The Spatter Code: Holographic Reduced Representation in the Binary Domain

Pentti Kanerva
RWCP Neuro SICS Laboratory, Real World Computing Partnership
Swedish Institute of Computer Science, Box 1263, SE-16429 Kista, Sweden
kanerva@sics.se

Holographic Reduced Representation

Distributed representation has been studied in the cognitive sciences as a brainlike alternative to the more conventional representations used in numeric and symbolic computing. Research on representation has been reviewed by Plate (1994) in introduction to his Holographic Reduced Representation (HRR), which is a way to encode compositional (recursive) structure with high-dimensional, random vectors of fixed width. Typically their dimensionality $N > 1{,}000$.

Structured information is usually encoded with records composed of fields. However, HRR vectors have no fields. Instead, the information of each "field" is distributed over the entire vector—hence "holographic." The main idea is to encode it so that the *similarity of meaning* is expressed as a correlation between HRR vectors. This makes HRRs attractive for modeling mental functions in artificial neural nets.

Encoding an HRR vector resembles data encryption, and decoding it resembles deciphering with a key. Let us encode 'name = Pat & sex = male & age = 66' as an example. Each of the components is itself represented by an N-dimensional random vector (N-vector), referred to by the boldface words **name**, **Pat**, **sex**, **male**, **age**, and **66**. HRRs are encoded in two steps called *binding* and *chunking*. Binding the variable 'name' to the value 'Pat' yields the N-vector **name*Pat** and, similarly, we get the N-vectors **sex*male** and **age*66** for the other two. The binding operation * depends on the kind of HRR, but it always combines two N-vectors into a single N-vector. Next, the N-vectors for the three bound pairs are chunked into a single N-vector, which is written as

$$\langle \mathbf{name*Pat} + \mathbf{sex*male} + \mathbf{age*66} \rangle$$

reminding us that it is a normalized sum vector (a mean vector) of some kind. We will refer to it as **PM6**.

HRRs are decoded in two steps called *probing* and *clean-up*. Probing **PM6** with **name**, written as **name#PM6**, produces an N-vector **Pat′** that is similar to—it correlates highly with—**Pat**, i.e.,

$$\begin{aligned}\mathbf{name\#PM6} &= \mathbf{name\#}\langle \mathbf{name*Pat} + \mathbf{sex*male} + \mathbf{age*66} \rangle \\ &= \mathbf{Pat'} \approx \mathbf{Pat}\end{aligned}$$

The "high" correlation is a matter of degree: The correlation of **Pat′** with **Pat** is significantly higher than with an unrelated HRR vector, which allows **Pat** to be identified among all HRR vectors that a system has encountered. Finding the "nearest neighbor" of the approximate result of probing—of **Pat′**—is called clean-up and it could be done with an auto-associative neural memory, for example.

Plate describes two kinds of HRR, one with *real* vectors and the other with *complex* vectors; he refers to the latter as "HRRs in the frequency domain." The N components of a real HRR vector are distributed normally with mean = 0 and variance = $1/N$, and of a complex HRR vector they are distributed uniformly around the unit circle of the complex plane. The binding (*) and probing (#) operators for real HRRs are circular convolution and "correlation," respectively; for complex HRRs they are coordinatewise complex multiplication and division. Chunking is by normalized vector sum, and clean-up is by some nearest-neighbor method, as has already been mentioned.

The Spatter Code

The Spatter Code (Kanerva, 1997; see its references for the development of the idea) works with random N-bit words. It grew out of the need to encode concepts at arbitrarily many levels in fixed-width words, and the initial concern was how to maintain the density of 1s over successive levels of composition. The spatter code realizes HRR with binary vectors and represents holographic reduced representation perhaps at its simplest.

Binary HRR vectors have N random bits that are mutually independent, and 0s and 1s are equally probable (the codewords are dense). Bitwise Exclusive-Or (XOR, ⊗) is used for both binding (*) and probing (#), and chunking is done with bitwise thresholded sum that realizes the majority rule. The result of probing can be cleaned up, at least in principle, with a neural associative memory.

The main virtue of the binary representation is simplicity. Many of its properties can be derived from the binomial distribution, and binary systems are the easiest of all to build or simulate. This helps make the idea of HRR accessible.

The binary representation has its peculiarities. There is no obvious way to realize the majority rule when the number of binary vectors that are chunked together is even, nor to give individual weights to the vectors when they are chunked together (weighting is not discussed in this paper). Binding and probing with the same operator, the XOR, that also is commutative can produce unwanted effects. It is possible to elaborate the binary representation and to remedy at least some of its apparent shortcomings.

References

Kanerva, P. (1997). Fully distributed representation. *Proceedings of 1997 Real World Computing Symposium, RWC '97* (pp. 358–365). Tsukuba-city, Japan: Real World Computing Partnership.

Plate, T.A. (1994). *Distributed representation and nested compositional structure.* Doctoral dissertation, Graduate Department of Computer Science, University of Toronto.