

1

Learning

We want to learn about the world (Hebbian learning, Ch 4).

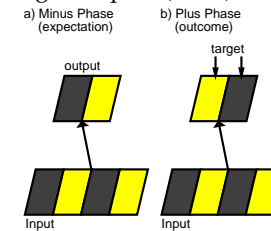
We also need to learn to do particular things (task-learning, Ch 5).

2

Task Learning: Good News/Bad News

Bad News: Hebbian bad at learning particular tasks.

Good News: Error-driven rule can learn tasks, using discrepancy between actual and target outputs (error).



$$\Delta w_{ik} = \epsilon(a_k^+ - a_k^-)a_i \quad (1)$$

Sims.

3

Task Learning

Key point: *Hidden units are crucial for learning difficult tasks, to allow activity patterns to be re-represented (transformed).*

Bad News: Hebbian works for learning about the world, but not for learning particular tasks.

Good News: Error-driven rule can learn tasks, using discrepancy between actual and target outputs (error).

We want it all: hidden units, Hebbian and error-driven learning.
What about biological plausibility?

4

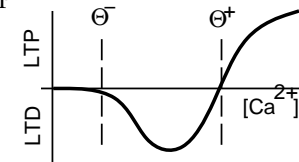
Biological Implementation

Minus Phase	Plus Phase					
	$x_i^+, y_j^+ \approx 0$			$x_i^+, y_j^+ \approx 1$		
	Err	Hebb	Combo	Err	Hebb	Combo
$x_i^-, y_j^- \approx 0$	0	0	0	+	+	+
$x_i^-, y_j^- \approx 1$	-	0	-	0	+	+

No Ca^{2+} → no learning

Mod Ca^{2+} → LTD

High Ca^{2+} → LTP



5

Learning

We want it all: hidden units, Hebbian and error-driven learning, biological plausibility.

Combination of Hebbian and error-driven learning consistent with LTP/LTD.

Can be implemented locally.

$$\Delta w_{ij} = \epsilon[(x_i^+ y_j^+) - (x_i^- y_j^-)]$$

6

Task Learning: Good News/Bad News

Bad News: Hebbian works for learning about the world, but not for learning particular tasks.

Good News: Error-driven rule can learn tasks, using discrepancy between actual and target outputs (error).

Bad News: The delta rule is fundamentally limited: no hidden units.

Good News: **Backpropagation** trains hidden units.

Bad News: Biology does not support backpropagation.

Good News: **GeneRec** can compute error in biologically plausible way as difference of two *activation* states.

7

Task Learning: Minimizing Error (Gradient Descent)

To minimize the error, take the *derivative* of the error with respect to the weights: indicates how error changes as weights change.

Task error = Summed-Squared Error:

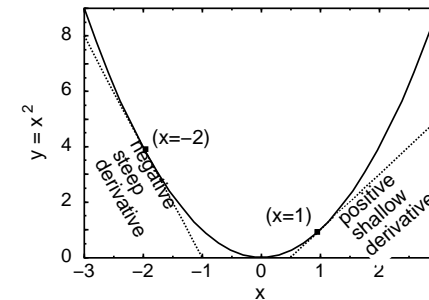
$$SSE = \sum_t \sum_k (t_k - o_k)^2 \quad (2)$$

Delta Rule minimizes SSE:

$$\Delta w_{ik} = \epsilon(t_k - o_k) s_i \quad (3)$$

8

Example: Minimizing $y = x^2$ via derivatives



How does y change w/ changes to x ?

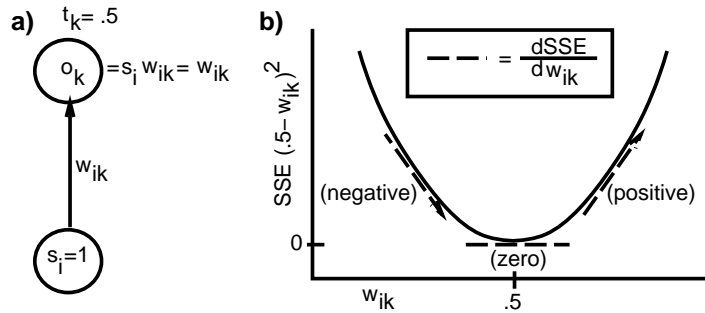
Derivative of y wrt x ; $\frac{dy}{dx}$ or $\frac{\partial y}{\partial x}$. Derivative of $x^2 = 2x$.

To minimize y , move x opposite the derivative.

9

Delta Rule in a Simple Network

$$SSE = \sum_t \sum_k (t_k - o_k)^2, \text{ derivative: } \Delta w_{ik} = \epsilon (t_k - o_k) s_i$$



10

Derivation of Delta for Linear Units

$$SSE = \sum_t \sum_k (t_k - o_k)^2. \text{ Activation: } o_k = \sum_i s_i w_{ik}$$

$$\frac{\partial SSE}{\partial w_{ik}} = \frac{\partial SSE}{\partial o_k} \frac{\partial o_k}{\partial w_{ik}} \quad (4)$$

$$\frac{\partial SSE}{\partial o_k} = -2(t_k - o_k) \quad (5)$$

$$\frac{\partial o_k}{\partial w_{ik}} = s_i \quad (6)$$

$$\frac{\partial SSE}{\partial w_{ik}} = -2(t_k - o_k) s_i \quad (7)$$

$$\Delta w_{ik} = \epsilon (t_k - o_k) s_i \quad (8)$$

11

Task Learning: Good News/Bad News

Bad News: Hebbian can't learn arbitrary input/output mappings.

Good News: Error-driven **delta rule** can, using discrepancy between actual and target outputs (error).

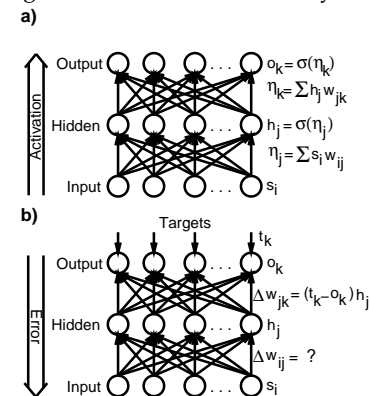
Bad News: The delta rule is fundamentally limited: no hidden units.

Good News: **Backpropagation** trains hidden units.

12

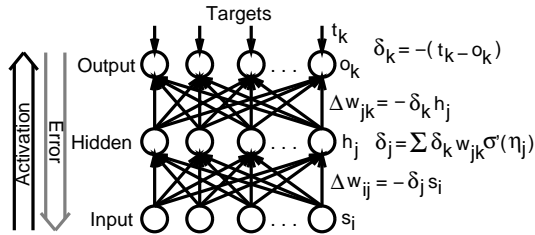
Error Backpropagation

Propagate error signals to hidden units so they can adjust weights:



13

Error Backpropogation



Major chain rule:

$$\frac{\partial CE}{\partial w_{ij}} = \sum_k \frac{dCE}{do_k} \frac{do_k}{d\eta_k} \frac{\partial \eta_k}{\partial h_j} \frac{dh_j}{d\eta_j} \frac{\partial \eta_j}{\partial w_{ij}} \quad (9)$$

For output units: $\delta_k = -(t_k - o_k)$

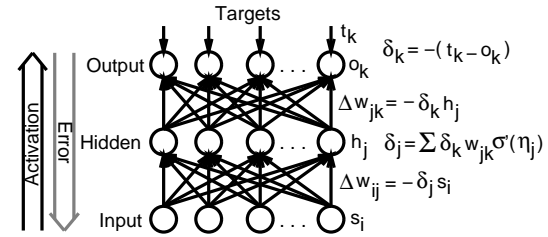
For hidden units: $\delta_j = \left(\sum_k \delta_k w_{jk}\right) (h_j(1 - h_j))$

14

The Problem with BP

For hidden units: $\delta_j = \left(\sum_k \delta_k w_{jk}\right) (h_j(1 - h_j))$

How does δ_k get propagated backwards across the synapse, down the axon, and out the dendrites? How do hidden units compute the derivative of their activation function?

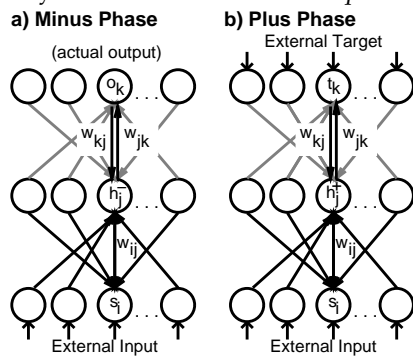


Most psychological models use this learning algorithm.

15

GeneRec: Biologically Plausible Bp

Use bidirectionally-connected network with 2 phases of settling:



$$\Delta w_{ij} = \epsilon[(x_i^+ y_j^+) - (x_i^- y_j^-)]$$

16

Task Learning

Hidden units are crucial for learning difficult tasks, to allow activity patterns to be re-represented (transformed).

Hebbian works for learning about the world, but not for learning particular tasks.

Error-driven rule can learn tasks, using discrepancy between actual and target outputs (error).

Combination of Hebbian and error-driven learning consistent w/biology, works w/hidden units.