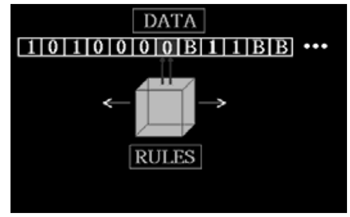
PSYCO 452

Week 2: Distributed Memory

•Introduction To Connectionism
•Building Associations
–Hebb Learning
–Delta Learning

The Classical Approach

The Classical approach adopts a strict "structure/rule" distinction in its view of information processing

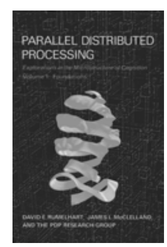


Some Classical Problems

- Poor for ill-posed problems
- Not damage resistant
- Does not degrade gracefully
- Serial -- therefore slow
- Not biologically plausible!

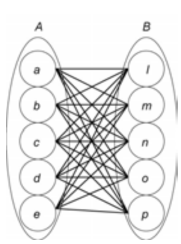
An Alternative

- Since the 1980s there has been an explosion of interest in parallel distributed processing (PDP) or connectionist architectures
- These architectures have been developed to solve the Classical problems



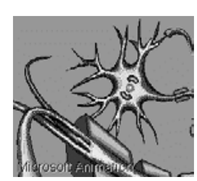
PDP Networks Are Parallel

- PDP models are networks of simple processors that operate simultaneously
- This causes fast computation, even if components are slow
- This is intended to fix the speed limitation of Classical models

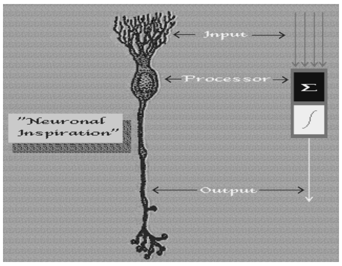


Biological Motivation

- PDP modelers pay more attention to the brain than do Classical researchers
- A PDP processor can be viewed as an abstract, simplified description of a neuron

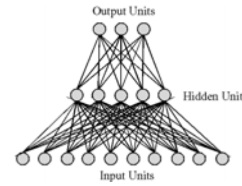


Neuronal Inspiration



Distributed Representations

- A PDP network's knowledge is stored as a pattern of weighted connections between processors
- These connections are analogous to a Classical program
- This knowledge is very distributed, providing damage resistance and graceful degradation

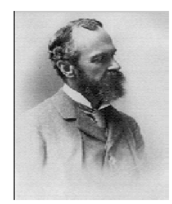


Synthetic Approach

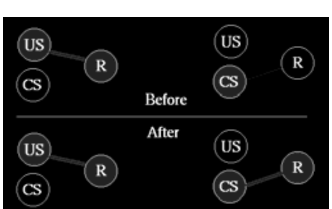
- Our first pass at synthetic psychology will be to use connectionist building blocks as our architecture
  - What are the building blocks?
  - What can we build with them?
- Over the next three weeks, we will consider three different building blocks:
  - Association
  - Decision
  - Trains of thought

First Building Block: Association

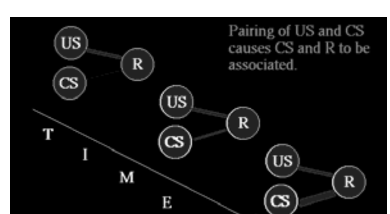
- One of the key building blocks for a connectionist system is a method for storing associations between and input and output pattern
- Let us begin by considering a couple of simple methods by which this sort of association could be achieved



Classical Conditioning



Simultaneity & Conditioning

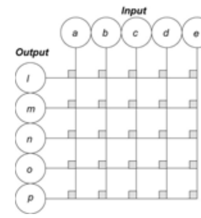


Hebb And Association

- "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes place in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" (Hebb, 1949)
- Principle of contiguity!

Content Addressable Memory

- Modern views of Hebb learning involve the strengthening of synapses (both excitatory and inhibitory) as well as the weakening of synapses
- These two processes have been combined to create many interesting models of content addressable memories



Memory Addressing Modes

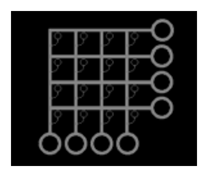
- "Address" addressable memory
  - Retrieve items by content-independent location

•Content Addressable Memory

- Content of item itself, or parts of it, represent the location or retrieval cue for the full item

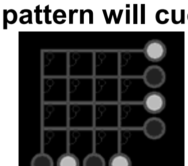
Distributed Memory

- A simple distributed memory system consists of two sets of processors, and a set of modifiable connections between them



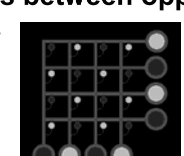
Hebb Rule 1

- Present two patterns of activity
- Associate the patterns because of their temporal contiguity
- Later, one pattern will cue the other



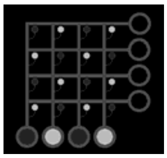
Hebb Rule 2

- Make more excitatory the connections between same-state processors
- Make more inhibitory the connections between opposite-state processors



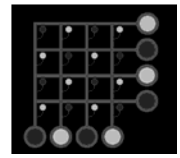
Hebb Rule 3

- To recall, activate processors with the cue
- Their activity sends a signal through existing connections



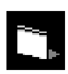
Hebb Rule 4

- The network signal should reconstruct the other pattern in the second set of processing units



Demonstrating Associative Learning

- Let's examine the Hebb rule in action
- Let us also determine some conditions in which Hebb learning does not work very well



James

Desired Weight Changes

Activity Of Input Unit	Activity Of Output Unit	Direction Of Desired Weight Change
Positive	Positive	Positive
Negative	Negative	Positive
Negative	Positive	Negative
Positive	Negative	Negative

Algebra Of The Hebb Rule

- Let W(t) be a matrix of connection weights at time t
- Let a and b be two to-be-associated vectors
- Hebb learning becomes:

$$W(t+1) = W(t) + a\,b'$$

- The outer product defines Hebb learning!

Learning Associations

$$\eta \bullet c \bullet d^T = \Delta_{t+1}$$

$$0.1 \bullet \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \bullet \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.2 & 0.4 & 0.6 & 0.8 \\ 0.3 & 0.6 & 0.9 & 1.2 \\ 0.4 & 0.8 & 1.2 & 1.6 \end{bmatrix}$$

Trial (t+1)	Operation	Equation Describing Weight Values
0	Start with the ***0*** matrix	$W_0 = 0$
1	Associate ***a*** with ***b***	$W_1 = W_0 + \Delta_1$ $= 0 + (b \bullet a^T)$ $= (b \bullet a^T)$
2	Associate ***c*** with ***d***	$W_2 = W_1 + \Delta_2$ $= (b \bullet a^T) + \Delta_2$ $= (b \bullet a^T) + (d \bullet c^T)$

Algebra Of Recall

- Recall from the memory involves filtering a signal through existing weights to produce output activity
- *r = Wc*

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix}$$

(A)

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \end{bmatrix}$$

(B)

Recall Proven

Cue	Recall Equation	Comments
a	$r = W_2 a$ $= ((b \bullet a^T) + (d \bullet c^T))a$ $= b \bullet a^T \bullet a + d \bullet c^T \bullet a$ $= b \bullet (a^T \bullet a) + d \bullet (c^T \bullet a)$ $= b(1) + d(0)$ $= b$	Equation 9-6 Expand W_2 from Table 9-2 Move vector ***a*** into the parentheses Identify the inner products with parentheses Compute inner products (orthonormal assumption) ***b*** is correctly recalled
c	$r = W_2 c$ $= ((b \bullet a^T) + (d \bullet c^T))c$ $= b \bullet a^T \bullet c + d \bullet c^T \bullet c$ $= b \bullet (a^T \bullet c) + d \bullet (c^T \bullet c)$ $= b(0) + d(1)$ $= d$	Equation 9-6 Expand W_2 from Table 9-2 Move vector ***c*** into the parentheses Identify the inner products with parentheses Compute inner products (orthonormal assumption) ***d*** is correctly recalled

Limitations Of Hebb Rule

- We can use linear algebra to reveal some interesting limitations of Hebb learning
- For instance, what if we relax the mutual orthogonality constraint?
- What if the correlation between c and a is equal to 0.5?

Correlation And Error

Cue	Recall Equation	Comments
a	$r = W_2 a$ $= ((b \bullet a^T) + (d \bullet c^T))a$ $= b \bullet a^T \bullet a + d \bullet c^T \bullet a$ $= b \bullet (a^T \bullet a) + d \bullet (c^T \bullet a)$ $= b(1) + d(½)$ $\neq b$	Equation 9-6 Expand W_2 from Table 9-2 Move vector ***a*** into the parentheses Identify the inner products with parentheses Compute inner products ***b*** is not correctly recalled!
c	$r = W_2 c$ $= ((b \bullet a^T) + (d \bullet c^T))c$ $= b \bullet a^T \bullet c + d \bullet c^T \bullet c$ $= b \bullet (a^T \bullet c) + d \bullet (c^T \bullet c)$ $= b(½) + d(1)$ $\neq d$	Equation 9-6 Expand W_2 from Table 9-2 Move vector ***c*** into the parentheses Identify the inner products with parentheses Compute inner products ***d*** is not correctly recalled!

Correcting The Hebb Rule

- We would like to develop a new kind of Hebb learning rule
- This rule would permit the network to correctly recall correlated patterns
- This rule would also allow the network to improve its performance with repeated presentations of patterns

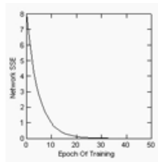
Error And Weight Change

Activity Of Input Unit	T - O	Implication	Operation To Reduce Error	Direction Of Desired Weight Change
Positive	**Positive**	**T > O**	**↑O**	**Positive**
Positive	**Negative**	**T < O**	**↓O**	**Negative**
Positive	**Zero**	**T = O**	**None**	**Zero**
Negative	**Positive**	**T > O**	**↑O**	**Negative**
Negative	**Negative**	**T < O**	**↓O**	**Positive**
Negative	**Zero**	**T = O**	**None**	**Zero**

The Delta Rule

- The delta rule can be viewed as a Hebb-style association between an input vector and an (output) error vector
- Repeated applications will reduce error
- The amount of learning depends on the amount of error
- The delta rule can be written as:

$$\Delta_{t+1} = \eta \; ((t - o) \bullet c^T)$$

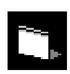


Linear Independence

- One vector can be created by combining (adding) two others
- If we have a set of vectors, and none of the vectors can be created by combining the others, the set of vectors is said to be linearly independent
- If the vectors are such that one can be created by combining some of the others, then the set is linearly dependent

Demonstrating The Delta Rule

- Let's examine the delta rule in action
- Let us note some instances in which it serves as an improvement over Hebb learning
- But let us also note that it is still subject to limitations



James

Two More Building Blocks

- How do we move beyond the sorts of limitations that we have noted in the simple distributed memory?
- First, we need to add nonlinearities into the processing units, letting them make decisions
- Second, we need to add some methods by which layers of these nonlinearities can be coordinated together
- These will be our topics for next week