

1

## Learning

Tuning the detectors locally with the overall network in mind.

Two main types: learning internal **model** of environment, & learning to solve a **task** (produce appropriate output from input).

Biology suggests associative or Hebbian learning: LTP and LTD. ("Units that fire together wire together!")

### Model Learning (aka self-organizing learning, Ch 4)

Based directly on Hebbian mechanisms.

Does principal components analysis (PCA) on input correlations.

2

## Task Learning (Ch 5)

Hebbian often can't learn to produce specific output for given input – boo.

Error-driven task learning can, using discrepancy between actual and target outputs (error) – yay.

The delta rule is fundamentally limited – boo.

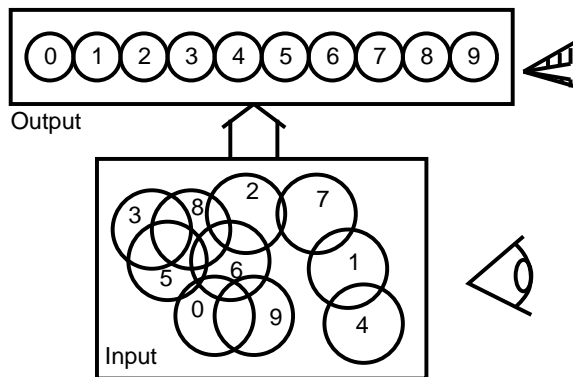
Backpropagation gets around these limitations – yay.

Biology does not support backpropagation – boo.

Error can instead be computed as difference of two *activation* states (consistent w/biology of LTP/D) – yay?

3

## Task Learning: Input-Output Mappings



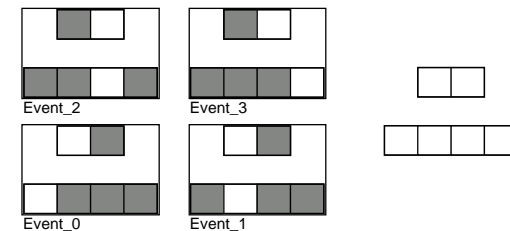
Make appropriate response given stimulus.

Make appropriate interpretation, expectation, plan...

4

## Consider some examples

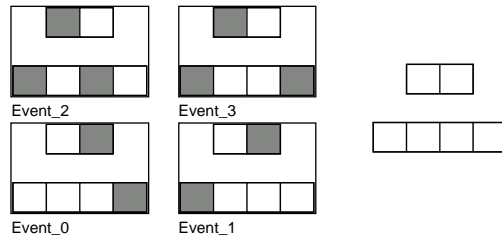
1.



5

Consider some examples

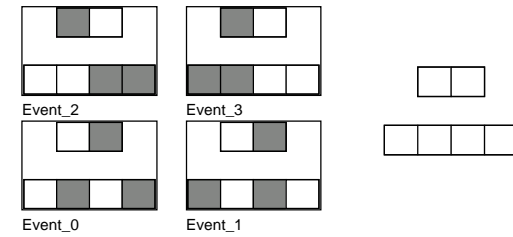
2.



6

Consider some examples

3.

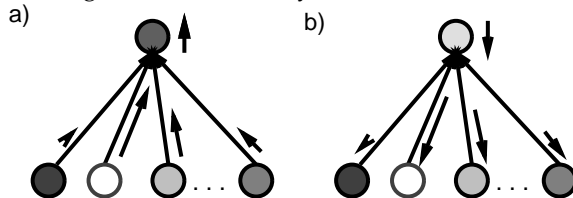


sims.

7

Task Learning: Minimizing Error (gradient descent)

Example: Optimize weight. Activation = caloric content of food.  
 Weight = how much you eat of that food. What do you need to know? Error measure.  
 How error changes w/how much you eat.



8

Task Learning: Minimizing Error (gradient descent)

Monitor with summed-squared error:

$$SSE = \sum_t \sum_k (t_k - o_k)^2 \quad (1)$$

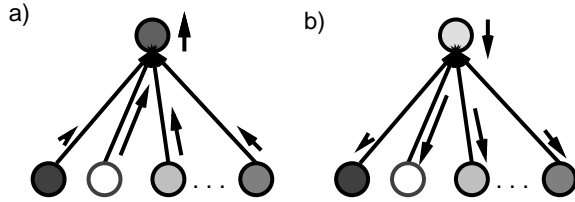
To minimize the error, take the *derivative* of the error with respect to the weights: indicates how the error changes as the weights change.

9

### Delta Rule

$$\Delta w_{ik} = \epsilon(t_k - o_k)s_i \quad (2)$$

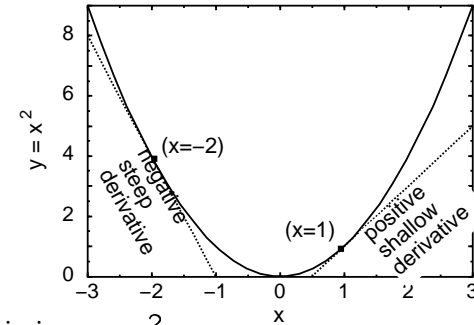
Supports credit and blame assignment:



Weights reflect strongest *solution* (vs. strongest *correlation* in Hebbian).

10

### Minimizing functions via derivatives



Want to minimize  $y = x^2$ .

Need to understand how  $y$  changes w/ changes to  $x$ .

Derivative of  $y$  wrt  $x$ , or  $\frac{\partial y}{\partial x}$ , for  $x^2 = 2x$ .

To minimize  $y$ , move  $x$  in direction opposite the derivative, and in step sizes according to absolute magnitude of derivative.

11

### In networks...

$$SSE = \sum_t \sum_k (t_k - o_k)^2, \quad \frac{\partial SSE}{\partial w_{ik}} = ?$$

$$\frac{\partial SSE}{\partial w_{ik}} = \frac{\partial SSE}{\partial o_k} \frac{\partial o_k}{\partial w_{ik}}$$

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

$$o_k = \sum_i s_i w_{ik}$$

$$\frac{\partial o_k}{\partial w_{ik}} = s_i$$

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

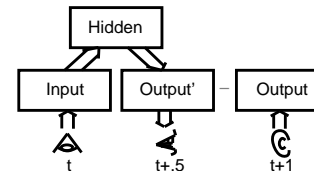
$$\Delta w_{ik} = \epsilon(t_k - o_k)s_i$$

Bias Wts:  $\Delta \beta_k = \epsilon(t_k - o_k)$

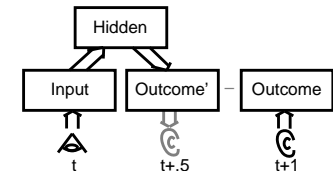
12

### Task Learning: Where does target come from?

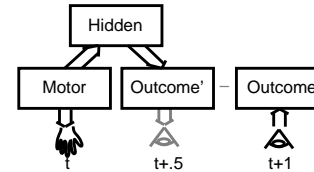
a) Explicit Teacher



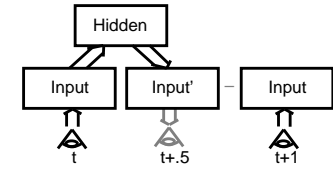
b) Implicit Expectation



c) Implicit Motor Expectation

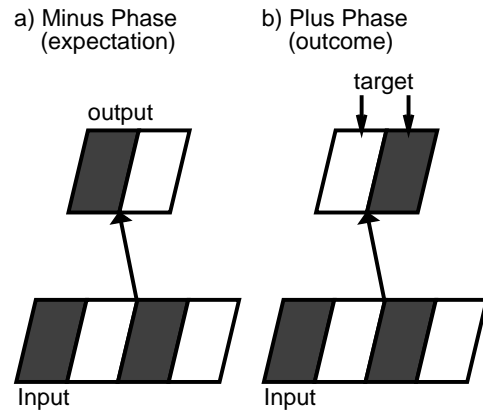


d) Implicit Reconstruction



13

## Activation Phases



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

14

## Soft Weight Bounding

$$\Delta w_{ik} = [\Delta_{ik}]_+(1 - w_{ik}) + [\Delta_{ik}]_-w_{ik} \quad (4)$$

sims.

15

## Task Learning (Ch 5)

Hebbian often can't learn to produce specific output for given input – boo.

Error-driven task learning can, using discrepancy between actual and target outputs (error) – yay.

The delta rule is fundamentally limited – boo.

Backpropagation gets around these limitations – yay.

Biology does not support backpropagation – boo.

Error can instead be computed as difference of two *activation* states (consistent w/biology of LTP/D) – yay?

16

## Re-representing and Hidden layers

Difficult tasks become easier when you re-represent using intermediate representations:

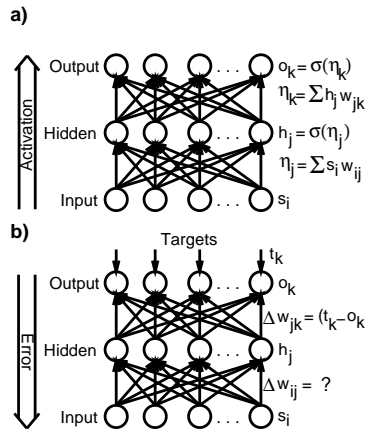
- Memorize digits using digit chunks.
- Read in terms of words, not letters.
- “Thinking outside the box”

Hidden layers enable this re-representation!  
(multiple levels of transformations).

17

### Error Backpropagation

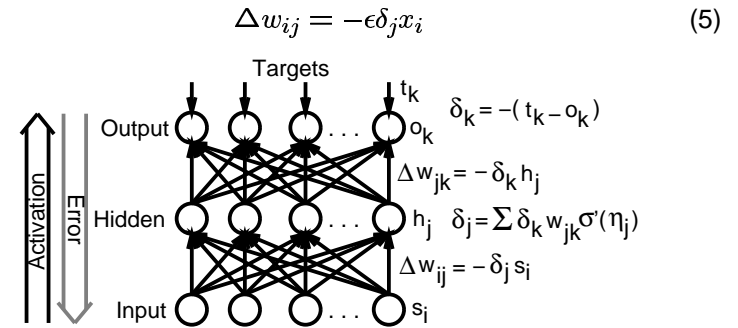
Key idea: propagate err sigs to hid units so they can adjust weights:



(weights from hidden to output can be trained by delta rule).

18

### BP: The Equations



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

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

19

### Problems with BP

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

How does that  $\delta_k$  get propagated backwards across the synapse, down the axon, and out the dendrites?

Who is computing the derivative of the activation function?

Most psychological models use this learning algorithm.

Thoughts?

20

### GeneRec to the Rescue

Errors in terms of activations.

Bidirectional connections are key.

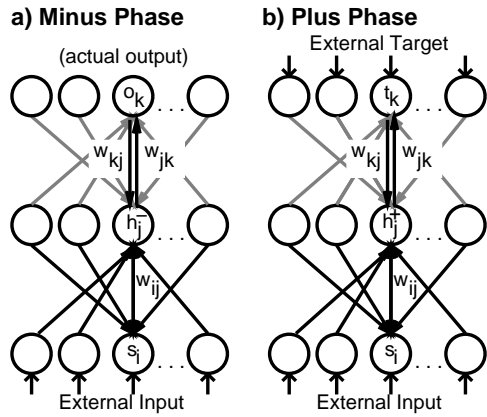
$$\Delta w_{ij} = \epsilon [(x_i^+ y_j^+) - (x_i^- y_j^-)]$$

#### Biological Implementation

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

21

GeneRec to the Rescue



Hidden units get actual output ( $o_k$ ) and target ( $t_k$ ) signals via activation propagation.

22<sup>BP</sup>

GeneRec

$$\delta_j = \sum_k (t_k - o_k) w_{jk} h_j (1 - h_j)$$

$$\begin{aligned} \delta_j &= \sum_k (t_k - o_k) w_{kj} h_j (1 - h_j) \\ &= (\sum_k t_k w_{kj} - \sum_k o_k w_{kj}) h_j (1 - h_j) \\ &= (\eta_j^+ - \eta_j^-) h_j (1 - h_j) \\ &= (y_j^+ - y_j^-) \end{aligned}$$

