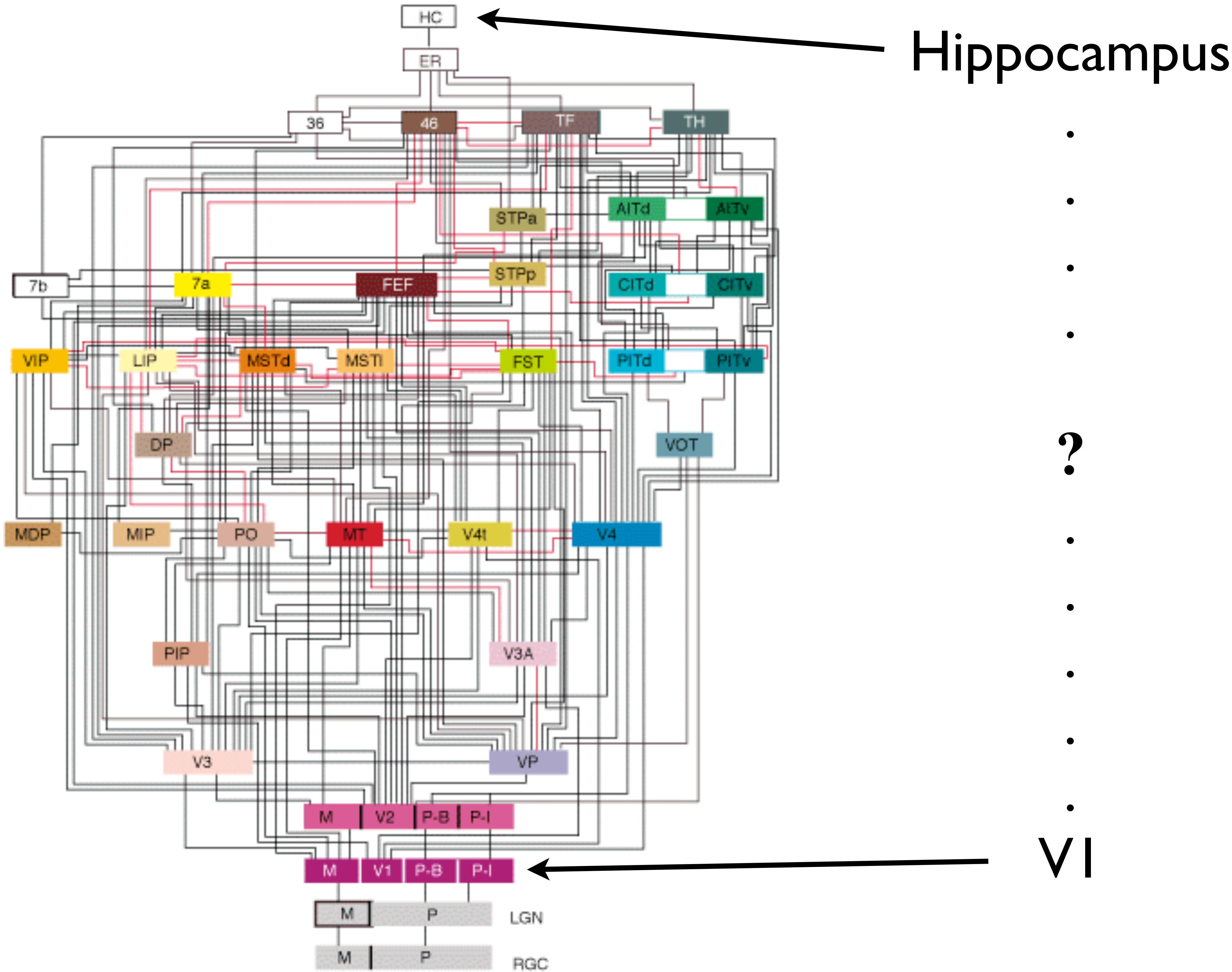
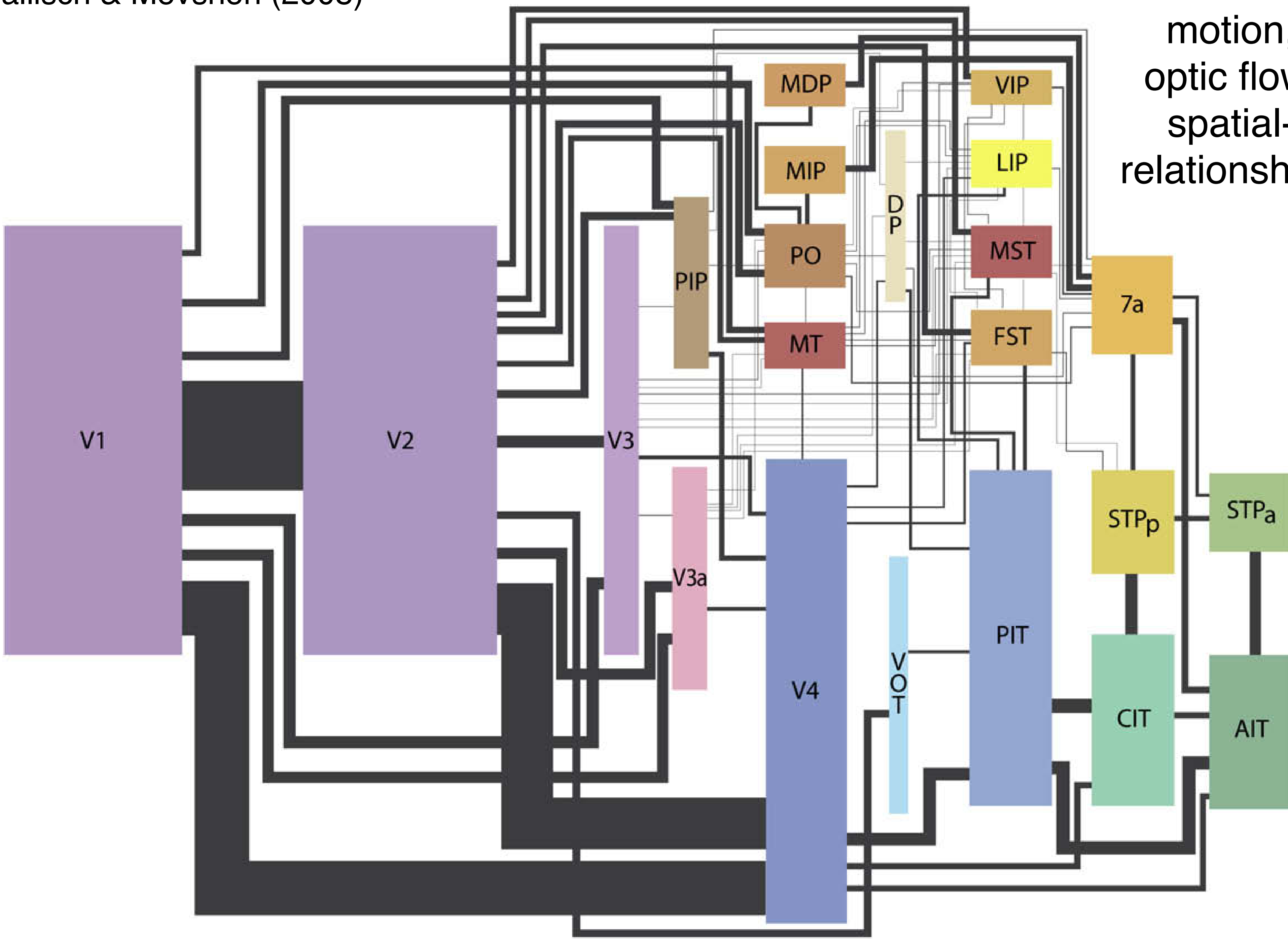
Representation Learning


HC
ER
36
46
TF
TH
STPa
AITd
AITv
7b
7a
FEF
STPp
CITd
CITv
VIP
LIP
MSTd
MSTl
FST
PITd
PITv
DP
VOT
MDP
MIP
PO
MT
V4t
V4
PIP
V3A
V3
VP
M
V2
P-B
P-I
M
V1
P-B
P-I
M
P
LGN
M
P
RGC
Hippocampus
?
V1

Wallisch & Movshon (2008)



motion, optic flow, spatial-relationships

lines, edges

surface boundaries

?

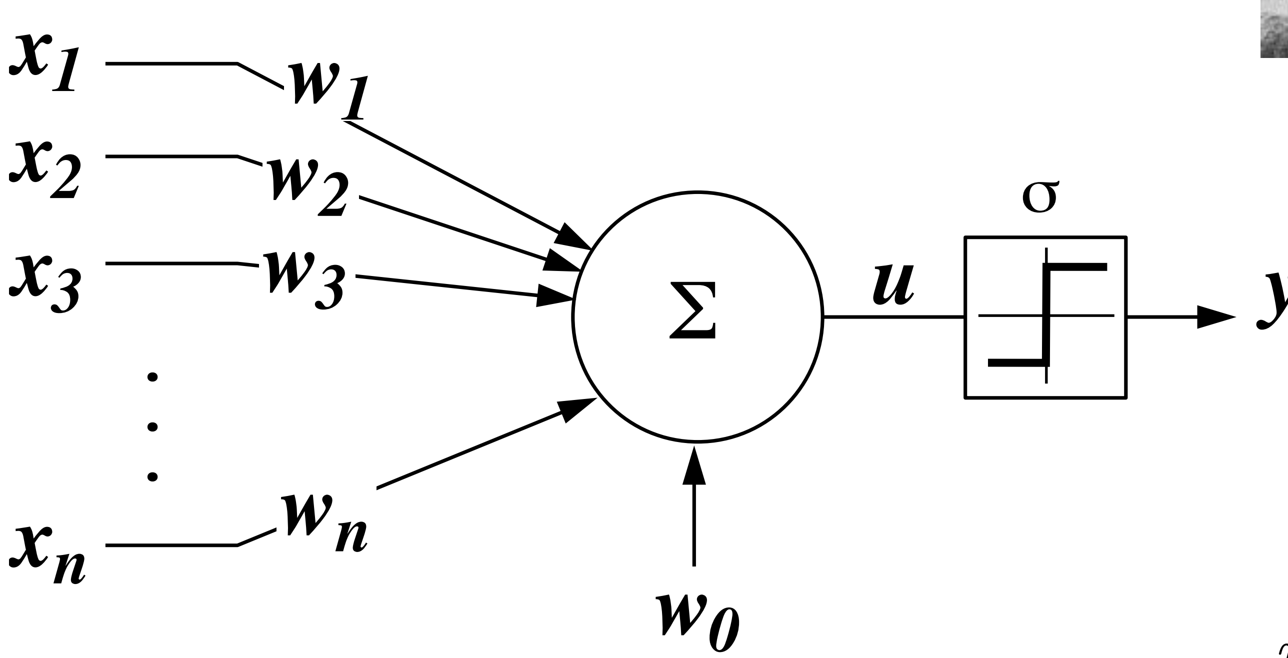
objects

faces

Supervised learning

Perceptron model

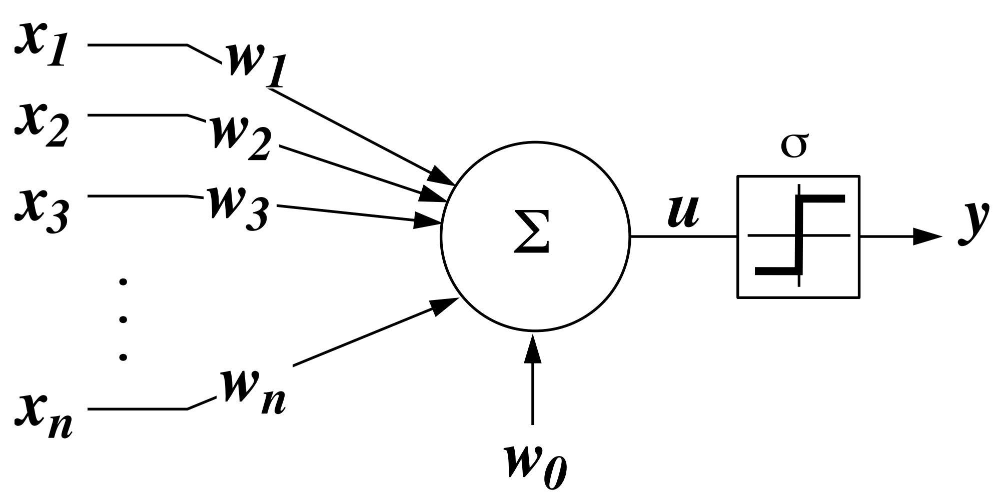
(Rosenblatt, ca. 1960)



$$u = w_0 + \sum_{i=1}^{n} w_i \, x_i$$

$$y = \sigma(u)$$

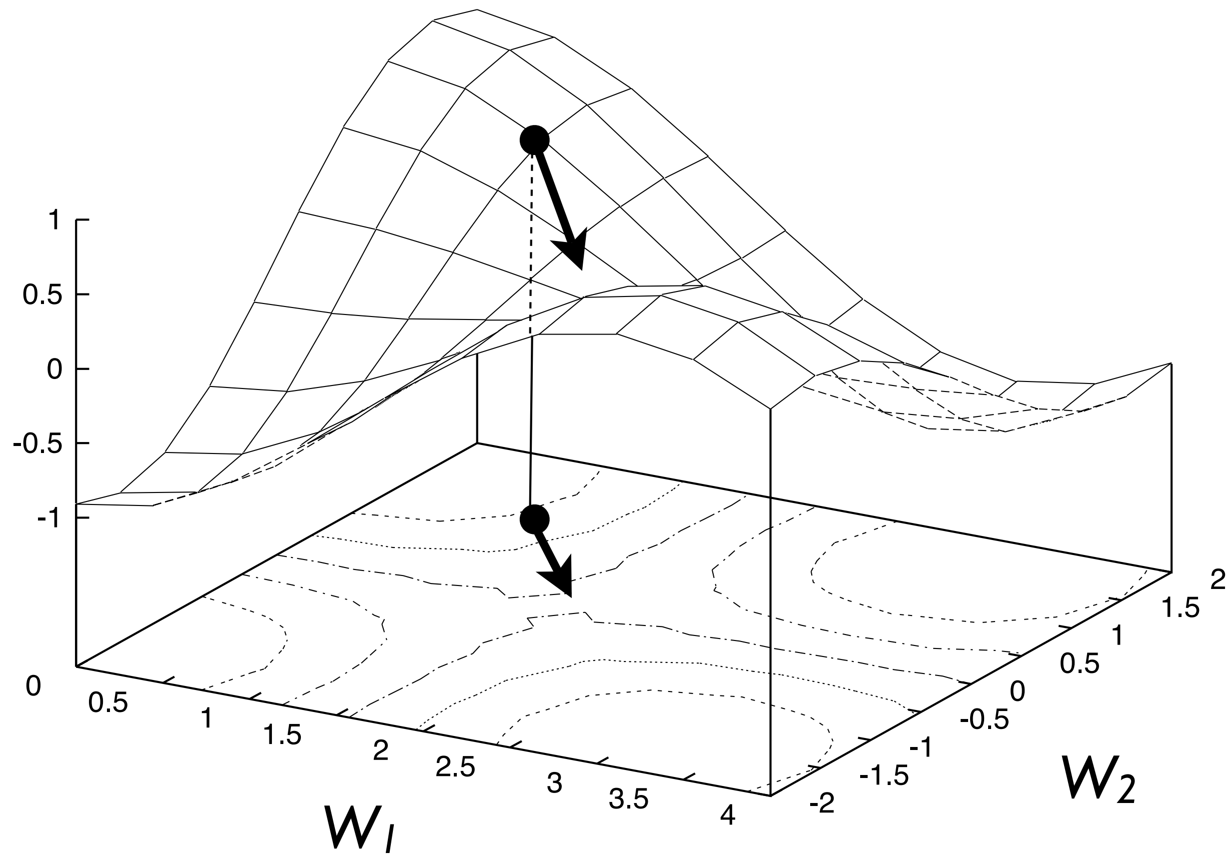
Perceptron learning rule
(Rosenblatt 1962)



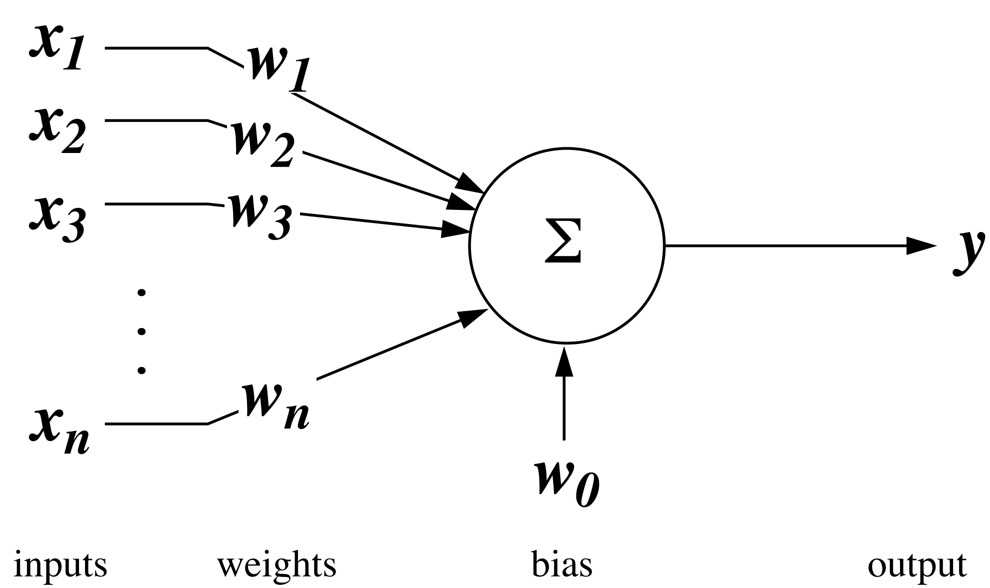
$$\Delta w_k = \begin{cases} 2\eta\, T^{(\alpha)}\, x_k^{(\alpha)} & y^{(\alpha)} \neq T^{(\alpha)} \\ 0 & \text{otherwise} \end{cases}$$

$$= \eta\left(T^{(\alpha)} - y^{(\alpha)}\right) x_k$$

Gradient descent in weight space



Linear neuron learning rule (Widrow & Hoff 1960)

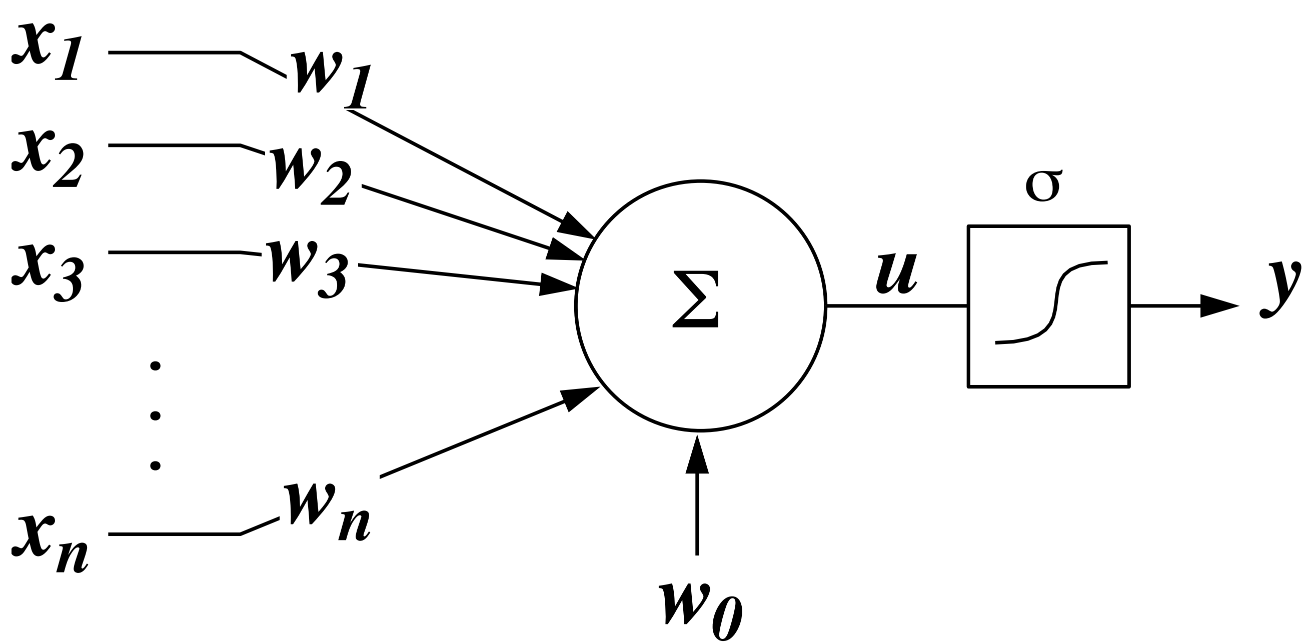


Objective function $\longrightarrow$ Learning rule

$$E = \frac{1}{2} \sum_{\alpha} \left[T^{(\alpha)} - y^{(\alpha)} \right]^2$$

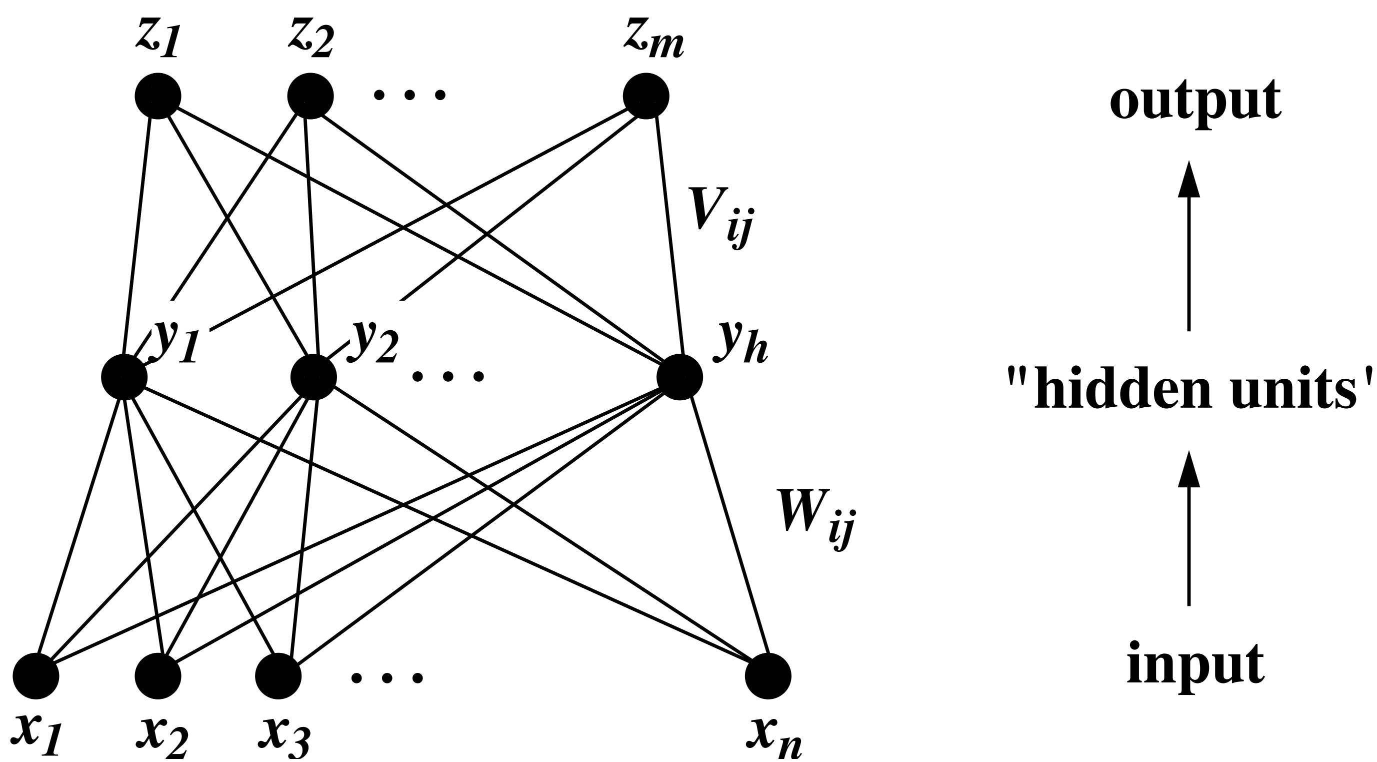
$$\begin{aligned} \Delta w_k &= -\eta \, \frac{\partial E}{\partial w_k} \\ &= \eta \sum_{\alpha} \delta^{(\alpha)} \, x_k^{(\alpha)} \\ \delta^{(\alpha)} &= T^{(\alpha)} - y^{(\alpha)} \end{aligned}$$

Linear neuron with output non-linearity



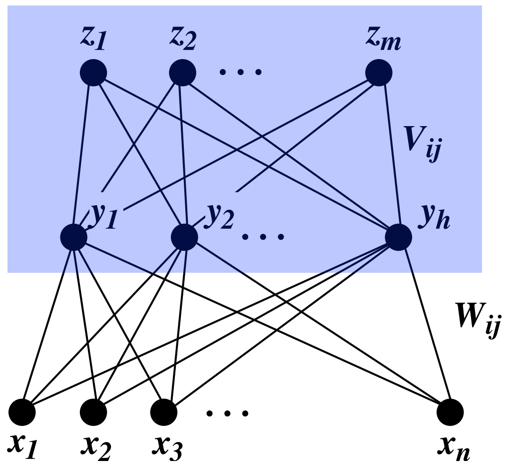
$$y = \sigma(u) \equiv \frac{1}{1 + e^{-\beta u}}$$

Two-layer network



$$
\begin{aligned}
z_i &= \sigma\Big(\sum_j V_{ij} y_j\Big) \\
y_i &= \sigma\Big(\sum_j W_{ij} x_j\Big)
\end{aligned}
$$

Learning rule for output layer



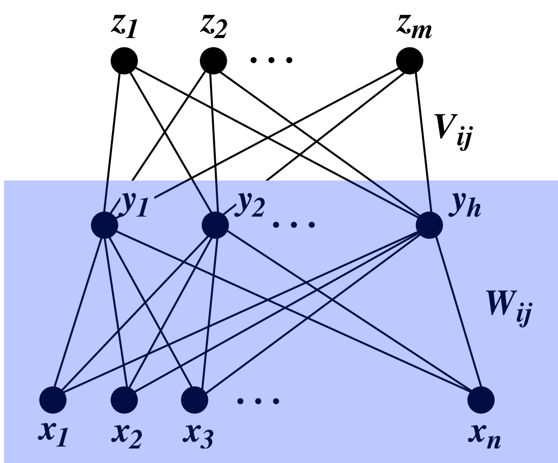
$$E^{(\alpha)} = \frac{1}{2} \sum_i \left[T_i^{(\alpha)} - z_i(\mathbf{x}^{(\alpha)}) \right]^2$$

$$\begin{aligned}
\Delta V_{ij} &= -\eta \frac{\partial E}{\partial V_{ij}} \\
&= [T_i - z_i(\mathbf{x})] \frac{\partial z_i(\mathbf{x})}{\partial V_{ij}} \\
&= [T_i - z_i(\mathbf{x})]\, \sigma'(u_{z_i})\, y_j \\
&= \boxed{\delta_{z_i}\, y_j}
\end{aligned}$$

where

$$\begin{aligned}
\delta_{z_i} &= [T_i - z_i(\mathbf{x})]\, \sigma'(u_{z_i}) \\
u_{z_i} &= \sum_j V_{ij}\, y_j
\end{aligned}$$

Learning rule for hidden layer



$$\begin{aligned} \Delta W_{kl} &= -\eta \frac{\partial E}{\partial W_{kl}} \\ &= \eta \sum_i [T_i - z_i(\mathbf{x})] \frac{\partial z_i(\mathbf{x})}{\partial W_{kl}} . \end{aligned}$$

$$\frac{\partial z_i(\mathbf{x})}{\partial W_{kl}} = \frac{\partial z_i(\mathbf{x})}{\partial y_k} \frac{\partial y_k}{\partial W_{kl}}$$

$$\begin{aligned} \Delta W_{kl} &= \eta \sum_i [T_i - z_i(\mathbf{x})] \sigma'(u_{z_i}) V_{ik} \sigma'(u_{y_k}) x_l \\ &= \boxed{\eta \, \delta_{y_k} x_l} \end{aligned}$$

where $\delta_{y_k} = \sigma'(u_{y_k}) \sum_i \delta_{z_i} V_{ik}$ ← back-propagation of error

Unsupervised learning

Redundancy Reduction as a Strategy for Unsupervised Learning

A. Norman Redlich

a l i c e w a s b e g i n n i n g t o g e t v e r y t i r e d o f s i t t i n g b y h e r s i s t e r o n
t h e b a n k a n d o f h a v i n g n o t h i n g t o d o o n c e o r t w i c e s h e h a d p e e p
e d i n t o t h e b o o k h e r s i s t e r w a s r e a d i n g b u t i t h a d n o p i c t u r e s o r
c o n v e r s a t i o n s i n i t a n d w h a t i s t h e u s e o f a b o o k t h o u g h t a l i c e w
i t h o u t p i c t u r e s o r c o n v e r s a t i o n s s o s h e w a s c o n s i d e r i n g i n h e
r o w n m i n d a s w e l l a s s h e c o u l d f o r t h e h o t d a y m a d e h e r f e e l v e r y
s l e e p y a n d s t u p i d w h e t h e r t h e p l e a s u r e o f m a k i n g a d a i s y c h a i
n w o u l d b e w o r t h t h e t r o u b l e o f g e t t i n g u p a n d p i c k i n g t h e d a i s i
e s w h e n s u d d e n l y a w h i t e r a b b i t w i t h p i n k e y e s r a n c l o s e b y h e r
t h e r e w a s n o t h i n g s o v e r y r e m a r k a b l e i n t h a t n o r d i d a l i c e t h i
n k i t s o v e r y m u c h o u t o f t h e w a y t o h e a r t h e r a b b i t s a y t o i t s e l f o
h d e a r o h d e a r

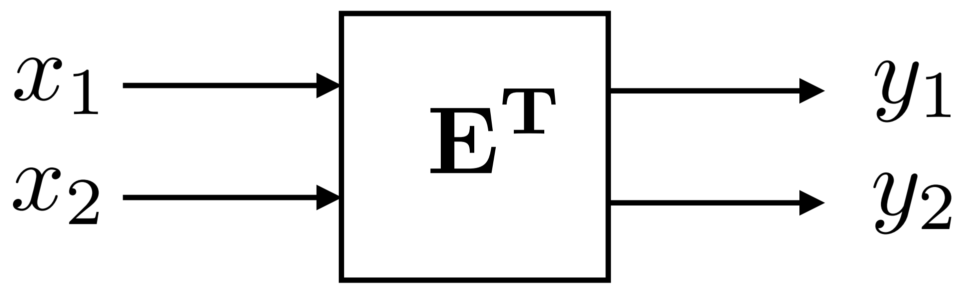
(Neural Computation, 1993)

alice was b e g in n ing t o g e t very t i r e d o f s i t t ing b y her s i s t e r o n the b a n
k and o f h a v ing n o th ing t o d o o n c e o r t w i c e she h a d p e e p e d in t o the b o
o k her s i s t e r was r e a d ing but i th a d n o p i c t u r e s o r c o n ver s a t i o n s in i
t and w h a t i s the u s e o f a b o o k th ough t alice with ou t p i c t u r e s o r c o n ver
s a t i o n s s o she was c o n s i d e r ing in her o w n m in d a s w e l l a s she c ould for
the h o t d a y m a d e her f e e l very s l e e p y and s t u p i d w he the r the p l e a s u r
e o f m a k ing a d a i s y c h a in w ould b e w o r th the t r ou b l e o f g e t t ing u p
and p i c k ing the d a i s i e s w he n s u d d e n l y a w h i t e r a b b i t with p in k e y e
s r a n c l o s e b y her the r e was n o th ing s o very r e m a r k a b l e in that n o r d i
dalic e th in k i t s o very m u c h ou t o f the w a y t o he a r the r a b b i t s a y t o i t s e
l f o h d e a r o h d e a r

alice was beginning toget verytiredof sitting by hersister ontheban k and of having nothingtodo onceortwice shehad peeped intothe book hersister was read ing but ithad no pictures or conversation s in it and what is theuseof a book thoughtalice without pictures or conversation s so shewas consider ing in her ow n mind aswell asshecould for the hot day made her feel very sleepy and stupid whether the plea sure of making ad a is y ch a in wouldbe wor th the trouble of gettingupand p i c king the d a is ies when suddenly a whiterabbit with p in k eyes r an close by her therewas nothing so very remark able in that n or did alice think it so very much outoftheway to hear therabbit saytoitself ohdear ohdear

Hebbian Learning and PCA

PCA
(Principal Components Analysis)



$$\begin{aligned} \langle x_1\, x_2 \rangle &= c_{12} \\ &\neq 0 \end{aligned}$$

$$\begin{aligned} y_1 &= \mathbf{e}_1 \cdot \mathbf{x} \\ y_2 &= \mathbf{e}_2 \cdot \mathbf{x} \end{aligned}$$

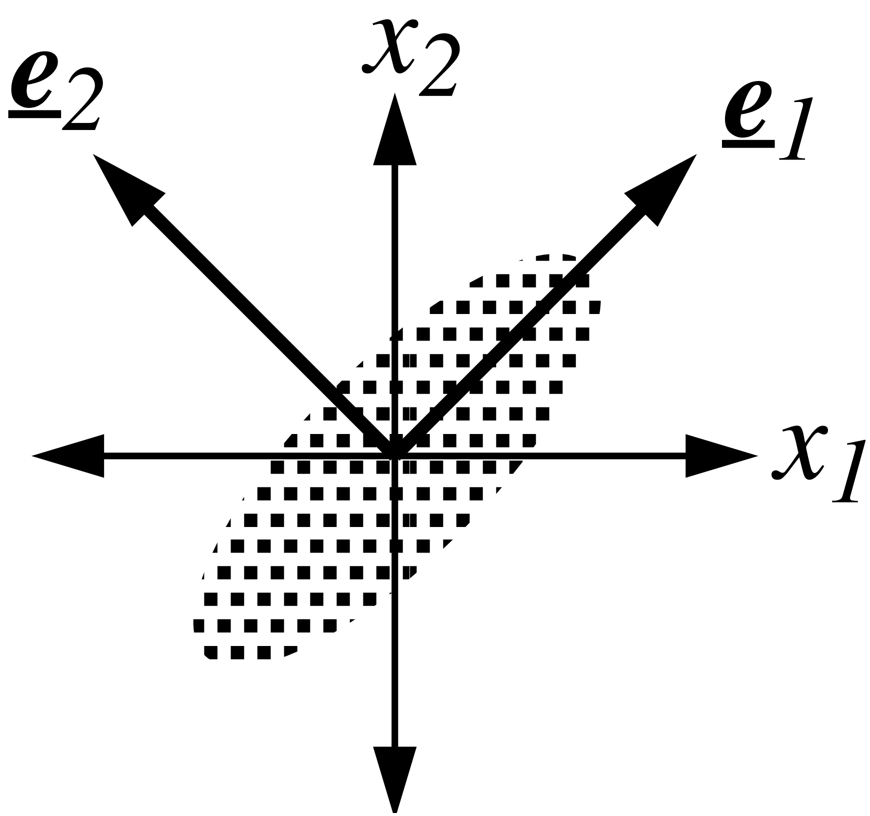
$$\begin{aligned} \langle y_1\, y_2 \rangle &= \langle y_1 \rangle \langle y_2 \rangle \\ &= 0 \end{aligned}$$

$$\mathbf{E} = \begin{bmatrix} | & | \\ \mathbf{e}_1 & \mathbf{e}_2 \\ | & | \end{bmatrix}$$

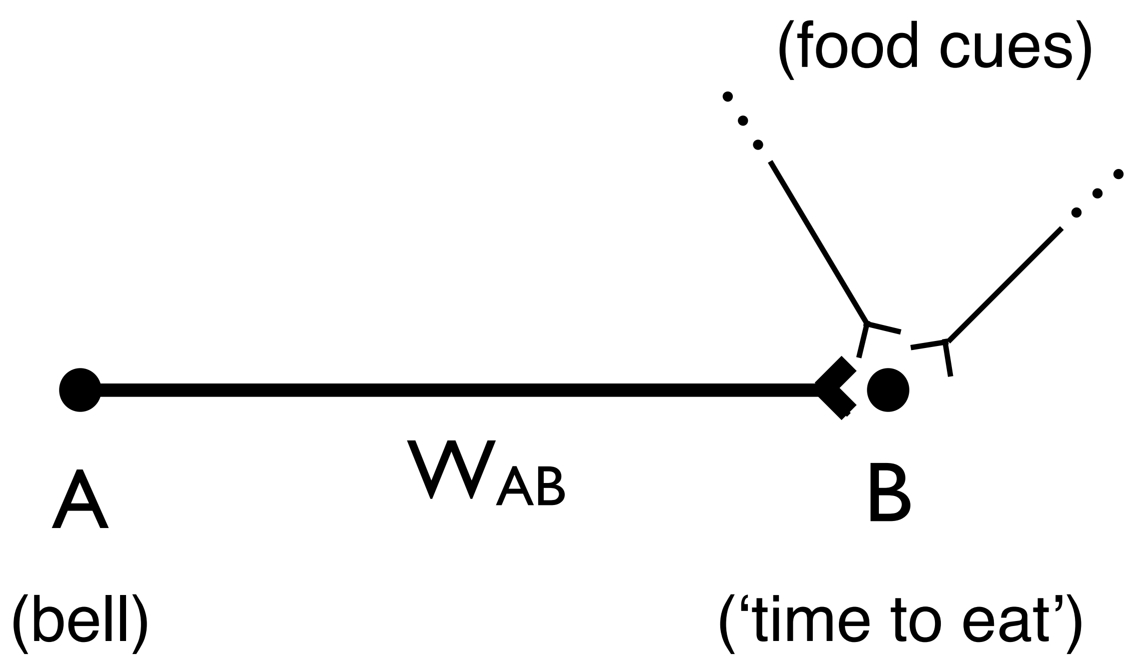
$$\mathbf{e}_1 \cdot \mathbf{e}_2 = 0$$

$$|\mathbf{e}_1| = |\mathbf{e}_2| = 1$$

PCA
(Principal Components Analysis)

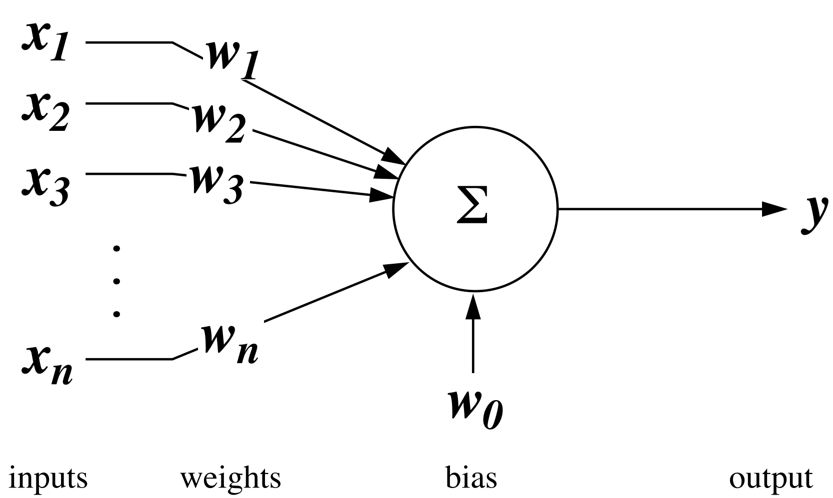


Hebbian learning



$$\Delta W_{AB} \propto < A\, B >$$

Linear Hebbian learning



$$\dot{w}_i \propto \langle y\, x_i \rangle$$

$$y = \sum_j w_j\, x_j$$

$$\dot{w}_i \propto \left\langle \sum_j w_j\, x_j\ x_i \right\rangle$$

$$= \sum_j w_j \,\langle\, x_j\ x_i \rangle$$

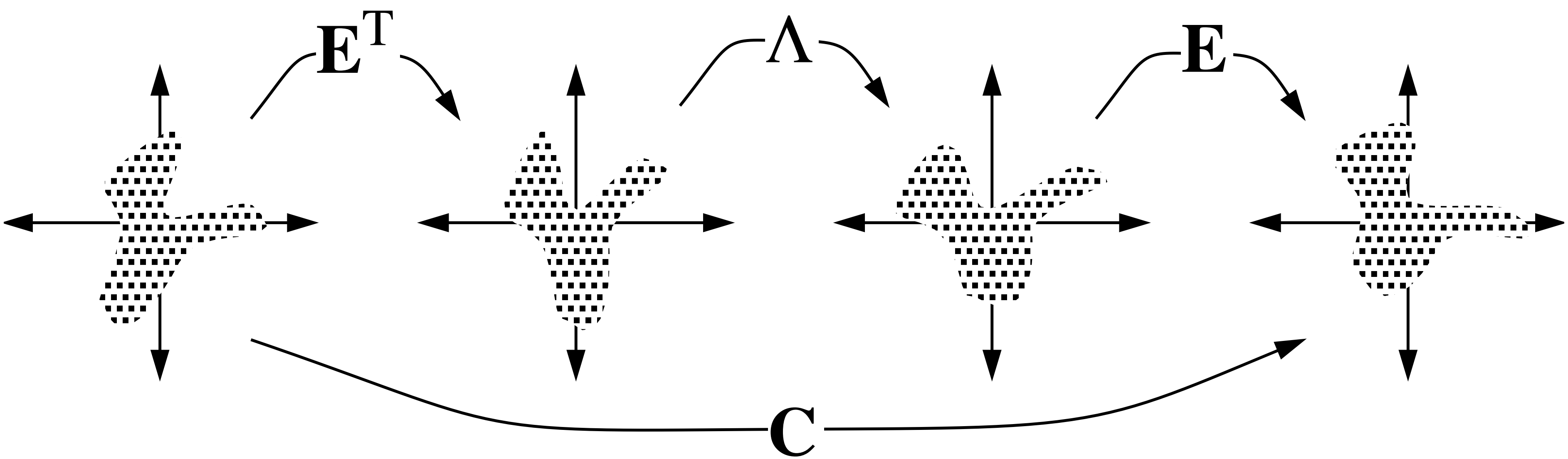
$$\boxed{\dot{\mathbf{w}} \propto \mathbf{C}\,\mathbf{w}} \qquad C_{ij} = \langle x_i\, x_j \rangle$$

$$\dot{\mathbf{w}} \propto \mathbf{C}\,\mathbf{w}$$

$$\dot{w}_1 \propto C_{11}\,w_1 + C_{12}\,w_2$$
$$\dot{w}_2 \propto C_{21}\,w_1 + C_{22}\,w_2$$

$$\mathbf{C} = \mathbf{E}\,\boldsymbol{\Lambda}\,\mathbf{E}^T$$

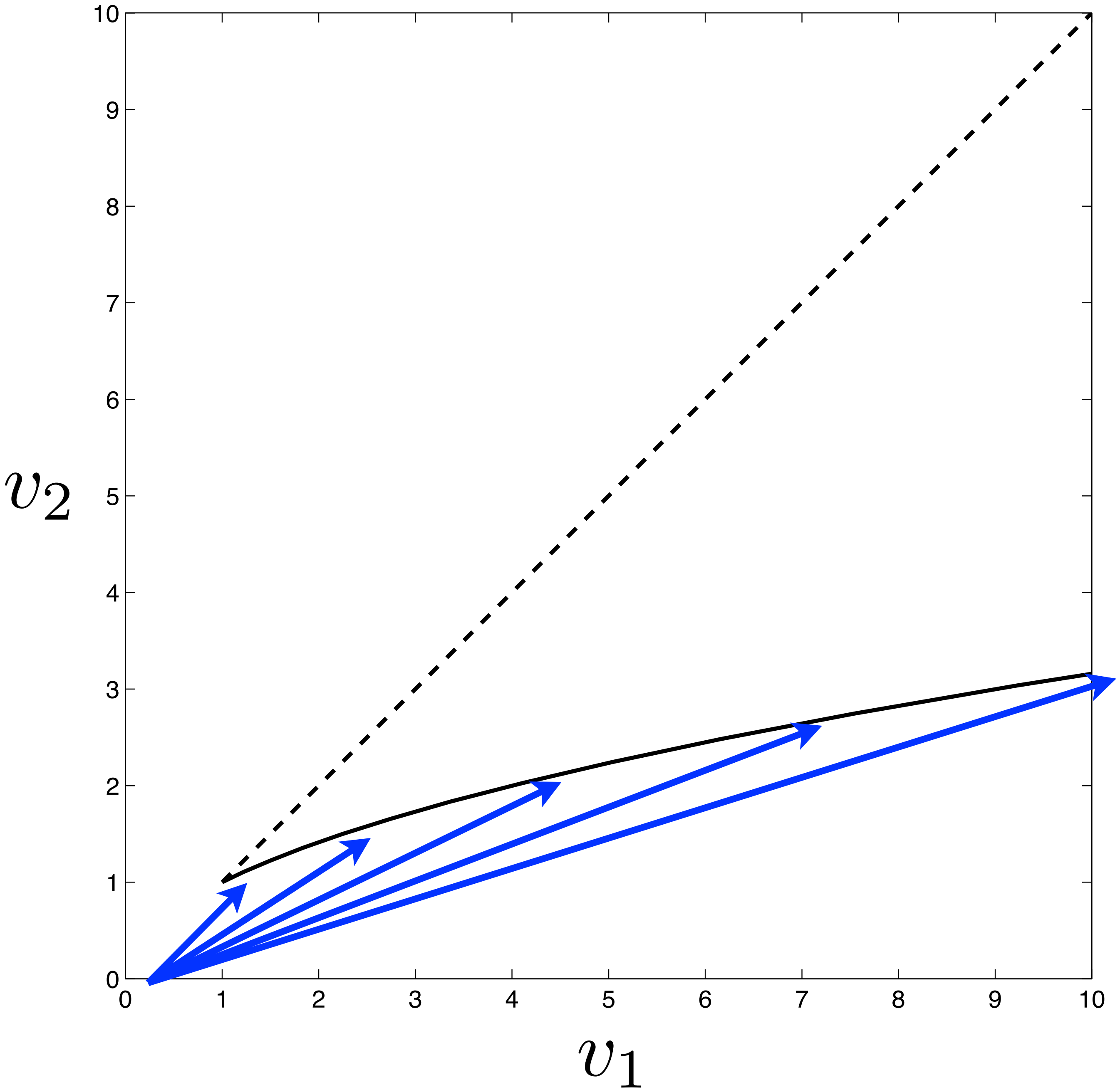
$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \\ | & | & & | \end{bmatrix} \qquad \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$



define: $\mathbf{v} = \mathbf{E}^{\mathbf{T}}\,\mathbf{w}$

$$
\begin{aligned}
\dot{\mathbf{w}} &\propto \mathbf{C}\,\mathbf{w} \\
&= \mathbf{E}\,\boldsymbol{\Lambda}\,\mathbf{E}^{T}\,\mathbf{w} \\
&= \mathbf{E}\,\boldsymbol{\Lambda}\,\mathbf{v} \\
\mathbf{E}^{T}\,\dot{\mathbf{w}} &\propto \mathbf{E}^{T}\,\mathbf{E}\,\boldsymbol{\Lambda}\,\mathbf{v} \\
\dot{\mathbf{v}} &\propto \boldsymbol{\Lambda}\,\mathbf{v}
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
\dot{v}_1 &\propto \lambda_1\,v_1 \\
\dot{v}_2 &\propto \lambda_2\,v_2
\end{aligned}
}
$$



$$v_1(t) = e^{\lambda_1 t}\, v_1(0)$$

$$v_2(t) = e^{\lambda_2 t} v_2(0)$$

$$\mathbf{w} = \mathbf{E}\,\mathbf{v}$$

$$\mathbf{w} \rightarrow \alpha\,\mathbf{e}_1$$

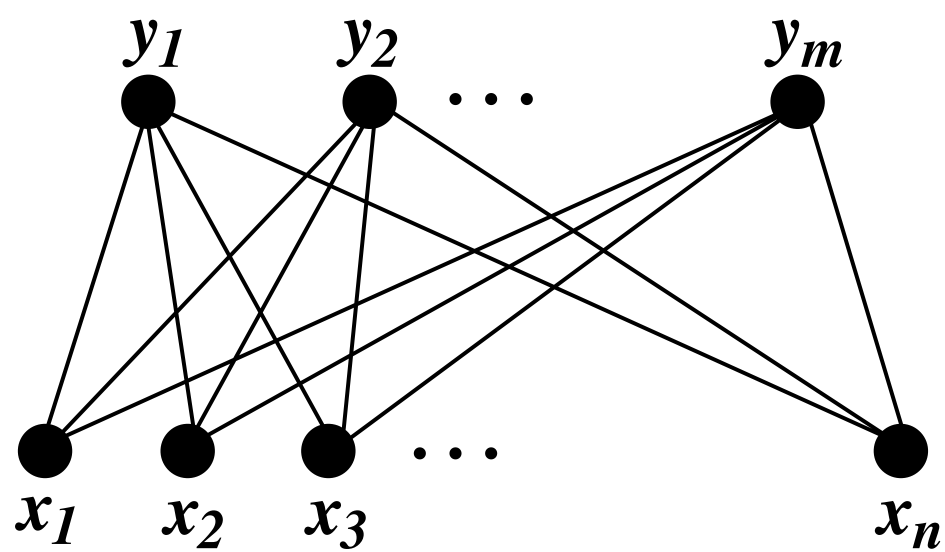
Constraining the growth of the weight vector

Oja's rule

$$\Delta w_i = \eta\, y\, (x_i - y\, w_i)$$

or $$\Delta \mathbf{w} = \eta\, y\, (\mathbf{x} - y\, \mathbf{w})$$

Multiple output units



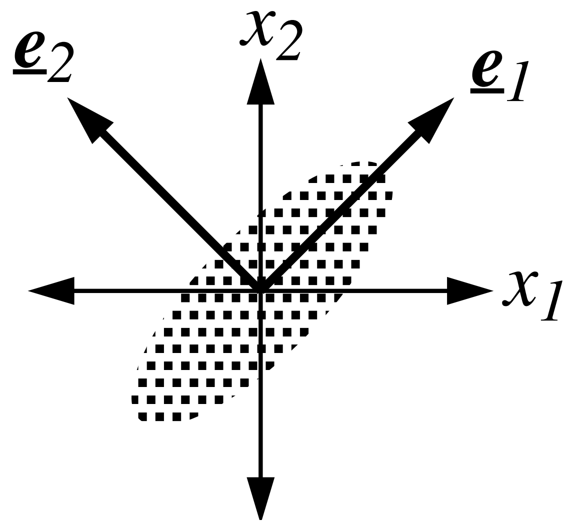
Sanger's rule: $$\Delta w_{ij} = \eta \, y_i \, (x_j - \sum_{k=1}^{i} y_k \, w_{kj})$$

Competitive Learning

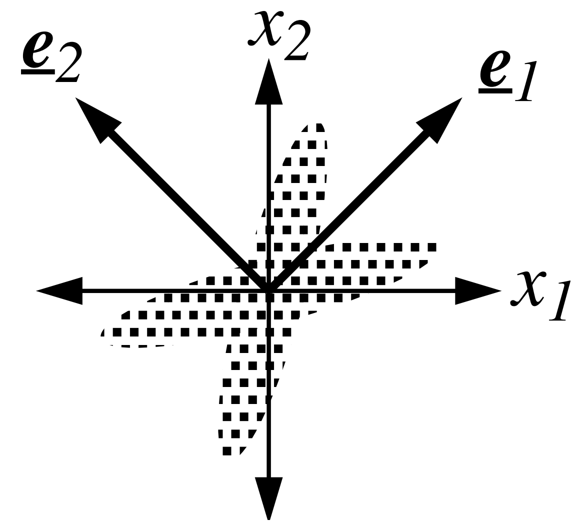
Gaussian

non-Gaussian

a.



b.



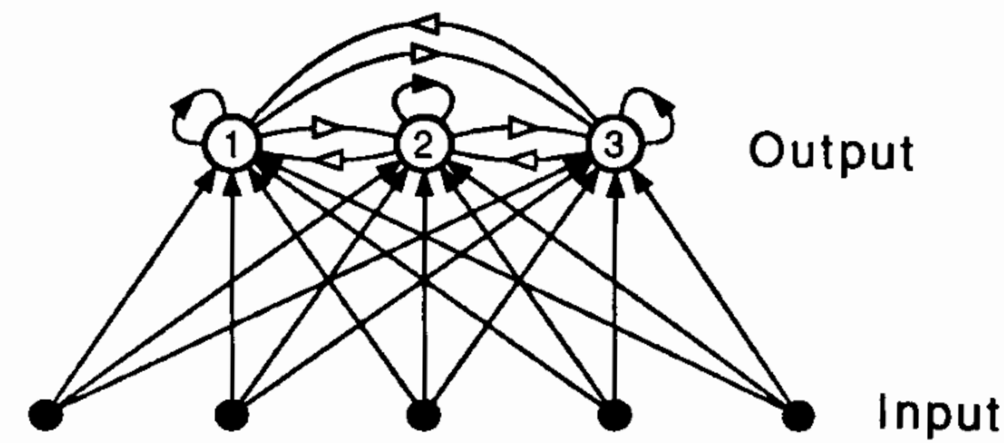
Non-linear Hebbian learning

Non-linear neuron: $$y = f(\sum_i w_i\, x_i)$$

Hebbian learning then yields:

$$\begin{aligned}
\Delta w_i &\propto y\, x_i \\
&= f(\sum_i w_j\, x_j)\, x_i \\
&= k_0\, x_i + k_1 \sum_j w_j\, x_j\, x_i \\
&\quad + k_2 \sum_{jk} w_j\, w_k\, x_k\, x_j\, x_i + \ldots
\end{aligned}$$

Winner-take-all learning



Network:

$$u_i = \sum_j w_{ij} x_j$$

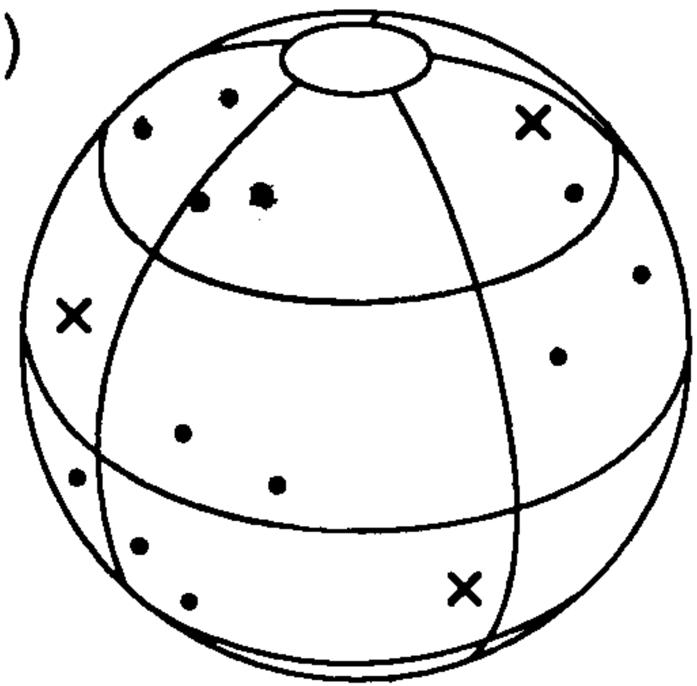
$$y_i = \begin{cases} 1 & u_i > u_j \;\; \forall j \neq i \\ 0 & \text{otherwise} \end{cases}$$

Learning rule: $$\Delta w_{ij} = \eta \, y_i \, (x_j - w_{ij})$$

Winner-take-all learning

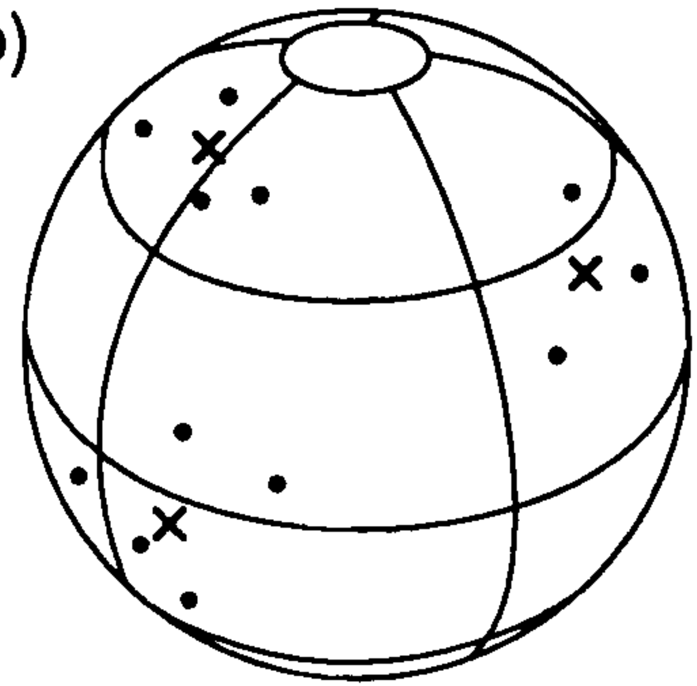
before learning

(a)



after learning

(b)



Winner-take-all learning

Energy function:

$$E(\{\mathbf{w}_i\}) = \frac{1}{2} \sum_{i,\mu} M_i^\mu \, |\mathbf{x}^{(\mu)} - \mathbf{w}_i|^2$$

Gradient descent:

$$\begin{aligned}
\Delta \mathbf{w}_i &= -\eta \, \frac{\partial E}{\partial \mathbf{w}_i} \\
&= \eta \sum_\mu M_i^\mu \, (\mathbf{x}^\mu - \mathbf{w}_i)
\end{aligned}$$

Biol. Cybern. 64, 165–170 (1990)

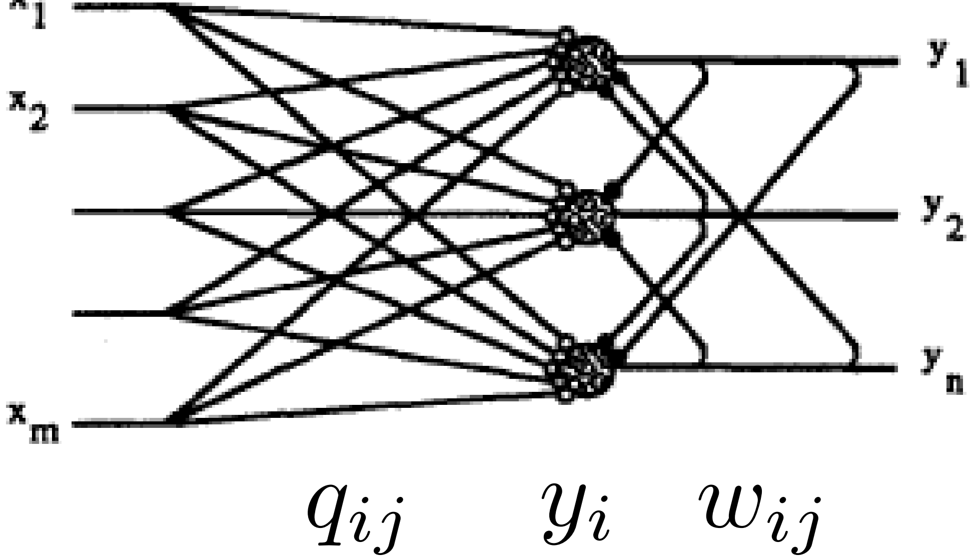
Biological Cybernetics

Forming sparse representations by local anti-Hebbian learning

P. Földiák

Physiological Laboratory, University of Cambridge, Downing Street, Cambridge CB2 3EG, United Kingdom

$$\frac{\mathrm{d}y_i^*}{\mathrm{d}t} = f\left(\sum_{j=1}^{m} q_{ij}x_j + \sum_{j=1}^{n} w_{ij}y_j^* - t_i\right) - y_i^*$$



anti-Hebbian rule–

$\Delta w_{ij} = -\alpha(y_i y_j - p^2)$

(if $i = j$ or $w_{ij} > 0$ then $w_{ij} := 0$)

Hebbian rule–

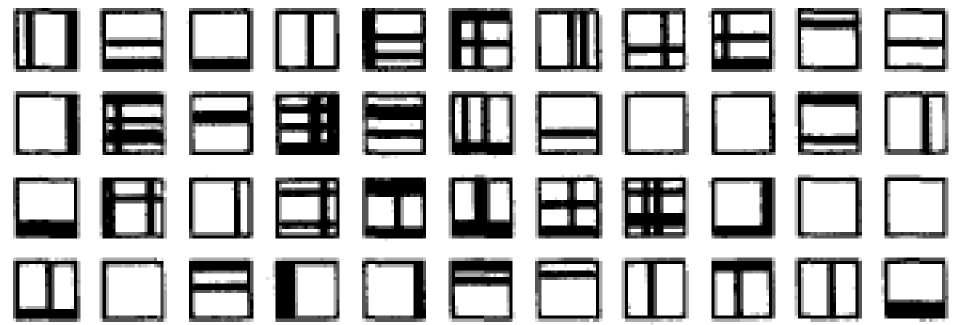
$\Delta q_{ij} = \beta y_i (x_j - q_{ij})$

threshold modification–

$\Delta t_i = \gamma(y_i - p)$.

Learning lines

Input patterns:



Learned weights: