Building a UTC from the brain up

Thad Polk
Department of Psychology
University of Michigan

Collaborators

Chuck Behensky
Eric Freedman
Rich Gonzalez
Matthew Jones
Rick Lewis
Pat Simen
Ed Smith
Scott Wallace
Brahm Windeler

UTCs: The Soar Approach

Soar's development driven by functional constraints

- Partly reflects its history as an AI system
- But also responds to flexibility of human behavior:

> If there is any property that a theory of cognition must explain it is how intelligence is actually possible. *Necessary* characteristics are well and good, but they are substantially less than half the story. *Sufficiency* is all-important. Intelligence itself is a sufficiency of capability. To be intelligent is to be able to do certain things… Unless the theory can actually demonstrate these capabilities, it cannot claim that is has explained intelligence.
>
> (Newell, 1990)

UTCs: The Soar Approach

Can view basic assumptions as providing functionality:

- Symbol systems: Universal computation
- Productions: Flexible, rather than fixed, control
- Problem space search: Handling uncertain problems
- Subgoaling: Handling lack of knowledge
- …

Essentially a *top-down* approach to UTC development

- TOP: Functionality/behavior you need to explain
- Work DOWN to a theory that provides that functionality
- Use that theory to explain cognitive phenomena

Building UTCs from the brain up

Bottom-up approach is complementary:

- BOTTOM: basic facts about neural computation
- Work UP to UTC by abstracting out critical principles
- Use those principles to explain cognitive phenomena

Connectionist modeling often has this goal

- But models make very different assumptions
- Don't use a common architecture

Overview

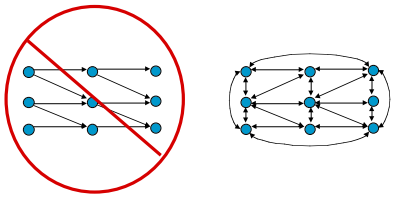
Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

Plan:

- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control

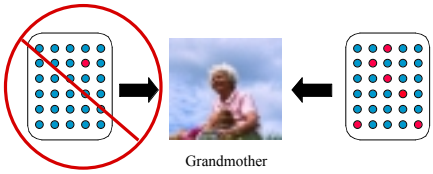
Neural Computation: Assumptions

1. Neural processing is *recurrent*, not feed-forward



Neural Computation: Assumptions

2. Neural representations are *distributed* across a population of neurons, not localized to single cells

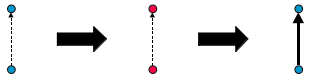


Grandmother

Neural Computation: Assumptions

3. Neural learning is *correlation-based*

Hebbian learning: "Neurons that fire together, wire together"



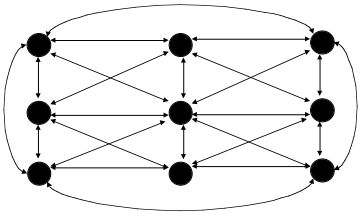
Overview

Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

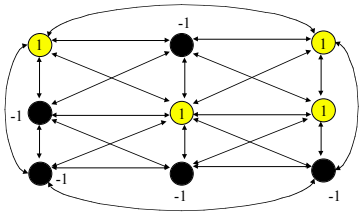
Plan:

- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control

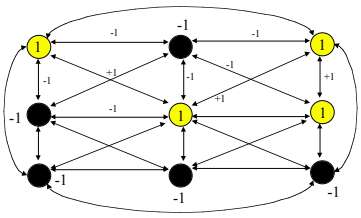
Recurrent Connectivity



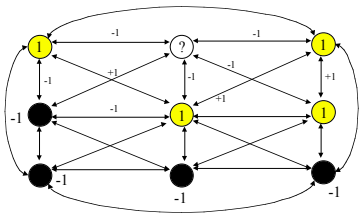
Distributed Representation



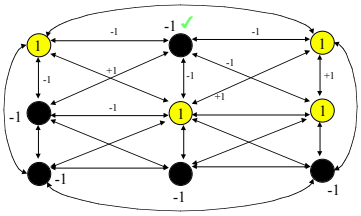
Correlation-based Learning



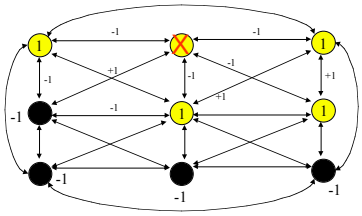
Emergent Properties: Completes Partial Patterns



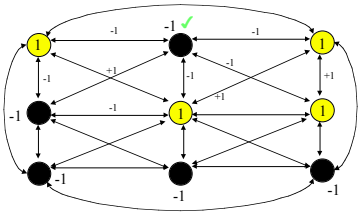
Emergent Properties: Completes Partial Patterns



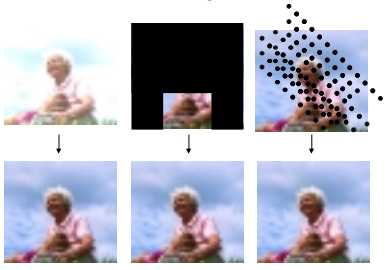
Emergent Properties: Cleans Up Noisy Patterns



Emergent Properties: Cleans Up Noisy Patterns

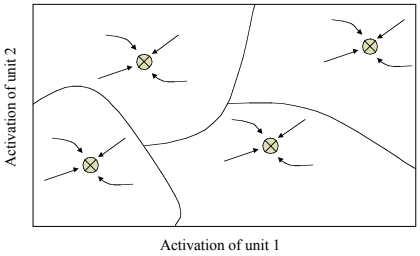


Emergent Properties: Associative Memory

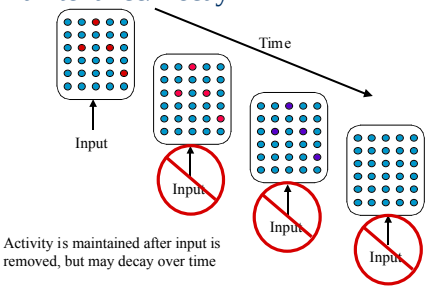


Attractors & Basins of Attraction

(Hopfield, 1982; 1984)

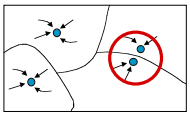


Emergent Properties: Maintenance/Decay



Activity is maintained after input is removed, but may decay over time

Emergent Properties: Similarity-Based Interference

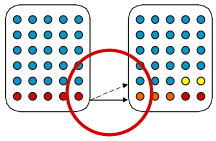


Similar attractors are close to each other

So noisy input can converge on wrong attractor

Similar patterns overlap each other

So Hebbian learning partially associates cues with incorrect items that are similar to target



The Attraction of Attractors

Any network that satisfies these assumptions (recurrence, distributed repr, Hebbian learning) will behave like an attractor net

- Associative memory, reverberatory maintenance, time-based decay, similarity-based interference…

Most of neocortex satisfies these assumptions

So most of neocortex should behave like attractor nets

- Properties should explain many cognitive phenomena
- Beginning of a UTC built from the brain up

Overview

Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

Plan:

- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control

Basic Questions about WM

How is information maintained after input is removed?

Why does information decay over time?

How does new information interfere with older information?

Attractor-based Answers

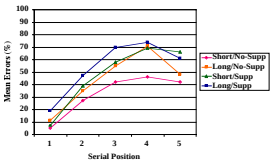
How is information maintained after input is removed?
- Reverberatory activity

Why does information decay over time?
- Reverberatory activity insufficient to maintain itself forever

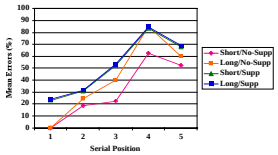
How does new information interfere with older information?
- Fall into nearby, but wrong, attractor basin
- Because patterns overlap, partially associate cues with both

Simulations

Baddeley et al. (1984)

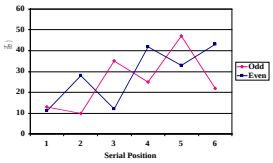


Simulation

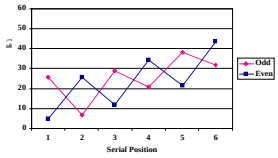


Simulations

Baddeley et al. (1968)



Simulation



Overview

Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

Plan:
- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control

Knowledge Representation

How do we represent concepts and their relationships?

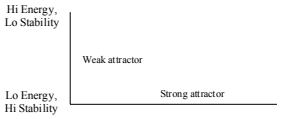
One view: Similar concepts have similar representations, dissimilar concepts have dissimilar
- E.g., Similar concepts are “nearby” in semantic space
- E.g., Similar concepts use overlapping distributed codes

Problem: Similarity judgments are asymmetric
- People judge North Korea to be more similar to China, than China is to North Korea
- More generally, non-protoypical concepts judged more similar to prototypical concepts than vice versa. Why?

Attractor-Based Explanation of Similarity Asymmetries

Prototypical concepts have stronger (more stable) attractors than do non-prototypical concepts

Easier to compare a weaker attractor to a stronger attractor (downhill) than vice versa (uphill):



Prediction

Exposure frequency (among other factors) increases attractor strength (via Hebbian learning)

Might therefore be able to *manipulate* similarity asymmetry by presenting stimuli with different frequency

- Should even work with simple perceptual stimuli (color patches) rather than complex concepts (countries, animals…) which are harder to control

Experiment: Materials

Blue Patches

Green Patches

Experiment: Pre-test

To establish baseline ratings/asymmetries for each participant:

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

. .
. .
. .

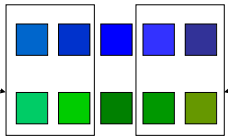
(All pairings in both directions)

Experiment: Training

Which is bigger? ■ ■

Which is bigger? ■ ■

1/2 subjects saw these hues most



1/2 subjects saw these hues most

Experiment: Post-test

To compare against pre-test baseline for each participant:

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

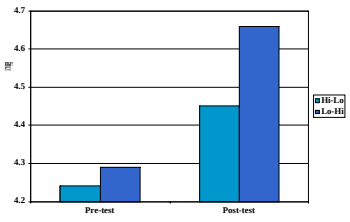
How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

How similar is ■ to ■ ? (0: very dissimilar, 9: very similar)

. .
. .
. .

Prediction: Relative to the pre-test, patches that were low freq will be rated more similar to patches that were high freq than vice versa

Results



Overview

Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

Plan:

- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control

Higher Cognition

Models of higher cognition (planning, problem solving) typically based on symbolic processing

- Symbols, productions, sequential behavior, search…

Most neural net models have little to say

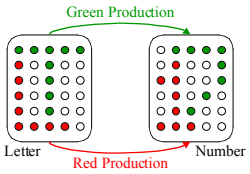
Do attractor nets have anything to contribute?

Mapping Productions Onto Attractors

Attributes = attractor nets/layers
Values = attractor patterns in the specified layer
Productions = associations/connections between layers

IF
Letter T
THEN
Number 7

IF
Letter L
THEN
Number 1



Mapping Productions Onto Attractors

Attributes = attractor nets/layers
Values = attractor patterns in the specified layer
Productions = associations/connections between layers

We've actually implemented this mapping in Lisp:

Input: Simplified production rules

Output: Matlab code that implements a set of attractor nets that behave like the production system (usually)

Can thus use the same attractor architecture for some higher cognitive tasks

Issues Raised by the Mapping

A number of features of production systems DON'T map easily:

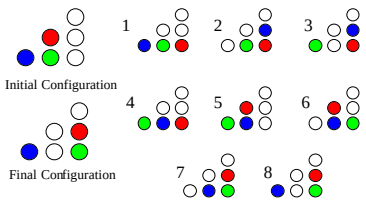
- Variables, independence/modularity of different productions, all-or-none matching, ...

Possible responses:

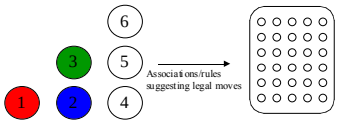
- Neural nets lack critical functionality for modeling cognition
  - Need to work on increasing their functionality
- Production systems have too much functionality
  - Might work with less powerful production systems that map more naturally
- Both are probably reasonable

Tower of London example

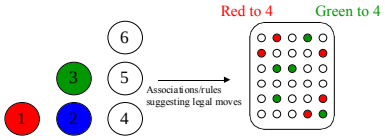
A simplified version of Tower of Hanoi



An Attractor Net for Blind Search

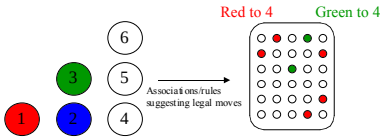


An Attractor Net for Blind Search



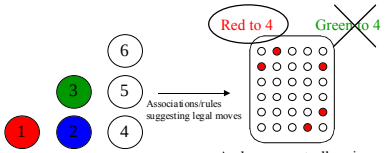
Initially, all legal moves get activated...

An Attractor Net for Blind Search



But attractors compete...

An Attractor Net for Blind Search



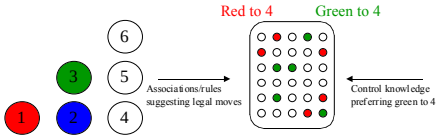
And one eventually wins.
(A kind of conflict resolution)

Adding Control Knowledge

Additional control knowledge can bias search

Adding Control Knowledge

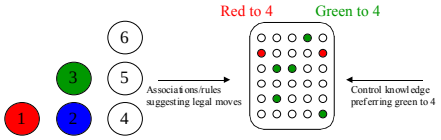
Additional control knowledge can bias search



Initially, all legal moves get activated...

Adding Control Knowledge

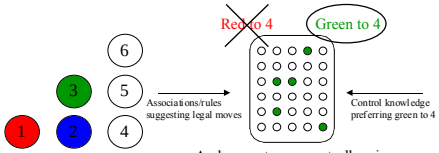
Additional control knowledge can bias search



But control knowledge biases competition...

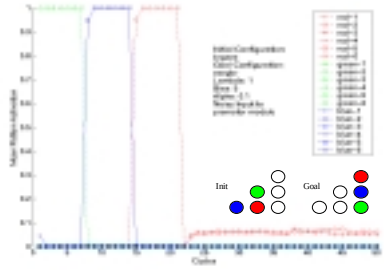
Adding Control Knowledge

Additional control knowledge can bias search



And correct one eventually wins.
(A kind of preference scheme)

Example



Overview

Goal: Derive architecture from facts about neural computation and apply it to cognitive phenomena

Plan:

- Simple assumptions about neural computation
- Emergent computational properties of the assumptions
- Relevance to cognition:
  - Working memory
  - Mental representation
  - Higher cognition & control