Motivation & Objectives

Analogy is really cool and central to cognition

Analogy is a good use case for the unique properties of VSA/HDC

· What makes analogy hard for conventional computing?
· Which VSA/HDC features might help with analogy?
· Not a solved problem

Use a set of attempts at aspects of analogy to highlight some VSA design issues
Outline

What is analogy?

Why is analogy hard for conventional computing?

VSA design examples:

- Plate - Similarity of hand crafted vectors
- Mikolov et al - Similarity of learned word vectors
- Kanerva - Simple substitution
- Emruli et al - Substitution with lookup
- Gayler & Levy - Settling on substitution
What is *ANALOGY*?

*ANALOGY* $\triangleq$ what analogy *really* is

Whatever it is, *ANALOGY* as a cognitive phenomenon is a complex, nuanced thing

- Everybody presents a different partial view of *ANALOGY*
  - Tendency to interpret the partial view as the whole thing
    - Analogical reasoning
    - Proportional analogies
    - Grand analogy (analogy as a party trick)
  - Please don’t do that

*ANALOGY* is too big to fit in this lecture, so I will resort to assertions and hand waving to explain enough of it for current purposes
Analogy is the core of cognition

Quote from Hofstadter (2006):

analogy-making $\triangleq$
the perception of common essence$^1$
between two things$^2$

$^1$ In one's current frame of mind
$^2$ Thing $\triangleq$ mental thing

See also
Chalmers et al (1992)
Blokpoel et al (2018)

I will jump off from Blokpoel:
cognition as inference to the best explanation
Inference to the Best Explanation

The cognitive loop:
Given some inputs (evidence $e$) and a set of potential explanations (hypotheses $H$) find the hypothesis ($h$) that best explains the evidence

Evidence and hypotheses are represented relationally (trees/graphs)

- A bet that natural regularities are “best” captured as transformations

“explains” is interpreted as graph structure matching - (sub)graph isomorphism

- structural similarity = literal similarity | optimal substitution of literals
- analogical “common essence” = common relational structure

Partial structure matching enables inference by carrying structure from one representation to another (pattern completion via autoassociative memory)
Where do the hypotheses come from?

Hypotheses are generated from all the agent’s relevant knowledge.

The hypothesis space must be open-ended, to allow for explaining novelty.

- Hypotheses must be compositional
  - Allows infinite productivity
  - Allows novel compositions of familiar components
  - Like a grammar for hypotheses
- Substitution enables composition (there may be other mechanisms)
Example: Relational representation

solar system = base structure = hypothesis (on this slide)
atom = target structure = evidence (on this slide)
structural similarity = literal similarity | \{ \text{sun} \mapsto \text{nucleus}, \text{planet} \mapsto \text{electron} \}

Chalmers et al (1992)
Example: Relational representation of evidence
Example: Relational representation of knowledge

Blokpoel et al (2018)
Example: Analogical augmentation

Relational Representation

Analogical Augmentation

Blokpoel et al (2018)
Example: Augmentation of evidence

Blokpoel et al (2018)
Example: Explanation

(all possible representations of evidence $e$) \longrightarrow (all possible representations of core $k$)

\[\text{--- analogical match}\]
Why is analogy hard for conventional computing?

Subgraph isomorphism, of two graphs, is NP-complete (intractable)

- Considers all possible vertex mappings
- The “obvious” approach is brute-force exhaustive enumeration
- Each vertex mapping provides very little information about the adequacy of the other vertex mappings

Considering all the base structures in the agent’s knowledge is much larger

Considering the transitive closure of analogical augmentations is much larger
Preview: Which VSA features might help?

Hardware parallelism (elementwise operations with small fan-in)

Mathematical parallelism (avoids explicit enumeration)

- The hardware only “sees” the total vector
- Distributive parallelism
  - $(A + B + C) \rho (P + Q + R) = A\rho P + A\rho Q + \ldots B\rho P + B\rho Q + \ldots$
- Equational parallelism
  - $T = (A + B + C) = (P + Q + R + S) = (X * Y * Z) = \ldots$
- Enables holistic transformations

Substitution is a primitive (via binding)

- Every value is potentially a variable
Plate (1994) - Hand crafted similarity

Focus of Chapter 6 of Plate’s thesis (1994) is the use of dot-product similarity as a measure of structural similarity of representations.

Reports experiments with hand-crafted representations aimed at qualitatively reproducing the results of psychology research into human judgement of analogical similarity under varying contributions of component similarity to overall similarity.

My take,

- superficial similarity $\approx$ similarity of arguments of relations
- structural similarity $\approx$ similarity of pattern of relations

but researchers are free to suit the details of their definitions to their needs.
Example stimuli

P (Probe) “Spot bit Jane, causing Jane to flee from Spot”

LS (Literal Similarity) “Fido bit John, causing John to flee from Fido.” (Has both structural and superficial similarity to the probe P.)

SF (Surface features) “John fled from Fido, causing Fido to bite John.” (Has superficial but not structural similarity.)

CM (Cross-mapped analogy) “Fred bit Rover, causing Rover to flee from Fred.” (Has both structural and superficial similarity, but types of corresponding objects are switched.)

AN (Analogy) “Mort bit Felix, causing Felix to flee from Mort.” (Has structural but not superficial similarity).

FOR (First-order-relations only) “Mort fled from Felix, causing Felix to bite Mort.” (Has neither structural nor superficial similarity, other than shared predicates.)
# Base and token vectors

<table>
<thead>
<tr>
<th>Base vectors</th>
<th>Token vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>jane = \langle \text{person} + \id_{jane} \rangle</td>
</tr>
<tr>
<td>dog</td>
<td>john = \langle \text{person} + \id_{john} \rangle</td>
</tr>
<tr>
<td>cat</td>
<td>fred = \langle \text{person} + \id_{fred} \rangle</td>
</tr>
<tr>
<td>mouse</td>
<td>spot = \langle \text{dog} + \id_{spot} \rangle</td>
</tr>
<tr>
<td>stroke</td>
<td>fido = \langle \text{dog} + \id_{fido} \rangle</td>
</tr>
<tr>
<td>bite&lt;sub&gt;agt&lt;/sub&gt;</td>
<td>rover = \langle \text{dog} + \id_{rover} \rangle</td>
</tr>
<tr>
<td>bite&lt;sub&gt;obj&lt;/sub&gt;</td>
<td>felix = \langle \text{cat} + \id_{felix} \rangle</td>
</tr>
<tr>
<td>flee&lt;sub&gt;agt&lt;/sub&gt;</td>
<td>mort = \langle \text{mouse} + \id_{mort} \rangle</td>
</tr>
<tr>
<td>flee&lt;sub&gt;from&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>cause&lt;sub&gt;antc&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>cause&lt;sub&gt;cnsq&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>stroke&lt;sub&gt;agt&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>stroke&lt;sub&gt;obj&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>lick&lt;sub&gt;agt&lt;/sub&gt;</td>
<td></td>
</tr>
<tr>
<td>lick&lt;sub&gt;obj&lt;/sub&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Stimulus episode representation construction

Probe episode (P): “Spot bit Jane, causing Jane to flee from Spot”

\[ P_{\text{bite}} = \langle \text{bite} + \text{bite}_{\text{agt}} \odot \text{spot} + \text{bite}_{\text{obj}} \odot \text{jane} \rangle \]
\[ P_{\text{flee}} = \langle \text{flee} + \text{flee}_{\text{agt}} \odot \text{jane} + \text{flee}_{\text{from}} \odot \text{spot} \rangle \]
\[ P_{\text{objects}} = \langle \text{jane} + \text{spot} \rangle \]
\[ P = \langle \text{cause} + P_{\text{objects}} + P_{\text{bite}} + P_{\text{flee}} + \text{cause}_{\text{antc}} \odot P_{\text{bite}} + \text{cause}_{\text{cnsq}} \odot P_{\text{flee}} \rangle \]

Note the addition of “lower level” components into the representations

- These are not strictly necessary for representing the structure

Construction of all the episode representations follows the same scheme
# Dot-product similarity with Probe

<table>
<thead>
<tr>
<th>P</th>
<th>Spot bit Jane, causing Jane to flee from Spot.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Episodes in long-term memory:</th>
<th>Object attributes</th>
<th>First-order relation names</th>
<th>Higher-order structure</th>
<th>HRR</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{LS}$ Fido bit John, causing John to flee from Fido.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.71</td>
<td>1.0</td>
</tr>
<tr>
<td>$E_{SF}$ John fled from Fido, causing Fido to bite John</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>0.47</td>
<td>1.0</td>
</tr>
<tr>
<td>$E_{CM}$ Fred bit Rover, causing Rover to flee from Fred.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.47</td>
<td>1.0</td>
</tr>
<tr>
<td>$E_{AN}$ Mort bit Felix, causing Felix to flee from Mort.</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>0.42</td>
<td>0.6</td>
</tr>
<tr>
<td>$E_{FOR}$ Mort fled from Felix, causing Felix to bite Mort.</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>0.30</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Plate (2000)
Reminders of dot-product similarity properties

\[ \text{sim}(A, A') = \text{sim}(A, (A + X)) > 0 \]

\[ \text{sim}(A, (A \otimes P)) \approx 0 \]

\[ \text{sim}((A \otimes P), (A' \otimes P)) = \text{sim}(A, A') > 0 \]

\[ \text{sim}((A \otimes P), (A' \otimes P')) = \text{sim}(A, A') \times \text{sim}(P, P') > 0 \]

\[ \text{sim}((A \otimes B \otimes \ldots \otimes X), (A' \otimes B \otimes \ldots \otimes X)) = \text{sim}(A, A') > 0 \]

\[ \text{sim}((A \otimes B \otimes \ldots \otimes X), (A' \otimes B \otimes \ldots \otimes X \otimes Y)) \approx 0 \]

If using self-inverse products:

\[ \text{sim}((A \otimes B \otimes \ldots \otimes X), (A' \otimes B \otimes \ldots \otimes X \otimes Y \otimes Y^\dagger)) = \text{sim}(A, A') \]

\( \dagger \). \( \otimes Y \) is equivalent to applying the substitution \( Y \mapsto 1 \)
My interpretation of Plate (1994) Chapter 6

Representation of structure requires product operations, which destroys dot-product similarity of result to arguments

- structural similarity $\neq$ only the dot-product similarity of core structure
- Needs something extra

Plate decorates the composite core structures with components to create similarity

- Might be *ad hoc* (depends on whether it is natural for the construction process)
- Might be missing necessary structure
  - Predicates are not represented as unique instances,
    - “Spot bit Fido causing Felix to bite John” is ambiguous
    - but representing them as unique instances might destroy their similarity
  
  \[
  \text{sim}((\text{bite}_1 \otimes \text{bite}_{\text{agt}} \otimes \text{Spot}), (\text{bite}_2 \otimes \text{bite}_{\text{agt}} \otimes \text{Spot})) \approx 0
  \]
Dot-product similarity is local

Dot-product similarity is at the heart of VSA system dynamics

Dot-product similarity is very “local”

- Almost all vectors are quasi-orthogonal to current state vector
- Only a tiny fraction of the vector space has nonzero similarity with state
- Miraculous luck if all directions of interest are local to the state
- Similarity driven dynamics alone can’t select between non-local directions

Relational structure encoded by Multiply and Permute, which are orthogonalising

- Something needs to be done to map relational structure into the local space so that it can engage the similarity dynamics
Mikolov et al (2013) - Learned word similarity

Proportional analogy with learned “semantic” vectors for words

- \( a : a' :: b : b' \)
- \( \text{man} : \text{woman} :: \text{king} :: \text{queen} \)
- \( V_{\text{woman}} - V_{\text{man}} + V_{\text{king}} = V_{\text{queen}} \)
Successful analogy? Not so much

 Doesn’t work as well as originally thought (Rogers et al, 2017)

· Relies on excluding $b$ from answer set (or using multiple choice)
· Works best when $a$, $a'$, and $b$ are relatively similar to each other and to $b'$
· Poor at some classes of relations, e.g. synonymy, antonymy

ANALOGY enables proportional analogy, but proportional analogy $\neq$ ANALOGY

“semantic” vectors don’t capture SEMANTICS

· Semantic vectors don’t know how to change a flat tyre
· Captures a narrow subset of linguistic regularities induced by SEMANTICS
· ANALOGY engages with SEMANTICS
Vector semantics and dot-product similarity

Proportional analogy via semantic vectors implies semantics $\equiv$ additive features

$$V_{woman} - V_{man} + V_{king}$$

$$= (V_{person} + V_{FvsM}) - (V_{person} - V_{FvsM}) + (V_{person} - V_{FvsM} + V_{royal})$$

$$= V_{person} + V_{FvsM} + V_{royal}$$

$$= V_{queen}$$

Additive features can’t capture SEMANTICS

ANALOGY is about structural relational similarity

- Representing relational structure requires product operators
- Static dot-product similarity structure is driven by additive structure
- Dot-product similarity (by itself) can’t fully capture structural similarity
Systematic substitution via binding

A binding can be used as a partial function for substitution
The substitution is applied uniformly across the components of the argument

With a commutative, self-inverse product operator, e.g. BSC, MAP
(I won’t discuss non-commutative or non-self-inverse products here):

\[ A \otimes X \equiv \{ A \mapsto X, \ X \mapsto A \} \]

Apply the substitution by binding it with the argument:

\[
\begin{align*}
(A \otimes X) \otimes (A + A \otimes B + X \otimes C + D) &= A \otimes X \otimes A + A \otimes X \otimes A \otimes B + A \otimes X \otimes X \otimes C + A \otimes X \otimes D \\
&= (A \otimes A) \otimes X + (A \otimes A) \otimes X \otimes B + A \otimes (X \otimes X) \otimes C + A \otimes X \otimes D \\
&= 1 \otimes X + 1 \otimes X \otimes B + A \otimes 1 \otimes C + A \otimes X \otimes D \\
&= X + X \otimes B + A \otimes C + A \otimes X \otimes D \\
\end{align*}
\]

\[
(A + A \otimes B + X \otimes C + D) \overset{(A \otimes X)}{\mapsto} (X + X \otimes B + A \otimes C + A \otimes X \otimes D)
\]
Subtlety of binding

Multiple interpretations of bindings \( A \otimes X \) depending on how they are used, e.g.:

- Key:value pair in a dictionary (note key and value are treated identically)
- Variable:value pair (note variable and value are treated identically)
- Virtual feature detector neuron \( A = \) pattern detected, \( X = \) identity of neuron
- Inference rule \( A = \) antecedent, \( B = \) consequent
- Substitution pattern \( A = \) search pattern, \( B = \) replacement pattern

\( A \) and \( X \) can be arbitrarily complex composites (everything is just a vector), e.g.:

- \( ( A + B + C ) \otimes X = A \otimes X + B \otimes X + C \otimes X \)
- \( A \otimes ( X + Y + Z ) = A \otimes X + A \otimes Y + A \otimes Z \)
- \( A \otimes B \otimes X \otimes Y = A \otimes ( B \otimes X \otimes Y ) = ( A \otimes X \otimes Y ) \otimes B = \ldots \)
Kanerva (2010) - What is the Dollar of Mexico?

\[ V_{\text{ustates}} = V_{\text{name}} \otimes V_{\text{USA}} + V_{\text{capital}} \otimes V_{\text{WDC}} + V_{\text{currency}} \otimes V_{\text{USD}} \]
\[ V_{\text{mexico}} = V_{\text{name}} \otimes V_{\text{MEX}} + V_{\text{capital}} \otimes V_{\text{MXC}} + V_{\text{currency}} \otimes V_{\text{MXN}} \]

\[ V_{U \otimes M} = V_{\text{ustates}} \otimes V_{\text{mexico}} \]
\[ = V_{\text{USA}} \otimes V_{\text{MEX}} + V_{\text{WDC}} \otimes V_{\text{MXC}} + V_{\text{USD}} \otimes V_{\text{MXN}} + \text{noise}_1 \]
(a sum of filler substitutions - fillers occupying the same role have been bound)

Warning: \textit{noise} is the sum of all terms you know will not be important

Apply the USA/Mexico mapping, e.g. “What is the Dollar of Mexico?”
\[ V_{U \otimes M} \otimes V_{\text{USD}} \]
\[ = (V_{\text{USA}} \otimes V_{\text{MEX}} + V_{\text{WDC}} \otimes V_{\text{MXC}} + V_{\text{USD}} \otimes V_{\text{MXN}} + \text{noise}_1) \otimes V_{\text{USD}} \]
\[ = \text{noise}_2 + \text{noise}_3 + V_{\text{USD}} \otimes V_{\text{MXN}} \otimes V_{\text{USD}} + \text{noise}_1 \otimes V_{\text{USD}} \]
\[ \approx V_{\text{MXN}} \]
Hand crafted substitution

This approach works because of identical roles, e.g. $V_{currency}$

The role representations are static and enumerated in advance

Great if that’s all you need

- Seriously, if that’s all you need, it’s the best way to do something analogy-like

$ANALOGY$ needs dynamic substitutions chosen in response to the context
“The analogical mapping unit (AMU) which learns mappings of the type $x_k \mapsto y_k$ from examples and uses bundled mapping vectors stored in the SDM to calculate the output vector $y'_k$” Emruli et al (2013)
How it works

$x_k$ is used as the address for the Sparse Distributed Memory (SDM)
The mapping $x_k \otimes y_k$ is used as the value to store in the SDM

Mappings are noncommutative because $x_k$ and $y_k$ are used differently

Write mode: mappings are written to SDM
Read mode: retrieves average mapping corresponding to $x_k$ and applies it to $x_k$

SDM does a sort of averaging memory over similar addresses
Interpolates over mappings
Can’t generate completely novel mappings

Note the “circuit based” approach, including non-VSA components (SDM)
There is usually an amount of plumbing and control to deal with
A purist would make the control distributed (VSA-like) but it’s usual to make the control localist as an engineering hack
Gayler & Levy (2009) - Settling on substitution

Graph isomorphism (not analogical mapping, but a proxy for necessary process)

Find vertex mappings that make the graphs identical

- \( \{ A \mapsto P, B \mapsto Q, C \mapsto S, D \mapsto R \} \)
- \( \{ A \mapsto P, B \mapsto Q, C \mapsto R, D \mapsto S \} \)
How it works: The long and winding road

The explanation is going to be long winded (sorry)

- What is a graph isomorphism (implementable definition)?
- Localist heuristic to find graph isomorphisms
- VSA distributed implementation of localist method
Adjacency matrix of a graph

How to represent a graph with a matrix

- Row and column indices correspond to vertices
- Cell entries indicate edges

```
A  B  C  D
A  0  0  0  0
B  0  0  1  1
C  0  1  0  0
D  0  1  0  0
```

Gayler & Levy (2009)
**Association graph**

The association graph is a graph product of the two graphs:

- Vertices correspond to vertex mappings of the two graphs
- Edges correspond to edge existence agreement of the vertex mappings
- A maximal clique corresponds to a maximal isomorphism of the two graphs

How to find a maximal clique of a graph

Replicator equations (from evolutionary game theory)
Also interpretable as Bayesian update

\[ x(t) = \text{prior distribution} = \text{support for each possible vertex mapping} \]
\[ x(t + 1) = \text{posterior distribution} = \text{support for each possible vertex mapping} \]
\[ w = \text{adjacency matrix of association graph} \]
\[ \pi(t) = \text{likelihood} = \text{multiplicative update to vertex mapping support given } w \]

\[ \pi_i(t) = \sum_{j=1}^{N} w_{ij} x_j(t) \]

\[ x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{N} x_j(t)\pi_j(t)} \]

Gayler (2009) Melbourne University presentation
Replicator equation circuit

Localist representation of mappings (potentially large)

$k$ number of vertices in each of the two graphs

$\dim(x) = \dim(\pi) = k^2$

$\dim(w) = k^2 \times k^2$

$\wedge \triangleq$ elementwise product
Settling of localist replicator equations

Gayler (2009) Melbourne University presentation
VSA representations for replicator equations

Vertices: $A, B, C, D, P, Q, R, S$

Edges: $B \otimes C, B \otimes D, Q \otimes R, Q \otimes S$

Graph vertex sets: $(A + B + C + D), (P + Q + R + S)$

Graph edge sets: $(B \otimes C + B \otimes D), (Q \otimes R + Q \otimes S)$

Initial potential vertex mappings $= x(t = 1) =$

$(A + B + C + D) \otimes (P + Q + R + S) =$

$A \otimes P + A \otimes Q + \ldots + B \otimes P + B \otimes Q + \ldots + D \otimes S$

Association graph edges (positive only) = potential edge mappings $= w =$

$(B \otimes C + B \otimes D) \otimes (Q \otimes R + Q \otimes S) =$

$(B \otimes C \otimes Q \otimes R) + (B \otimes C \otimes Q \otimes S) + (B \otimes D \otimes Q \otimes R) + (B \otimes D \otimes Q \otimes S)$
VSA replicator equation circuit

Interpret all vectors as being of the form $kV$, where $k$ is the magnitude/support for $V$ (the unit magnitude direction).

Analog computing: $V$ is the labelled wire, $k$ is the voltage on the wire.

Gayler (2009) Melbourne University presentation
Multiset intersection

\[ \land \triangleq \text{multiset intersection} \]

A multiset is a set with a nonnegative magnitude of membership for each element (i.e. the magnitudes of the component vectors vary across components)

\[
\begin{align*}
arg_1 &= aV_1 + bV_2 + cV_3 \\
arg_1 &= pV_1 + qV_2 + rV_4
\end{align*}
\]

\[ \land (arg_1, arg_2) = apV_1 + bqV_2 \]

The elementwise multiplication of the magnitudes of the component vectors

This corresponds to the elementwise multiplication of support for the vertex mappings in the localist version

Won’t go into the implementation here
Evidence propagation

Association graph edges have the form: \( B \otimes C \otimes Q \otimes R \)
and can be interpreted as mappings:
\( (B \otimes C) \otimes (Q \otimes R) \) \# interpret as mapping between edges in the graphs
\( (B \otimes Q) \otimes (C \otimes R) \) \# interpret as mapping between vertex mappings

Association graph edges applied as inference rules:
\[
\pi = x \otimes w \\
= (B \otimes Q + \ldots) \otimes (B \otimes C \otimes Q \otimes R + \ldots) \\
= (\text{vertex mappings}) \otimes (\text{mappings between vertex mappings}) \\
= (k(B \otimes Q) + \ldots) \otimes ((B \otimes Q) \otimes (C \otimes R) + \ldots) \\
= k(C \otimes R) + \ldots
\]

Interpret \( (B \otimes Q) \otimes (C \otimes R) \) as the rule:
“\( \text{To the extent } k \text{ that } B \otimes Q \text{ is supported as part of the solution} \)
Increase the support for \( C \otimes R \text{ as part of the solution by } k \)”
Settling of VSA replicator equation circuit

Gayler (2009) Melbourne University presentation
References / Reading


B. Emruli, R. W. Gayler, and F. Sandin (2013) *Analogical mapping and inference with binary spatter codes and sparse distributed memory*. The 2013 International Joint Conference on Neural Networks (IJCNN)


