Simulating Mirror Neurons

Kate Derosier and Jackie Kay

Presented by: Enes Özipek
OUTLINE

- Background and Related Work
- ARMS
- Experiments
- Results
- Conclusion
Mirror Neurons

(A) Monkey at rest

(B) Grasping execution

(C) Observation of grasping movements

A mirror neuron is in a resting state

A mirror neuron fires

A mirror neuron fires

(No electrical signals)
Properties

Visual and Motor Stimuli

Goal-oriented

PMVr (premotor ventral): view-invariant

Superior temporal sulcus (STS)

Similar architecture in humans.
HAMMER

- Inverse and Forward models/pairs
- The most salient part of the image
- Arbitration module
- Associated forward model predicts
- Prediction error calculated
- Confidence values updated

The most salient part of the image
Advantages

- Recognizing a known action

Disadvantages

- Hard-coded
- Not biologically plausible
Self-perception

- Might be functional in real robots.
- Not suitable for simulating MNS.
ARMS (Adaptive Robotic Mirror System)

Figure 1: Schematic of the ARMS system.
Simulated iCub performing a power grasp, a side grasp, and a precision grasp (L to R)
ARMS vs Previous Work

✓ Reinforcement learning
✓ Offline/Online learning
Merge Self-Organizing Maps

How do SOMs work?

1. SOM weights initialized
2. Take random input and calculate Euclidean distance.
3. Find BMU and update the weights (including neighbours)
MSOM Update Rules

\[ q(t) = (1 - \beta)w^i_{u(t-1)}(t) + \beta w^c_{u(t-1)}(t) \]

\[ w^i_v(t) = w^i_v(t) + \gamma \Phi(u, v, t)(s(r) - w^i_v(t)) \]

\[ w^c_v(t) = w^c_v(t) + \gamma \Phi(u, v, t)(q(t) - w^c_v(t)) \]

\[ y_v(t) = e^{-d_v(t)} \]

\[ d_v(t) = (1 - \alpha\|s(r) - w^i_v(t)\|^2 + \alpha\|q(t) - w^c_v(t)\|^2). \]
Bidirectional activation-based learning

• Not using traditional error backpropogation
• Two-phase update
• Three layer network

Figure 3: A sample BAL network, with forward activations shown in red and backward activations shown in orange. Each connection is labeled with the input the connection passes to the postsynaptic unit. In addition, each unit has a bias, which is omitted for simplicity.
Update Rules

The network weights are updated using only local variables.
- \( a_p \) : presynaptic unit activation
- \( a_q \) : post synaptic unit activation
- learning rate: \( \lambda \)

### Table: Update Rules

<table>
<thead>
<tr>
<th>Phase</th>
<th>Layer</th>
<th>Net Input</th>
<th>Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>x</td>
<td>( \eta_i^F = \sum_i w_{ij,i} x_i^F )</td>
<td>( x_i^F = \text{stimulus} )</td>
</tr>
<tr>
<td>F</td>
<td>h</td>
<td>( \eta_i^F = \sum_j w_{ij,k} h_j^F )</td>
<td>( h_i^F = \sigma(\eta_i^F) )</td>
</tr>
<tr>
<td>F</td>
<td>y</td>
<td>( \eta_k^F = \sum_j w_{jk,i} h_j^F )</td>
<td>( y_k^F = \sigma(\eta_k^F) )</td>
</tr>
<tr>
<td>B</td>
<td>y</td>
<td>( \eta_i^B = \sum_i w_{ij,i} y_i^B )</td>
<td>( y_i^B = \text{stimulus} )</td>
</tr>
<tr>
<td>B</td>
<td>h</td>
<td>( \eta_i^B = \sum_k w_{kj,k} y_k^B )</td>
<td>( h_i^B = \sigma(\eta_i^B) )</td>
</tr>
<tr>
<td>B</td>
<td>x</td>
<td>( \eta_i^B = \sum_j w_{ji,j} h_j^B )</td>
<td>( x_i^B = \sigma(\eta_i^B) )</td>
</tr>
</tbody>
</table>

\[
\delta w_{pq}^F = \lambda a_p^F (a_q^B - a_q^F)
\]

\[
\delta w_{pq}^B = \lambda a_p^B (a_q^F - a_q^B)
\]

Differences in net inputs times the activation function derivative can be approximated by differences in the activation values.
Experiments

- Ten variations on each of the three grasp types
- Collect motor and visual data
- Each input trajectory flattened and binarized
  - Take max and min values
  - Calculate the midpoint
  - Set coordinate value
    - 0 if less than the midpoint
    - 1 if greater than the midpoint
- Calculate a sphere in n-dimensional space (10 variations)
- Evaluate the model
Training Configurations

- Both MSOMs -> 3x3 g, BAL -> 9 m, 7 h, 9 v, Iteration: 5k
- Motor MSOM -> 12x12 g, Visual MSOM -> 16x16 g, BAL -> 144 m, 160 h, 256 v, iteration: 1k
- Motor MSOM -> 12x12 g, Visual MSOM -> 16x16 g, BAL -> 144 m, 160 h, 256 v, iteration: 5k

- g: Grid
- m: motor units
- h: hidden units
- v: visual units
## Results

<table>
<thead>
<tr>
<th></th>
<th>Forward (motor to visual)</th>
<th>Backward (visual to motor)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power</td>
<td>Precision</td>
</tr>
<tr>
<td>Small MSOMs, long training</td>
<td>Power</td>
<td>3 (0)</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>3 (0)</td>
</tr>
<tr>
<td></td>
<td>Side</td>
<td>3 (0)</td>
</tr>
<tr>
<td>Big MSOMs, short training</td>
<td>Power</td>
<td>0 (0)</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0 (0)</td>
</tr>
<tr>
<td></td>
<td>Side</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Big MSOMs, long training</td>
<td>Power</td>
<td>0 (3)</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0 (3)</td>
</tr>
<tr>
<td></td>
<td>Side</td>
<td>0 (3)</td>
</tr>
</tbody>
</table>
Conclusions

ARMS failed to learn any association

It suffers underfitting

Third configuration took 48 hours.

Believe ARMS would perform better when trained more.
References