Terms

Given lowercase letters are scalar values and uppercase letters are vectors/matrices...

- X = vector of presynaptic membrane potentials
- Y = vector of postsynaptic membrane potentials
- T = vector of target postsynaptic membrane potentials
- E = vector of error between target and actual postsynaptic membrane potentials
- W_{XY} = vector of axon conductance factors between X and Y
- b = forward propagation interference factor
- S = vector of axon potentials
- η = learning factor
- t_{hresh} = axon potential action threshold

Forward Propagation / $(X, W_{XY}, b) \Rightarrow Y$

Assuming: $X = [2,3]$, $\quad W_{XY} = \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix}$, $\quad b = 0$

Compute Axon Potential $(X, W_{XY}, b) \Rightarrow S$

$$\begin{aligned} S &= X \cdot W_{XY} + b \\ &= [2,3] \cdot \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix} + 0 \\ &= [(2)(4) + (3)(0), (2)(5) + (3)(-2)] \\ &= [11, 4] \end{aligned}$$

Compute Activation Functions / $S \Rightarrow Y$

Assuming $\forall_i s_i \in S$...

- **Linear:** $f_{act}(S) = S$
- **Threshold:** $f_{\text{act}}(S) = \left[\begin{cases} 0 & \text{if } s_i \leq t_{\text{hreshold}} \\ 1 & \text{if } s_i > t_{\text{hreshold}} \end{cases} \right]_{i=0}^{n-1} = \begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}$
- **Sigmoid:** $f_{act}(S) = \left[\frac{1}{1+\exp(-s_i)} \right]_{i=0}^{n-1} = \begin{bmatrix} s_0 \\ s_1 \\ \dots \\ s_{n-1} \end{bmatrix}$

Unsupervised Hebbian Learning / $(\eta, X, Y, W_{XY}) \Rightarrow W'_{XY}$

- Replicates classical conditioning
- Biologically inspired
- Learns associations, but does not recognize right from wrong

Assuming: $\eta = 0.1 \quad X = [2, 3], \quad Y = [11, 16], \quad W_{XY} = \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix},$

Calculate weight delta matrix / $(\eta, X, Y) \Rightarrow \Delta W_{XY}$

$$\begin{aligned} \Delta W_{XY} &= \eta \times X^T \times Y \\ &= 0.1 \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 11, 16 \end{bmatrix} \\ &= 0.1 \times \begin{bmatrix} (2)(11) & (2)(16) \\ (3)(11) & (3)(16) \end{bmatrix} \\ &= \begin{bmatrix} 2.2 & 3.2 \\ 16.5 & 24 \end{bmatrix} \end{aligned}$$

Apply weight delta matrix / $(W_{XY}, \Delta W_{XY}) \Rightarrow W'_{XY}$

$$\begin{aligned} W'_{XY} &= W_{XY} + \Delta W_{XY} \\ &= \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix} + \begin{bmatrix} 2.2 & 3.2 \\ 16.5 & 24 \end{bmatrix} \\ &= \begin{bmatrix} 6.2 & 8.2 \\ 16.5 & 22 \end{bmatrix} \end{aligned}$$

Semi-Supervised Hebbian Learning / $(\eta, X, T, W_{XY}) \Rightarrow W'_{XY}$

- Combines classical (unsupervised) and operant (supervised) conditioning principles
- Does unsupervised learning on the whole network, then does supervised learning on the final axon conductance vector
- Biologically inspired
- Error-driven reinforcement guided by target output and ignoring current output on final axon conductance vector

Assuming: $\eta = 0.1, \quad X = [2, 3], \quad T = [13, 17], \quad W_{XY} = \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix}$

- First, perform Unsupervised Hebbian Learning on all axon conductances up to the network's output membrane potential vector.
- Next, perform the Supervised Hebbian Learning on the axon conductance vector connecting the final hidden layer's membrane potential vector and the output membrane potential vector like so:

Calculate weight delta vector / $(\eta, X, T) \Rightarrow \Delta W_{XY}$

$$\begin{aligned} \Delta W_{XY} &= \eta \times X^T \times T \\ &= 0.1 \times \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 13, 17 \end{bmatrix} \\ &= 0.1 \times \begin{bmatrix} (2)(13) & (2)(17) \\ (3)(13) & (3)(17) \end{bmatrix} \\ &= \begin{bmatrix} 2.6 & 3.4 \\ 3.9 & 5.1 \end{bmatrix} \end{aligned}$$

Calculate final weight vector / $(W_{XY}, \Delta W_{XY}) \Rightarrow W'_{XY}$

$$\begin{aligned} W'_{XY} &= W_{XY} + \Delta W_{XY} \\ &= \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix} + \begin{bmatrix} 2.2 & 3.2 \\ 16.5 & 24 \end{bmatrix} \\ &= \begin{bmatrix} 6.2 & 8.2 \\ 16.5 & 22 \end{bmatrix} \end{aligned}$$

Gradient Descent / *Theorem*

- Used to calculate ΔW_{XY} given an error expression

Assuming: $E'_{wh} = \frac{1}{2}(T-Y)^2$ Widrow-Hoff error with $\frac{1}{2}(...)^2$ for derivative convenience

Perform Gradient Descent on Widrow-Hoff error / $E_{wh} \Rightarrow \Delta W_{XY}$

$$\begin{aligned}
\frac{\partial E'_{wh}}{\partial Y} &= \frac{\partial}{\partial Y} E'_{wh} \\
&= \frac{\partial}{\partial Y}\frac{1}{2}(T-Y)^2 \\
&= \frac{1}{2} \cdot \frac{\partial}{\partial Y}(T-Y)^2 \\
&= \frac{1}{2} \cdot \frac{\partial}{\partial Y}u^2 \qquad \text{where } u = T-Y \text{ and } \frac{d}{dY}(T-Y) = -1 = \frac{du}{dY} \\
&= \frac{1}{2} \cdot -\frac{\partial}{\partial u}u^2 \qquad \text{where } \frac{d}{dY} = \frac{d}{dY}\frac{du}{du} = \frac{du}{dY}\frac{d}{du} = -\frac{d}{du} \\
&= \frac{1}{2} \cdot -2u \\
&= -u \\
&= -(T-Y)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial Y}{\partial W} &= \frac{\partial}{\partial W} Y \\
&= \frac{\partial}{\partial W} f_{act}(WX^T) \quad \text{Assuming linear activation function where } f_{act}(S_i) = S_i \\
&= \frac{\partial}{\partial W} WX^T \\
&= X^T
\end{aligned}$$

$$\begin{aligned}
\Delta E_{wh} &= \frac{\partial E'_{wh}}{\partial W} \\
&= \frac{\partial E'_{wh}}{\partial W} \cdot \frac{\partial Y}{\partial Y} \\
&= \frac{\partial E'_{wh}}{\partial Y} \cdot \frac{\partial Y}{\partial W} \\
&= -(T-Y) \cdot X^T \\
&= -X^T(T-Y)
\end{aligned}$$

$$\begin{aligned}
\Delta W_{wh} &= -\eta \times \Delta E_{wh} \\
&= -\eta \times -X^T(T-Y) \\
&= \eta \times X^T \times (T-Y) \\
&= \eta \times X^T \times E_{wh} \qquad \text{Assuming } E_{wh} = T-Y
\end{aligned}$$

Now, given error signal $\frac{1}{2}(T-Y)^2$, we can simply reference the above formula when updating weights during backpropagation assuming linear forwardpropagation.

For us to compute ΔW_{wh} with other activation functions, we would have to recompute $\frac{\partial Y}{\partial W}$ where $Y = f_{act}$ is not defined as $f_{act}(s_i) = S$.

Widrow-Hoff Learning / $(\eta, X, T, Y, W_{XY}) \Rightarrow W'_{XY}$

- AKA Delta Learning Rule
- Replicates operant conditioning
- Not biologically plausible
- Algorithm for backpropagation

Assuming: $\eta = 0.1, \quad X = [2,3], \quad T = [13,17], \quad Y = [11,16], \quad W_{XY} = \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix}$

Calculate error vector / $(T, Y) \Rightarrow E$

$$\begin{aligned} E &= T - Y \\ &= [13,17] - [11,16] \\ &= [2,1] \end{aligned}$$

Calculate weight delta vector / $(\eta, E, X) \Rightarrow \Delta W_{XY}$

$$\begin{aligned} \Delta W_{XY} &= \eta \times X^T \times E \\ &= 0.1 \times \begin{bmatrix} 2 \\ 1 \end{bmatrix} [2,3] \\ &= 0.1 \times \begin{bmatrix} (2)(2) & (2)(3) \\ (1)(2) & (1)(3) \end{bmatrix} \\ &= \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.3 \end{bmatrix} \end{aligned}$$

Calculate final weight vector / $(W_{XY}, \Delta W_{XY}) \Rightarrow W'_{XY}$

$$\begin{aligned} W'_{XY} &= W_{XY} + \Delta W_{XY} \\ &= \begin{bmatrix} 4 & 5 \\ 0 & -2 \end{bmatrix} + \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.3 \end{bmatrix} \\ &= \begin{bmatrix} 4.4 & 5.6 \\ 0.2 & -1.7 \end{bmatrix} \end{aligned}$$