CISC 3250
Systems Neuroscience

Neuroplasticity: Learning in Neurons

Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

Review of weights

RI(t)=\sum_k w_k \alpha_k(t)

Weights indicate
• Connection (0 or not)
• NT effect
  – w>0 excitatory
  – w<0 inhibitory
• Magnitude of impact of input

Association

We recall information through associations with other information
• Pneumonics:
  Roy G. Biv
  Please Excuse My Dear Aunt Sally (PEMDAS)
• Memories of experiences:
  Lake -> Summer vacation 2014
  Dealy -> Final exam Fall 2013
• Complex objects
  ::Bark:: -> Dog, fur, happy/fear

Features of associators

• Pattern completion/generalization

• Recognizing prototypes
  – Neuron firing for common combinations
• Fault tolerance
  – Selected dendrites miss input, post-synaptic neuron still fires
**Pattern completion**

Activation requires only a subset of desired inputs.

<table>
<thead>
<tr>
<th>r (^{in})</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>leg1</td>
<td>0.5</td>
</tr>
<tr>
<td>leg2</td>
<td>0.5</td>
</tr>
<tr>
<td>body</td>
<td>0.5</td>
</tr>
<tr>
<td>ears</td>
<td>0.5</td>
</tr>
<tr>
<td>mouth</td>
<td>0.5</td>
</tr>
<tr>
<td>tail</td>
<td>0.5</td>
</tr>
</tbody>
</table>

How many inputs needed to fire?

Define input \( h = \sum_k w_k r_k^{in} \)

Neuron fires at rate \( r_{out}=1 \) when \( h > 1.2 \)

Assume \( r^{in}=1 \) when active, \( r^{in}=0 \) when inactive.

**Prototypes**

Activation requires all desired inputs.

<table>
<thead>
<tr>
<th>r (^{in})</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>lights</td>
<td>?</td>
</tr>
<tr>
<td>wheel</td>
<td>?</td>
</tr>
<tr>
<td>trunk</td>
<td>?</td>
</tr>
<tr>
<td>door</td>
<td>?</td>
</tr>
<tr>
<td>window</td>
<td>?</td>
</tr>
</tbody>
</table>

**0.3 < w ≤ 0.375**

\( h = \sum_k w_k r_k^{in} \)

Neuron fires at rate \( r_{out}=1 \) when \( h > 1.5 \)

**Assume inputs at \( r^{in}=1 \) or \( r^{in}=0 \)**

**Example 1:**

\( \text{leg1} = 1; \quad \text{leg2} = 1; \quad \text{body} = 1; \)
\( \text{ears} = 1; \quad \text{mouth} = 0; \quad \text{tail}=0; \)
\( h=0.5+0.5+0.5+0.5+0+0 \rightarrow h=2 \)
\( r^{out}=g(h)=g(2) \rightarrow r_{out}=1 \)

**Example 2:**

\( \text{leg1} = 0; \quad \text{leg2} = 0; \quad \text{body} = 0; \)
\( \text{ears} = 0; \quad \text{mouth} = 1; \quad \text{tail}=1; \)
\( h=0+0+0+0.5+0.5 \rightarrow h=1 \)
\( r^{out}=g(h)=g(1) \rightarrow r_{out}=0 \)

**Define input \( h = \sum_k w_k r_k^{in} \)**

**Assume \( r^{in}=1 \) when active, \( r^{in}=0 \) when inactive.

**Example 1:**

\( \text{example} 1: \)
\( \text{leg1} = 1; \quad \text{leg2} = 1; \quad \text{body} = 1; \)
\( \text{ears} = 1; \quad \text{mouth} = 0; \quad \text{tail}=0; \)
\( h=0.5+0.5+0.5+0.5+0+0 \rightarrow h=2 \)
\( r^{out}=g(h)=g(2) \rightarrow r_{out}=1 \)

**Example 2:**

\( \text{example} 2: \)
\( \text{leg1} = 0; \quad \text{leg2} = 0; \quad \text{body} = 0; \)
\( \text{ears} = 0; \quad \text{mouth} = 1; \quad \text{tail}=1; \)
\( h=0+0+0+0.5+0.5 \rightarrow h=1 \)
\( r^{out}=g(h)=g(1) \rightarrow r_{out}=0 \)
Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work

Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work

Learning to associate: Conditioning

Associating both smell and whistle with food

- **Unconditioned stimulus**: smell – already associated with food
- **Conditioned stimulus**: whistle – indicates food coming

Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work

Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work

Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work

Fault tolerance

Activation requires only a subset of desired inputs

How many inputs needed to fire?

In this one case, we assume some of the inputs (e.g., from moo) can fail to communicate over synapse, while other copies of input still work fine. Only need one moo input to work
Two forms of plasticity

- **Structural plasticity**: generation of new connections between neurons
- **Functional plasticity**: changing strength of connections between neurons

**Hebbian plasticity**: “cells that fire together, wire together”

Chemical level: NT receptors

Increase weight by improving NT detection

Post-synaptic:
- Insert more receptors into dendrite membrane
- Improve performance of receptors

Pre-synaptic:
- Increase amount of NT released

Marr’s levels of analysis

- **Computational theory**: Learn associations among sensations
- **Representation and algorithm**: Associate each sense with set of neural outputs, adjust weights on these outputs into another neuron
- **Hardware implementation**: Insert/remove NT receptors from dendrites

Math of Hebbian rate learning

“Cells that fire together, wire together”

\[ \Delta w_{ij} = \epsilon(w)r_i r_j \]

i.e.: \( \Delta w_j = \epsilon(w)r_{out} r_{in} \)

\( \epsilon \): learning speed

Time (sec)
Using the learning rule

Define input \( h = \sum_k w_k r_k^{in} \)

Neuron fires at rate \( r_{out} = 1 \) when \( h > 1 \)

\[
\varepsilon(w) = \begin{cases} 
-0.5 & w < 0 \\
0.5 & w \geq 0 
\end{cases}
\]

\[
\Delta w_j = \varepsilon(w) r_{out} r_j^{in}
\]

Weight control and decay

- Synaptic weights are finite
- Propose learning rules that keep weights bounded

\[
\Delta w_{ij} = r_i r_j - c w_{ij}
\]

\[
\Delta w_j = r_{out} (r_j - w_j) \quad \text{Willshaw}
\]

- Or, preserve total synaptic weight across network: “normalization”

\[
w_j \leftarrow \frac{w_j}{\sum_k |w_k|}
\]

Side note:

Weight control with Hebb

\[
\Delta w_{ij} = \varepsilon(w) r_{out} r_j^{in}
\]

- Higher weight – suppressed weight update
Using weight control and decay

Define input $h = \sum_k w_k r_k^{in}$
Neuron fires at rate $r_{out}=1$ when $h > 1$

$$\Delta w_j = r_{out}(r_j - w_j)$$

<table>
<thead>
<tr>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$r_{out}=1$ $r_{out}=1$ $r_{out}=1$ $r_{out}=1$

Using weight control and decay

Define input $h = \sum_k w_k r_k^{in}$
Neuron fires at rate $r_{out}=1$ when $h > 0.5$

$$e(w) = \begin{cases} 
-0.5 & w < 0 \\
0.5 & w \geq 0 
\end{cases}$$

$$w_j \leftarrow \frac{w_j}{\sum_k |w_k|}$$

<table>
<thead>
<tr>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
<th>$r_{in}$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$r_{out}=1$ $r_{out}=1$ $r_{out}=1$ $r_{out}=1$
Hebb + normalization

Step 1: Compute output at time $t$

Step 2: Use Hebb learning based on $r_{out}^t$, $w_j^t$, $r_j^t$ to find new $w_j^{t+1}$

Step 3: Divide new $w_j^{t+1}$'s by $\sum_k |w_k^{t+1}|$ so new $|w_j|$'s add to 1

AI Neural Net Learning

Computed output: $g_{sigmoid}(\sum_i w_j r_j^{in})$

Desired output: $y_{out} \in [0, 1]$

$\Delta w_j = \epsilon r_j^{in} r_{out} (y_{out} - r_{out})(1 - r_{out})$

Using weights

Learning weights