Title: Hyperdimensional computing

URL: https://en.wikipedia.org/wiki/Hyperdimensional_computing

PageID: 73654860

Categories: Category:Artificial neural networks, Category:Deep learning

-----

Hyperdimensional computing (HDC) is an approach to computation, particularly Artificial General Intelligence . HDC is motivated by the observation that the cerebellum operates on high-dimensional data representations. [1] In HDC, information is thereby represented as a hyperdimensional (long) vector called a hypervector. A hyperdimensional vector (hypervector) could include thousands of numbers that represent a point in a space of thousands of dimensions, [2] as vector symbolic architectures is an older name for the same approach. Research extenuates for creating Artificial General Intelligence .

Process

Data is mapped from the input space to sparse HD space under an encoding function $\phi : X \to H$. HD representations are stored in data structures that are subject to corruption by noise/hardware failures. Noisy/corrupted HD representations can still serve as input for learning, classification, etc. They can also be decoded to recover the input data. H is typically restricted to range-limited integers (-v-v) [3]

This is analogous to the learning process conducted by fruit flies olfactory system. The input is a roughly 50-dimensional vector corresponding to odor receptor neuron types. The HD representation uses ~2,000-dimensions. [3]

Transparency

HDC algebra reveals the logic of how and why systems makes decisions, unlike artificial neural networks . Physical world objects can be mapped to hypervectors, to be processed by the algebra. [2]

Performance

HDC is suitable for "in-memory computing systems", which compute and hold data on a single chip, avoiding data transfer delays. Analog devices operate at low voltages. They are energy-efficient, but prone to error-generating noise. HDC's can tolerate such errors. [2]

Various teams have developed low-power HDC hardware accelerators. [3]

Nanoscale memristive devices can be exploited to perform computation. An in-memory hyperdimensional computing system can implement operations on two memristive crossbar engines together with peripheral digital CMOS circuits. Experiments using 760,000 phase-change memory devices performing analog in-memory computing achieved accuracy comparable to software implementations. [4]

Errors

HDC is robust to errors such as an individual bit error (a 0 flips to 1 or vice versa) missed by error-correcting mechanisms. Eliminating such error-correcting mechanisms can save up to 25% of compute cost. This is possible because such errors leave the result "close" to the correct vector. Reasoning using vectors is not compromised. HDC is at least 10x more error tolerant than traditional artificial neural networks , which are already orders of magnitude more tolerant than traditional computing. [2]

Example

A simple example considers images containing black circles and white squares. Hypervectors can represent SHAPE and COLOR variables and hold the corresponding values: CIRCLE, SQUARE, BLACK and WHITE. Bound hypervectors can hold the pairs BLACK and CIRCLE, etc. [2]

Orthogonality

High-dimensional space allows many mutually orthogonal vectors. However, If vectors are instead allowed to be nearly orthogonal , the number of distinct vectors in high-dimensional space is vastly larger. [2]

HDC uses the concept of distributed representations, in which an object/observation is represented by a pattern of values across many dimensions rather than a single constant. [3]

Operations

HDC can combine hypervectors into new hypervectors using well-defined vector space operations.

Groups , rings , and fields over hypervectors become the underlying computing structures with addition, multiplication, permutation, mapping, and inverse as primitive computing operations. [4] All computational tasks are performed in high-dimensional space using simple operations like element-wise additions and dot products . [3]

Binding creates ordered point tuples and is also a function $\otimes : H \times H \rightarrow H$. The input is two points in H , while the output is a dissimilar point. Multiplying the SHAPE vector with CIRCLE binds the two, representing the idea “SHAPE is CIRCLE”. This vector is "nearly orthogonal" to SHAPE and CIRCLE. The components are recoverable from the vector (e.g., answer the question "is the shape a circle?"). [3]

Addition creates a vector that combines concepts. For example, adding “SHAPE is CIRCLE” to “COLOR is RED,” creates a vector that represents a red circle.

Permutation rearranges the vector elements. For example, permuting a three-dimensional vector with values labeled x , y and z , can interchange x to y , y to z , and z to x . Events represented by hypervectors A and B can be added, forming one vector, but that would sacrifice the event sequence. Combining addition with permutation preserves the order; the event sequence can be retrieved by reversing the operations.

Bundling combines a set of elements in H as function $\oplus : H \times H \rightarrow H$. The input is two points in H and the output is a third point that is similar to both. [3]

History

Vector symbolic architectures (VSA) provided a systematic approach to high-dimensional symbol representations to support operations such as establishing relationships. Early examples include holographic reduced representations, binary spatter codes, and matrix binding of additive terms. HD computing advanced these models, particularly emphasizing hardware efficiency. [3]

In 2018, Eric Weiss showed how to fully represent an image as a hypervector. A vector could contain information about all the objects in the image, including properties such as color, position, and size. [2]

In 2023, Abbas Rahimi et al., used HDC with neural networks to solve Raven's progressive matrices . [2]

In 2023, Mike Heddes et Al. under the supervision of Professors Givargis, Nicolau and Veidenbaum created a hyper-dimensional computing library [5] that is built on top of PyTorch .

Applications

Image recognition

HDC algorithms can replicate tasks long completed by deep neural networks , such as classifying images. [2]

Classifying an annotated set of handwritten digits uses an algorithm to analyze the features of each image, yielding a hypervector per image. The algorithm then adds the hypervectors for all labeled images of e.g., zero, to create a prototypical hypervector for the concept of zero and repeats this for the other digits. [2]

Classifying an unlabeled image involves creating a hypervector for it and comparing it to the reference hypervectors. This comparison identifies the digit that the new image most resembles. [2]

Given labeled example set S = { (x i , y i) } i = 1 N , where x i ∈ X and y i ∈ { c i } i = 1 K {\displaystyle S=\{(x_{i},y_{i})\}_{i=1}^{N},\ {\scriptstyle {\text{where}}}\ x_{i}\in X\ {\scriptstyle {\text{and}}}\ y_{i}\in \{c_{i}\}_{i=1}^{K}} is the class of a particular x i . [3]

Given query x q ∈ X the most similar prototype can be found with k ∗ = k ∈ 1 , . . . , K a r g m a x p (φ (x q)) , φ (c k)) {\displaystyle k^{*}=_{k\in 1,...,K}^{argmax}\ p(\phi (x_{q})),\phi (c_{k}))} . The similarity metric ρ is typically the dot-product. [3]

Reasoning

Hypervectors can also be used for reasoning. Raven's progressive matrices presents images of objects in a grid. One position in the grid is blank. The test is to choose from candidate images the one that best fits. [2]

A dictionary of hypervectors represents individual objects. Each hypervector represents an object concept with its attributes. For each test image a neural network generates a binary hypervector (values are +1 or −1) that is as close as possible to some set of dictionary hypervectors. The generated hypervector thus describes all the objects and their attributes in the image. [2]

Another algorithm creates probability distributions for the number of objects in each image and their characteristics. These probability distributions describe the likely characteristics of both the context and candidate images. They too are transformed into hypervectors, then algebra predicts the most likely candidate image to fill the slot. [2]

This approach achieved 88% accuracy on one problem set, beating neural network–only solutions that were 61% accurate. For 3-by-3 grids, the system was 250x faster than a method that used symbolic logic to reason, because of the size of the associated rulebook. [2]

Other

Other applications include bio-signal processing, natural language processing, and robotics. [3]

See also

Support vector machine

References

Kleyko, Denis; Rachkovskij, Dmitri A.; Osipov, Evgeny; Rahimi, Abbas (2023-07-31). "A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part I: Models and Data Transformations" . ACM Computing Surveys . 55 (6): 1– 40. arXiv : 2111.06077 . doi : 10.1145/3538531 . ISSN 0360-0300 .

Kleyko, Denis; Rachkovskij, Dmitri; Osipov, Evgeny; Rahimi, Abbas (2023-09-30). "A Survey on Hyperdimensional Computing aka Vector Symbolic Architectures, Part II: Applications, Cognitive Models, and Challenges" . ACM Computing Surveys . 55 (9): 1– 52. arXiv : 2112.15424 . doi : 10.1145/3558000 . ISSN 0360-0300 .

External links

Stock, M., Van Criekinge, W., Boeckaerts, D., Taelman, S., Van Haeverbeke, M., Dewulf, P., De Baets, B. (2024), Dutt, V. (ed.), "Hyperdimensional computing: a fast, robust, and interpretable paradigm for biological data", PLOS Computational Biology , 20 (9), Public Library of Science (PLOS): e1012426, arXiv : 2402.17572 , doi : 10.1371/journal.pcbi.1012426 , PMID 39316621 {{ citation }} : CS1 maint: article number as page number (link)

Cumbo, F., Chicco, D. (2025), "Hyperdimensional computing in biomedical sciences: a brief review", PeerJ Computer Science , 11 (e2885) e2885, doi : 10.7717/peerj-cs.2885 , PMC 12192801 , PMID 40567746

Kanerva, Pentti (2009-06-01). "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors" . Cognitive Computation . 1

(2): 139– 159. doi : 10.1007/s12559-009-9009-8 . ISSN 1866-9964 . S2CID 733980 .

Neubert, Peer; Schubert, Stefan; Protzel, Peter (2019-12-01). "An Introduction to Hyperdimensional Computing for Robotics" . KI – Künstliche Intelligenz . 33 (4): 319– 330. doi : 10.1007/s13218-019-00623-z . ISSN 1610-1987 . S2CID 202642163 .

Neubert, Peer; Schubert, Stefan (2021-01-19). "Hyperdimensional computing as a framework for systematic aggregation of image descriptors". arXiv : 2101.07720v1 [cs.CV].

Stock, Michiel (2022-10-04). "Tutorial on Hyperdimensional Computing" . Retrieved 2023-07-29 .

"HD/VSA" . www.hd-computing.com . 2023-03-13 . Retrieved 2023-04-15 .

Ananthaswamy, Anil (2023-04-13). "A New Approach to Computation Reimagines Artificial Intelligence" . Quanta Magazine . Retrieved 2023-06-13 .