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 outline

1.2 A historical perspective on bioinformatics

Many decades ago, around the 1950s, we did not know much about the molecules that carry our genetic information and how this would be translated into higher levels of biology. All that was known about deoxyribonucleic acid (DNA) was that it carries nucleotides in equimolar proportions so that there is as much guanine as cytodine and as much adenine as thymine. A major breakthrough came with Watson and Crick's discovery of the structure of DNA in 1953[1]. Despite that, it took some more decades before the genetic code was deciphered and how this information is further transferred. Researchers came to know that DNA is essentially build up of a linear sequence of the 4 aforementioned nucleic acids. This sequence encodes information that undergoes transcription into ribonucleic acid (RNA) that in turn gets translated into proteins. The encoding of proteins is done in groups of three, known as codons. All of this was later stated as the *centra dogma of molecular biology*, also by Crick in 1958[2]. This was hugely important for later research since it gives us more insight on how the genetic code is translated and transferred.

1.2.1 History of bioinformatics-based methods

In the same decade, major leaps were made in the research of protein structure and sequences. The first three-dimensional protein structures were determined via X-ray crystallography [3], which is still mostly the preferred method to this day. On top of that, the arrangement of the primary structure of a protein has been resolved after the first sequencing of a polypeptide. Sanger determined by sequencing insulin in 1953[4] that a protein is build up of a sequence of amino acids, all connected by a peptide bond

into a polypeptide. This established the idea that proteins are biological macromolecules that carried lots of information[5] and so came a boom of research on more efficient methods for obtaining protein sequences. The most popular method of that time was the Edman degradation method. A major issue with this method was that only a theoretical maximum of 50 to 60 sequential amino acids could be sequenced. Larger proteins had to be cleaved into fragments that were small enough to be sequenced. Tracing back the input sequence from this data was a cumbersome process and thus published Dayhoff the first computational program applied to biological data, COMPROTEIN[6] in 1962. This program was essentially a *de novo* sequence assembler for Edman degradation data. Furthermore, sequencing amino acids was also rapidly made automated later in the 1960s. These innovations assisted the creation of the first published protein sequence database[7] in 1965.

Further in the 1960s, researchers discovered the evolutionary value of having protein sequence data of different species. The problem to be solved back then was the quantification of the similarity between sequences. Pairwise alignment algorithms such as the algorithm of Needleman & Wunsch [8] for global alignments and that of Smith & Waterman [9] for local alignments from the 1970s solved this issue and are now considered to be traditional bioinformatics tasks. Together with this, mathematical frameworks for amino acid substitutions in the context of evolution such as PAMs and BLOSUMs also contributed to bioinformatics. These pairwise alignment algorithms are sufficient for comparing two sequences but unfeasible for searching databases for homologous sequences, hence faster algorithms like FASTA [10] and, as of yet widely used for simple searches for similar sequences, BLAST [11] were developed in the 1980s and 1990s. These methods were and are still important for discovering functional, structural and evolutionary information in biological sequences since sequences that are in some way similar, have a high chance to have the same biological function. This also means that such sequences might derive from a common ancestor and it became commonplace that sequence patterns may lead to structural and functional relevance. A natural extension of pairwise alignments is multiple sequence alignment (MSA) [12], which is to align multiple related sequences. The most popular and still widely used tools today include Clustal [13] and MUSCLE [14]. This reveals much more information than pairwise alignment can. It allows for the identification of conserved sequence patterns and critical amino acid residues with much more statistical

significance which is of great value for constructing phylogenetic profiles of gene families. This all showed the importance of computational biology and established bioinformatics as a beneficial field of science.

Development of DNA-based applications took some more time since the genetic code and how it translates to amino acids was not deciphered yet until 1968 [15]. Early DNA and RNA sequencing methods were first demonstrated in the 1970s [16, 17] and like with protein sequencing methods, these methods only became faster, more efficient and more scalable. The 1990s saw the appearance of whole genome sequencing and internet-accessible databases that we still use to this day such as Genbank, Genomes and PubMed and so computational biology had to follow the ever increasing amount of data that it had to process. With the advent of second-generation sequencing in the 2000s came even more Big Data issues since this allowed us to sequence millions of DNA and RNA molecules in a single run, further challenging bioinformaticians and computer scientists.

Bioinformatics has now embraced a more holistic approach with many subdisciplines such as proteomics, transcriptomics and genomics with each having its own challenges but still all very intertwined. From here now on, the focus will be placed on protein research in the realm of bioinformatics.

1.3 Protein research

1.3.1 Protein structures

Proteins are an essential part of molecular biology and are responsible for a wide variety of functions. They are part of the important biological macromolecules that make up life, hence a lot of effort has gone towards trying to understand the functions of protein and disruptions in its mechanisms that lead to many kinds of diseases. To start, proteins are composed of a linear chain of amino acids (AA) with a length ranging from 50 to tens of thousands of AAs, all connected by peptide bonds into a polypeptide. This is also referred to as the *primary structure* of a protein as mentioned earlier [5]. A sequence of amino acids is mostly determined by the genetic code without considering post-translational and post-transcriptional modifications etc. In the genetic code of all living organisms, there are 20 different kinds of amino acids coded in that make up the ’language’ of proteins. This se-

quence of amino acids does not occur as a mere linear chain of peptides, however. A protein consists of much more intricasies. The polypeptide can also form locally folded structures due to chemical interactions within the backbone (the polypeptide chain without the R-group), referred to as the *secondary structure* of a protein. The most well-known and common types of these are α-helices and β-sheets. The overall three-dimensional structure of a protein is referred to as the *tertiary structure*.

The most common way to determine the 3D structure of a protein has remained to be X-ray crystallography for more than half a century [3], with cyro-electron microscopy now catching up rapidly.[18] However, these kinds of laboratory approaches for structure determination of proteins are not simple, expensive and in some cases not possible for the protein in question whilst sequence determination is relatively much easier to perform. For this reason, the number of verified three-dimensional structures has not kept up with the explosive growth in sequence information. On top of that, structure prediction is highly in demand for researchers in applications such as drug design. Therefore, a lot of effort has gone into computational methods for structure and function predictions from protein sequences. While a protein's structure and function are dynamic and dependent on its surroundings such as the cellular state and other proteins and molecules, it is still defined by its underlying sequence. This means that a lot of the 3D-structural and functional information of a protein should be retrievable from its amino acid sequence [19], but our computational modeling abilities continue to be challenged by the complexity of the sequence-structure-function relationship in proteins. A lot of methods have been developed to tackle this problem, until recently, these methods were mainly based on statistical sequence models and physics-based structural simulations.

Physics-based approaches such as Rosetta [20] solves this problem by using atom energy functions and minimizing the total free energy of the system. These kind of methods had much succes, but also assume simplified energy models, are extremely computationally intensive and have a limited accuracy.[21]

and can help in the prediction of secondary and tertiary 3-dimensional structures. To cope with the number of recorded protein sequences rising exponentially, far more compute-wise efficient methods based on multiple sequence alignments had to be developed like PSI-BLAST [22], HHblits [23]

and MMseqs [24]. However, these methods might not be able to keep up with the ever-increasing number of protein sequences stored in databases.

1.3.2 State-of-the-art protein language modeling

It is intuitive to represent a protein as a sequence of letters with each letter corresponding to an amino acid. Likewise to natural languages, we can find common elements between naturally evolved proteins. These motifs and domains are essential to many biological processes and can easily be represented as words, phrases and sentences of amino acids.

1.4 Hyperdimensional computing

Hyperdimensional computing (HDC) is a relatively new paradigm of computing developed by **Kanerva** [25] that tries to mimic the workings of a (human) brain by computing with vectors of tens of thousands of 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. Computers use a large and complicated set of arithmetic operations in the form of deep learning networks which require terabytes of data and thousands of Watts of computing power to come close to mastering a language whilst a human can recognize other languages relatively easily 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 very large efficiency by consuming only roughly 20 W of energy.

Achieving these kinds of flexible brain-like 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 [26],

Spatter Code [27] etc. A hyperdimensional vector (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 fully randomly. 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.

1.4.1 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.

1.4.1.1 Similarity measurement

For many kinds of problems, it will be necessary to quantify the similarity between two HDVs. The method 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 distance* 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.

1.4.1.2 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. [28] 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.

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

1.4.1.3 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 is represented by 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 to binding every element before bundling.

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

1.4.1.4 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 [29] 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.

$X =$	1	0	1	1	0	0
$\Pi(X) =$	0	1	0	1	1	0

1.4.2 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.

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

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

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 close to 0 just by stochastics. This result is not comparable to state-of-the-art accuracies but very efficient nonetheless

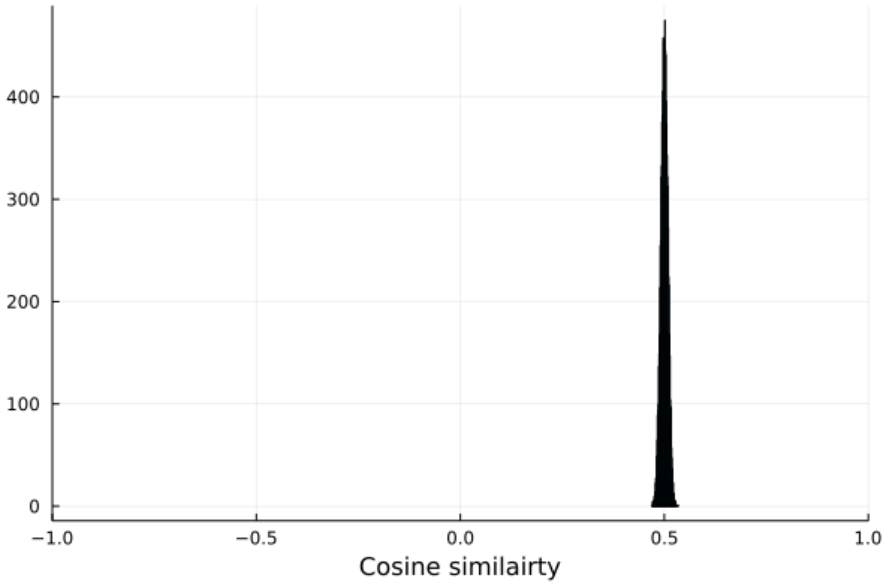

Figure 1.1: 10000 cases of random 10000-dimensional bipolar vectors are made and each time implemented following example 1.8. The resulting cosine similarities between A and A' are then plotted in a histogram.

as all of these calculations were done in less than 2 seconds. This same experiment was done with random binary 10000-dimensional vectors and it finished even faster as expected but retained the accuracy (similar *Hamming distance* peak around 0.25).

BIBLIOGRAPHY

[1] J. D. WATSON and F. H. C. CRICK. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, April 1953.

[2] FRANCIS CRICK. Central dogma of molecular biology. *Nature*, 227(5258):561–563, August 1970.

[3] J. C. Kendrew, G. Bodo, H. M. Dintzis, R. G. Parrish, H. Wyckoff, and D. C. Phillips. A three-dimensional model of the myoglobin molecule obtained by x-ray analysis. *Nature*, 181(4610):662–666, March 1958.

[4] F. Sanger and E. O. P. Thompson. The amino-acid sequence in the glycyl chain of insulin. 1. the identification of lower peptides from partial hydrolysates. *Biochemical Journal*, 53(3):353–366, February 1953.

[5] JOSEPH S. FRUTON. Early theories of protein structure. *Annals of the New York Academy of Sciences*, 325(1):1–20, 1979.

[6] J Wesley Leas. *Proceedings of the December 4-6, 1962, Fall Joint Computer Conference*. ACM, 1962.

[7] Robert T. Hersh, Richard V. Eck, and Margaret O. Dayhoff. Atlas of protein sequence and structure, 1966. *Systematic Zoology*, 16(3):262, September 1967.

[8] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[9] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[10] David J. Lipman and William R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[11] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[12] Michio Murata, Jane S Richardson, and Joel L Sussman. Simultaneous comparison of three protein sequences. *Proceedings of the National Academy of Sciences*, 82(10):3073–3077, 1985.

[13] Desmond G Higgins and Paul M Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988.

[14] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[15] Francis HC Crick. The origin of the genetic code. *Journal of molecular biology*, 38(3):367–379, 1968.

[16] F Sanger, G M Air, B G Barrell, N L Brown, A R Coulson, C A Fiddes, C A Hutchison, P M Slocombe, and M Smith. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 265(5596):687–695, February 1977.

[17] W. MIN JOU, G. HAEGEMAN, M. YSEBAERT, and W. FIERS. Nucleotide sequence of the gene coding for the bacteriophage MS2 coat protein. *Nature*, 237(5350):82–88, May 1972.

[18] Ka Man Yip, Niels Fischer, Elham Paknia, Ashwin Chari, and Holger Stark. Atomic-resolution protein structure determination by cryo-EM. *Nature*, 587(7832):157–161, October 2020.

[19] O.B. Ptitsyn. How does protein synthesis give rise to the 3d-structure? *FEBS Letters*, 285(2):176–181, 1991.

[20] Andrew Leaver-Fay, Michael Tyka, Steven M. Lewis, Oliver F. Lange, James Thompson, Ron Jacak, Kristian W. Kaufman, P. Douglas Renfrew, Colin A. Smith, Will Sheffler, Ian W. Davis, Seth Cooper, Adrien Treuille, Daniel J. Mandell, Florian Richter, Yih-En Andrew Ban, Sarel J. Fleishman, Jacob E. Corn, David E. Kim, Sergey Lyskov, Monica Berrondo, Stuart Mentzer, Zoran Popović, James J. Havranek, John Karanicolas, Rhiju Das, Jens Meiler, Tanja Kortemme, Jeffrey J. Gray, Brian Kuhlman, David Baker, and Philip Bradley. Rosetta3. In *Computer Methods, Part C*, pages 545–574. Elsevier, 2011.

[21] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell Systems*, 12(6):654–669.e3, June 2021.

[22] S. Altschul. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[23] Martin Steinegger, Markus Meier, Milot Mirdita, Harald Vöhringer, Stephan J. Haunsberger, and Johannes Söding. HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, 20(1), September 2019.

[24] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*, 35(11):1026–1028, October 2017.

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

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

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

[28] 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.

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