Cognitive Control as a Gated Cortical Network

Jared Sylvester, Jim Reggia and Scott Weems
Depts. of Computer Science & Psychology, UMIACS, CASL
University of Maryland, College Park, MD

6 Nov 2011
Goal

- Generalizable model of cognitive control
  - Learned, not hard-wired into architecture
- Base behavior on memory contents
- Two type of memory/learning:
  - Memory of perceptual stimuli
  - Memory of task procedures
- Biological inspiration of our approach:
  - Network of regions, recurrent attractor nets, gating, distributed representations, Hebbian learning
Attractor Net Memories

\[ w_{ij}^t = w_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^t (1 - \delta_{ij}) \]

- Stored patterns are attractors
  - Form auto-associative memory
- But fixed-point attractors
  - Network gets “stuck” in attractor basin
Sequentially Visit Attractor States

- Dynamic thresholds
  - Increase when node’s state remains unchanged
  - Harder for node to stay in the same state

For details, see:

Visiting Attractors in Order

- Asymmetric weights
  - Correlate activity with other nodes’ previous activity
    \[ v_{ij}^t = (1 - k_D)v_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^{t-1} \]
  - Network transitions between attractors in order

For details, see:
Adding Cognitive Control

- Modeled Running Memory Span task
  - Can match human behavioral results
  - But all control was exogenous
- For internal control, use multiple networks
  - Network of attractor networks
  - Controlled by gating
  - Learn processing of sequences
Control Mechanism

- Built around attractor networks
- Trained prior to task beginning
- Directs the model by operating gates
- Core is “instruction memory”
Control Mechanism

- Built around attractor networks
- Trained prior to task beginning
- Directs the model by operating gates
- Core is “instruction memory”
  - Stores sequence of steps to do subtasks
  - Multiple sequences stored simultaneously
  - Divided into cue & response sections
Instruction Memory

Distributed ‘cue’ pattern

Make tea
Instruction Memory

Distributed ‘cue’ pattern

- Make tea

Distributed ‘response’ patterns

- Boil water
- Steep tea bag
- Add sugar
Instruction Memory

Distributed ‘cue’ pattern | Distributed ‘response’ patterns
---|---
Make tea | Make tea
Boil water | Steep tea bag
Add sugar |
Instruction Memory

Distributed ‘cue’ pattern

Make tea ↔ Boil water

Distributed ‘response’ patterns

Make tea ↔ Steep tea bag

Make tea ↔ Add sugar

\[
 w_{ij}^t = (1 - k_{CTRL})w_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^t (1 - \delta_{ij})
\]
Instruction Memory

Distributed ‘cue’ pattern                  Distributed ‘response’ patterns

Make tea                                  Boil water

Make tea                                  Steep tea bag

Make tea                                  Add sugar

\[ v_{ij}^t = (1 - k_{CTRL})v_{ij}^{t-1} + \frac{1}{N} a_i^t a_j^{t-1} \]
Task: “Store/Recognize”

- Visual stimulus
- Mode input

- “load” – add visual stimulus to W.M.; output “complete” when done
- “evaluate” – is visual stimulus in W.M?
  - If so, output “present.”
  - If not, add it to W.M. & output “not present”

<table>
<thead>
<tr>
<th>Mode input</th>
<th>load</th>
<th>load</th>
<th>evaluate</th>
<th>evaluate</th>
<th>evaluate</th>
<th>evaluate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual input</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>X</td>
<td>Y</td>
<td>X</td>
</tr>
<tr>
<td>Correct response</td>
<td>complete</td>
<td>complete</td>
<td>present</td>
<td>not present</td>
<td>not present</td>
<td>present</td>
</tr>
</tbody>
</table>
How to add item to working memory
Visual Input

Working Memory

Memory Input Gate

Memory Training Gate

Compare Output Gate

Control Input Gate

Compare Module

Control Module

Complete Output

Present Output

Not Present Output

Mode Input

(LOAD)
Visual Input

Working Memory

Memory Input Gate

Memory Training Gate

Compare Output Gate

Control Input Gate

Compare Module

Control Module

Complete Output

Present Output

Not Present Output

Mode Input

(LOAD)
$$w_{ij}^t = (1 - k_{WM})w_{ij}^{t-1} + \frac{1}{N}a_i^t a_j^t (1 - \delta_{ij})$$
Accuracy with Varying Instruction Memory Decay

$k_{CTRL}$ is the “decay” rate in the controller’s instruction memory layer

Negative decay (i.e. gain) performs better. 

$k_{CTRL}$ is the “decay” rate in the controller’s instruction memory layer. Negative decay (i.e. gain) performs better.
Conclusion

- Modeled proof-of-concept task
- Behavior determined by control module’s memory contents
  - i.e., learned, not hard-wired
  - $n$-Back model done
- Biologically plausible
  - Network of regions, recurrent attractor nets, gating, distributed representations, Hebbian learning