Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors

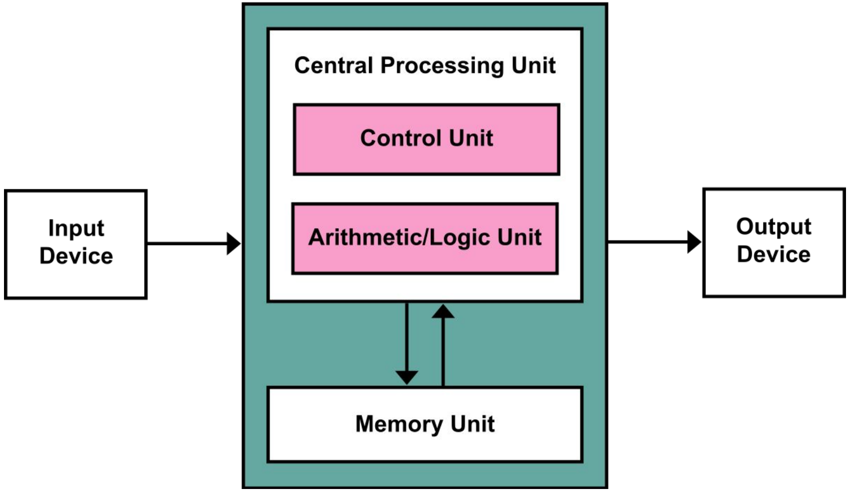
Andrew Larson, Michael Wang, Neel Dutta, Zechao Li

Introduction: The Brain as a Computer

- No two brains are identical
  - Yet, they can be functionally equivalent
- “Hardware” differs
- “Software” established over time
- Brain vs. Computer tradeoffs

The von Neumann Architecture

- Instructions and data are treated the same
- Modified von Neumann Architecture
  - Computing with large random patterns/high-dimensional random vectors
- Consists of:
  - RAM
  - Input and output channels
  - CPU
    - Sequencing Unit
    - ALU



An Engineering View of Representation

- Representation
  - Computers use binary representation
- Different choices offer different tradeoffs
- Brain’s representations are largely nonbinary
- Dimensionality in the context of numbers
  - Entity vs. Representation

Properties of Neural Representation

Hyperdimensionality

- The brain's circuits contain huge numbers of neurons and synapses
- Begs the need for high dimensional computing
- Hyperdimensional spaces have unique properties

Robustness

- Extremely tolerant of component failure
- Proportion of allowable errors increases with dimensionality

Randomness

- Internal wiring of circuits is left largely up to chance
- Authors start with random vectors in hyper dimensional space
- Randomness has been always in NN to some degree
- Tolerance to randomness ≠ Requiring randomness

Hyperdimensional Computer

Hyperdimensional computer

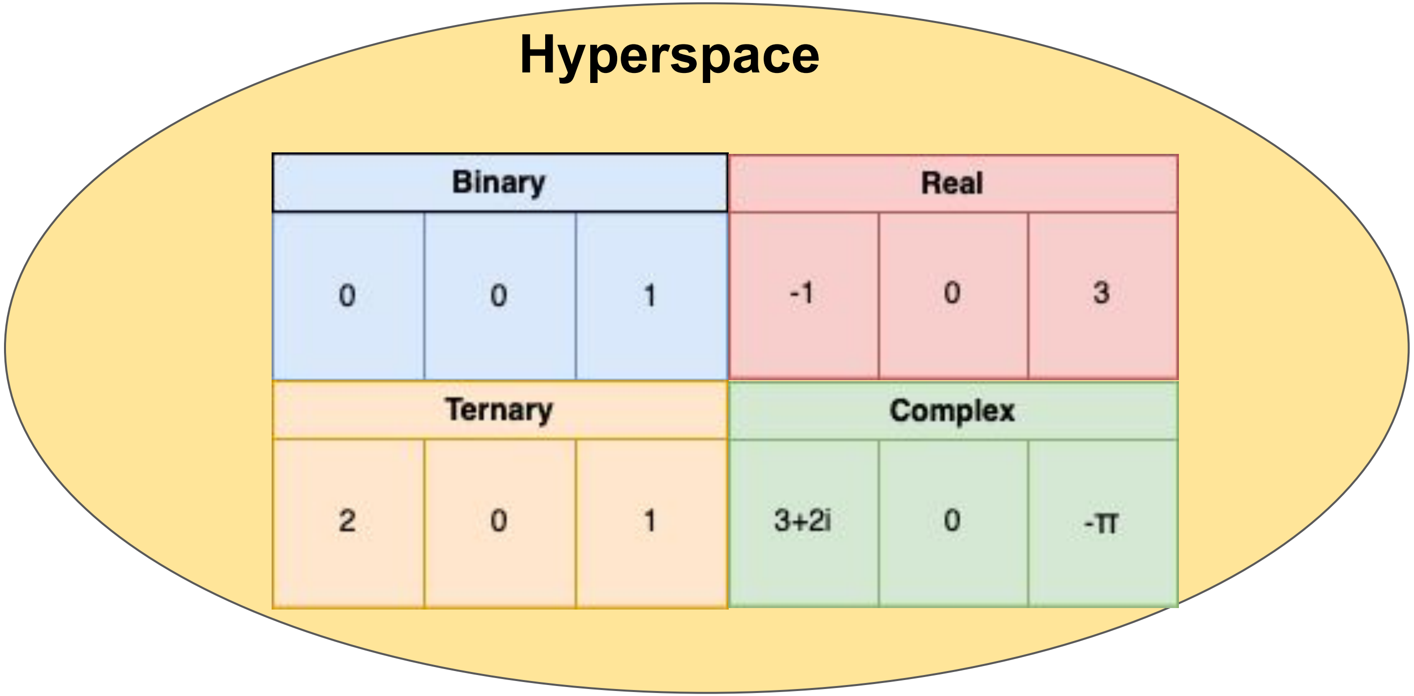
A possible solution to perform higher dimensional math computations

Modern day computers: low-dimensional binary vectors

Inherit concepts from modern day computing and apply to higher dimensions

Hyperdimensional representation

- Hyperdimensional representational space unit: **vector**



Hyperdimensional representation

- Vectors have several important properties

- **These properties can be hard to imagine based on intuition**

Vector Example

- Say we are given 10000 element bit vector
- **Point**: a possible vector in the hyperspace
  - 2^10000
- **Distance**: Number of places two points differ
  - Ex. <0, 0, 0…> and <1, 1, 1…>
  - Since all points differ, distance = 10000bits
- **Dimensionality**: 10000/10000 = 1

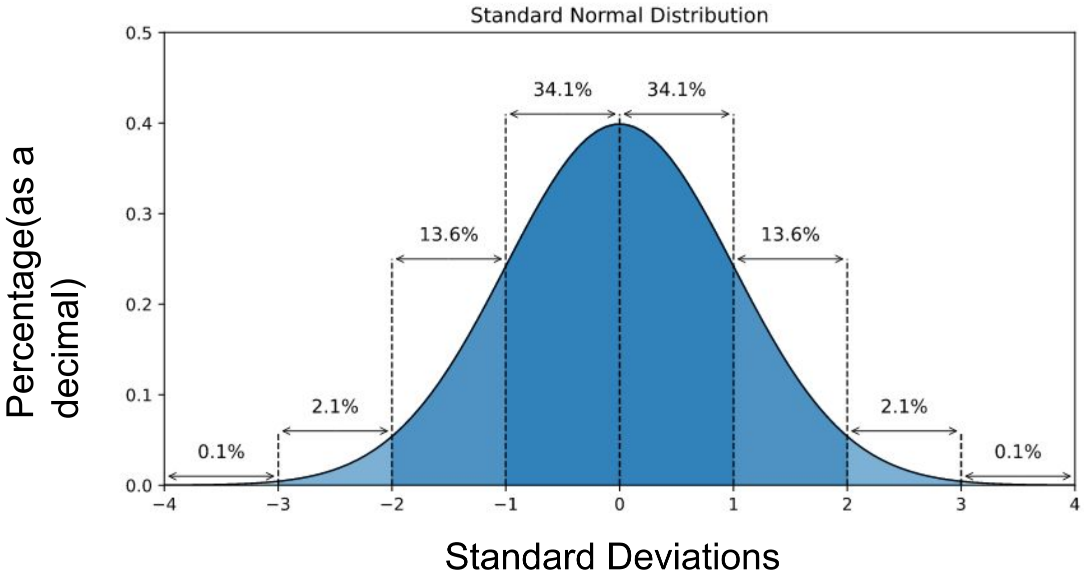
Vector Example continued

- For 10000 binary vector, mean distance is 5000
- Standard Deviation: $\sigma=\sqrt{(np(1-p))}$
    - $=\sqrt{(10000)(0.5)(1-0.5)}$
    - $= \sqrt{2500} = 50$
- **Almost all** points are 4700-5300 bits away
    - So called 600 bit wide "bulge"

# of Standard Deviations	% of Total
1	68.27
2	95.45
3	99.73
4	99.9937
5	99.99994
6 →	99.9999998

Hyperdimensional representation: distance

- Most points are more similar to each other than different
  - Distance between points is relatively the same
- Distance increases as we move **away** from midpoint
- Why does this matter?



Hyperdimensional representation: Robustness

- Distribution of points enables more robust model
    - Can tolerate large changes(>33%) without affecting the result
- Could we possibly falsely identify a vector that arrives at the wrong result?
    - Yes, but not as easy as you may think

Example: Robustness

- Suppose we are given 10000 bit binary vectors A and B that differ by 2500 bits with 3333 of points changed at random:
  - **$D = d + e - 2de$**

D = relative distance from noisy A to B

d = # of differing bits

e = # of bits changed at random

  - D_B = 2500 + 3333 - 2(2500)(3333) = on **average** 4166 bits away!
  - **Adding noise increases distance from noisy A to B 4166>3333 bits**

Similarity

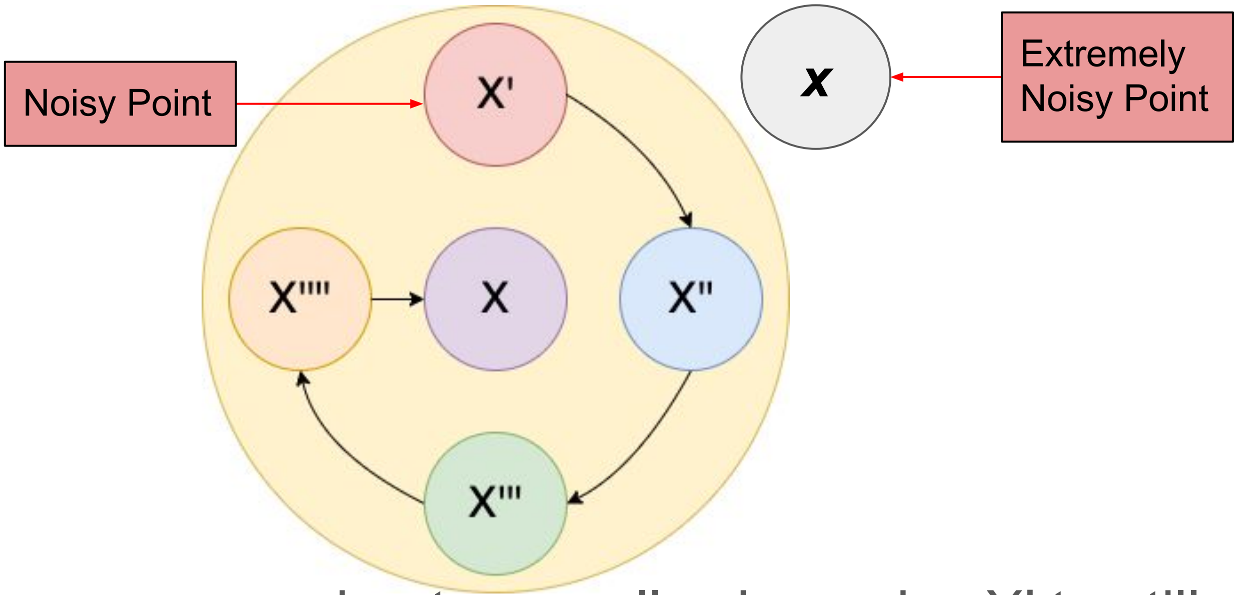
- **Distance**: Number of places two points differ
- Points are considered “similar” when within 0.5 dimensionality
- **Lower Distance does not imply similarity**
- **Similar patterned points can be unrelated**

Hyperdimensional Memory

- Need a new way to store large vectors
- Current architecture not suited to scale for large bit vectors
- Solution: Autoassociative memory

Autoassociative memory

- **Retrieve original data X from noisy version X’**



- Provides memory robustness, allowing noisy X’ to still retrieve data iteratively

Alternative

- Heteroassociative: Retrieve data X by addressing memory with pattern A’, or A
  - **Problem: cannot deal with noise as well(pattern X’ will have some noise)**

Hyperdimensional Arithmetic

- ALU needs to focus on the **principles** rather than hardware
  - Functionality doesn't change compared to modern computing
- Homogeneous operations enable special operations:
  - Invertible, distributive property, preserves distance, dissimilar to input vectors
- **These properties allow us to encode internal representations to form systems: cognitive code**

Constructing a Cognitive Code

Constructing a Cognitive Code

A collection of building blocks based on most basic operations and representations.

Storage

Autoassociative Memory

↓

Nearest Neighbor Search

Reversible Mappings → **Near Orthogonality**

Example: Sets and Storage

Set Construction

$$\vec{S} = \vec{A} + \vec{B} + \vec{C} + \vec{D} + \vec{E} + \vec{F}$$

↓

Reversible Mapping

$$\vec{S} \rightarrow \vec{S'}$$

↓

Storage

$$\vec{S'} \rightarrow Storage$$

Retrieval

Noisy: $\vec{S'}_n$

Retrieve $\vec{S'}$

Inverse Mapping: $\vec{S'} \rightarrow \vec{S}$

Retrieve One: $\vec{A}$ .... $\vec{F}$

Compute: $\vec{X} = \vec{S} - \vec{A}$

Retrieve: $\vec{B}$

Compute: $\vec{Y} = \vec{X} - \vec{B}$

…..

XOR Multiplication and Permutations

XOR Multiplication

Randomness: $X_A \stackrel{\text{def}}{=} A * X \rightarrow d(X_A, X) = |A|$

Permutation Matrix Π

Randomness, Distance Preservation

Sequences

Goal: Flatten sequence ABC into single hypervector

Storing

Construct

$S_1 = \Pi A + B$

$S_2 = \Pi B + C$

Fill In Memory

S_1
S_2
A
B
C

Can Do Better

Traversing

1. Given B
2. Compute ΠB
3. Probe with $\Pi B \rightarrow$ Get S_2
4. Compute $Z = S_2 - \Pi B$
5. Probe with $Z \rightarrow$ Get C

Cognitive Code: The Upshot

- Blocks: Sets, Sequences, Bindings, Substitutions

- Whole point was to build system which functions like brain
  - Does brain even use these? Seems to be the case…

- Makes hyperdimensional computing system easier to reason about and use

3 Examples of Cognitive Connotations

1. Context Vectors

- **High-dimensional vector that represents a word's context information**
- Context can be context windows or documents on a single topic
- Context information collected in large **matrixes of frequencies**
    - Rows are context vectors
    - Transformations used to obtain better vectors
    - Latent semantic analysis (LSA)
    - Random vector method

	Doc1	Doc2		Doc # 200,000
Word1	1	0	...	1
Word2	0	0		1
...	...	...		...
Word # 100,000	0	1		0

Context Vector Example:

- Example: A vocabulary of 100,000 words, a corpus of 200,000 documents, each has 500 words
- Latent Semantic Analysis (LSA):
  - 100,000 rows x 200,000 cols
  - Drawback: High cost, impractical to scale
- Random Vector:
  - Each Document randomly activates 20 columns
  - Document’s random index vector: 20 1s
  - 100,000 rows x 10,000 cols

	Col1	Col2		Col # 10,000
Word1	1	0	...	1
Word2	0	0		1
...	...	...		...
Word # 100,000	0	1		0

Context Vector continued: Takeaways

- Words with similar meanings have similar context vectors
- Linguistically impoverished and crude, disregards grammar
- Large random patterns can capture regularities in data
- Random-vector method is suited for incremental on-line learning

2 & 3. Inference from holistic mapping; Learning from example

- **Rules can be encoded in distributed representation, learnt from examples**
- Example: **Rule:** ‘If x is the mother of y and y is the father of z then x is the grandmother of z.”
  - $R_{xyz} = G_{xz} * (M_{xy} + F_{yz})$
- Apply on specific case: “Anna(a) is the mother of Bill(b)” and “Bill(b) is the father of Cid (c)”
  - $R_{xyz} * (M_{ab} + F_{bc}) = G_{xz} * (M_{xy} + F_{yz}) * (M_{ab} + F_{bc})$, Resulting vector G'_{ac}
- “Dollar of United States” – {Peso: “Dollar of Mexico”}

Looking Forward

- Math can be used to abstract properties of neural systems similar to cognitive models
- Ideas presented hopefully lead to more comprehensive cognitive models
- Challenges:
    - Future computers will be more well equipped to handle vectors
    - Chips will not be identical duplicates
    - Finding mathematical system to mirror cognitive models

Questions?

Ideas / Questions

Possible to have hypervectors which are not nearly-orthogonal in autoassociative memory.

Assumption that a von Neumann architecture is a good starting point.

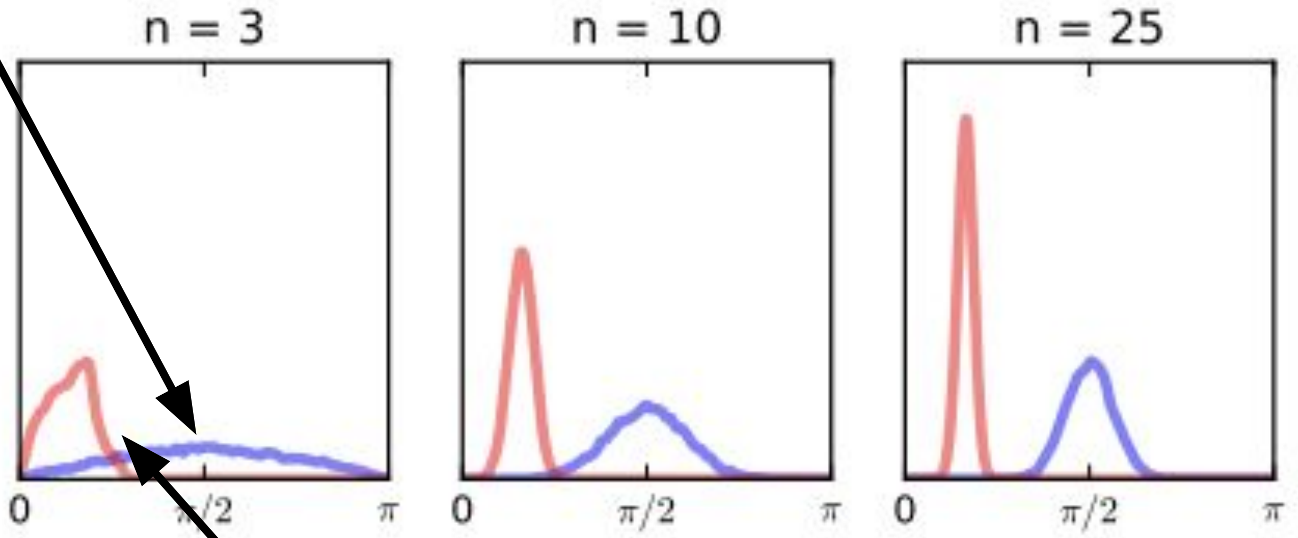
Do we necessarily need the building blocks mentioned in the cognitive code section?

Cool Idea: Holistic Representation

From *High Dimensional Geometry of Binary Neural Networks*

Angle between two continuous random vectors ([-1, 1])

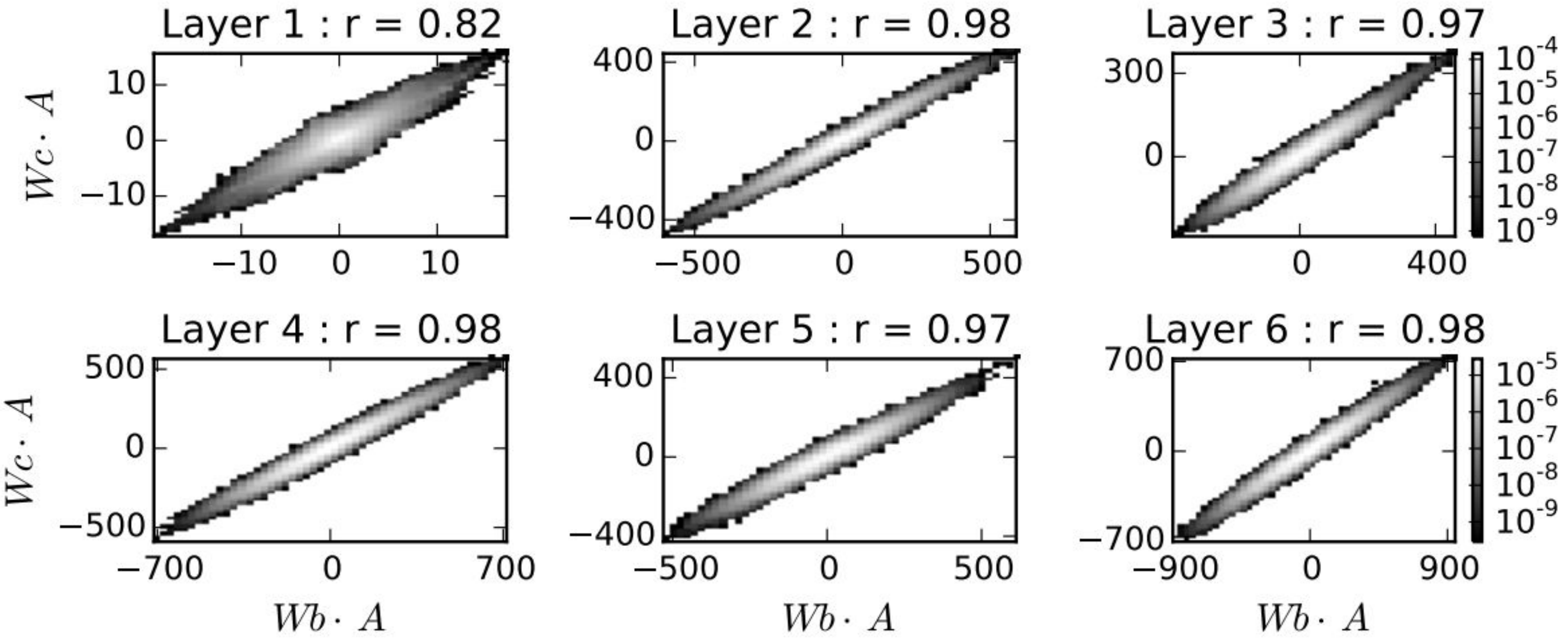
Probability



Angle

Angle between random continuous vector ([-1, 1]) and binarized version

Dot Product Preserves Direction



Bindings and Data Records

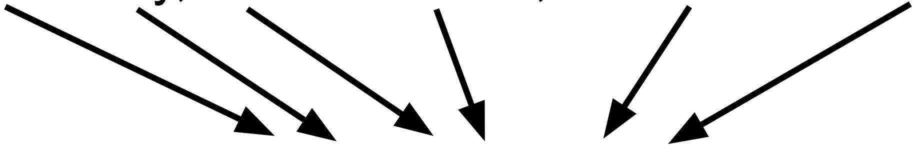
Binding → $C = \mathbf{\Pi} A * B$

Data Record

Combines several variable-value pairs into a single record

Example

Name = Mary, Sex = Female, Birth Year = 1966



$H = X*A + Y*B + Z*C$

3. What is the dollar of Mexico:

- **Infer the literal meaning from figurative expression**
- Prototyping: “Dollar of Mexico”, first language
- Expanding structure: “Peso of France”, second language

Looking Back

- Neural Net associative memories: cognitive models with high dimensionality
- Holographic reduced representation (HRR): reduced representation using circular convolution
- LSA vs random indexing
- So far only very general properties of hyperdimensional spaces are explored