Connectionist Symbol Processing

15-496/782: Artificial Neural Networks
David S. Touretzky
Spring 2004

What Is a Symbolic System?

- *Symbols* are arbitrary tokens.
- They combine to form *structures*.
- *Rules* govern derivation of new structures from old.

That's it!

Most of AI and CS is built on this framework.

Also, all of mathematics and logic.

Inference in Symbolic Systems

A *grammar* defines legal symbol structures.
A *production rule* in an expert system derives new working memory elements from old ones.
Logical deduction (Prolog, theorem provers).
Inductive reasoning (BACON).
Case-based reasoning.
Alpha-beta search.

All are part of the “symbolic paradigm.”

What Do Neural Net Models Offer?

- Feature vector representations.
  - Dot product = similarity measure.
- A different approach to compositional structure.
- Different types of inference:
  - statistical learning (PCA, Boltzmann, Helmholtz, etc.)
  - nonlinear mapping (backprop)
  - parallel constraint satisfaction (Hopfield/Boltzmann)

A statistical, vector-based alternative to the symbolic framework. *More like the brain???
Feature Vector Representations

The representation of “chair” should be more similar to “table” than to “banana”.

Shouldn’t need axioms and rules to “deduce” this!

Ideally, the network should learn its own feature vector representations based on statistics.

See Hinton’s “family trees” model on next slide.

LSA (Latent Semantic Analysis):

Use SVD (like PCA) to derive “meaningful” feature vectors.

Hinton's Family Trees Example

English:

Italian:

Hinton's Family Trees Network

Hidden Layer Representation
Hinton's Family Trees: Generalization Performance

- Domain:
  - 24 people (12 English, 12 Italian)
  - 12 relations (mother, father, husband, wife, son, daughter, aunt, uncle, brother, sister, nephew, niece)
  - 104 true relations over these family trees

- Training set:
  - Trained on 100 relations

- Test set:
  - Tested on 4 relations.
  - Got 3/4 right on one run, 4/4 right on another run.

Latent Semantic Analysis (LSA)

- Method of constructing feature vectors for words.
  - Build a frequency count table of words x contexts. A context can be a sentence, a paragraph, or a document.

- Apply semantic/contextual weighting functions.

- Perform SVD (Singular Value Decomposition).

- Use resulting values as feature vectors for words or documents.

Inference by Nonlinear Mapping

- Elman's SRN (Simple Recurrent Network) predicts next word in a sentence.

- NEITalk text to speech system learns complex mapping.

- Pollack's recurrent network learned regular grammars.

- McClelland & Rumelhart's past tense model.

McClelland & Rumelhart's Past Tense Model

![Past Tense Model Diagram]

Weights (LMS rule)

Regular forms:
- flip --> flipped
- fib --> fibbed
- fit --> fitted

Irregular forms:
- go --> went
- sing --> sang
- buy --> bought

bring --> brought / brang / broughted
**Parallel Constraint Satisfaction**

"The astronomer married the star." (Pollack & Waltz)

![Diagram](image)

"Astronomer primes "stellar body", but "movie star" eventually wins.

---

**More Parallel Constraint Satisfaction**

- Hopfield nets for Traveling Salesman Problem
- Boltzmann machines
- Recurrent backprop nets that learn attractor states
- McClelland & Rumelhart word perception model.

---

**Problems of Compositional Structure**

- Structures are (in principal) unbounded, but vectors have finite length.
- Is dot product still a meaningful similarity measure?
- How can we get the systematicity that comes so easily to symbolic systems?
- Can we learn composite representations? **[Conflict?]**
- Construct new structures on the fly?

---

**Compositional Structure (cont.)**

- Can we do "more" than symbolic systems do with compositional structure?
- The "Turing tarpit": implementation doesn't matter.
- Drew McDermott: the "Touretzky tarpit". Using connectionist architectures to implement symbolic ones (in a possibly inefficient way.)
**Reduced Descriptions (Hinton)**

"Pointer following is expensive, so avoid it."

```
<table>
<thead>
<tr>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Frame</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Handle</td>
</tr>
</tbody>
</table>
```

Does a room have a keyhole? Does a cat have nostrils?

---

**Holographic Reduced Representations (Tony Plate)**

- Use large \( N = 2000 \) elements vectors.
  - Elements drawn independently from \( \text{Normal}(0, 1/N) \)
  - Vectors have roughly zero mean.
  - No similarity: \( a \cdot b = 0 \)
  - Not a feature vector representation.

- Combine vectors using circular convolution operator.

\[
C = a \circ b
\]

- Sort of like “cons” in Lisp: associates \( a \) with \( b \).

---

**Circular Convolution Operator**

\[
z = x \circ y
\]

\[
z_k = \sum_{k=0}^{N-1} x_k y_{(k-k)\mod N}
\]

---

**Circular Convolution Properties**

- **Commutative** \( a \circ b = b \circ a \)
- **Associative** \( a \circ (b \circ c) = (a \circ b) \circ c \)
- **Bilinear** \( a \circ (b + y \cdot c) = b \circ a + y \circ c \)
- **Identity** \( 1 \circ a = a \)
- **Zero** \( 0 \circ a = 0 \)
- **Inverse** \( a^{-1} \circ a = I \)
Inverse of a Vector

If \( \mathbf{x} \) is drawn from \( \text{Normal}(0,1/N) \) then approximate inverse \( \mathbf{x}^T = \mathbf{x}_{\lfloor \text{mod} N \rfloor} \). 

Inverse \( \mathbf{x}^{-1} \) exists under other conditions, but may be numerically unstable.

Taking Apart a Pair

Given: \( \mathbf{a} \odot \mathbf{b} \)  
To get back \( \mathbf{b} \), multiply by \( \mathbf{a}^T \):

\[
\mathbf{a}^T \odot (\mathbf{a} \odot \mathbf{b}) = (\mathbf{a}^T \odot \mathbf{a}) \odot \mathbf{b} \approx \mathbf{1} \odot \mathbf{b} = \mathbf{b}
\]

Since \( \mathbf{a}^T \) only approximates the inverse, we really get

\[
\hat{\mathbf{b}} = \mathbf{b} + \eta.
\]

Use a separate 'cleanup' process to eliminate noise:

\[
\mathbf{b} + \eta \rightarrow \mathbf{b}
\]

How to do cleanup?  
Associative memory  
Nearest-neighbor match

Encoding Propositions

Nouns: John, Mary, Spot  
Verbs: bite, kiss  
Roles: \( \text{bite}_{\text{agt}}, \text{bite}_{\text{obj}}, \text{kiss}_{\text{agt}}, \ldots \)

'Spot bit Mary'

\[
\mathbf{C} = \text{bite} + \text{bite}_{\text{agt}} \odot \text{Spot} + \text{bite}_{\text{obj}} \odot \text{Mary}
\]

Who was the biter? \( \mathbf{C} \odot \text{bite}^T_{\text{agt}} \approx \text{Spot} + \text{noise} \)

Who was bitten? \( \mathbf{C} \odot \text{bite}^T_{\text{obj}} \approx \text{Mary} + \text{noise} \)

What was the action? \( \mathbf{C} \approx \text{bite} + \text{noise} \)

What did Spot do? \( \mathbf{C} \odot \text{spot}^T \approx \text{bite}_{\text{agt}} \)

Convolution Preserves Similarity

\[
\text{sim}(\mathbf{a} \odot \mathbf{b}, \mathbf{c} \odot \mathbf{d}) \approx \text{sim}(\mathbf{a}, \mathbf{c}) \cdot \text{sim}(\mathbf{b}, \mathbf{d})
\]

Superpositions are similar to each of their components, to a decreasing degree as more patterns are added.

\[
\text{sim}(\mathbf{a}, \mathbf{a+b}) > \text{sim}(\mathbf{a}, \mathbf{a+b+c}) > \text{sim}(\mathbf{a}, \mathbf{b})
\]

Similarity can be measured by dot product.
Normalization

All vectors should be normalized to unit length:

\[ z = x \odot y \]

\[ \langle z \rangle = \frac{z}{\|z\|} \]

Advantages:
- unbiased dot product similarity scores
- equal weighting of all component patterns
- reduced noise in decoded slot fillers

Feature Representations

John = animal + human + id_{John}
Jane = animal + human + id_{Jane}
Spot = animal + dog + id_{Spot}

John is more similar to Jane than to Spot.

Recursive Composition

'Spot bit Jane, causing Jane to flee from Spot.'

\[ P_{\text{bite}} = \langle \text{bite} + \langle \text{spot} + \text{jane} \rangle + \text{bite}_{\text{agl}} \odot \text{spot} + \text{bite}_{\text{obj}} \odot \text{jane} \rangle \]

\[ P_{\text{flee}} = \langle \text{flee} + \langle \text{spot} + \text{jane} \rangle + \text{flee}_{\text{agl}} \odot \text{jane} + \text{flee}_{\text{from}} \odot \text{spot} \rangle \]

\[ P = \langle \text{cause} + (P_{\text{bite}} + P_{\text{flee}}) + \text{cause}_{\text{avc}} \odot P_{\text{bite}} + \text{cause}_{\text{csq}} \odot P_{\text{flee}} \rangle \]

P contains a 'reduced description' of Jane being bitten.

Analogy Retrieval

Describe a situation in terms of predicates and objects.

'Spot bit Fred' = bit(Spot, Fred)

Find an 'analogous' situation:

Same predicate? bit(Fido, John)
Same objects? saw(Fred, Spot)
Same predicate & objects? bit(Fred, Spot)
Same object-roles? bit(Spot, Jane)
**Analogical Retrieval**

- Ken Forbus' MAC/FAC is a symbolic analogy-making program.
- Use content vectors to summarize a description by counting the # of occurrences of each object and predicate.
- Dot product of content vectors heuristically estimates similarity.
- High scoring descriptions are then explored symbolically to verify the analogy.
- But content vectors ignore structural similarity: “John bit Fido” treated the same as “Fido bit John.”

**A Better Retrieval Heuristic**

Holographic Reduced Representations (HRRs) encode structural similarity:

\[ \langle \text{bite} + \langle \text{John} + \text{Fido} \rangle + \text{bite}_{\text{agt}} \otimes \text{John} + \text{bite}_{\text{obj}} \otimes \text{Fido} \rangle \]

---

**Structural Similarity**

People are sensitive to structural similarity.

“Fido bit John, causing John to flee from Fido.”

**LS (Literal Similarity)**

Fido bit John, causing John to flee from Fido.

**SF (Surface Features)**

John fled from Fido, causing Fido to bite John.

**CM (cross-mapped analogy) – roles switched**

Fred bit Rover, causing Rover to flee from Fred.

**Structural Similarity (cont.)**

**AN (analogy)**

Mort bit Felix, causing Felix to flee from Mort.

**FOR (first-order relations only)**

Mort fled from Felix, causing Felix to bite Mort.
How Humans Handle Similarity

For LTM retrieval in humans:

\[
LS > CM \geq SF > AN \geq FOR
\]

<table>
<thead>
<tr>
<th>Episodes in long-term memory</th>
<th>Commonalities with probe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>E_LS: Fido bit John, causing John to flee from Fido.</td>
<td>✓</td>
</tr>
<tr>
<td>E_SF: John fled from Fido, causing Fido to bite John</td>
<td>✓</td>
</tr>
<tr>
<td>E_CM: Fred bit Rover, causing Rover to flee from Fred.</td>
<td>✓</td>
</tr>
<tr>
<td>E_AN: Mort bit Felix, causing Felix to flee from Mort.</td>
<td>✓</td>
</tr>
<tr>
<td>E_FOR: Mort fled from Felix, causing Felix to bite Mort.</td>
<td>✓</td>
</tr>
</tbody>
</table>

Summary

- Connectionist symbol processing proposes a statistical, vector-based alternative to the "classic" symbolic paradigm.
- Issues:
  - feature vector encoding
  - compositionality (difficult)
  - nonlinear mapping
  - parallel constraint satisfaction
- Interesting ideas in all four areas.
- No one has figured out how to combine all four. Still in the very early days of this research.