angles. As we will discuss in more detail in Section V, circular data is present in many fields of research, including astronomy, medicine, engineering, biology, geology and meteorology [17]–[19]. Embedded systems and IoT also deal with a wide variety of circular data, such as in robotics where the joints generate angular data [20], and satellites that fly in elliptical orbits [21]. The importance is such that the study and interpretation of this type of data gave rise to a subdiscipline of statistics called *directional statistics* (also known as circular or spherical statistics) [19]. Despite this great relevance, our work is the first to address learning from circular data in HDC.

To assess the practical impact of these contributions, we conducted experiments with publicly available datasets that contain circular data relevant to real-world applications. We compared the circular-hypervector set with the two existing basis sets in regression and classification tasks. Circular-hypervectors, like level-hypervectors, add no additional runtime overhead as they are generated once before training. We obtained an improvement of 7.2% in the classification of 15 types of surgical gestures. In regression, the error is reduced by 67.7% in temperature and power level prediction tasks.

II. Hyperdimensional Computing

Hyperdimensional Computing (HDC) is a computation model that relies on high dimensionality and randomness. Inspired by neuroscience, it seeks to mimic important characteristics of the animal brain while balancing accuracy, efficiency and robustness [9]. The central idea is to represent inputs $x \in \mathcal{X}$ by projecting them onto a hyperspace $\mathcal{H} = \{0,1\}^d$, with $d \approx 10{,}000$ dimensions. This mapping $\phi : \mathcal{X} \to \mathcal{H}$ is called *encoding*, and the resulting representations $\phi(x)$ are named *hypervectors*.

The intuitive principle that guides the design of encoding functions is that similar objects in input space need to be mapped to similar hypervectors in $\mathcal{H}$. We use the normalized Hamming distance as

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.

a measure of distance between hypervectors, which takes the form $\delta : \mathcal{H} \times \mathcal{H} \to \{\sigma \in \mathbb{R} \mid 0 \leq \sigma \leq 1\}$, and define the hypervector *similarity* to be $1 - \delta$. All cognitive tasks in HDC are ultimately based on similarity. Predictions or decisions are inferred from a model which is created by transforming and combining information using HDC arithmetic.

A. Operations

The arithmetic in HDC is based on a simple set of three element-wise operations [9], illustrated in Figure 1.

a) Binding: Operation used to "associate" information. The function $\otimes : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$ multiplies two input hypervectors to produce a third vector dissimilar to both operands. Binding is commutative and distributive over bundling and serves at its own inverse, i.e., $A \otimes (A \otimes B) = B$. The binding operation is efficiently implemented as element-wise XOR.

b) Bundling: Operation is used to aggregate information. The function $\oplus : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$ performs addition on its inputs to produce a hypervector that is similar to its operands. Bundling is implemented as an element-wise majority operation. The output then represents the mean-vector of its inputs.

c) Permuting: The operator $\Pi : \mathcal{H} \to \mathcal{H}$ is often used to differentiate permutations of a sequence. The exact input can be retrieved with the inverse operation. The most used permutation is the cyclic shift, and with $\Pi^i(A)$ we denote a cyclic shift of the elements of A by i coordinates.

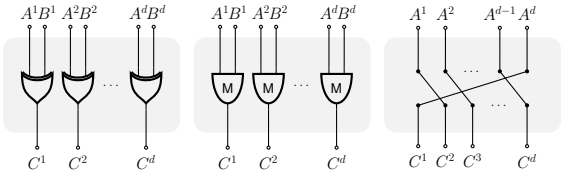

Fig. 1. *Binding*, *bundling*, and *cyclic shift* permutation operations illustrated on binary hypervectors A and B where the superscript denotes the element index of the hypervector. The logical gates in bundling are *majority gates*.

Much of HDC's ability to learn comes from the fact that the very high dimension of the $\mathcal{H}$-space allows combining information with these operations while preserving the information of the operands with high probability, due to the existence of a huge number of quasi-orthogonal vectors in the space [22].

B. Classification

An overview of the HDC classification framework is illustrated in Figure 2. For each class $i \in \{1, \ldots, k\}$ in the training set, we construct a hypervector M_i as follows:

$$M_i = \bigoplus_{j:\ell(x_j)=i} \phi(x_j)$$

where each $x_j \in \mathcal{X}$ is a training sample and $\ell(x_j) \in \{1, \ldots, k\}$ its respective label. The $\bigoplus$ symbol represents the bundling of hypervectors. The resulting M_i is named *class-vector*, and is used as a "prototype" of class i. A trained HDC classification model is therefore denoted by $\mathcal{M} = \{M_i, \ldots, M_k\}$, and consists of a class-vector for each class.

Given an unlabeled input $\hat{x} \in \mathcal{X}$, i.e., a test sample, and a trained model $\mathcal{M}$, we can compare $\phi(\hat{x})$, the *query-vector*, with each class-vector and infer that the label of $\hat{x}$ is the one that corresponds to the most similar class-vector:

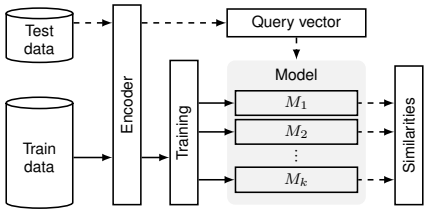

Fig. 2. Overview of the Hyperdimensional Computing classification framework. Solid lines indicate training steps, dashed lines indicate inference steps.

$$\ell^\star(\hat{x}) = \underset{i \in \{1,\ldots,k\}}{\arg\min}\ \delta\left(\phi(\hat{x}), M_i\right)$$

where $\ell^\star(\hat{x}) \in \{1, \ldots, k\}$ is the predicted class for $\hat{x}$.

C. Regression

In a regression setting, an invertible encoding function ϕ_ℓ is used to map labels to hypervectors in a set $\mathbf{L} = \{L_1, \ldots, L_k\}$, whose generation is discussed in Section IV. The model $\mathcal{M}$ consists of a single hypervector, which memorizes training samples $x \in \mathcal{X}$ with their associated label $\ell(x) \in \mathbb{R}$:

$$\mathcal{M} = \bigoplus_i \phi(x_i) \otimes \phi_\ell(\ell(x_i))$$

A prediction can be made given a trained model $\mathcal{M}$ and an unlabeled input $\hat{x} \in \mathcal{X}$. First the approximate label hypervector is obtained by binding the model with the encoded sample $\mathcal{M} \otimes \phi(\hat{x}) \approx \phi_\ell(\ell(\hat{x}))$ [9], [22]. The precise label hypervector is then the most similar label hypervector L_l, where:

$$l = \underset{i \in \{1,\ldots,k\}}{\arg\min}\ \delta\left(\mathcal{M} \otimes \phi(\hat{x}), L_i\right)$$

Finally, the label is obtained by decoding the label hypervector:

$$\ell^\star(\hat{x}) = \phi_\ell^{-1}(L_l)$$

The encoding functions ϕ and ϕ_ℓ, are domain specific and use the HDC operations to encode complex information (e.g., a word) by combining simpler, atomic pieces of information (e.g., the letters of a word). An example is discussed in Section III-A. The first important decision in designing an HDC encoding is how to represent atomic information as hypervectors. These hypervectors represent the smallest pieces of meaningful information and are referred to as *basis-hypervectors*, the central subject in this paper. Our main goal is to show that a special set of basis-hypervectors (described in Section V) is more appropriate for dealing with circular data.

III. Basis-Hypervectors

In this section we describe the two existing basis-hypervector sets used to encode unitary of information. Their main feature is the pairwise distances as illustrated in Figure 3.

A. Encoding symbols

Early applications in HDC focused on sequences of symbols such as text data [14]. The units of information, in this case characters, were mapped (one-to-one) to hypervectors $\mathbf{R} = \{R_1, \ldots, R_m\}$ sampled *uniformly* at random from the hyperspace $\mathcal{H}$, called *random-hypervectors*. From this, a word $w = \{\alpha_1, \ldots, \alpha_n\}$ is typically encoded as:

$$\phi(w) = \bigoplus_{i=1}^{n} \Pi^i\left(\phi_{\mathbf{R}}(\alpha_i)\right)$$

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.

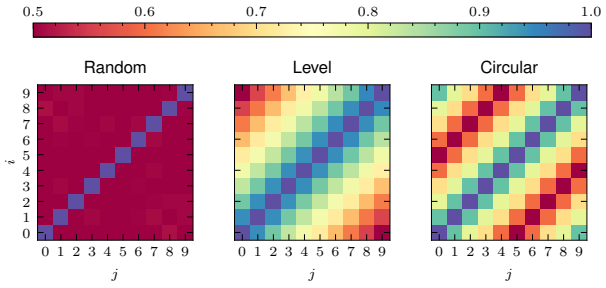

Fig. 3. Pairwise similarity of each i-th and j-th hypervector in different basis-hypervector sets of size 12.

where $\phi_{\mathbf{R}}(\alpha_i) \in \mathbf{R}$ maps the character α_i to its corresponding random-hypervector R_i, and Π is the permutation operator. In general, any sequence or set comprised of symbolic or categorical data can be encoded using random-hypervectors.

Because of the high dimensionality of $\mathcal{H}$, each pair of random-hypervectors is quasi-orthogonal with high probability [8], i.e., they are not correlated. While this seems suitable for encoding symbols, clearly it is not as adequate for other kinds of unitary information, such as real numbers.

B. Encoding real numbers

Many domains use real numbers to represent information such as distance and time. We encode real values to $\mathcal{H}$ with the function $\phi_{\mathbf{L}}$, which maps them to a discrete set of hypervectors $\mathbf{L} = \{L_1, \ldots, L_m\}$. First, we place m points $\{\xi_1, \ldots, \xi_m\}$ evenly over the real interval $[a, b]$. Any real number x is then represented in the hyperspace by $\phi_{\mathbf{L}}(x) = L_l$, where:

$$l = \underset{i \in \{1,\ldots,m\}}{\arg\min} |x - \xi_i|$$

The central question here is how to construct the set of hypervectors $\mathbf{L}$. There is clearly a stronger relation between neighboring points when compared to ξ_1 and ξ_m due to the distance between them. Encoding strategies capable of capturing such relationships lead to better models (see Secs. VI and V). In the next section, we present how hypervectors with this relationship have been created thus far. Then, we propose an improved method that yields more representational power.

IV. Generating level-hypervectors

A method for representing real numbers with linearly correlated hypervectors was first described by Rahimi et al. [13] and Widdows and Cohen [23]. These sets are widely used in HDC and are generally referred to as *level-hypervectors*. The generation of $\mathbf{L} = \{L_1, \ldots, L_m\}$ starts by assigning a uniform random vector to L_1. Each subsequent vector is obtained by flipping $d/2/m$ bits. Flipped bits are never unflipped and, therefore, the vectors L_1 and L_m share *exactly* $d/2$ bits, making them *precisely* orthogonal. In this section, we argue that if the precise distance constraint is relaxed, a set with greater representation power can be created.

A. The importance of quasi-orthogonality

From an information theory perspective, the amount of information conveyed in the outcome of a random trial is a function of the probability of that outcome. More formally, for a given random variable with possible outcomes $\varepsilon_1, \ldots, \varepsilon_n$, which occur with probability $\mathbb{P}(\varepsilon_1), \ldots, \mathbb{P}(\varepsilon_n)$, the *Shannon information content* $\mathscr{I}$ of an outcome ε_i is defined as [24]:

$$\mathscr{I}(\varepsilon_i) = \log_2 \frac{1}{\mathbb{P}(\varepsilon_i)}$$

If we think of a random-hypervector set as a random sample, the probability of each realization is extremely low. Thus the entropy, or information content, is very high. This is one of the main theoretical foundations of HDC.

Note that random-hypervectors are independently and uniformly sampled, resulting in *quasi*-orthogonal vectors. These vectors are simple to create, but more importantly, they have greater representational power than *precisely*-orthogonal vectors. In mathematical terms, while the number of orthogonal vectors in $\mathcal{H}$ is d, the number of quasi-orthogonal vectors is almost 2^d [8]. Given that each set is sampled uniformly, a much larger sample space implies a much lower probability of occurrence per outcome. By the definition above, this results in greater information content.

B. Level-hypervectors revisited: applying the notion of "quasi"

As discussed above, the key to the representational power of random-hypervectors—and more generally of HDC—comes from the relaxed notion of distance: quasi-orthogonality. In contrast, the level-hypervectors created with the existing method have a fixed distance between each pair of hypervectors. This limits the number of possible outcomes of their generation which is equivalent to constraining their representation power. Instead, we want the distance between two hypervectors L_i and L_j in $\mathbf{L}$, with $i < j$, to be proportional to $j - i$ *in expectation*. If we denote by $\Delta^{i,j}$ the desired value for $\mathbb{E}[\delta(L_i, L_j)]$, then:

$$\Delta^{i,j} = \frac{j - i}{2(m - 1)}$$

To achieve this goal, we propose the Algorithm 1 presented below. The method starts by assigning two uniformly random hypervectors to L_1 and L_m, and a d-dimensional vector whose elements are sampled uniformly from [0,1] to Φ. Then, for each remaining level L_l an interpolation threshold value τ_l is set and Φ acts as a filter to determine each bit, copied either from L_1 or from L_m. Proposition 1 establishes that the obtained set of level-hypervectors meets the previously motivated property.

Algorithm 1: Level-hypervectors using interpolation filter

Input : Two positive integers m and d
Output: A set of m d-dimensional level-hypervectors $\mathbf{L} = \{L_1, \ldots, L_m\}$

$L_1 \leftarrow$ uniform_sample($\{0,1\}^d$)
$L_m \leftarrow$ uniform_sample($\{0,1\}^d$)
$\Phi \leftarrow$ uniform_sample($[0,1]^d$)
for *each remaining level* $l \in \{2, \ldots, m-1\}$ **do**
  $\tau_l \leftarrow \frac{m-l}{m-1}$
  for *each dimension* $\partial \in \{1, \ldots, d\}$ **do**
    if $\Phi^{(\partial)} < \tau_l$ **then**
      $L_l^{(\partial)} \leftarrow L_1^{(\partial)}$
    else
      $L_l^{(\partial)} \leftarrow L_m^{(\partial)}$
return $\{L_1, \ldots, L_m\}$

Proposition 1. *Let* $\mathbf{L} = \{L_1, \ldots L_m\}$ *denote a set of hypervectors generated by Algorithm 1. For all* i *and* $j > i$ *in* $\{1, \ldots, m\}$*, we have* $\mathbb{E}[\delta(L_i, L_j)] = \Delta^{i,j}$.

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.

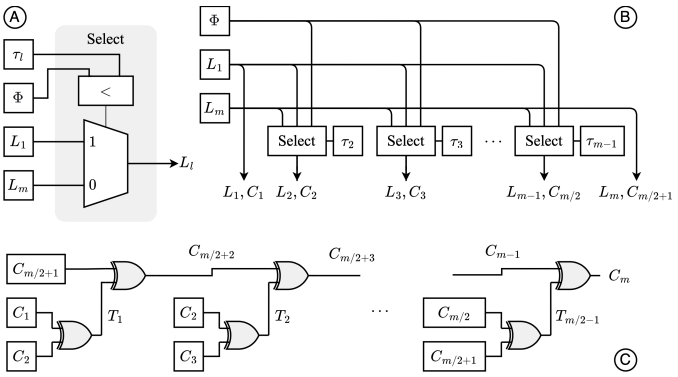

Fig. 4. Illustration of the circuit for creating the proposed basis-hypervector sets: (A) shows the "Select" building block responsible for creating each level-hypervector and (B) shows how they are interconnected to generate the set of size m. These are the same steps as for creating the "forward path" of circular-hypervectors, but then it is necessary to create the "backward path," whose circuit is illustrated in part (C).

Proof. First, from the definition of the normalized Hamming distance, we have:

$$\delta\left(L_i, L_j\right) = \frac{1}{d}\sum_{\partial=1}^{d} \mathbb{1}\left(L_i^{\partial} \neq L_j^{\partial}\right)$$

where $\mathbb{1}$ is the indicator function. By applying the linearity of expectation property, the i.i.d. property for all dimensions of L_i, and considering that the expectation of an indicator function equals the probability of the event, we get:

$$\mathbb{E}\left[\delta(L_i, L_j)\right] = \mathbb{P}\left(L_i^{\bar{\partial}} \neq L_j^{\bar{\partial}}\right) \tag{1}$$

where $\bar{\partial} \in \{1, \ldots, d\}$ indicates that the probability is dimension independent. Then, from Algorithm 1 we have:

$$\mathbb{P}(L_i^{\bar{\partial}} = L_j^{\bar{\partial}}) = \mathbb{P}(\Phi^{\bar{\partial}} < \tau_i)\begin{pmatrix} \mathbb{P}(\Phi^{\bar{\partial}} < \tau_j | \Phi^{\bar{\partial}} < \tau_i)\mathbb{P}(L_1^{\bar{\partial}} = L_1^{\bar{\partial}}) \\ +\mathbb{P}(\Phi^{\bar{\partial}} \geq \tau_j | \Phi^{\bar{\partial}} < \tau_i)\mathbb{P}(L_1^{\bar{\partial}} = L_m^{\bar{\partial}}) \end{pmatrix}$$
$$+\mathbb{P}(\Phi^{\bar{\partial}} \geq \tau_i)\begin{pmatrix} \mathbb{P}(\Phi^{\bar{\partial}} \geq \tau_j | \Phi^{\bar{\partial}} \geq \tau_i)\mathbb{P}(L_1^{\bar{\partial}} = L_1^{\bar{\partial}}) \\ +\mathbb{P}(\Phi^{\bar{\partial}} < \tau_j | \Phi^{\bar{\partial}} \geq \tau_i)\mathbb{P}(L_1^{\bar{\partial}} = L_m^{\bar{\partial}}) \end{pmatrix}$$

Given that $\Phi^{\bar{\partial}}$ is uniform in $[0, 1]$ and $\tau_i = \frac{m-i}{m-1}$ according to the algorithm, we can calculate this probability to be:

$$\mathbb{P}\left(L_i^{\bar{\partial}} = L_j^{\bar{\partial}}\right) = 1 - \frac{j-i}{2(m-1)} \tag{2}$$

Considering that the event is binary, from Equations 1 and 2, we get: $\mathbb{E}\left[\delta\left(L_i, L_j\right)\right] = \Delta^{i,j}$. □

In Figure 4 we illustrate how this algorithm can be implemented using a simple circuit. Part (A) shows that, for each level L_l, the output dimensions are the result of a 2-to-1 multiplexer between L_1 and L_m, whose select line is the result of a comparator between the filter Φ and the threshold τ_l. The filter and threshold value can be represented as random integers instead of floating points for efficiency. We call each of these blocks a *Select*. In part (B), we show how the vector corresponding to each level is obtained as the output of a Select block by changing only one of the inputs which is the corresponding τ_l threshold.

Note that creating basis-hypervectors is an offline process that runs before any training or inference steps, and once created, they can be reused in different applications. We emphasize that their creation, with the proposed method or with the existing one, does not introduce any concrete overhead.

V. Encoding Circular Data

Symbols and real numbers can be represented in the hyperspace with random and level-hypervector sets. However, not every type of data falls into these two categories. Consider, for instance, angular data in $\Theta = [0, 2\pi]$. The distance $\rho \in [0, 1]$ between two angles α and β in Θ is defined as [25]:

$$\rho(\alpha, \beta) = \frac{1}{2}\left(1 - \cos(\alpha - \beta)\right)$$

If we use level-hypervectors to encode the Θ-interval, the distances between the hypervectors would not be proportional to the distance between the angles. Notice that the endpoints of an interval represented with level-hypervectors are completely dissimilar, while a circle has no endpoints.

Angles are widely used to represent information in meteorology [26], ecology [27], medicine [20], [28], astronomy [29] and engineering [30]. Moreover, many natural and social phenomena have circular-linear correlation on some time scale. Consider for example the seasonal temperature variations over a year or the behavior of fish with respect to the tides in a day. In these cases, it makes sense to represent the time intervals (e.g., Jan 1st - Dec 31st) as cyclic intervals [25], [27].

Given the multitude of applications using circular data, unsurprisingly there has been great scientific effort to adapt statistical and learning methodologies to handle it appropriately [18]. This gave rise to a branch of statistical methodology known as *directional statistics* [19]. Despite all this effort, to the best of our knowledge, our work is the first to address the adaptation in the context of HDC learning.

A. Circular-hypervectors

We propose a method for creating a basis-hypervector set, called circular-hypervectors, suitable for learning from circular data in HDC. Our method is based on Heddes et al. [12], which proposes the use of hypervectors for a dynamic hashing system. The algorithm for generating equidistant vectors on the circle is improved and extended for the learning context through our analysis of level-hypervectors and the information content control mechanism presented in section V-B.

We want to build a set of hypervectors $\mathbf{C} = \{C_1, \ldots, C_m\}$[1] such that for all C_i and C_j in $\mathbf{C}$ their distance δ in $\mathcal{H}$ is proportional to the distance between the angles they represent:

$$\mathbb{E}\left[\delta(C_i, C_j)\right] = \frac{1}{2}\,\rho\left((i-1)\frac{2\pi}{m}, (j-1)\frac{2\pi}{m}\right)$$

This relationship is in terms of expected value to improve the information content as discussed in Section IV.

The creation of circular-hypervectors, shown in Figures 5 and 4, is divided into two phases, one for each half of the circle. The first half is simply a set of $m/2 + 1$ level-hypervectors, with C_1 and $C_{m/2+1}$ quasi-orthogonal. The second half is created by applying the transitions between the levels of the first half, in order, from the last hypervector:

$$C_i = C_{i-1} \otimes T_{i-m/2-1}, \text{ for } i \in \{m/2+2, \ldots, m\} \tag{3}$$

where the transition $T_i = C_i \otimes C_{i+1}$ are the flipped bits between levels i and $i + 1$. The circuit for this process is provided in

[1]We assume m to be even to simplify discussions. Sets of odd cardinality can be generated as subsets $\{C_1, C_3, C_5, \ldots, C_{2m-1}\}$ of a set of size $2m$.

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.

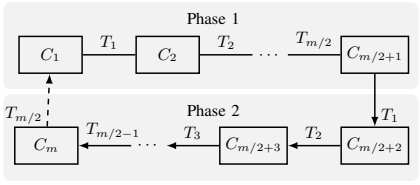

Fig. 5. Diagram of the creation of circular-hypervectors. Phase 1 shows the level-hypervectors and the transformations between them. Phase 2 shows the use of transformations for the second half of the circle.

Figure 4 (C). The combined transitions $\{T_1, \ldots, T_{m/2}\}$ are equal to the transition from C_1 to $C_{m/2+1}$ such that:

$$C_{m/2+1} = C_1 \otimes \bigotimes_{i=1}^{m/2} T_i$$

Since binding is its own inverse, the transition bound to C_{i-1} in Equation 3 makes the new hypervector C_i closer to C_1. Moreover, the transitions $\{T_1, \ldots, T_{m/2}\}$ occur in any half of the circle, ensuring that the hypervector at the opposite side from any hypervector is always quasi-orthogonal to it.

B. Controlling the trade-off between correlation preservation and information content

The discussion in Section IV-A illustrated the benefit of relaxing the distance between generated hypervectors, resulting in greater information content. Another concern is that, while level and circular-hypervectors have the ability to translate important correlations into hyperspace, from the perspective of information content this diminishes their representational power: by forcing the vectors to be correlated, the probability distribution over the possible outcomes becomes more concentrated, decreasing the entropy. We argue that more powerful models can be created if this constraint is relaxed as well.

The random-hypervectors are the most capable at representation as they are sampled uniformly, without any restriction. However, for this very reason they are unable to map existing correlations in the input space to the hyperspace. The level and circular sets preserve correlation but are restricted in information content. The ideal basis set is then expected to have a balance between its ability to preserve correlation and its information content.

To address this trade-off, we introduce a hyperparameter $r \in [0, 1]$ that interpolates between level or circular-hypervectors and the random set. As shown in Figure 6, the parameter changes the similarity of neighboring nodes. Intermediate values of r enable higher information content while preserving local correlation.

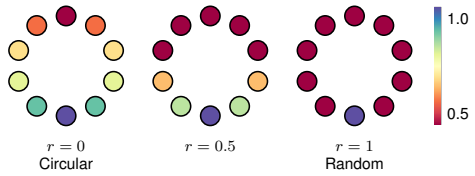

Fig. 6. Effect of the r-hyperparameter on the similarities between each node and the bottom reference node in a circular set of 10 hypervectors.

To interpolate, we concatenate multiple level-hypervector sets created with Algorithm 1. The last hypervector of one set becomes the first hypervector of the next set. The number of transitions n per level-hypervector set is given by $n = r + (1 - r)(m - 1)$, where m is the total number of hypervectors. Levels in the concatenated set are obtained by using the threshold value:

$$\tau_l = 1 - \frac{(l-1) \bmod n}{n}$$

When $r = 1$, each level set contains only two hypervectors, resulting in a set identical to a random-hypervectors. For circular-hypervectors the change only applies to phase 1.

TABLE I
CLASSIFICATION ACCURACY RESULTS

Dataset	Random	Level	Circular
Knot Tying	76.6%	75.9%	**84.0%**
Needle Passing	76.0%	76.0%	**83.6%**
Suturing	73.0%	60.4%	**78.7%**

VI. EXPERIMENTS

We evaluated circular-hypervectors in classification and regression settings where they are compared to random and level-hypervectors. The r-hyperparameter is evaluated by observing its effect on the same two settings. The data was randomly split in 70% for training and 30% for testing in all experiments. The experiments were implemented using Torchhd [31], a high-performance python library for HDC research.

A. Classification

To evaluate the performance of circular-hypervectors in a classification setting we used JIGSAWS [20], a dataset containing kinematic data from surgeons operating the *da Vinci* robot. We consider 18 kinematic variables of the rotation matrices of the tool manipulators and three different surgical tasks: *Knot Tying*, *Needle Passing* and *Suturing*. Each sample has a label which indicates one out of 15 surgical *gestures* and the task is to classify which gesture is being performed. For this, we used the standard classification framework presented in Section II-B. A sample is encoded as $\bigoplus_{i=1}^{18} K_i \otimes V_i$, where the key K_i is a random-hypervector that represents the index i, and the value V_i is encoded as a random, level or circular-hypervector. The circular-hypervectors have $r = 0.1$.

The results comparing each basis-hypervector set in each surgical task are shown in Table I. The circular-hypervectors perform 7.2% better, on average, than random-hypervectors, which in-turn slightly outperform level-hypervectors. The execution times are practically equivalent because the one-time hypervector generation cost is linear in the number of hypervectors for all basis sets. Moreover, the majority of the execution time is spent on training steps which are equivalent once the hypervectors are generated.

B. Regression

We also evaluated circular-hypervectors on two regression tasks. The first contains temperature data measured at the Aotizhongxin station, Beijing, available on the UCI ML Repository [32]. We hypothesize that the circular-hypervectors are better for representing the days and hours because their cyclic nature and high correlation with temperature. Each model was trained on the using the regression framework described in Section II-C. The samples were encoded as $Y \otimes D \otimes H$ with the year Y, day D, and hour H hypervectors. The year is encoded as a level-hypervector. The day and hour hypervectors changed between random, level, and circular.

The second dataset contains power level measurements from the Mars Express satellite, provided by the European Space Agency [33]. A training sample consists of the elapsed fraction of Mars' orbit

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.

TABLE II
REGRESSION MEAN SQUARED ERROR RESULTS

Dataset	Random	Level	Circular
Beijing	441.1	126.8	**21.9**
Mars Express	1294.1	715.6	**339.1**

around the sun, called the mean anomaly. The encoding of the mean anomaly changes between random, level, and circular-hypervectors. The label is the power level at a given time, encoded as a level-hypervector.

The mean squared error for both regression tasks are presented in Table II. Circular-hypervectors reduce the error by 67.7% and 84.4% on average compared to level and random-hypervectors, respectively. These results, combined with those for classification, indicate that circular-hypervectors are indeed more suitable for encoding circular data.

C. r-value

We evaluated separately the effect of the r parameter on the tasks above by varying its value to interpolate between circular and random-hypervectors. We use the normalized mean squared error for the regression tasks and the normalized accuracy error, defined as $\frac{1-\alpha}{1-\bar{\alpha}}$ where α is the accuracy and $\bar{\alpha}$ the reference accuracy, for classification. The reference for all tasks is set to the performance of random-hypervectors.

Figure 7 shows that better performance can be achieved when $r > 0$, as is inline with the theoretical analysis presented in Section IV-A. These results indicate the importance of considering the information content of a basis-hypervector set as we propose. In addition, it shows the value of the proposed r parameter to control the trade-off between representation power and the ability to preserve correlations in the hyperspace.

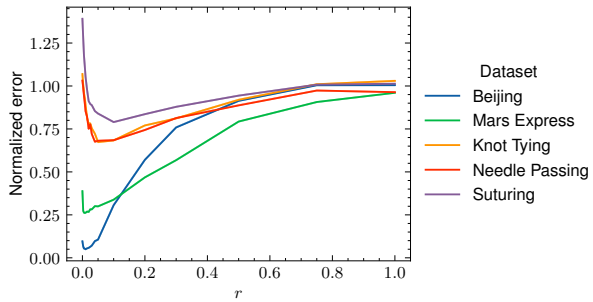

Fig. 7. Error of the circular-hypervectors with varying r-hyperparameter, normalized against the random-hypervectors performance.

VII. CONCLUSION

We study basis-hypervectors: stochastically created vectors used to represent atomic information in HDC. Taking inspiration from information theory, we propose a method for creating level-hypervectors with greater representational power. Furthermore, we introduce a method to handle circular data in HDC. This method, which uses the improved level-hypervectors, is the first approach to learning from circular data in HDC. We believe that these contributions have the potential to benefit HDC in general, as they improve the accuracy of models based on circular and real data, present in most learning applications.

REFERENCES

[1] S. Branco *et al.*, "Machine learning in resource-scarce embedded systems, fpgas, and end-devices: A survey," *Electronics*, vol. 8, no. 11, p. 1289, 2019.
[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
[3] F. Crick *et al.*, "The recent excitement about neural networks," *Nature*, vol. 337, no. 6203, pp. 129–132, 1989.
[4] A. H. Marblestone *et al.*, "Toward an integration of deep learning and neuroscience," *Frontiers in Computational Neuroscience*, p. 94, 2016.
[5] J. C. Whittington and R. Bogacz, "Theories of error back-propagation in the brain," *TiCS*, vol. 23, no. 3, pp. 235–250, 2019.
[6] Y. LeCun *et al.*, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[7] C. D. James *et al.*, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," *BICA*, vol. 19, pp. 49–64, 2017.
[8] P. Kanerva, *Sparse distributed memory*. MIT press, 1988.
[9] ——, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.
[10] L. Ge and K. K. Parhi, "Classification using hyperdimensional computing: A review," *Circuits and Systems Magazine*, vol. 20, no. 2, pp. 30–47, 2020.
[11] M. Hersche *et al.*, "Integrating event-based dynamic vision sensors with sparse hyperdimensional computing: a low-power accelerator with online learning capability," in *ISLPED*, 2020, pp. 169–174.
[12] M. Heddes *et al.*, "Hyperdimensional hashing: A robust and efficient dynamic hash table," in *DAC*, 2022.
[13] A. Rahimi *et al.*, "Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition," in *ICRC*. IEEE, 2016, pp. 1–8.
[14] ——, "A robust and energy-efficient classifier using brain-inspired hyperdimensional computing," in *ISLPED*, 2016, pp. 64–69.
[15] A. X. Manabat *et al.*, "Performance analysis of hyperdimensional computing for character recognition," in *ISMAC*. IEEE, 2019, pp. 1–5.
[16] I. Nunes *et al.*, "Graphhd: Efficient graph classification using hyperdimensional computing," in *DATE*, 2022.
[17] N. I. Fisher, *Statistical analysis of circular data*. Cambridge University Press, 1995.
[18] A. Lee, "Circular data," *WIRE: Computational Statistics*, vol. 2, no. 4, pp. 477–486, 2010.
[19] K. V. Mardia *et al.*, *Directional statistics*. Wiley Online Library, 2000, vol. 2.
[20] Y. Gao *et al.*, "Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling," in *MICCAI workshop: M2cai*, vol. 3, 2014, p. 3.
[21] L. Lucas and R. Boumghar, "Machine learning for spacecraft operations support-the mars express power challenge," in *SMC-IT*. IEEE, 2017, pp. 82–87.
[22] A. Thomas *et al.*, "Theoretical foundations of hyperdimensional computing," *JAIR*, vol. 72, pp. 215–249, 2021.
[23] D. Widdows and T. Cohen, "Reasoning with vectors: A continuous model for fast robust inference," *Logic Journal of the IGPL*, vol. 23, no. 2, pp. 141–173, 2015.
[24] D. J. MacKay *et al.*, *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.
[25] U. Lund, "Least circular distance regression for directional data," *J.Appl.Stat.*, vol. 26, no. 6, pp. 723–733, 1999.
[26] H. Holzmann *et al.*, "Hidden markov models for circular and linear-circular time series," *Environmental and Ecological Statistics*, vol. 13, no. 3, pp. 325–347, 2006.
[27] R. Kempter *et al.*, "Quantifying circular–linear associations: Hippocampal phase precession," *J. of Neuroscience Methods*, vol. 207, no. 1, pp. 113–124, 2012.
[28] F. Gao *et al.*, "On the application of the von mises distribution and angular regression methods to investigate the seasonality of disease onset," *Statistics in Medicine*, vol. 25, no. 9, pp. 1593–1618, 2006.
[29] D. Marinucci and G. Peccati, *Random fields on the sphere: representation, limit theorems and cosmological applications*. Cambridge University Press, 2011, vol. 389.
[30] G. S. Chirikjian, *Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups*. CRC press, 2000.
[31] M. Heddes, I. Nunes *et al.*, "Torchhd: An open-source python library to support hyperdimensional computing research," *arXiv preprint arXiv:2205.09208*, 2022.
[32] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[33] T. Dressler, "Thp 2.0 - thermal power model redesigned," 2019.

Authorized licensed use limited to: Access paid by The UC Irvine Libraries. Downloaded on December 14,2023 at 18:31:21 UTC from IEEE Xplore. Restrictions apply.