ISSN 1870-4069

Exploring Storing Capacity of Hyperdimensional Binary Vectors

Job Isaias Quiroz Mercado, Ricardo Barrón Fernández,
Marco Antonio Ramírez Salinas

Instituto Politécnico Nacional (IPN), Centro de Investigación en Computación,
Mexico City, Mexico
jobquiroz@hotmail.com, barron2131@gmail.com, marco.a.ramirez.s@gmail.com

Abstract. Hyperdimensional computing is an emergent model of computation based on the manipulation of high-dimensional vectors which are used not only to represent variables and values, but also to represent complex structures such as relations, sets and sequences. All vectors in the model are always the same size either if they represent a single concept or a sequence of objects. Hyperdimensional computing uses reduced representations, since there is a compression process to encode complex structures while maintaining the same size on the output vector. In this paper we explore the storing capacity of hyperdimensional vectors that encode semantic feature norms. We describe a method for encoding and retrieving feature information of concrete concepts and present experimental results of the successful retrieval of such features.

Keywords: hyperdimensional computing, vector symbolic architectures, reduced representations, semantic pointer architecture.

1 Introduction

Hyperdimensional computing (HD computing) is based on the properties of high-dimensional vectors and arithmetical operations perform on them. HD computing takes ideas from artificial neural networks in the sense that processing is performed in a distributed fashion; it is also inspired on symbolic computing because complex structures, such as hierarchical trees or sequences, can be formed by manipulating symbols (vectors) that represent simpler objects. Additionally, HD computing also "includes ideas from probability theory, statistics and abstract algebra" [14].

The use of high-dimensional vectors comes implicit with Artificial Neural Networks (ANN), but in HD computing vectors are not only part of the architecture but are the basic computing entities itself. Hyperdimensional vectors can be manipulated to make associations, form hierarchies and perform other types of cognitive computations, formation of these types of complex structures is one of the weak points of ANN [4].

The main property of HD computing is that it can operate with approximate patterns, providing flexibility to the computing system and allowing it to scale to large learning applications. HD computing is becoming more relevant as a cognitive modeling tool.

Semantic features "represent the basic conceptual components of meaning for any lexical item" [5]. Any lexical term is associated with a set of semantic features each of which constitutes one component of a word's intension, semantic features try to establish the meaning of a word in terms of its relationships with other words. Since semantic feature are subjective they must be obtained empirically. In [11] McRae et al, describe a set of semantic feature norms collected from approximately 725 participants for 541 concrete concepts.

In this work, we explore the storing capacity of hyperdimensional binary vectors that encode semantic features norms from McRae's dataset. Our aim is to measure the maximum number of vectors that can be stored, and later retrieved, within a single vector, our hypothesis is that for straightforward encoding methods it is not necessary to have vectors above 5,000 dimensions.

Our experimental results will be used for selecting the appropriate dimensionality of vectors within a HD computing system still in development.

The rest of the paper is organized as follows: Section 2 summarizes several related works to HD computing for concept representation. Section 3 explains the general properties of HD computing and describes how to represent semantic features. In Section 4 we present the experimental results and finally, Section 5 draws the conclusions and future work.

2 Related Work

2.1 Hyperdimensional Computing

In [9] Kanerva introduces the term Hyperdimensional Computing as a model for cognitive computing, in his work he summarizes the main properties of the model, and some current applications for it. But even before that, during the mid-1990s, a class of connectionist network architectures called Vector Symbolic Architectures (VSA) started to be developed. These architectures use arithmetical operations on vectors for encoding structure using distributes representations [7].

The Semantic Pointer Architecture is a VSA developed to model a high-level cognitive system that is also biologically plausible. The main hypothesis for the model is that "Higher-level cognitive functions in biological systems are made possible by semantic pointers." [1]. A semantic pointer is a high dimensional vector that can be used to access large amounts of information within memory, it has the function of a pointer, but unlike conventional pointers, semantic pointers are similar to the information they point to.

The Semantic Pointer Architecture (SPA) has extensively been in [1], and it has been used in "Spaun", a large-scale spiking neuron model capable to integrate perception, cognition and action across several different tasks [2]. SPA provide a unified framework to study cognition, to model working memory and to encode natural language sentences, all with the same representation mechanisms: hyperdimensional vectors.

Crawford et al. implemented a large-scale knowledge base called WordNet using the SPA. WordNet is a manually constructed lexical database of the English language [3],

where each word is associated with multiple meanings forming a complex hierarchical structure of concepts. This work is relevant to Hyperdimensional Computing because it demonstrates how hyperdimensional vectors can be used to encode complex cognitive structures while maintaining flexibility and efficiency.

2.2 Semantic Feature Norms

Semantic feature norms are another way to describe a concept, that unlike WordNet definitions, are empirically obtained by interviewing people: each person is presented with a set of concept names and are asked to list the features he thinks "are important for each concept".

These feature representations have been used to "provide insight into a number of phenomena involving semantic memory and categorization". Additionally, semantic feature norms can be useful to develop models of semantic memory and concept representation. Since semantic feature norms are directly obtained by asking people they tend to be ambiguous, however this is the type of communication that prevails in the real world.

In this work we use binary high dimensional vectors for encoding concepts described by semantic feature norms from the largest dataset in literature, we explore the capacity for storing and later retrieving each of the features composing a concept description.

3 Hyperdimensional Computing Background

One of the most relevant properties of the high-dimensional spaces is that most of the space is nearly orthogonal to any given point. This means that if two random vectors are generated, it is highly probable (more than 99.999%) they are mutually orthogonal, new symbols (vectors) can be stored into memory without clashing with preexisting elements.

These properties where exploited in the SDM model developed by Kanerva in 1988 [10]. However, the properties of high-dimensional spaces can also be used to perform other type computations, for which is necessary to define a set of HD computing operators.

3.1 HD Computing Operations

HD computing is mainly about manipulating and comparing patterns, such patterns are stored in an associative memory where all original vectors are stored and where new generated vectors can be cleaned up and be approximated to one or more of the original vectors.

HD computing has three main operations: addition, multiplication and permutation. In this work we use binary vectors, addition is an element-wise binary average, multiplication is an element-wise exclusive-or and permutation is realized by a logical shift. A more detailed explanation of HD Computing operations can be found in work of Kanerva [13].

All previous operators allow us to encode, map and retrieve hyperdimensional patters, but in most cases the retrieval is not going to be exact. For example, $X = X_1 * A + X_2 * B$ is storing the association of A with X_1 and B with X_2. In order to retrieve A we can multiply X by X_1:

$$X * X_1 = (X_1 * A + X_2 * B) * X_1 \Rightarrow X * X_1 = A + X_1 * X_2 * B. \quad (4)$$

The resulting vector contains the sum of the desire value (A) and an unknown vector $(X_1 * X_2 * B)$, to discriminate this last vector, we can use a clean-up memory that approximates $X * X_1$ to A:

$$Read(A + X_1 * X_2 * B) = Read(A + noise) = A. \quad (5)$$

When a noisy version of an item is given as input, the memory must either output the most similar item, according to a distance metric, in this case hamming distance, or indicate that the input is not close enough to any of the store items.

3.2 Encoding Feature Representations

Feature representation of a concept consists of a list of features that, according to people, are the most important to that concept. A concept feature usually consists of two or more words, usually a verb followed by another concept, depending on these words a feature can be classified within a feature type. In Table 1, we show an example for the feature representation for *knife* found in McRae's dataset.

Table 1. Feature representation for *knife*.

Concept name	Feature	Production frequency	Brain Region Classification
Knife	has a handle	14	Visual-form and surface
	made of steel	8	Visual-form and surface
	is shiny	5	Visual-form and surface
	used for cutting	25	Function
	used for killing	7	Function
	is sharp	29	Tactile
	is dangerous	14	Encyclopedic
	found in kitchens	8	Encyclopedic
	a weapon	11	Taxonomic
	a utensil	9	Taxonomic

The most straightforward way to encode feature representation is by generating two vectors, one for the name of the feature, i.e. 'has', and another for its value, i.e. 'handle', and then using the multiplication operator for binding them together. Once each feature is represented as a pair of binded vectors the next step is to add all the features to produce a single vector that encodes all the features associated with the concept. By applying the proper operations, we can extract the value for each of the features encoded in the vector:

$$knife = has * blade + made.of * steel + texture * shiny + \cdots + is * utensil. \quad (6)$$

In the McRae's dataset all concepts have a different number and type of features, and in most cases, there are two or more features of the same type, either because an object is composed by several parts and therefore the 'has' feature type appears more than once, or because and object has several options for some features, i. e. an apple can be red, green or yellow. These cases do not represent a problem in the encoding process because hyperdimensional multiplication distributes over addition [9] and therefore we can encode each feature independently without knowing if there are more features that share the same type:

$$\begin{aligned} apple &= is * fruit + has * skin + color * (red + green + yellow) \\ &= is * fruit + +has * skin + color * red + color * green + color * yellow. \end{aligned} \tag{7}$$

From equation 7 we can see the semantic pointer $apple$, it is a pointer because is a mean to access more information in memory, we can extract each feature value and, since a feature value can be a semantic pointer itself, we can proceed to explore them in more detail. Semantic Pointers can be used to represent hierarchies, Figure 1, where every level in the hierarchy is a deeper exploration within a semantic pointer.

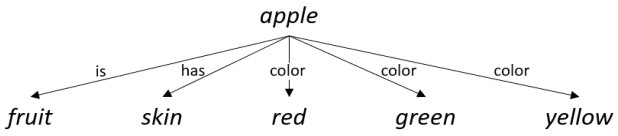

Fig. 1. Semantic Pointer Vectors can represent hierarchical structures.

3.3 Retrieving encoded feature values

Once a concept has been encoded, following equation 6, we have to be able to retrieve the feature values associated with it, to do this the inverse operation has to be applied to the semantic pointer, in the case of binary vectors the XOR operation is its own inverse.

Since in our approach we can have more than one feature value associated to a feature type our clean-up memory outputs, not only the closest vector to the feature value obtained, but all the vectors that are similar enough according to certain threshold value:

$$is * apple \cong fruit\ color * apple \cong (red, green, yellow). \tag{8}$$

To extract all feature values from a semantic pointer it is necessary to know its correspondent feature types, however once a vector is already encoded there is no way to know which feature types are present within this semantic pointer. In order to extract the feature values, we multiply the semantic pointer with all the feature type vectors in the system and, by using the clean-up memory, verify which vectors are meaningful after the multiplication is done. This process is explained in Algorithm 1.

Algorithm 1. Extracting feature value vectors from a semantic pointer.

```
Function GetFeatureValues (SPvector):
  FeatTypes = [is, has, color, shape, …]  -- All feature types
  FeatValues = []
  For each FtType in FeatTypes:
    ValueVec = FtType * SPvector
    ValueVec = CleanUpMemory (ValueVec)
    If ValueVec is not null:
      ValueVec.push(FeatValeus)
  Return ValueVec
```

The function CleanUpMemory returns a list of all the FeatureValue vectors that are close enough to the input vector. Two vectors have a very high probability of being the same when its normalized hamming distance is around 0.45.

4 Experimental Results

We performed several experiments to test the maximum capacity for storing and retrieving feature values from semantic pointer vectors representing definitions of concepts from the McRae's dataset.

During the experiment we randomly select a set of concepts from the dataset and encode them into our associative memory, using binary hyperdimensional vectors. Additionally, we store a list of all the feature values for each selected concept to later compare them with the values extracted from the semantic pointers.

The average number of feature values for the concepts in the McRae dataset is around 17 and the maximum being 22, however, to explore the maximum storing capacity of the semantic vectors we defined additional concepts by randomly selecting features from the dataset, the maximum number of features for a concept was 70.

Table 2 shows the percentage of successful retrievals of feature values for N-size semantic pointers and the number of features in vector ranging from 10 to 70. Figure 2 illustrates these same results.

Based on the obtained results and considering that the maximum amount of features in the McRae dataset is 22, we can state that a binary vector of size N = 2,500 is enough to store the feature representations of concepts from the used dataset. Even though some sources [8, 9] indicate that 10,000 bits can be convenient for most applications, in our case the size of our vectors might be smaller which has a direct impact on the processing time of our system.

We can also observe how the percentage of retrieved features increases as the dimensionality of the vector increases but not in a linear fashion, for example for a vector size of N = 5,000 we can retrieve 86% of all values in a semantic pointer composed by 50 features, if we double the size of the vector, N = 10,000, we can retrieve 82% of the features.

Even though this might seem a small increment it is very relevant to our work because, by the way in which we encode all the features, there is no way to know if the

retrieved features are the most important, in terms of frequency for example, for the concept. There might be ways to ensure that the most important features are retrieved first, but for the characteristics of this dataset it is unnecessary.

Table 2. Storing capacity of binary semantic pointers.

Number of Features in Vector	Successfully retrieved features (%)				
	N = 1,000	N = 2,500	N = 5,000	N = 7,500	N = 10,000
10	100 %	100 %	100 %	100 %	100 %
20	95 %	100 %	100 %	100 %	100 %
30	90 %	100 %	100 %	100 %	100 %
40	72.5 %	92.5 %	95.0 %	100 %	100 %
50	70.0 %	68.0 %	86.0 %	92.0 %	82.0 %
60	58.3 %	46.7 %	61.7 %	56.7 %	61.7 %
70	38.6 %	42.9 %	34.3 %	34.3 %	27.1 %

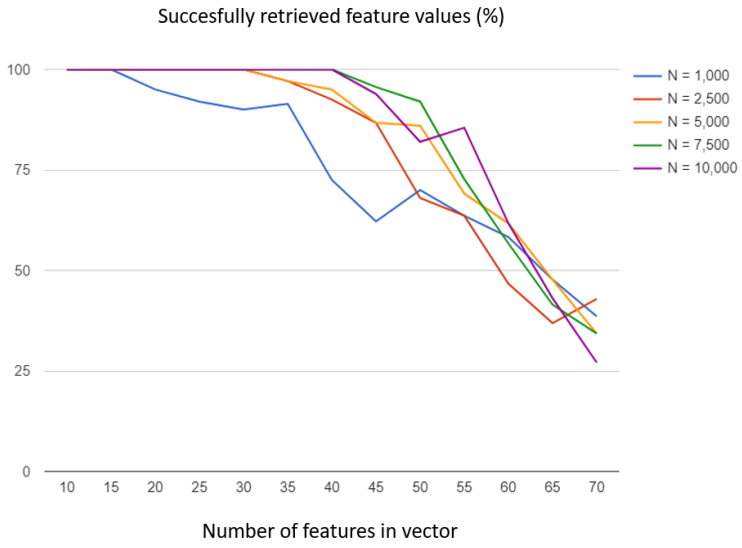

Fig. 2. Storing capacity for N-size semantic pointers.

5 Conclusions and Future Work

This work presents an empirical exploration of the storing capacity of binary semantic pointers as a mean to represent sematic feature norms. We presented some highlights of Hyperdimensional Computing, an emergent model of computation based on the

manipulation of high-dimensional vectors. This type of computation allows us to encode from single concepts up to more complex data structures such as sequences.

We describe how feature representations might be encoded into high-dimensional vectors and we describe a straightforward method to retrieve such vectors. We perform experiments for encoding randomly chosen vectors from the largest known dataset of semantic feature norms.

This works presents preliminary results from a larger project that pretends to encode the entire McRae dataset into a Vector Symbolic Architecture for a Semantic Network, the results presented allow us to decide the size of the vectors to be used in the complete system.

Acknowledgements. This work has been funded by SIP-IPN under grant SIP-20181698 and also by CONACYT scholarship number 666415.

References

1. Eliasmith, C.: How to build a brain: A neural architecture for biological cognition. USA: Oxford University Press (2013)
2. Eliasmith, C., Stewart, T., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., *et al.*: A large-scale model of the functioning brain. Science, 338 (6111), 1202–1205 (2012)
3. Fellbaum, C.: WordNet and wordnets. Encyclopedia of Language and Linguistics. Second Edition, Oxford: Elsevier, pp. 665–670 (2005)
4. Fodor, J. A., Pylyshyn, Z. W.: Connectionism and Cognitive Architecture: A Critical Analysis. Cognition 28(1):3–71 (1988)
5. Fromkin, V., Rodman, R., Hyams, N.: An Introduction to Language. Boston, MA: Wadsworth, Cengage Learning, p. 578 (2014)
6. Gallant, S., Okaywe T.: Representing Objects, Relations and Sequences. Neural Computation 25(8):2038–2078 (2013)
7. Gayler, R.: Vector Symbolic Architectures answer Jackendoff's challenges for Cognitive Neuroscience. In: ICCS/ASCS International Conference on Cognitive Science. CogPrints, Sydney, Australia, University of New South Wales, 133–138 (2003)
8. Kanerva, P.: Computing with 10,000-bit words. In: 52nd Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, pp. 304–3010 (2014)
9. Kanerva, P.: Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High Dimensional Random Vectors. Cognitive Computation 1(2), 139–159 (2009)
10. Kanerva, P.: Sparse Distributed Memory. Cambridge, MA: Bradford/MIT Press (1988)
11. McRae, K., Cree. G., Seidenberg, M., McNorgan, C.: Semantic feature production norms for a large set of living and nonliving things. Behavior Research Methods, Instruments & Computers, 37(4):547–559 (2005)
12. Plate, T.: Holographic reduced representation: distributed representation of cognitive structure. Stanford: CSLI (2003)
13. Quiroz, K., Barrón, R., Ramírez, M.: Sequence Prediction with Hyperdimensional Computing. Research in Computing Science, 138 (2017)
14. Rahimi, A., Datta, S., Kleyko, D., Paxon, E., Olshausen, B., Kanerva, P., Rabaey, J.: High-Dimensional Computing as a Nanoscalable Paradigm. IEEE Transactions on Circuits and Systems: Regular Papers (99), 1–14 (2017)