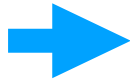
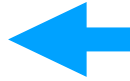


Learning in One-Layer Networks

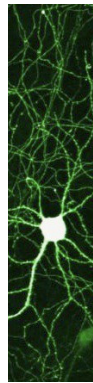
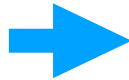
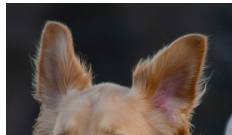
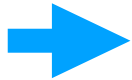
Psych 209
January 9, 2020

Input-output mapping

Simplest model of learning: input-output mapping



Input-output mapping

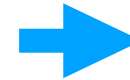


Pattern associator

a1
a2
·
·
·
an



b1
b2
·
·
·
bn



Input stimulus

Input pattern

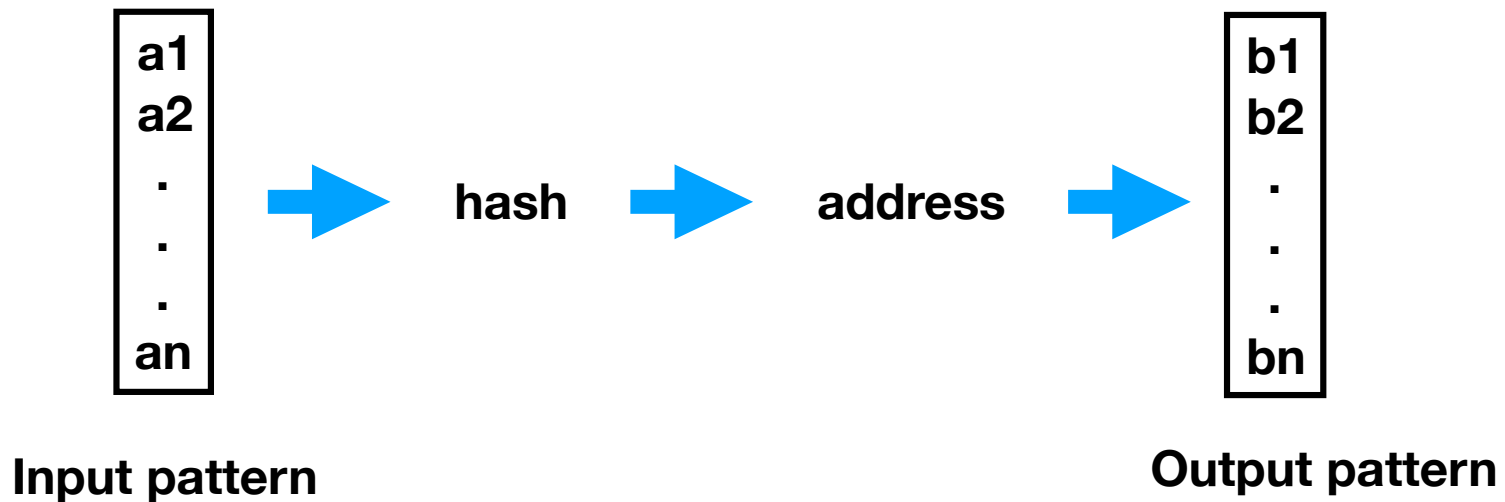
Output pattern

Output behavior

Input-output mapping

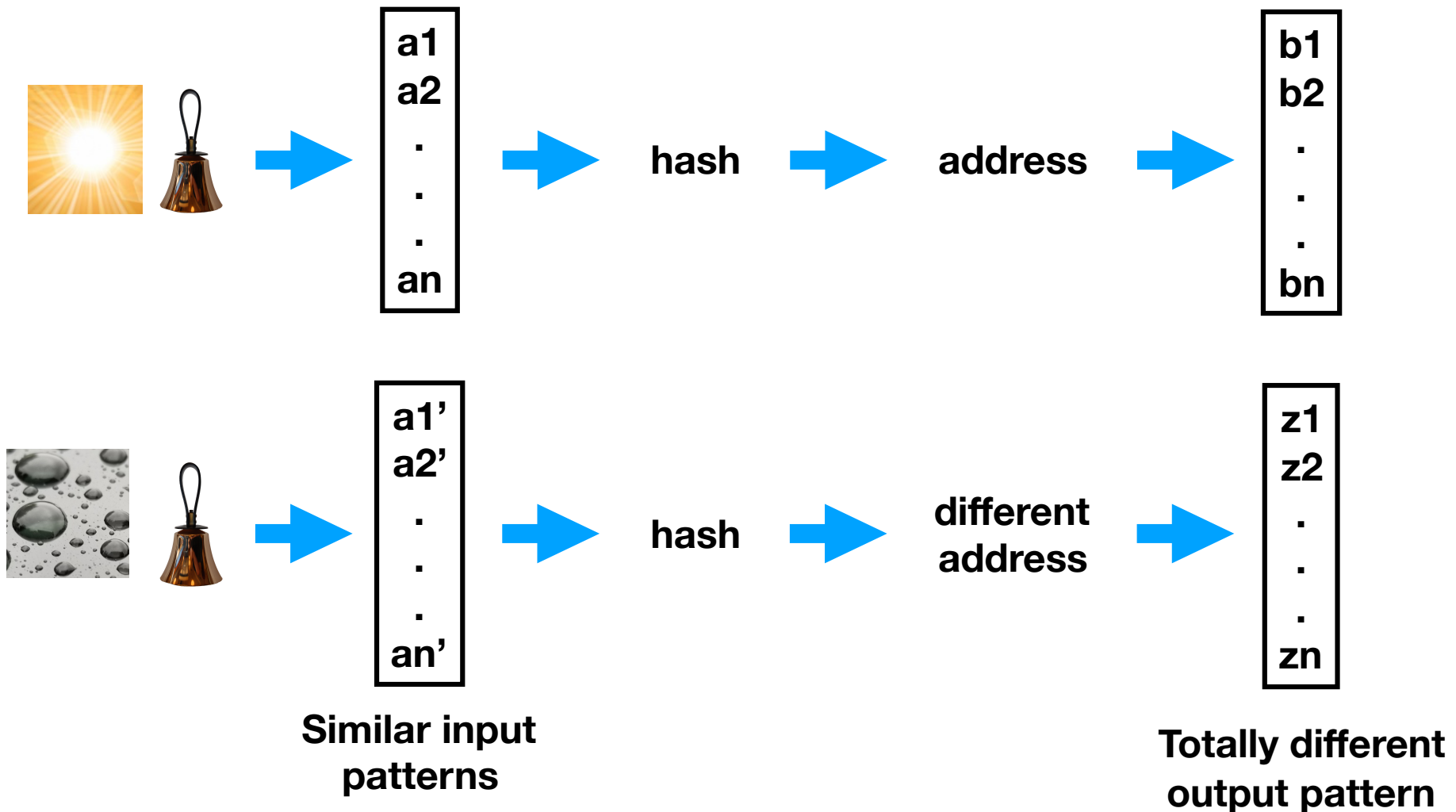
How can we learn input-output associations?

We know how to store (key, value) pairs on a computer



Input-output mapping

But location-based memory is brittle

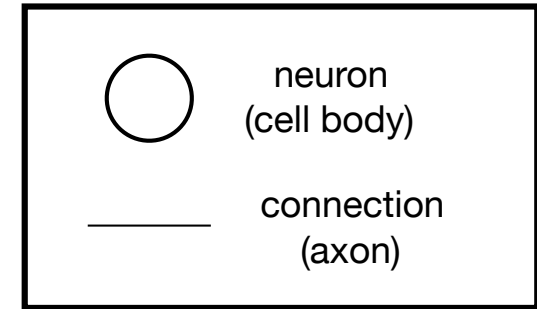
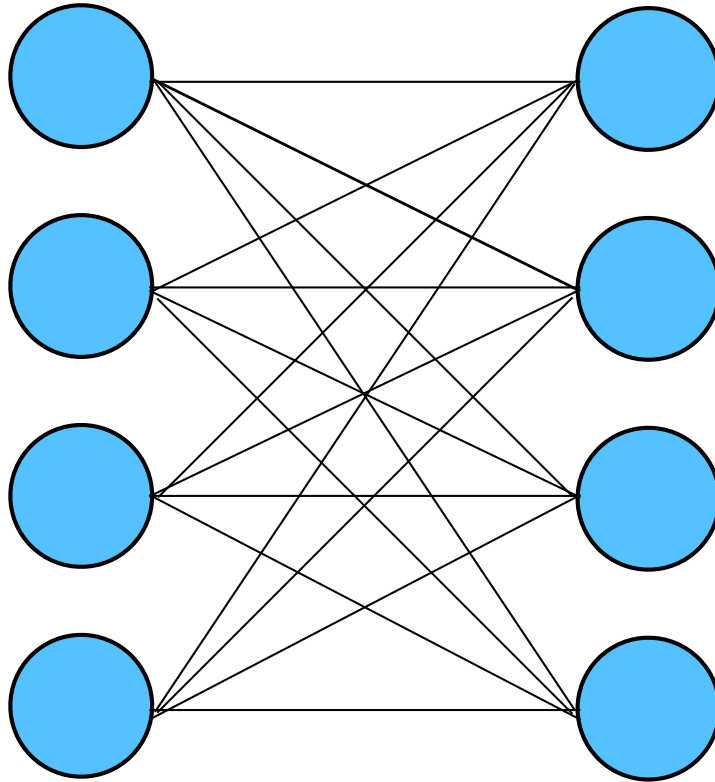


Pattern Associator

Content-addressable memory: keys directly “produce” values without location lookup

Input pattern

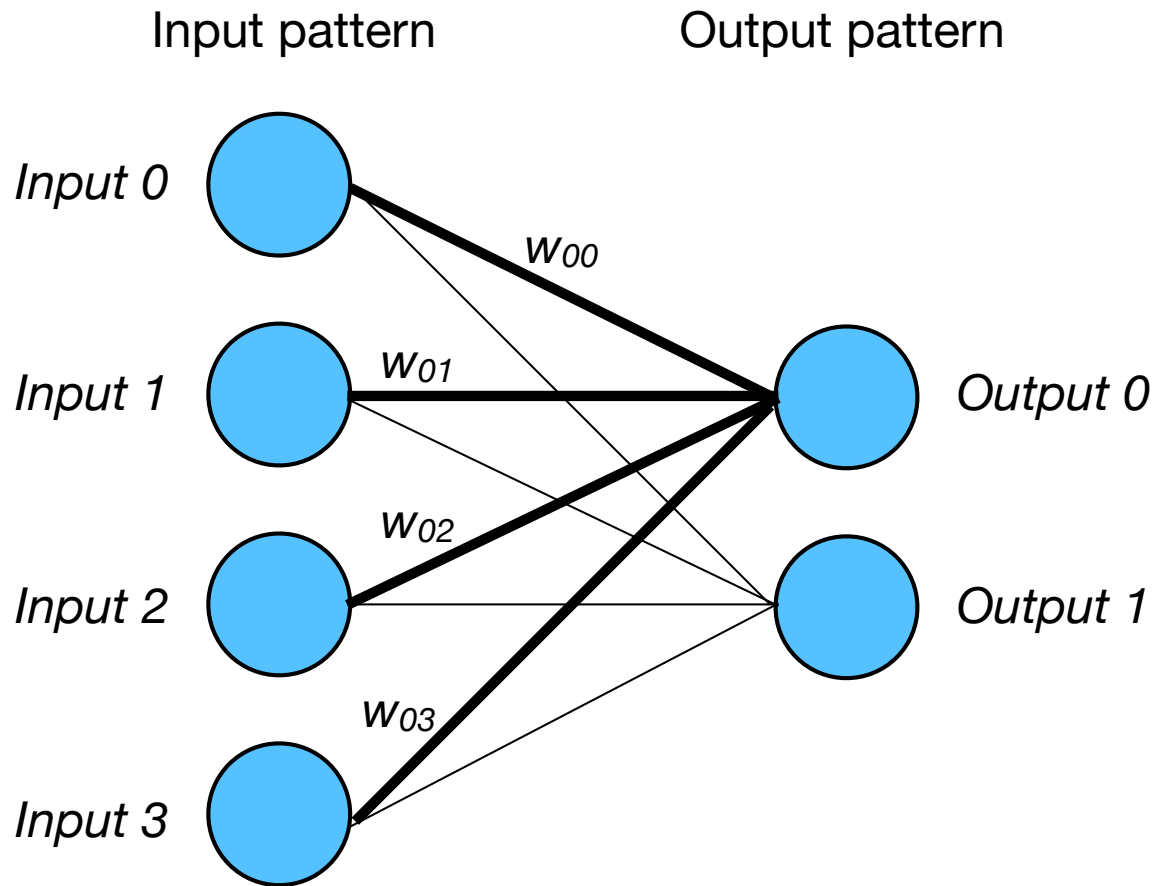
Output pattern



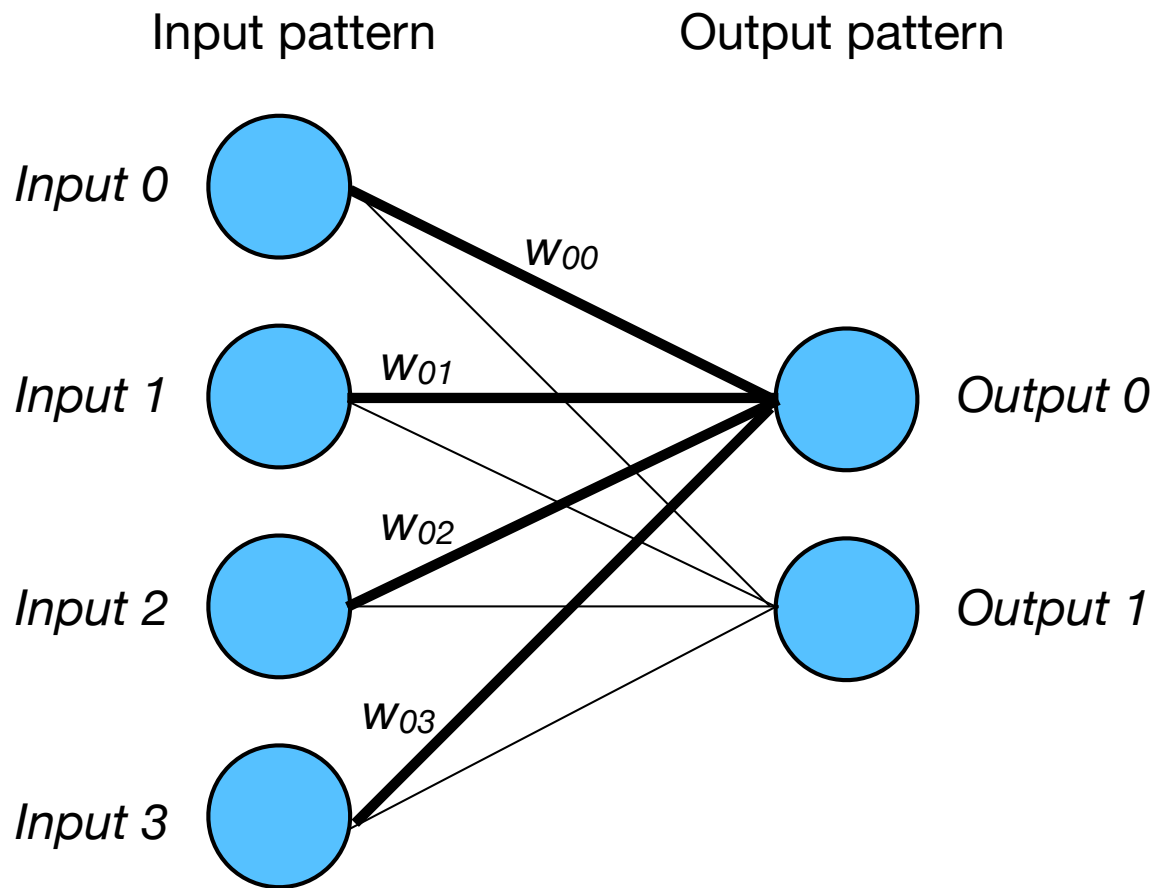
Desirable properties

Fully connected

Forward Pass

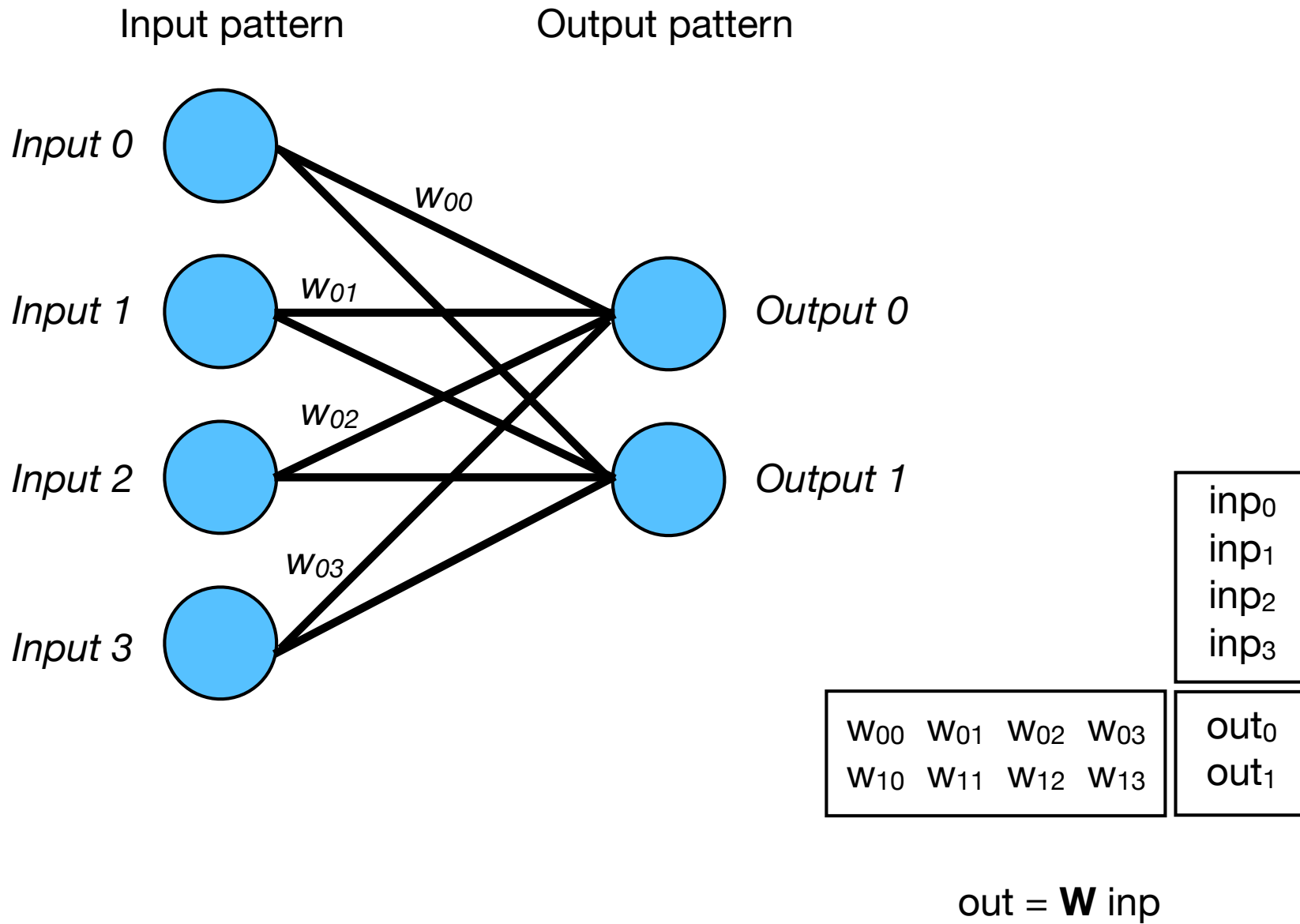


Forward Pass

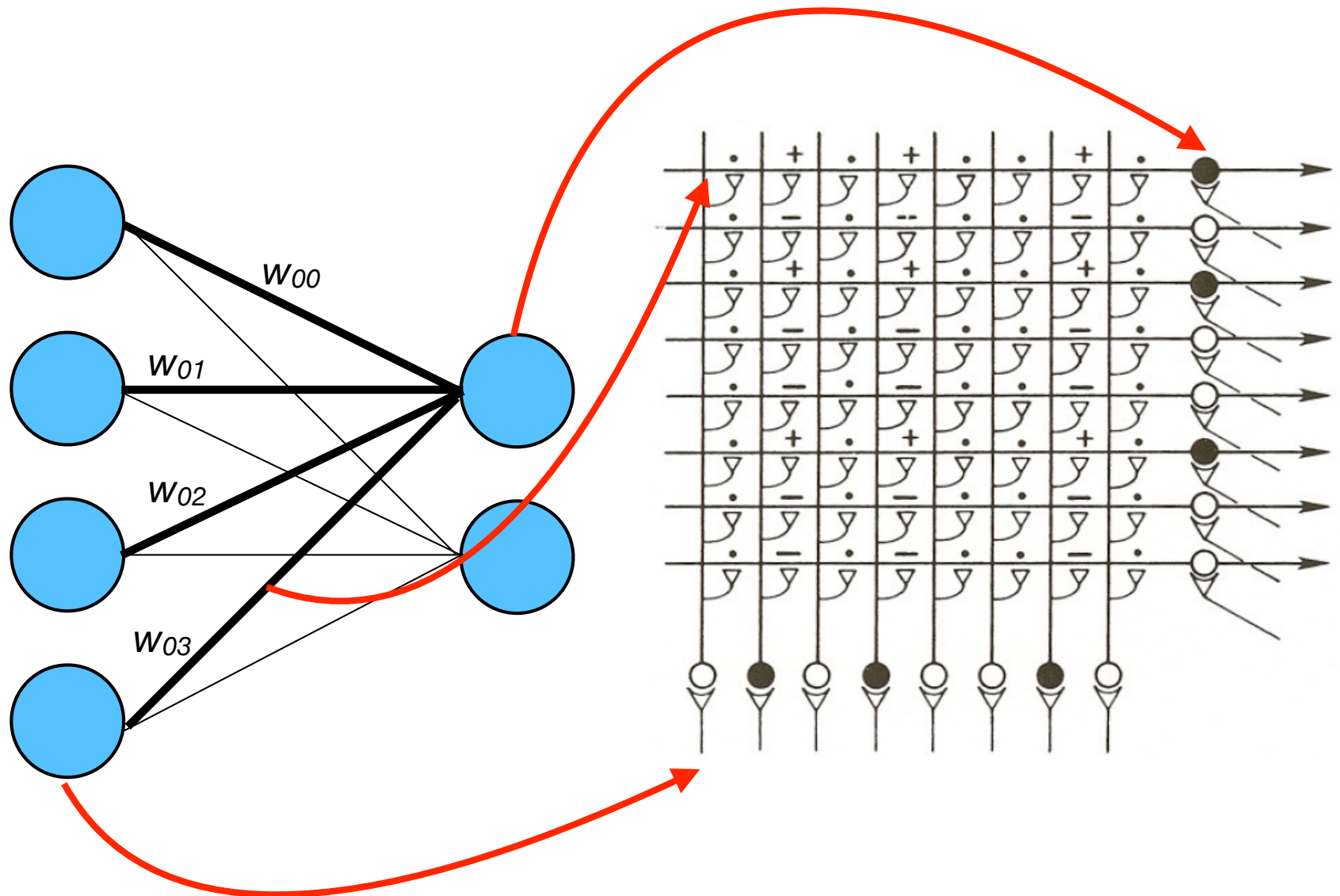


$$a_i = \sum_j w_{ij} a_j$$

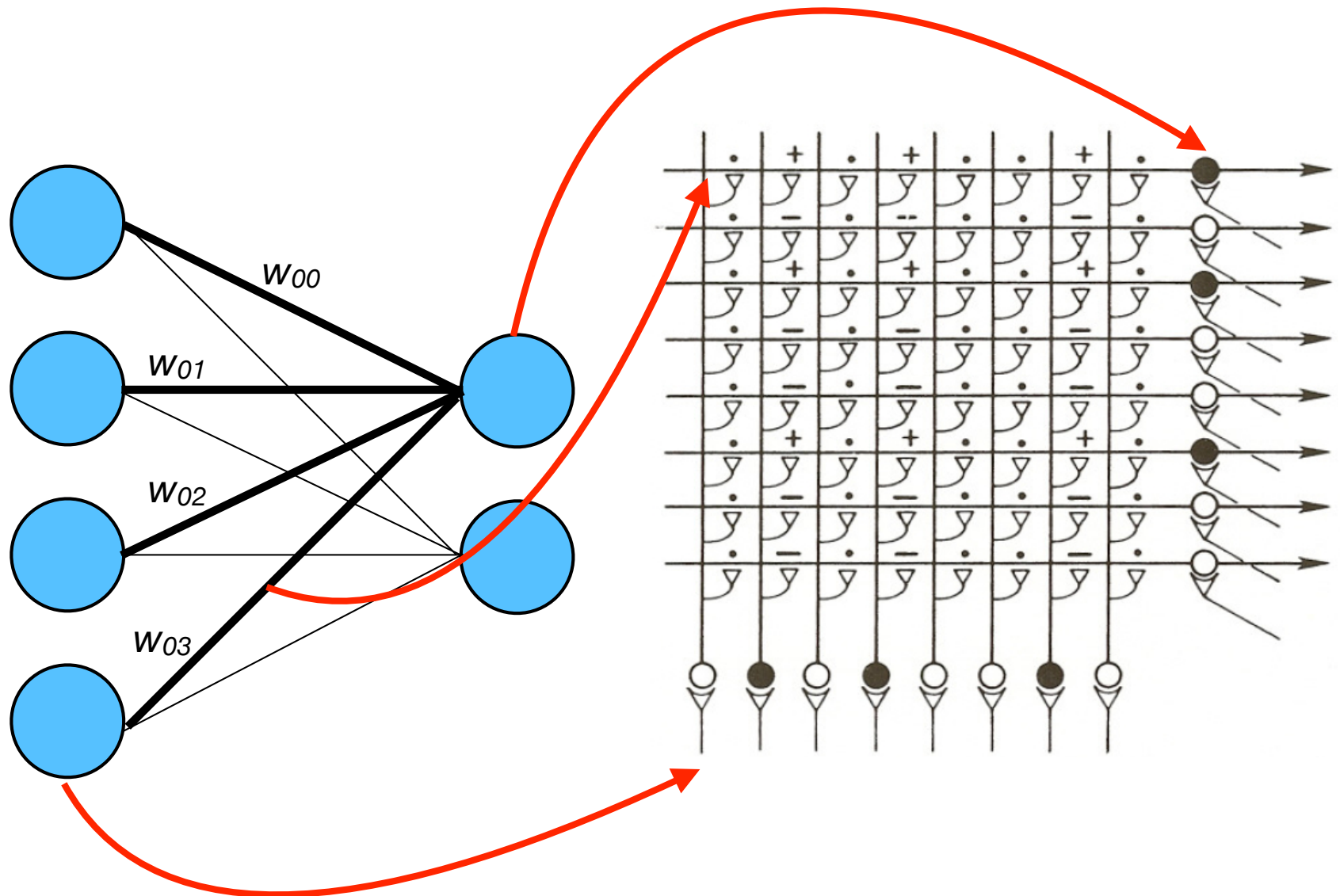
Forward Pass



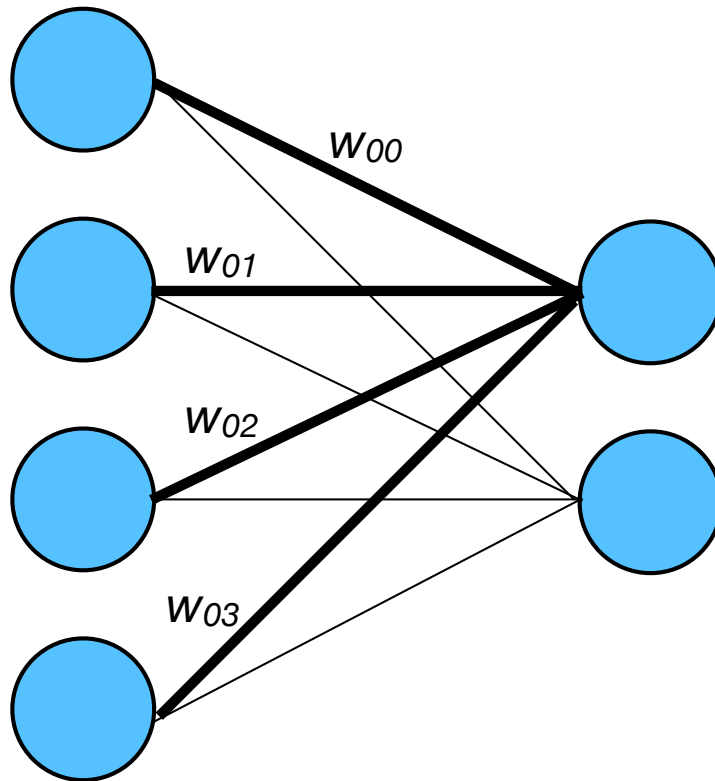
Matrix Representation



How do we learn the weights?



How do we learn the weights?

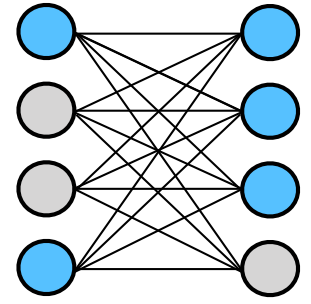


Learning rules:

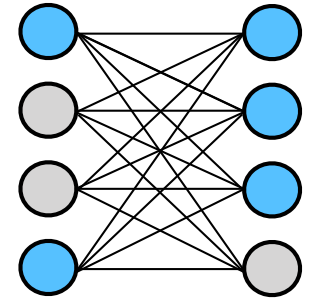
1. Hebb's rule
2. Delta rule

Hebb's Rule

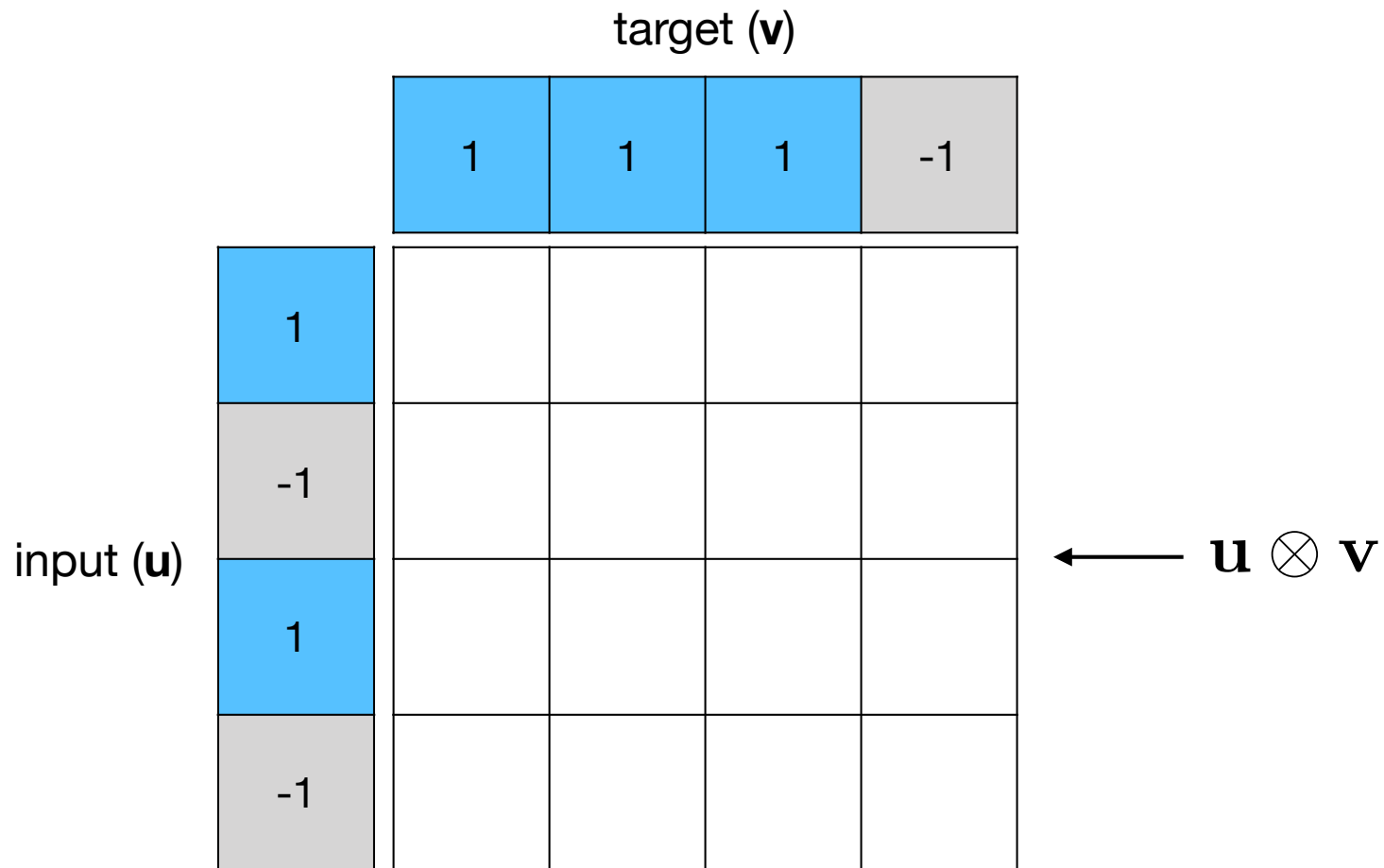
- “Neurons that fire together wire together”



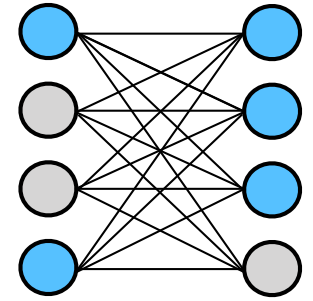
Hebb's Rule



- “Neurons that fire together wire together”



Hebb's Rule



- “Neurons that fire together wire together”

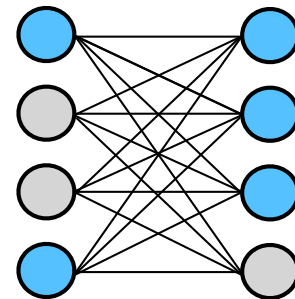
Storing a memory

		target (v)			
		1	1	1	-1
input (u)	1	1	1	1	-1
	-1	-1	-1	-1	1
	1	1	1	1	-1
	-1	-1	-1	-1	1

← $\mathbf{u} \otimes \mathbf{v}$

Storing multiple
memories

Hebb's Rule



targets
 V^T

-1	1	-1	-1
1	1	1	-1

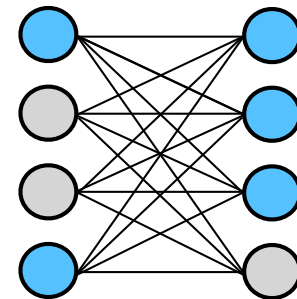
inputs
 U

-1	1	2	0	2	0
-1	-1	0	-2	0	2
-1	1	2	0	2	0
-1	-1	0	-2	0	2

← UV^T

Storing multiple
memories

Hebb's Rule



targets

V^T

-1	-1	1	-1
-1	1	-1	-1
1	1	1	-1

inputs

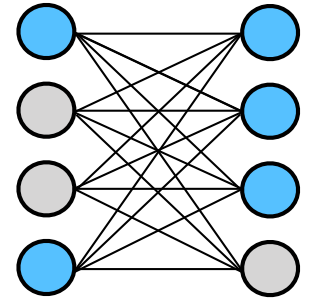
U

1	-1	1	1	-1	3	-1
1	-1	-1	-1	-3	1	1
-1	-1	1	3	1	1	1
-1	-1	-1	1	-1	-1	3

← UV^T

Hebb's Rule

Retrieval



1	-1	3	-1
-1	-3	1	1
3	1	1	1
1	-1	-1	3

UV^T

query

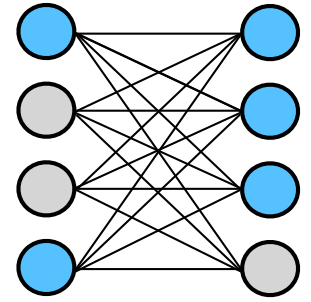
1	-1	1	-1	4	4	4	-4
---	----	---	----	---	---	---	----



threshold (clip to -1, 1)

1	1	1	-1
---	---	---	----

Hebb's Rule



targets
 V^T

-1	-1	1	-1
-1	1	-1	-1
1	1	1	-1

inputs
 U

1	-1	1	1	-1	3	-1
1	-1	-1	-1	-3	1	1
-1	-1	1	3	1	1	1
-1	-1	-1	1	-1	-1	3

Observe that the input
patterns here are an
orthogonal set

← UV^T

Hebb's Rule

Orthogonality

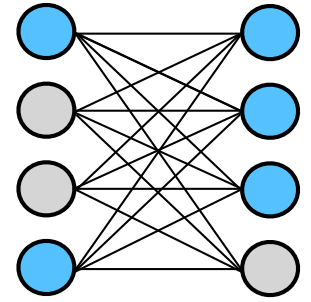
$$a \perp b \iff \cos(\theta_{ab}) = 0 \iff \langle a, b \rangle = \|a\| \|b\| \cos(\theta_{ab}) = 0$$

Say we want to query the memory with input $\mathbf{u}_i$:

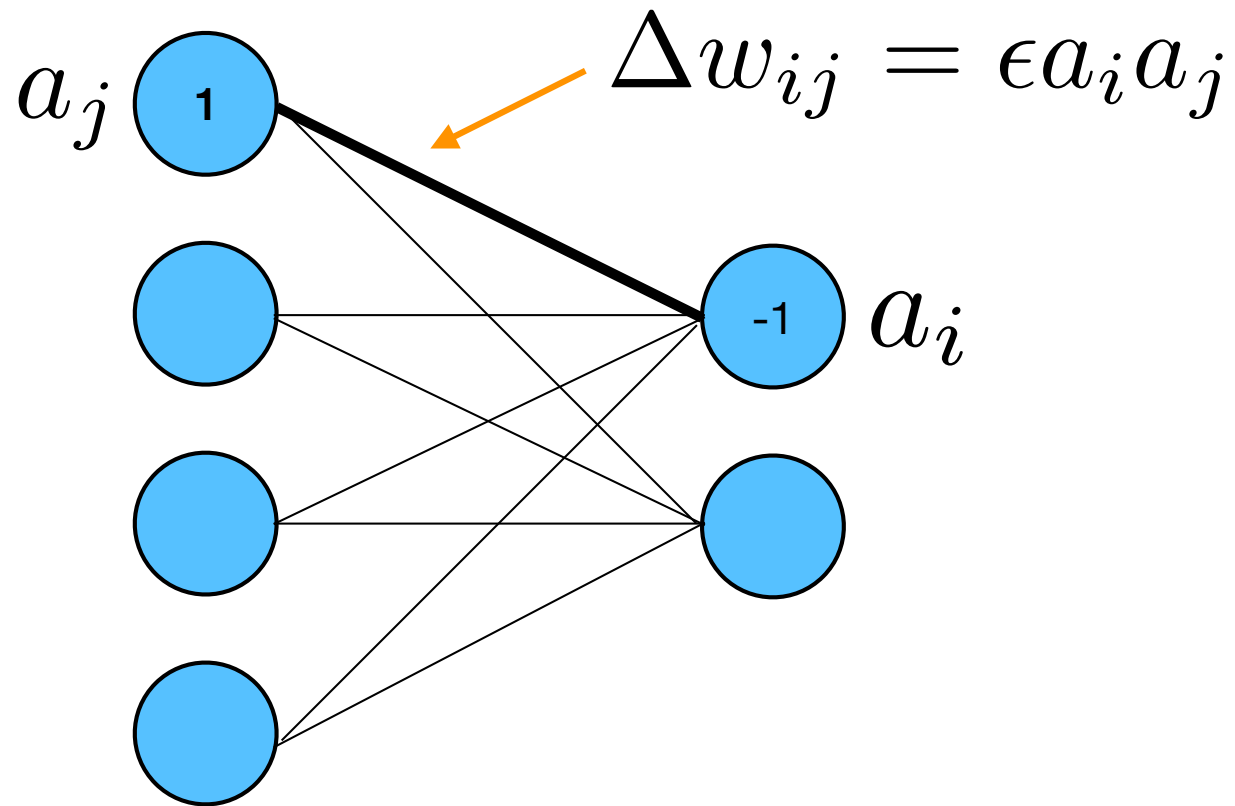
$$\begin{aligned} \mathbf{u}_i^T \mathbf{U} \mathbf{V}^T &= [\langle \mathbf{u}_i, \mathbf{u}_0 \rangle, \dots, \langle \mathbf{u}_i, \mathbf{u}_i \rangle, \dots, \langle \mathbf{u}_i, \mathbf{u}_n \rangle] \mathbf{V}^T \\ &= [0, \dots, \|\mathbf{u}_i\|^2, \dots, 0] \mathbf{V}^T \\ &= \|\mathbf{u}_i\|^2 \mathbf{v}_i^T \end{aligned}$$

So if the inputs are orthogonal, retrieval will be correct up to a positive scale factor, which could be corrected by thresholding in our binary case.

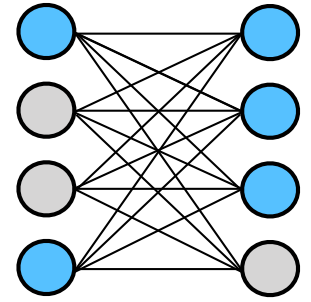
Hebb's Rule



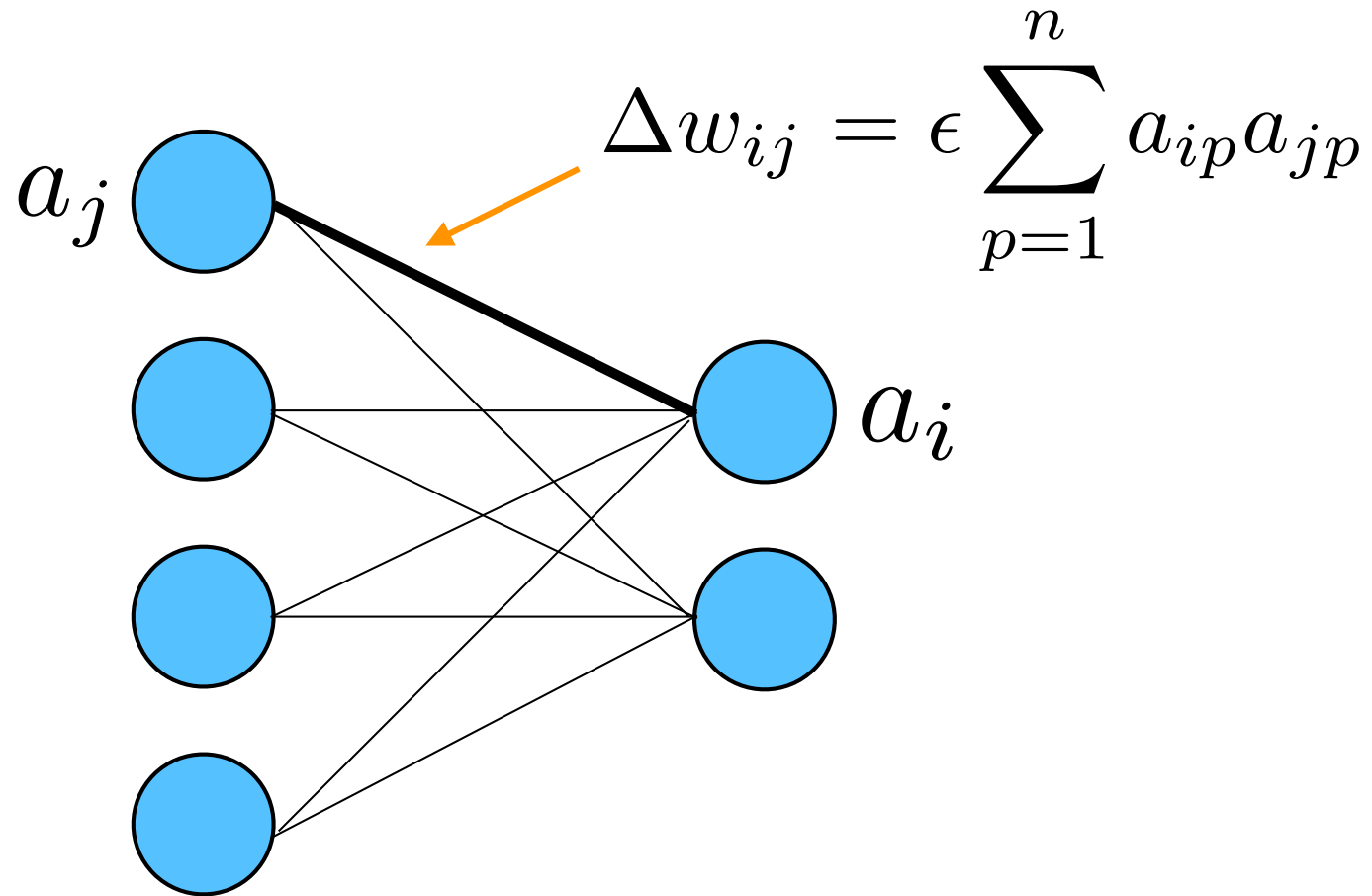
- Incremental version:



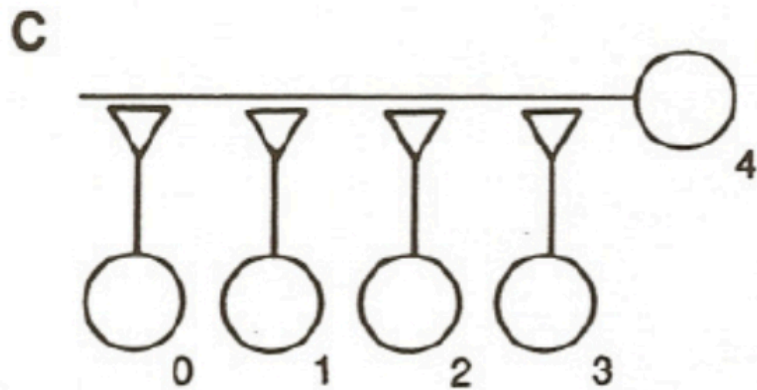
Hebb's Rule



- Incremental version:



Hebb's Rule



D

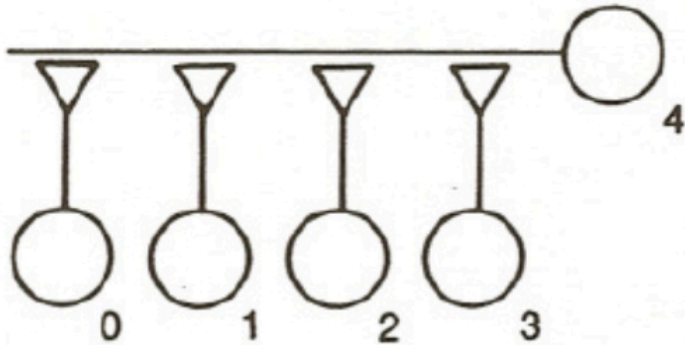
	Input				Output
	0	1	2	3	4
	+	-	+	-	+
	+	+	+	+	+
	+	+	+	-	-
	+	-	-	+	-

$$\Delta w_{40} = \epsilon \sum_{p=0}^3 a_{4p} a_{0p}$$

$$= \epsilon + \epsilon - \epsilon - \epsilon = 0$$

Hebb's Rule

C



D

Input				Output
0	1	2	3	4
+	-	+	-	+
+	+	+	+	+
+	+	+	-	-
+	-	-	+	-

$$\Delta w_{40} = 0$$

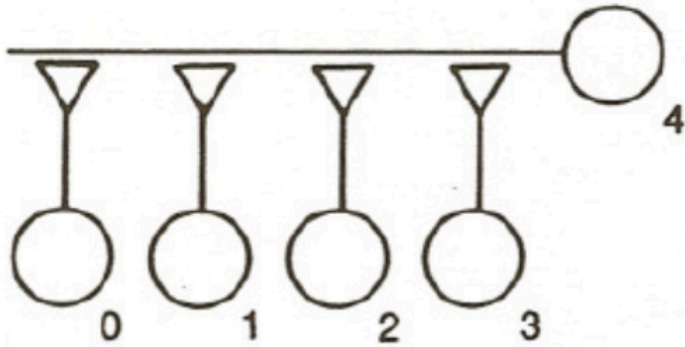
$$\Delta w_{41} = 0$$

$$\Delta w_{42} = 2\epsilon$$

$$\Delta w_{43} = 0$$

Hebb's Rule

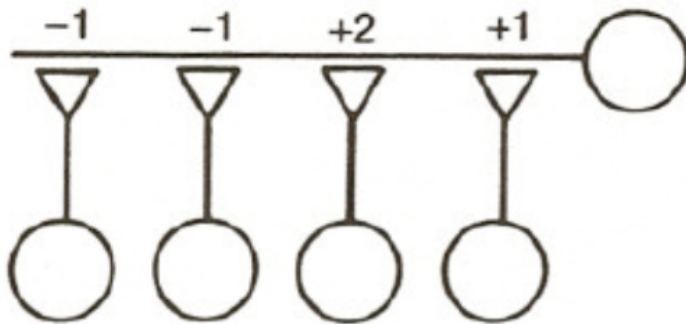
C



D

Input				Output
0	1	2	3	4
+	-	+	-	+
+	+	+	+	+
+	+	+	-	-
+	-	-	+	-

E



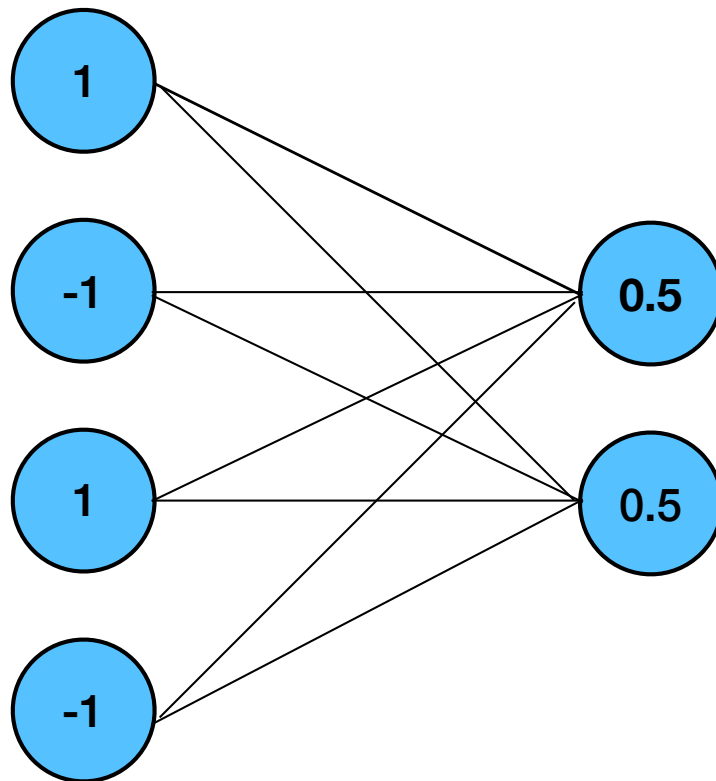
Delta Rule

input: [1, -1, 1, -1]

output: [0.5, 0.5]

target: [1, 1]

error, or tss: $(1-0.5)^2 + (1-0.5)^2 = 0.25 + 0.25 = 0.5$



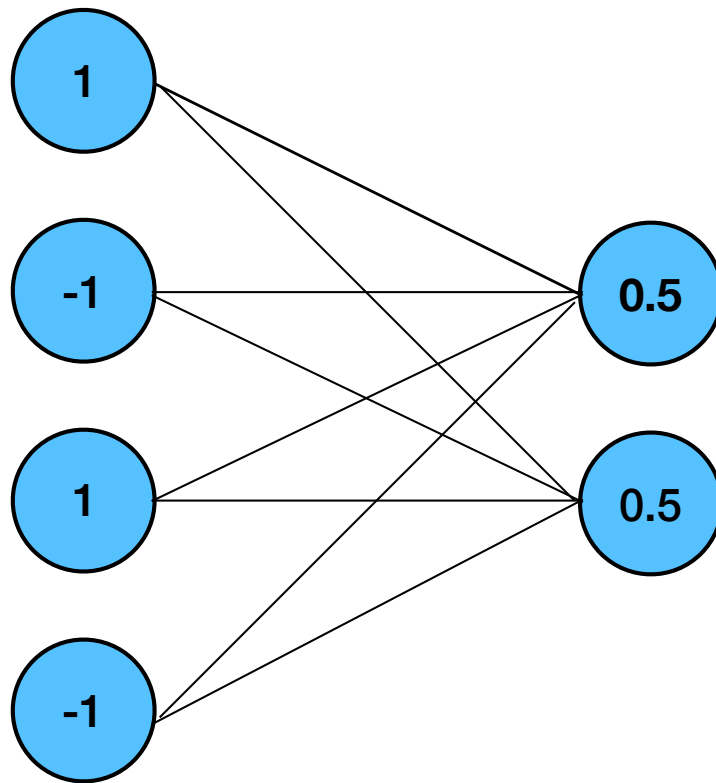
Delta Rule

input: [1, -1, 1, -1]

output: [0.5, 0.5]

target: [1, 1]

error, or tss: $(1-0.5)^2 + (1-0.5)^2 = 0.25 + 0.25 = 0.5$



$$\Delta w_{ij} = \epsilon \frac{\partial E}{\partial w_{ij}}$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}}$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}}$$



$$\frac{\partial E}{\partial \hat{y}_i} = \frac{1}{2} \sum_k \frac{\partial}{\partial \hat{y}_i} (y_k - \hat{y}_k)^2$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}}$$

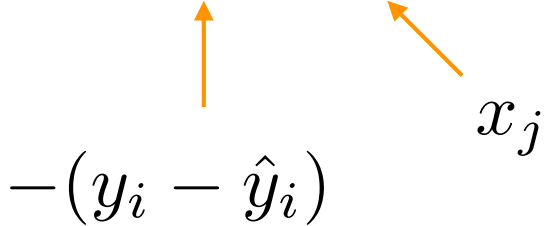


$$\begin{aligned} \frac{\partial E}{\partial \hat{y}_i} &= \frac{1}{2} \sum_k \frac{\partial}{\partial \hat{y}_i} (y_k - \hat{y}_k)^2 \\ &= \frac{1}{2} \frac{\partial}{\partial \hat{y}_i} (y_i - \hat{y}_i)^2 = \frac{1}{2} * 2(y_i - \hat{y}_i) * -1 \\ &= -(y_i - \hat{y}_i) \end{aligned}$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}}$$


$-(y_i - \hat{y}_i)$

x_j

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}} = -(y_i - \hat{y}_i) x_j$$

Delta Rule

$$E = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2$$

$$\Delta w_{ij} = -\epsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_{ij}} = -(y_i - \hat{y}_i) x_j$$

Update rule:

$$\Delta w_{ij} = \epsilon (y_i - \hat{y}_i) x_j$$

Homework question + Ir demo

- (...)

Linear Predictability Constraint

- These are linear models: outputs can only be expressed as a linear combination of inputs
- Usually, we need nonlinear functions, e.g. to capture mapping from images to class labels (not just a linear combination of pixels)