COMPUTING WITH HYPERVECTORS

Pentti Kanerva

Redwood Center for Theoretical Neuroscience
UC Berkeley

What are Hypervectors?

. High-dimensional, e.g., vectors of 10,000
  - bits or
  - integers or
  - reals or
  - complex numbers (phase angles)

. Random or pseudo-random

. Vector components are
  - independent and
  - identically distributed (i.i.d.)

It is possible to build a fully *general* system of computing based on hypervectors. Computing is *transparent* and has many positive qualities we associate with *brains*:

. Robust and noise-tolerant

. Learns from data/example, learns by analogy

. Can learn fast: "One-shot" learning ("1" ≈ 5)

. Integrates signals from disparate senses

. Allows simple algorithms that scale to large problems efficiently

. Allows high degree of parallelism

Example from analogy:

The question

What is the Dollar of Mexico?

is readily understood by us but is a constant challenge to traditional AI and neural nets

It can be solved with hypervector mapping

This talk is about the **mathematical theory of hypervector computing**

The theory dates to the 1990s and is referred to variously as Holographic Reduced Representation (Plate), Binary Spatter Code (Kanerva), and Vector-Symbolic Architecture (Gayler & Levy)

The underlying math dates to the late 1800s and early 1900s and is referred to as abstract algebra or "modern" algebra

The math is subtle but simpler than the name "abstract algebra" might suggest

The theory can be made practical by nanotechnology

. Requires very large circuits

. Tolerates variation in the components

. Allows high degree of parallelism, reducing
  the need for fast switching

  - Hence energy-efficient

Computing with hypervectors complements
traditional numeric, symbolic, and neural-net
computing/deep learning

Computing with Hypervectors resembles ordinary computing with Booleans and numbers, including

. A memory for the vectors: High-capacity long-term storage that is content-addressable

. Operations on the vectors akin to addition and multiplication

- The operations form an **algebraic system** that is **richer than a field**

NOTE: The power and utility of *number arithmetic* is based on the *algebra of fields*

CAUTION: The algebra of hypervectors is *not* identical to, or part of, *linear algebra*

Operations on Hypervectors: An example

. **Seed vectors**: 10,000 randomly placed 1s and -1s

```
              1  2  3          ....          10,000
            .------------------------------------.
    A =     | +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1 |
            '------------------------------------'
```

. A seed vector can represent a letter of the alphabet, for example

. **Addition** (+): Coordinate by coordinate

```
      A =  +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1
      B =  +1 +1 +1 +1 -1 +1 .... -1 +1 +1 +1
      C =  -1 -1 +1 -1 -1 +1 .... -1 -1 -1 +1
  --------------------------------------------
  A+B+C =  +1 -1 +1 +1 -3 +1 .... -1 +1 -1 +3
```

. **Multiplication** (*): Coordinate by coordinate

```
    A =  +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1
    B =  +1 +1 +1 +1 -1 +1 .... -1 +1 +1 +1
-------------------------------------------
  A*B =  +1 -1 -1 +1 +1 -1 .... -1 +1 -1 +1

  NOTE: A*A = 1 1 1 1 ... 1 1 -> self-inverse
```

. **Permutation** (*r*): Rotation of coordinates

```
    A =  +1 -1 -1 +1 -1 -1 .... +1 +1 -1 +1
            /  /  /  /  /        /  /  /  /
           /  /  /  /  /        /  /  /  /
   rA =  -1 -1 +1 -1 -1 .... +1 +1 -1 +1 +1
```

. **Similarity** between vectors: Cosine

cos(**A**, **A**) = 1

cos(**A**,-**A**) = -1

cos(**A**, **B**) = 0 if **A** and **B** are **orthogonal**

The **blessing of dimensionality**: A *randomly* chosen hypervector is *approximately orthogonal* (dissimilar) to any vector seen so far

Key features of hypervector algebra; [*] denotes a property *not* shared by number fields

. Addition **commutes: A + B = B + A**

. Addition, multiplication and [*]permutation are **invertible**

. Multiplication **distributes** over addition

. Permutation **distributes** over both addition and multiplication [*]

. The output of addition is **similar** to the inputs [*]

. The outputs of multiplication and permutation are **dissimilar** to the inputs [*]

. Multiplication and permutation **preserve similarity** [*]

How are the operations used?

A, **B**, **C**, **P**, **R**, **S**, **X**, **Y**, **Z** are 10,000-D random

. Encoding a **pair** with multiplication (associating variable *x* with value *a*, also called **binding**): *p* = (*x*,*a*)

P = **X*A**

. Extracting the value of *x* from the pair:

```
X*P = X*(X*A) = (X*X)*A = A
                (X*X canels out)
```

. Encoding a **set** with addition: $s = \{a, b, c\}$

$$\mathbf{S} = \mathbf{A} + \mathbf{B} + \mathbf{C}$$

. Encoding a **data record** with a set of bound pairs: d = ‘($x = a$) & ($y = b$) & ($z = c$)'

$$\mathbf{D} = \mathbf{X}*\mathbf{A} + \mathbf{Y}*\mathbf{B} + \mathbf{Z}*\mathbf{C}$$

. Extracting the value of x from the record:

$$\begin{aligned}
\mathbf{X}*\mathbf{D} &= \mathbf{X} * (\mathbf{X}*\mathbf{A} + \mathbf{Y}*\mathbf{B} + \mathbf{Z}*\mathbf{C}) \\
&= \mathbf{X}*\mathbf{X}*\mathbf{A} + \mathbf{X}*\mathbf{Y}*\mathbf{B} + \mathbf{X}*\mathbf{Z}*\mathbf{C} \\
&= \mathbf{X}*\mathbf{X}*\mathbf{A} + (\mathbf{X}*\mathbf{Y}*\mathbf{B} + \mathbf{X}*\mathbf{Z}*\mathbf{C}) \\
&= \quad \mathbf{A} + \quad \text{noise} \\
&\approx \mathbf{A}
\end{aligned}$$

. Encoding a **sequence** with rotation and multiplication: (a,b)

$$\mathbf{AB} = r\mathbf{A} * \mathbf{B}$$

. Extending **AB** with **C**: (a,b,c)

$$\begin{aligned}\mathbf{ABC} &= r(\mathbf{AB}) * \mathbf{C} \\ &= rr\mathbf{A} * r\mathbf{B} * \mathbf{C}\end{aligned}$$

. Extracting the first element of **ABC**:

$$\begin{aligned}ss(\mathbf{ABC} * \mathbf{BC}) &= ss(rr\mathbf{A} * r\mathbf{B} * \mathbf{C} * r\mathbf{B} * \mathbf{C}) \\ &= ss(rr\mathbf{A}) \\ &= \mathbf{A}\end{aligned}$$

where s is the inverse (counter-rotate) of r

The representation is **holographic**: Every piece of information is distributed over every coordinate

- Every subset of coordinates is computing the same thing, only less accurately

- No coordinate is critical

  - Hence *robustness*

The set of operations is **complete**:

- We could implement hypervector Lisp

Examples of Hypervector Computing

1. Language Vectors: We made 10,000-D language vectors for 21 EU languages from seed vectors representing letters. Projected onto a plane, the languages cluster according to known families:

```
                                      Italian
                                       *    *Romanian
                                      Portuguese
                                       *    *Spanish
     *Slovene                                   *French
*Bulgari *Czech
      *Slovak                                    *English
                           *Greek
   *Polish                      *Lithuanian
                                 *Latvian
                           *Estonian
                    *       *Finnish
                   Hungarian


                                                *Dutch
                                        *Danish  *German
                                           *Swedish
```

We tested the language vectors' ability to identify languages by comparing them to vectors for 21,000 test sentences (1,000 sentences from each language). The best match agreed with the correct language 97.8% of the time.

2. Semantic Vectors with Random Indexing

We computed semantic vectors for words from seed vectors representing documents and achieved TOEFL scores (Test of English as a Foreign Language) on par with LSA's (Latent Semantic Analysis). LSA relies on compute-heavy Singular Value Decomposition; its complexity grows with the square of the number of documents. Random Indexing is linear in the number of documents and scales easily to millions of documents.

. Random indexing processed through 37,000 documents in 10 minutes vs. LSA's 2+ hours (in 2000). Extended to a million documents: 10 hours for Random Indexing vs. 300 days for LSA.

Random indexing is a form of Random Projections. A company in Sweden has scanned news articles in several languages with random indexing since 2008

3. Analogical Mapping with Multiplication by Hypervector

What is the Dollar of Mexico?

Encoding of **USA** and **MEX**ico: **Na**me of country, **Ca**pital city, **Mon**etary unit

USA = Nam*Us + Cap*Dc + Mon*$
MEX = Nam*Mx + Cap*Mc + Mon*P

Pairing up the two--binding

Pair = USA*MEX

Analyzing the pair

Pair = Us*Mx + Dc*Mc + $*P + noise

Literal interpretation of *Dollar of Mexico*
produces nonsense:

$$\begin{aligned}
\mathbf{\$}*\mathbf{MEX} &= \mathbf{\$} * (\mathbf{Nam}*\mathbf{Mx} + \mathbf{Cap}*\mathbf{Mc} + \mathbf{Mon}*\mathbf{P}) \\
&= \mathbf{\$}*\mathbf{Nam}*\mathbf{Mx} + \mathbf{\$}*\mathbf{Cap}*\mathbf{Mc} + \mathbf{\$}*\mathbf{Mon}*\mathbf{P} \\
&= \text{noise} + \text{noise} + \text{noise} \\
&\quad \text{(nothing cancels out)}
\end{aligned}$$

However, what in Mexico corresponds to Dollar in USA?

$$\begin{aligned}
\mathbf{\$}*\mathbf{Pair} &= \mathbf{\$} * (\mathbf{USA}*\mathbf{MEX}) \\
&= \mathbf{\$} * (\mathbf{Us}*\mathbf{Mx} + \mathbf{Dc}*\mathbf{Mc} + \mathbf{\$}*\mathbf{P} + \text{noise}) \\
&= \mathbf{\$}*\mathbf{Us}*\mathbf{Mx} + \mathbf{\$}*\mathbf{Dc}*\mathbf{Mc} + \mathbf{\$}*\mathbf{\$}*\mathbf{P} + \mathbf{\$}*\text{noise} \\
&= \text{noise} + \text{noise} + \mathbf{P} + \text{noise} \\
&= \mathbf{P} + \text{noise} \\
&\approx \mathbf{P}
\end{aligned}$$

Parting Thoughts

. The cognitive/computational powers of the brain are intimately related to the mathematical properties of *high-dimensional* spaces

  - Truly high: D = 10,000 ... 100,000

. Such spaces can be understood in terms of their *geometry* and *algebra*

. A mode of computing that exploits these properties is made practical by *nanotechnology*

With *thanks* to my colleagues and collaborators in exploring high-D representation: Aditya Joshi, Johan Halseth, Andrew Liu, Anna Pham, Quinn Tran, Mika Laiho, Paxon Frady, Abbas Rahimi, Jan Rabaey, Fritz Sommer, and Bruno Olshausen, and to funding by DARPA/SRC/SONIC and Intel

Computing with Hypervectors

Abstract:
Hypervectors are high-dimensional (e.g., $D = 10{,}000$), (pseudo)random, with independent identically distributed (i.i.d.) components. Computing with hypervectors is an alternative to conventional (von Neumann) computing with Booleans and numbers, and to neural nets and deep learning trained with gradient descent (error back-propagation). At the core is an algebra of operations on vectors, resembling the algebra of numbers that makes computing with numbers so useful. New representations are computed from existing ones very fast compared to arriving at them through gradient descent, and the algebra allows composed vectors to be factored into their constituents. Computing with hypervectors resembles traditional neural nets in its reliance on distributed representation, making it tolerant of noise and component failure. It fills the gap between traditional and neural-net computing, and the architecture is ideal for realization in nanoelectronics.

****
2016 Neuro-Inspired Computational Elements Workshop (NICE 2016)
Tuesday 8 March 2016
10:10-10:35
****