Superpositions with Sparse Distributed Memories, High Dimensional and Dense Vectors

JW

Abstract

Vector representations are standard for representing data. Aside of the intrisic quality of the representations, various questions arise as they are used in interplay with other data, algorithms and systems.

For three frameworks, we focus here on the superposition operation and present a technical review of the observed behaviours.

1 Introduction

For modelling and representing data, various attempts have been made and vector models have been considered for instance with vector symbolic architectures or to model data learned from algorithms like neural networks.

One standard way to represent data and even language has became the standard, dense vectors, a number of floats within a range.

Alternative representations are based on the ideas of Hyperdimensional or high dimensional (HD) computing, which are binary vectors of large size. As an interesting variation, we consider here also a sparse version of these: sparse distributed memories (SDM). The above framework are introduced in [1] and [2] respectively.

1.1 Operation with vector representations

We focus here mainly on simple, standard operations on vectors:

- similarity: an operation measuring similarity between two vectors $sim(\vec{v}_1, \vec{v}_2)$, e.g. cosine similarity for dense vectors.
- superposition: given a set of vectors $S = (\vec{v}_1, v_2, \ldots)$, the operation of adding an additional vector to the set above.

- proning: the possibility to identify the presence of a sub constituent using similarities. This is achieved by setting thresholds or selecting the most similar elements.

Notice that HD and SDM computing allow for a vector symbolic architecture framework, with the design of sets, records and sequences (Kanerva cognitive code [1]) with properties that are naturally inherited by SDM's.

2 Experiments with set and random vectors

Below, for the three vector formalisms, we produce visualisations of sets robustness under superposition. The experiment consists in building sets by the superposition of 1 to N random elements.

As a baseline, we start to evaluate the averaged similarity for random vectors. Then we compare the superposition with N elements to the one with N+1. Finally, we compare the initial random vector with the various superpositions.

2.1 Similarities and Superpositions

The code is available online [1].

2.1.1 Dense Vectors

$superpose_{Dense}(\vec{v}_1, \vec{v}_2, \ldots) \to \|\vec{v}_1 + \vec{v}_2\|$, whereas $\| \ldots \|$ is a normalisation to vector of length 1.
Similarity is the standard cosine similarity.

2.1.2 HD Computing

$superpose_{HD\ C.}(\vec{v}_1, \vec{v}_2, \ldots) = [f_{HD}(\vec{v}_1[1], \vec{v}_2[1], \ldots), \ldots, f_{HD}(\vec{v}_1[N], \vec{v}_2[N], \ldots)]$, with $f_{HD}(x_1, x_2, \ldots, x_n) = round_{HD}(\sum_i^n x_i/n)$, $round_{HD}(x)$ is randomized toward $\{0, 1\}$ when $x = 0.5$.
Similarity is based on the XOR operation between bits at the same position, which gives the Hamming distance after normalisation $d_H(A, B) = (\vec{A}\ XOR\ \vec{B})/length(\vec{A})$. We set $similarity(A, B) = 1 - 2 * d_H(A, B)$.

[1] `github`

2.1.3 SDM

$superpose_{SDM}(\vec{v}_1, \vec{v}_2, \ldots) = Set(Indices(\vec{v}_1) + Indices(\vec{v}_2) + \ldots)$, indices that allows to characterise SDM. Similarity is as for HD computing, however, in practice, one uses sparse frameworks.

2.2 Similarity and Comparison Measures for Search

In Fig.1, we are comparing the various vectors behaviour under superposition. On one hand, we are comparing similarities for:

- vectors of same sizes (superpositions of N dissimilar sub-elements)
- superposition 1 VS superpose N (similar sub-elements)
- superposition N VS superpose N+1 (similar sub-elements)

We also want to evaluate, given a similarity 1 VS N or N VS N+1, the chance to retrieve the correct superposition. Obviously, as the superposition is built on random vectors, there is a probability of incidental overlap that will diminish with the vector's size.

Even though the chance incidental error can be exactly computed for SDM's, we approach it via our simple experiment with random vectors. Here, as a first approximation, we compute:

$$\begin{aligned} \Delta E(N,\, N+1) &= \frac{(s_{N,\,N+1} - s_{N+1,\,N+2}) + \Delta E_N + \Delta E_{N+1}}{\Delta E_N + \Delta E_{N+1}}, \\ \Delta E(N,\, 1) &= \frac{(s_{N,\,1} - s_{N+1,\,1}) + \Delta E_N + \Delta E_1}{\Delta E_N + \Delta E_1}, \end{aligned}$$

with $s_{i,\,j} \equiv similarity(superposition_j, superposition_j)$ and $stdd_{i,\,j}$ for the standard deviation for similarities $s_{N,\,N+1}$ for all the compositions considered. This approximates or evaluates the relative distance to incidental superposition in standard deviations, thus giving a measure of how likely it is to retrieve the correct superposition for search.

3 Memory Requirements

We here simply review the models memory scaling behaviours:

- HD vectors: binary vectors of length L would need $L \times 2$. However, in practice one use sparse framework with the indices that are Int16, Int32 or Int64. So one needs $L \times 64$ bits.

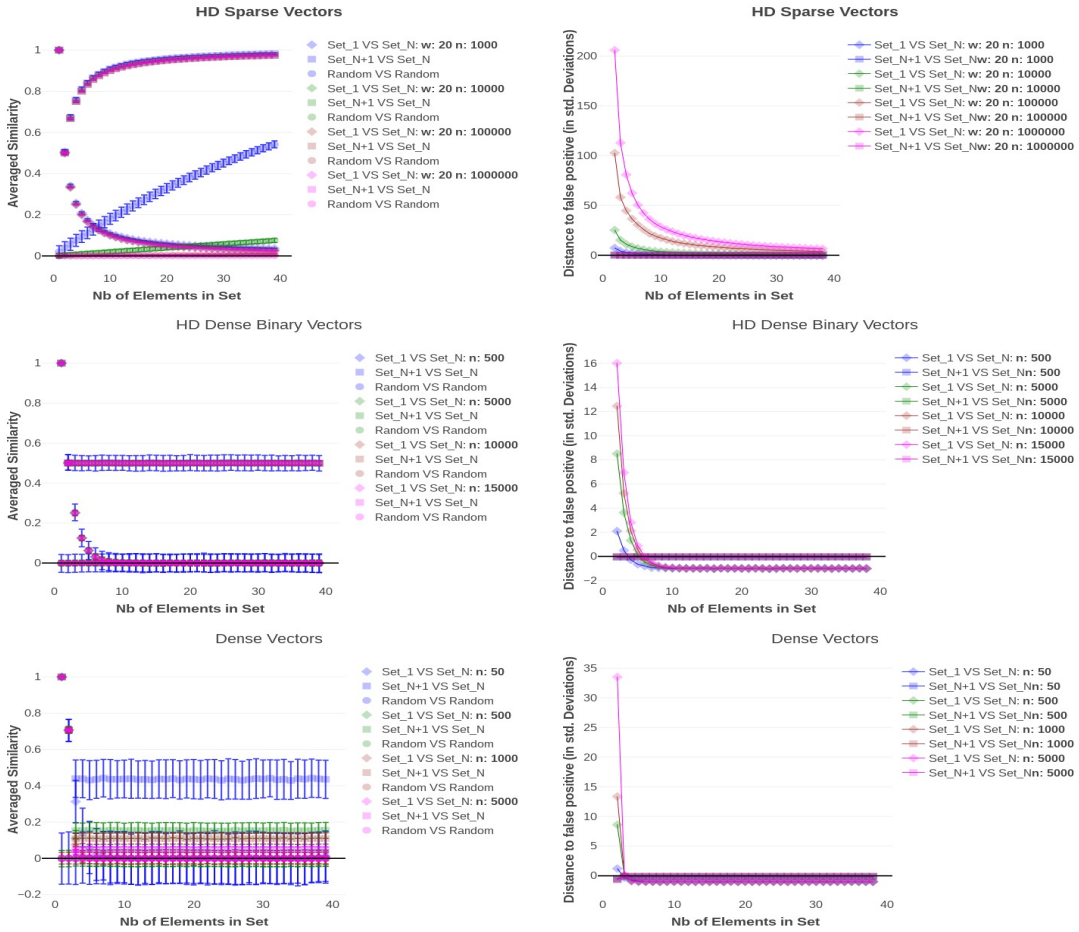

Figure 1: After building sets of sizes 1 to N by superposition, one compares similarities of random elements of same sizes and of sets of sizes 1 VS N, N VS $N+1$ with similar subconstituents. This is done on the left side for sparse and dense binary vectors as well as for dense vectors. On the right, the approximated relative distance to an incidentally wrong superposition in standard deviations are plotted (see subsection 3 for details). The number of standard deviations is on the y axis and secure retrievals are above zero. While dense vectors information collapses under superposition, large SDM would allow to identify sub-elements and distinguish correctly the nearest structures, thus confirming the theory.

- Dense: A dense Vector of length L filled with $Float64$ needs $L \times 64$ bits.
- SDM's: binary vectors of length L and W non zero elements needs $W \times 2$.

The dense vectors space needed is constant for both HD computing and dense vectors. Inversely, it is linear and proportional to the number of elements superposed for Sparse vectors. Notice also that for all the example shown in Fig.1, the SDM's have by far the lowest memory requirements.

4 Discussion and Conclusion

We have experimented with various vector types and sizes, focusing on the superpositions of atomic elements, we have illustrated and confirmed that large sparse vectors are the more robust containers for storing information. We observed that a large number of events could be encoded with minimal space and computational costs.

As a results, super sparse/superlarge SDM's are very well suited for one shot learning of events combinations storage.

References

[1] *Hyperdimensional Computing*, Kanerva Pentti, Addison Wesley, Massachusetts, 2nd edition, 1994.

[2] *Sparse Distributed Memory* Kanerva Pentti, The MIT Press (1988). ISBN 978-0-262-11132-4