FACULTY OF BIOSCIENCE ENGINEERING

HYPERDIMENSIONAL COMPUTING FOR PROTEIN LANGUAGE MODELING

...

Michael Fatjanov
Student ID: ...

Supervisor(s): Prof. Dr. Bernard De Baets

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of master in Bioinformatics.
Academic year: 2022-2023

GHENT UNIVERSITY

Gent, FILL IN THE DATE

The promotor,

Prof. Dr. Bernard De Baets

The author,

Michael Fatjanov

ACKNOWLEDGEMENTS

Thank you, all of you!

CONTENTS

SAMENVATTING

nederlandse samenvatting

SUMMARY

insert english summary here...

1. INTRODUCTION

1.1 Digital biology, protein sequence research and traditional bioinformatics tools

1.2 State-of-the-art, deep learning and protein language modeling

1.3 Hyperdimensional computing

Hyperdimensional computing is a relatively new paradigm of computing developed by Kanerva.(1) that tries to mimic the workings of a (human) brain by computing with vectors of tens of thousands elements long, so in the realm of hyperdimensionality. The human brain consists of about 100 billion neurons (nerve cells) and 1000 trillion synapses that connect these neurons. Each neuron is connected to up to 10000 other neurons, creating massive circuits. This is likely fundamental to the workings of the human brain and what separates our brains from modern von Neumann computer architectures which operate on 8 to 64-bit vectors. This becomes clear when we compare the relative simplicity for a human to learn a language compared to computers which require a large and complicated set of arithmetic operations in the form of deep learning networks together with terabytes of data and thousands of Watts of computing power to try to come close mastering a language whilst a human can recognize other languages relatively easy when they don't even speak it. Likewise languages, we can very easily memorize and compare other intrinsically complex and contextual concepts such as images. A computer would have a hard time finding similarities between a set of images and faces because this requires very

complex machine-learning models. The human brain can do this all with a huge efficiency by consuming only roughly 20 W of energy.

To achieve this kind of flexible intelligence, we might have to step away from the restrictive von Neumann architecture and so Kanerva proposes the use of hyperdimensional vectors, a different form of representation for entities by which a computer can compute with. The use of models based on high dimensionality is not entirely new and is being explored since the 1990s. Some of these earlier models include Holographic Reduced Representations (2), Spatter Code (3) etc. An HDV can represent anything from a scalar number to any kind of concept. This vector is initially made up of totally random elements, but with a simple set of operations which will be explained later, we can use other vectors to combine some concepts into new similar or dissimilar concepts. For example, to show the essence of HDC and how it tries to simulate the brain, we can compare the concept of a *table* to the concept of a *brocolli*. We would not immediately conclude that they are in any way similar but as humans, we can trace back *table* to *plate* which has some similarities with *food* from which we can easily extract the concept of *brocolli*. These kinds of operations are not very obvious for a classical computer but creating these semantic pathways are rather easy for humans.

The elements in an HDV can be made up of binary bits like in classical computing but also of bipolar or real numbers. The choice of the nature of the elements has also implications on the nature of the different operations and the results. Highly efficient bit operations could be used on binary vectors but then the amount of information stored in such a vector would be drastically lessened compared to bipolar or real vectors, leading to lower accuracy.

An initial HDV is made up totally random. This *holistic* or *holographic* representation of a concept smeared out over a vector consisting of thousands of bits gives rise to interesting properties such as its robustness. These kinds of systems are very tolerant to noise and failure of bits since we introduce a lot of redundancy in the vector just by stochastics. This is very unlike classical computing where every bit counts and one failure in a bit can lead to disasters.

1.3.1 Operations on hyperdimensional vectors

The interesting properties of HDC are based on only four basic operations we can perform on HDVs. We will discuss these for bipolar and binary vectors.

1.3.1.1 Similarity measurement

For many kinds of problems, it will be necessary to quantify the similarity between two HDVs. This depends on the nature of the vectors. For binary vectors, the *Hamming* distance defined as in equation 1.1 is widely used.

$$Ham(A,B) = \frac{1}{d}\sum_{i=1}^{d} 1_{A_{(i)} \neq B_{(i)}} \tag{1.1}$$

The *cosine* similarity as defined in equation 1.2 is most commonly used for bipolar vectors.

$$cos(A,B) = \frac{A \cdot B}{||A|| * ||B||} \tag{1.2}$$

The results of both of these measurements are summarized in table 1.1. It

Measurement	Dissimilar	Orthogonal	Similar
Hamming distance	1	0.5	0
Cosine similarity	-1	0	1

Table 1.1: Overview of similarity measurements in HDC depending on the nature of the HDVs

is important to note that two random HDVs will be orthogonal to each other just by stochastics. Also notice that the first quantifies a distance and the latter a similarity.

1.3.1.2 Addition

Also referred to as *bundling*, the element-wise addition as in equation 1.3 of n input vectors $\{X_1 + X_2 + \cdots + X_n\}$ creates a vector X that is maximally similar to the input vectors.

$$X = X_1 + X_2 + \cdots + X_n \tag{1.3}$$

For bipolar vectors this is straightforward. The input vectors are added element-wise but the resulting vector is restricted to a bipolar nature too

depending on the sign of each element, thus containing only −1, 1 but allowing 0 for elements that are in disagreement as shown in the following 6-dimensional example.

$$\begin{array}{rcccccc} X_1 = & +1 & -1 & +1 & +1 & -1 & -1 \\ X_2 = & +1 & +1 & +1 & -1 & -1 & -1 \\ X_3 = & -1 & -1 & +1 & +1 & -1 & +1 \\ X_4 = & -1 & -1 & -1 & +1 & -1 & +1 \\ \hline X_1+X_2+X_3+X_4 = & 0 & -1 & +1 & +1 & -1 & 0 \end{array}$$

For binary vectors, the vectors are element-wise bundled based on the majority element. This is no problem if an odd number of input vectors are considered but ambiguity rises when bundling an even set of vectors. This can be solved by setting the element in question randomly. (4) Another possibility is to add another random vector however this may seem to add more unnecessary noise, especially when bundling a low number of vectors. We can also reverse this by an *inverse addition*. For bipolar vectors, this means just multiplying the vector of interest by -1. A binary vector can be flipped bit-wise.

Similar to an ordinary arithmetic summation, the bundling addition of hyperdimensional vectors is commutative so the result is not dependent on the order of addition.

$$X_1 + X_2 = X = X_2 + X_1 \tag{1.4}$$

1.3.1.3 Multiplication

Also referred to as *binding*, we can element-wise multiply two vectors resulting in a vector maximally dissimilar to the input vectors. Vectors X and Y are bound together forming Z being orthogonal to X and Y as shown in equation 1.5.

$$Z = X * Y \tag{1.5}$$

This *binding* operation translates to a simple arithmetic element-wise multiplication for bipolar vectors. For binary vectors, this represents a *XOR*

bit-operation shown as follows.

$$\begin{array}{rcccccc} X = & 1 & 0 & 1 & 1 & 0 & 0 \\ Y = & 1 & 1 & 0 & 1 & 0 & 1 \\ \hline X * Y = & 0 & 1 & 1 & 0 & 0 & 1 \quad 0 \end{array}$$

This operation can also be undone by multiplying with the same vector again. It is its own inverse so that

$$A * A = O \text{ where } O \text{ is a vector containing only 0s} \tag{1.6}$$

Likewise an ordinary multiplication, this operation is commutative and distributive over additions, meaning that transforming a bundle of concepts with binding is equivalent of binding every element before bundling.

$$A = Z * (X + Y) = XZ + YZ \tag{1.7}$$

1.3.1.4 Permutation

The permutation operation of an HDV, also known as *shifting*, is a simple reordering of the HDV. This can be random but a circular shift is widely employed (5) and makes the operation easily reversible. This results in a vector technically dissimilar from the input vector but still encoding its information. This will become important later when it will be used to encode sequential information such as tokens in a text. This operation will be denoted by Π.

$$\begin{array}{rcccccc} X = & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline \Pi(X) = & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

1.3.2 Examples

There are many interesting possibilities given the relative simplicity of all these operations. We shall illustrate some applications and examples. In the following example, the robustness of these hyperdimensional vectors is shown. Assume A, B, C, X, Y, Z to be random 10000-dimensional bipolar

hypervectors and $D = X * A + Y * B + Z * C$. We will try to retrieve A from D.

$$A' = X * D \tag{1.8}$$
$$= X * (X * A + Y * B + Z * C) \tag{1.9}$$
$$= \underbrace{X * X * A}_{A} + \underbrace{X * Y * B + X * Z * C}_{\text{noise}} \tag{1.10}$$
$$\approx A \tag{1.11}$$

This example was implemented in a Julia script, the results are illustrated in figure 1.1.

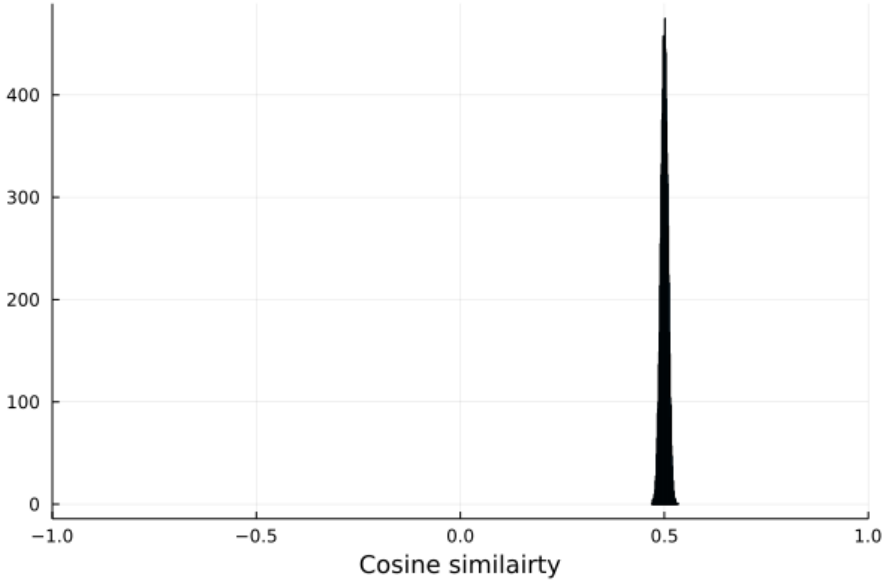

Figure 1.1: 10000 cases of random 10000-dimensional bipolar vectors were made and each time implemented following example 1.8. The resulting cosine similarities between A and A' are then plotted in a histogram.

We see that we can retrieve a lot of information with most of the cosine similarities centering around 0.5. Notice that two completely random HDVs would have a cosine similarity of 0 just by stochastics. Not comparable to state of-the-art accuracies but very efficient nonetheless as all of these calculations were done in less than 2 seconds.

BIBLIOGRAPHY

[1] Pentti Kanerva. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. *Cognitive Computation*, 1(2):139–159, jun 2009.

[2] T.A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

[3] Pentti Kanerva. The spatter code for encoding concepts at many levels. pages 226–229, 1994.

[4] Manuel Schmuck, Luca Benini, and Abbas Rahimi. Hardware optimizations of dense binary hyperdimensional computing: Rematerialization of hypervectors, binarized bundling, and combinational associative memory. *J. Emerg. Technol. Comput. Syst.*, 15(4), oct 2019.

[5] Lulu Ge and Keshab K. Parhi. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 20(2):30–47, 2020.