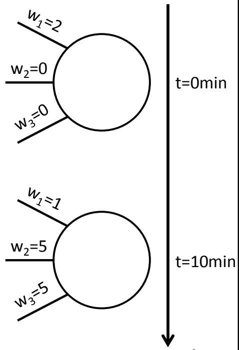
CISC 3250
Systems Neuroscience

Neuroplasticity: Learning in Neurons

Professor Daniel Leeds
dleeds@fordham.edu
JMH 332



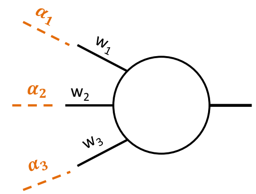
1

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



2

Association

We recall information through associations with other information

- Pneumonics:
  - Roy G. Biv
  - Please Excuse My Dear Aunt Sally () Exp x / + -
- Memories of experiences:
  - Lake -> Summer vacation 2014 
  - Dealy -> Final exam Fall 2013 
- Complex objects
  -  ::Bark:: -> Dog, fur, happy/fear

3

Features of associators

- Pattern completion/ generalization 
- Learning prototypes 
  - Neuron firing for common combinations
- Fault tolerance
  - Selected dendrites miss input, post-synaptic neuron still fires

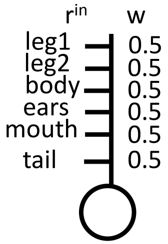
4

Assume inputs at r^{in}=1 or r^{in}=0

Pattern completion

dog detector

Activation requires only a subset of desired inputs



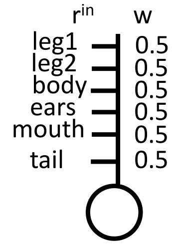
How many inputs needed to fire?

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

5



Example 1:

leg1 = 1; leg2 = 1; body = 1;
ears = 1; mouth = 0; tail=0;
h= 0.5+0.5+0.5+0.5+0+0 -> **h=2**
r^{out}=g(h)=g(2) -> **r^{out} = 1**

Example 2:

leg1 = 0; leg2 = 0; body = 0;
ears = 0; mouth = 1; tail=1;
h= 0+0+0+0+0.5+0.5 -> **h=1**
r^{out}=g(h)=g(1) -> **r^{out} = 0**

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

Neuron fires at rate r^{out}=1 when $h > 1.5$

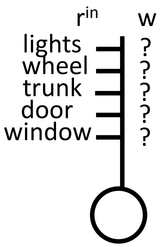
Assume r^{in}=1 when active, r^{in}=0 when inactive

6

Prototypes

car detector

Activation requires all desired inputs

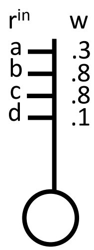


$\mathbf{0.3 < w \leq 0.375}$

$h = \sum_k w_k r_k^{in}$

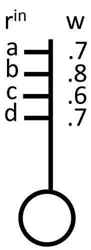
Neuron fires at rate r^{out}=1 when $h > 1.5$

8



Best labeled as prototype detector for b and c together

b and c are the two desired inputs



Best labeled as pattern completion (just need any three of the inputs to fire)

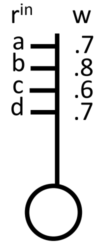
$h = \sum_k w_k r_k^{in}$

Neuron fires at rate r^{out}=1 when $h > 1.5$

9

Example for pattern completion:
a=0; b=1;
c=1; d=0;
h=0+0.8+0.6+0 -> **h=1.4**
r^{out}=g(1.4) -> **r^{out}=0**

Example 2:
a=1; b=1;
c=0; d=1;
h=0.7+0.8+0+0.7 -> **h=2.2**
r^{out}=g(2.2) -> **r^{out}=1**



Best labeled as pattern completion (just need any three of the inputs to fire)

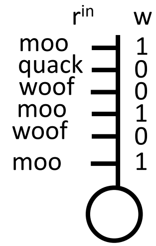
$$h = \sum_k w_k r_k^{in}$$

Neuron fires at rate r^{out}=1 when $h > 1.5$

Fault tolerance

cow detector

Activation requires only a subset of desired inputs



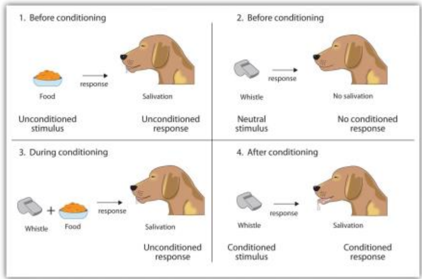
How many inputs needed to fire?

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

$$h = \sum_k w_k r_k^{in}$$

Neuron fires at rate r^{out}=1 when $h > 1.5$

Learning to associate: Conditioning



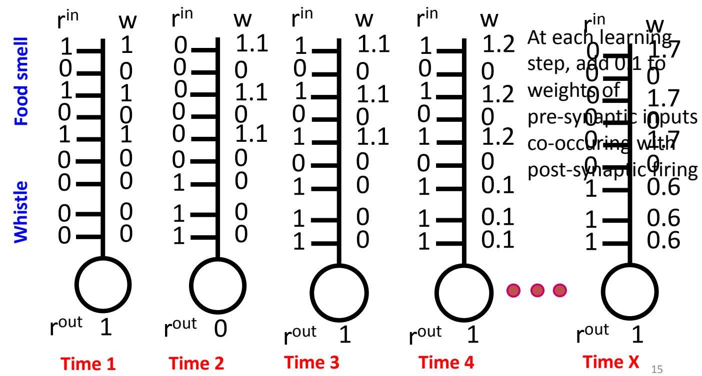
Associating both smell and whistle with food

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

Computing level: Associator network

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

Neuron fires at rate r^{out}=1 when $h > 1.5$



Two forms of plasticity

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

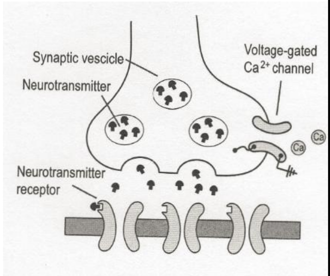
Hebbian plasticity:
"cells that fire together, wire together"

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

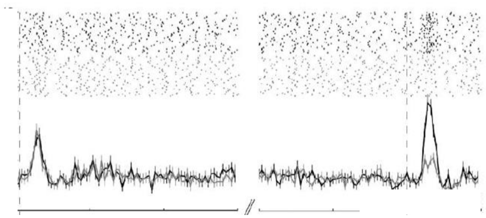
Math of Hebbian rate learning

"Cells that fire together, wire together"

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

r_i --- r^{out}
r_j --- r^{in}
ϵ learning speed

i.e.: $\Delta w_{ij} = \epsilon(w) r^{out} r_j^{in}$



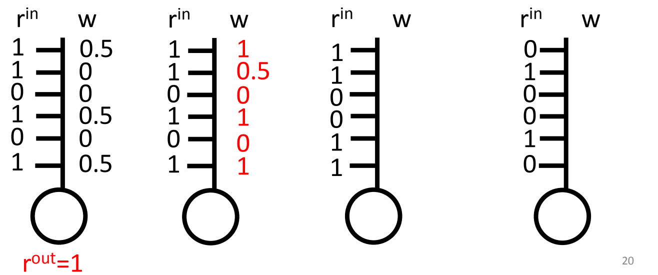
Time (sec)

Using the learning rule

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

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

$$\boldsymbol{\Delta w_{ij} = \epsilon(w) r^{out} r_j^{in}}$$



20

Some more math

$$w_j^{t=2} = w_j^{t=1} + \Delta w_j^{t=1}$$
$$\Delta w_j^{t=1} = \epsilon\left(w_j^{t=1}\right) \times r_{out}^{t=1} \times r_1^{t=1}$$

$$w_1^{t=2} = w_1^{t=1} + \Delta w_1^{t=1}$$
$$= w_1^{t=1} + \epsilon(w_1^{t=1}) \times r_{out}^{t=1} \times r_1^{t=1}$$
$$= 0.5 + \epsilon(0.5) \times 1 \times 1 = 0.5 + 0.5 = 1$$

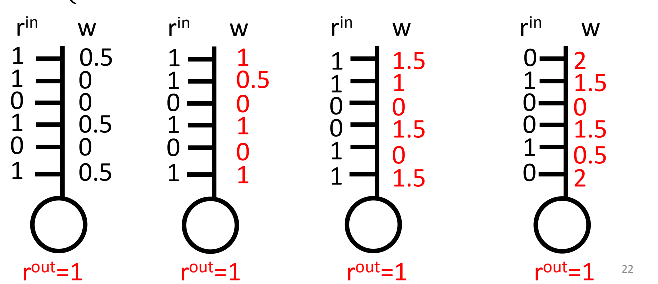
21

Using the learning rule

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

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

$$\boldsymbol{\Delta w_{ij} = \epsilon(w) r^{out} r_j^{in}}$$



22

Weight control and decay

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

$$\Delta w_{ij} = r_i r_j - c w_{ij}$$
$$\Delta w_j = r_{out}(r_j - w_j) \qquad \textit{Willshaw}$$

- Or, preserve total synaptic weight across network: **"normalization"**

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

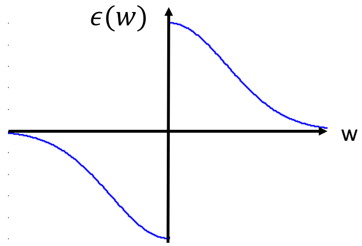
23

Side note:

Weight control with Hebb

$\Delta w_{ij} = \epsilon(w) r^{out} r_j^{in}$

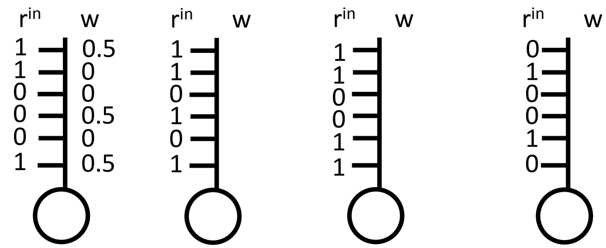
- Higher weight – suppressed weight update



Using weight control and decay

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

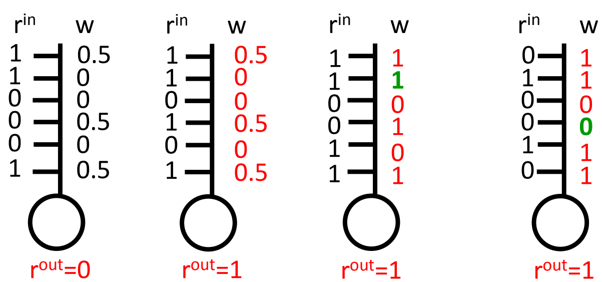
$\boldsymbol{\Delta w_j = r_{out}(r_j - w_j)}$



Using weight control and decay

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

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



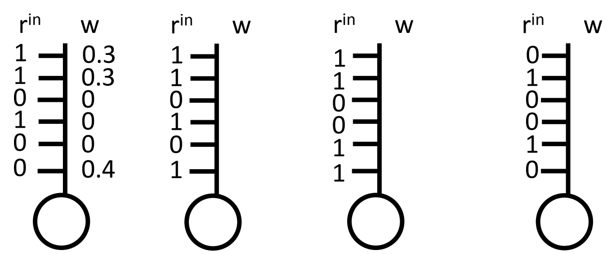
Using weight control and decay

Define input $h = \sum_k w_k r_k^{in}$
Neuron fires at rate r^{out}=1 when $h > .5$

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

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

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



Using weight control and decay

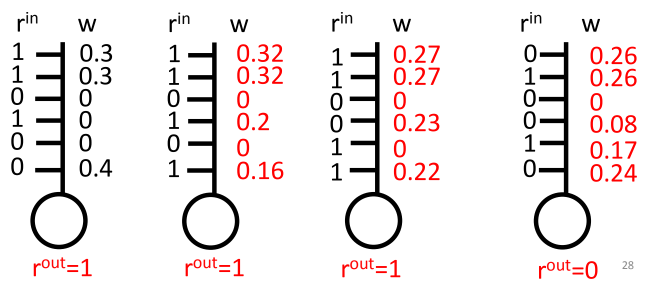
Define input $h = \sum_k w_k r_k^{in}$

Neuron fires at rate r^{out}=1 when $h > .5$

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

$$\boldsymbol{\Delta w_{ij} = \epsilon(w) r_i r_j}$$

$$\boldsymbol{w_{ij} \leftarrow \frac{w_{ij}}{\sum_j |w_{ij}|}}$$



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}'s

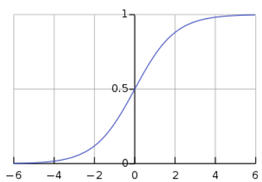
Step 3: Divide new w_j^{t+1}'s by $\sum_k |w_k^{t+1}|$ so new $|w_j|$'s add to 1

AI Neural Net Learning

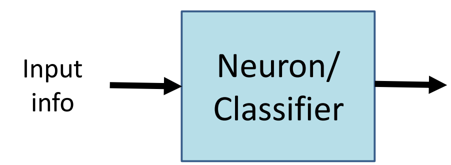
Computed output: $g^{sigmoid}\left(\sum_i w_j r_j^{in}\right)$

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