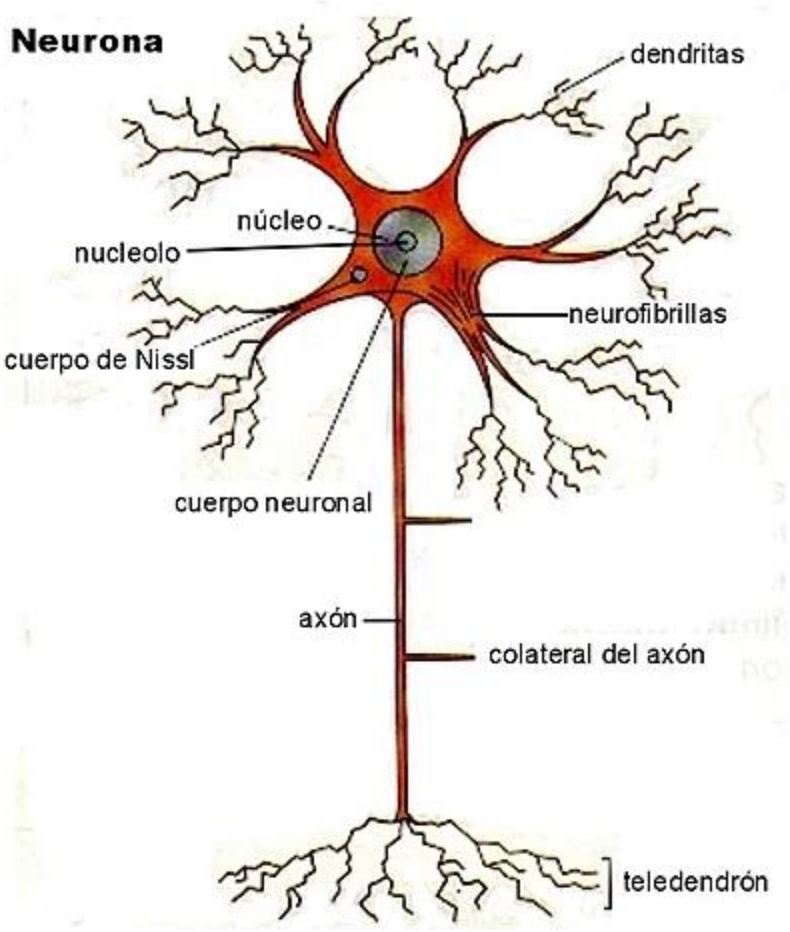
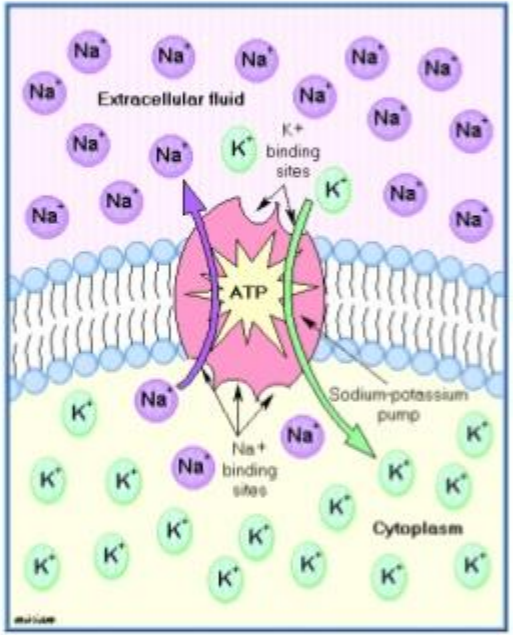
Redes Neuronales

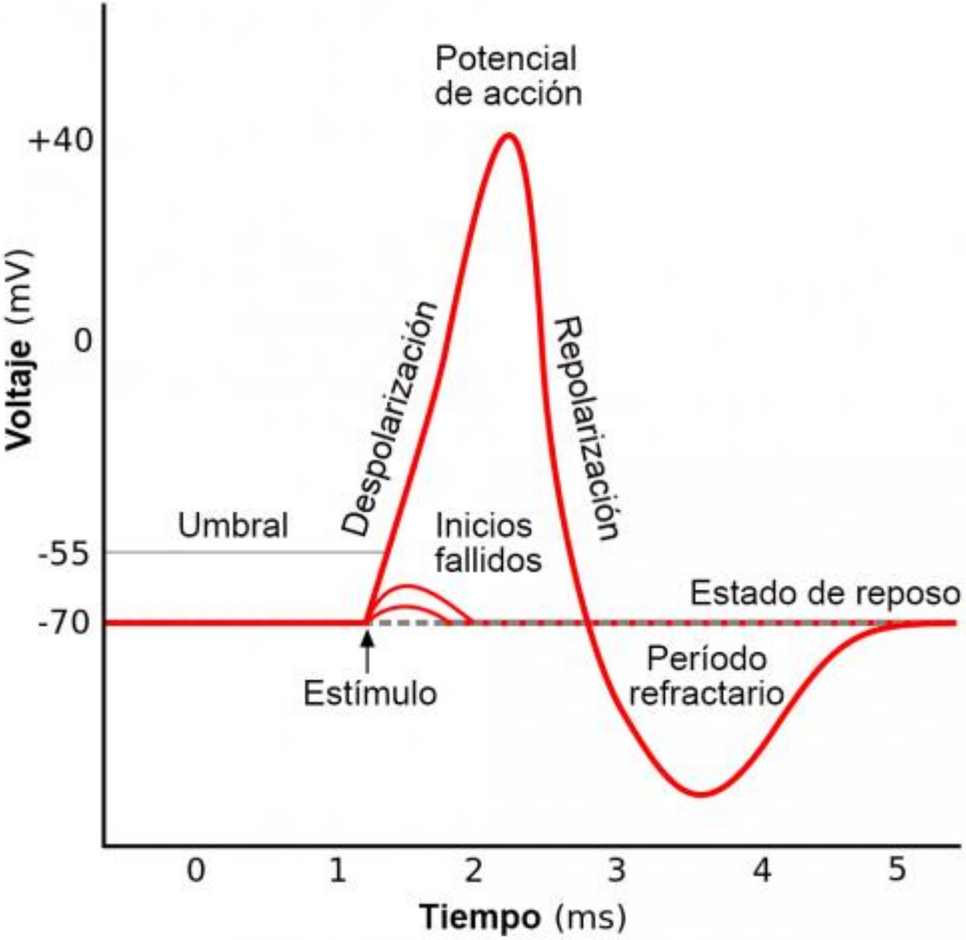
Ing. Juan M. Rodriguez

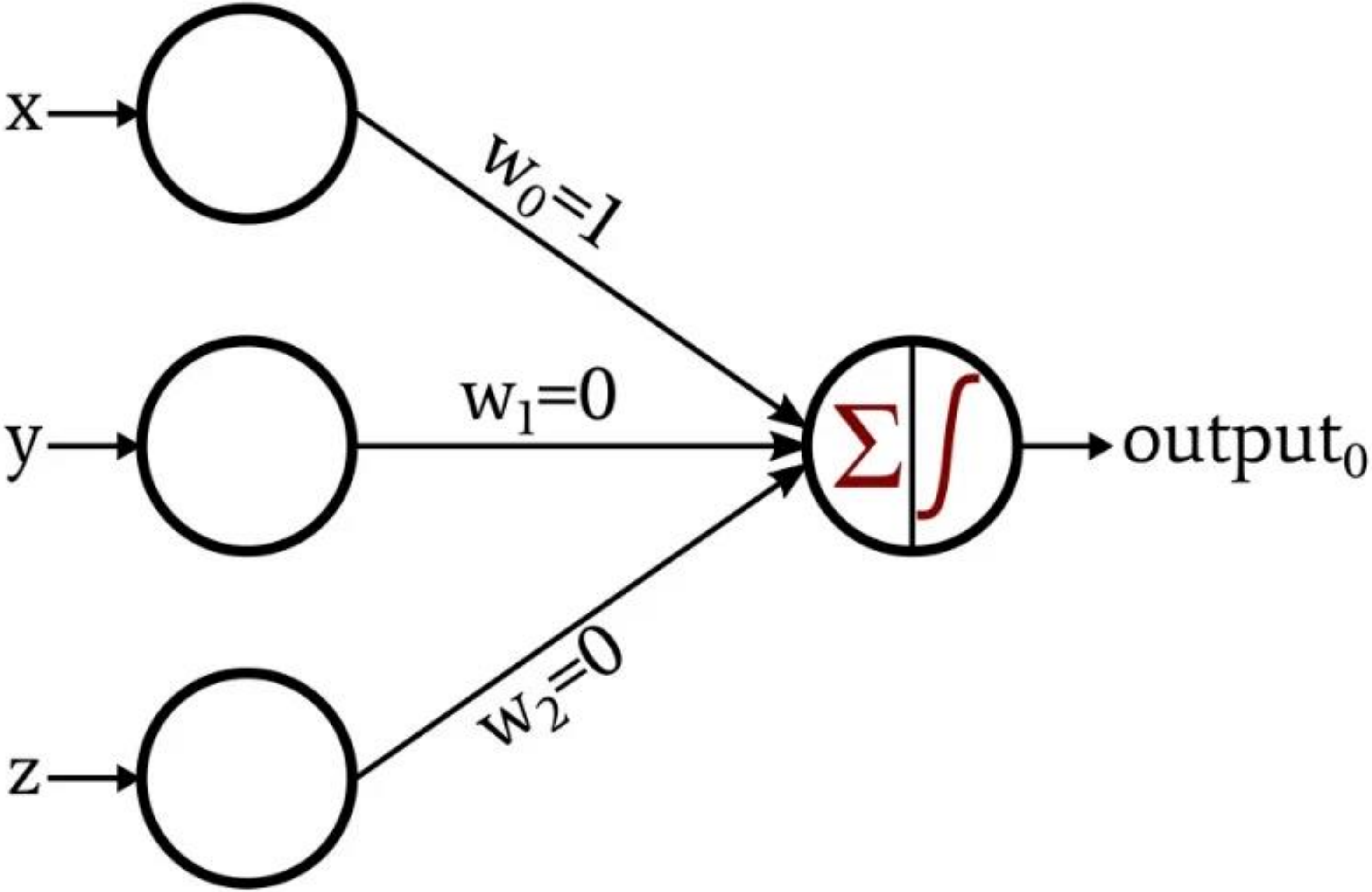

Neurona
dendritas
núcleo
nucleolo
neurofibrillas
cuerpo de Nissl
cuerpo neuronal
axón
colateral del axón
teledendrón

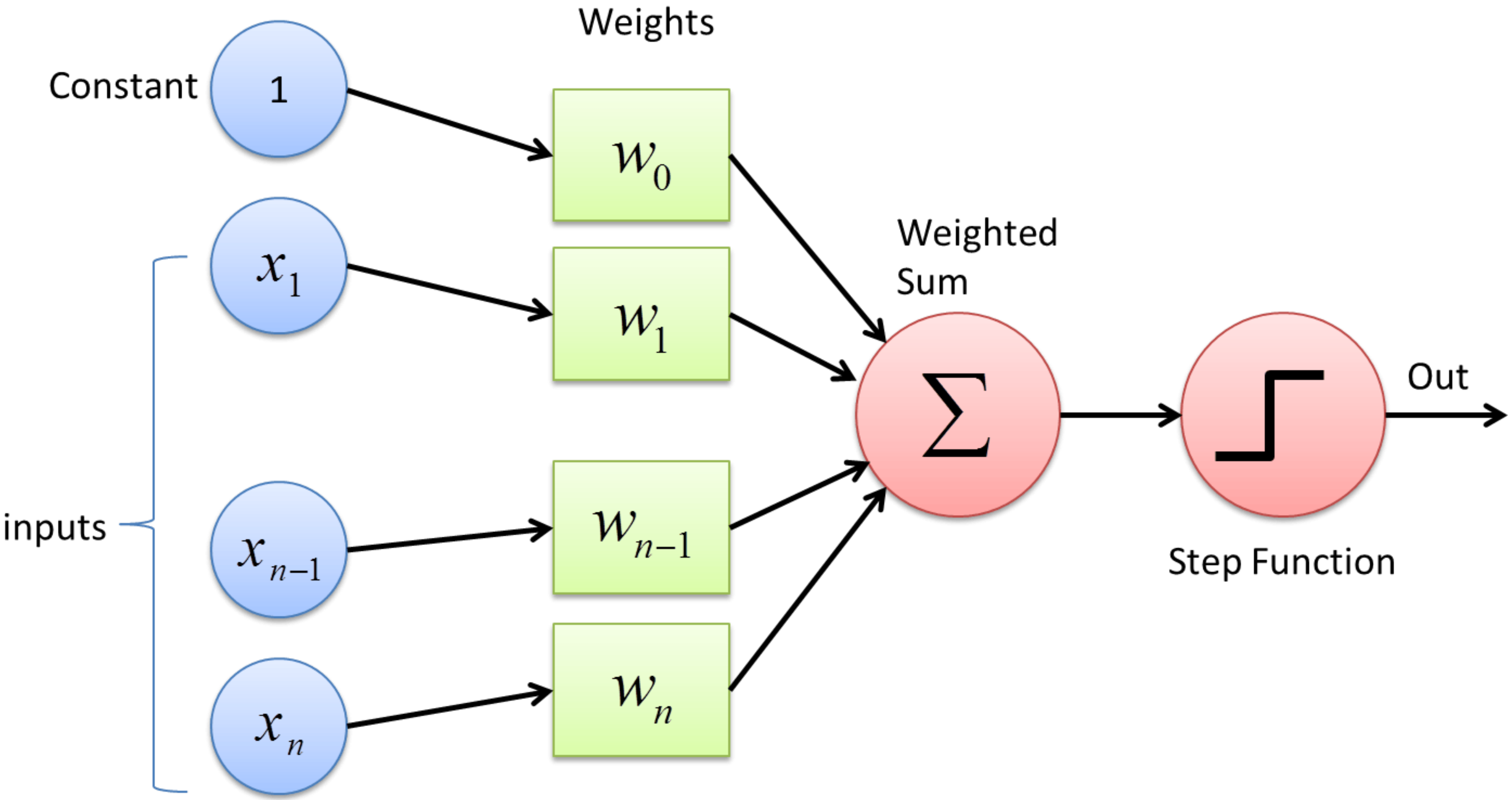


- Lo anterior permite que haya diferencias de cargas entre el exterior (+) y el interior (-) de la neurona: **POLARIDAD.**
- La diferencia de carga está dada por la concentración de iones.
- Hay mayor concentración de Na^+ fuera de la membrana y mayor concentración de K^+ dentro de la misma
- Esto es posible gracias a la **bomba de sodio-potasio (transporte activo).**

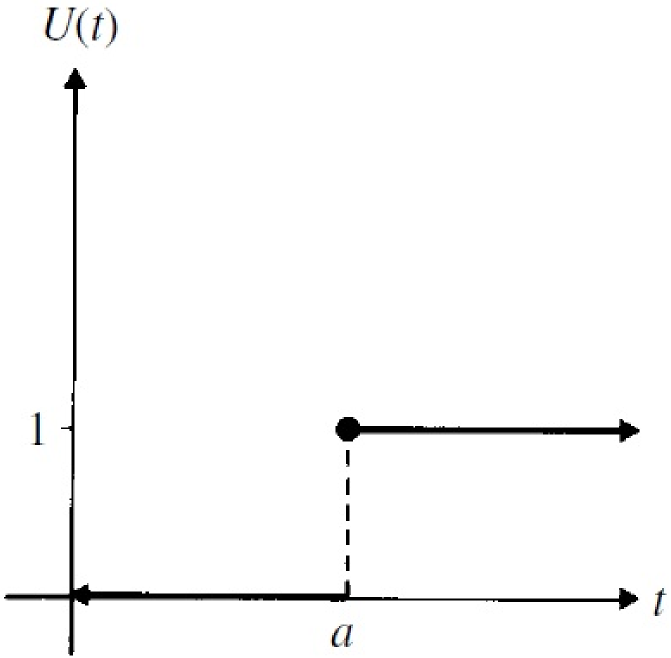
EL GRADIENTE IONICO LO LOGRA GRACIAS A LA BOMBA DE SODIO-POTASIO


Potencial
de acción
+40
0
-55
-70
Voltaje (mV)
Despolarización
Repolarización
Umbral
Inicios
fallidos
Estado de reposo
Estímulo
Período
refractario
0
1
2
3
4
5
Tiempo (ms)


x
y
z
w0=1
w1=0
w2=0
output0

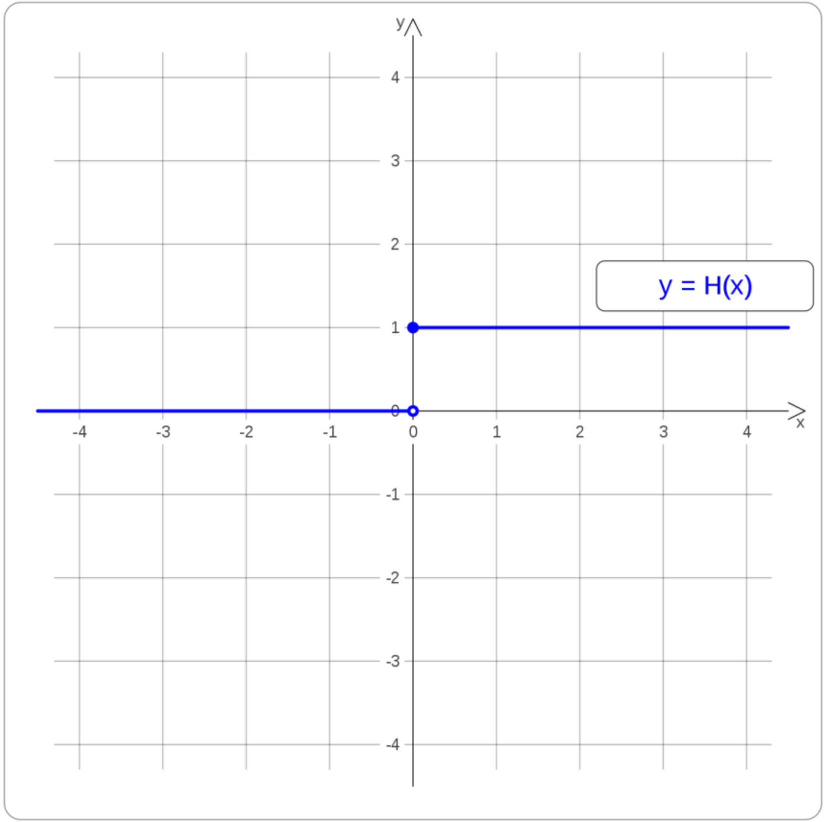

Weights
Constant
1
w_0
x_1
w_1
Weighted
Sum
Σ
Out
inputs
x_{n-1}
w_{n-1}
x_n
w_n
Step Function

Función escalón



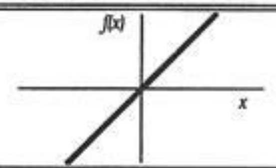
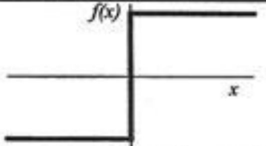
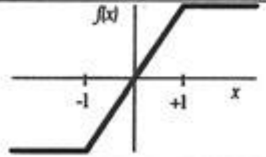
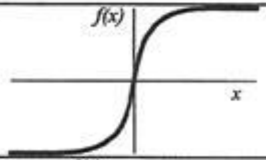
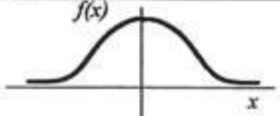
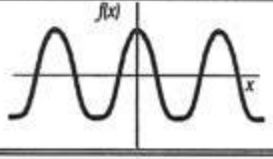
a = 1

Heaviside - Función escalón



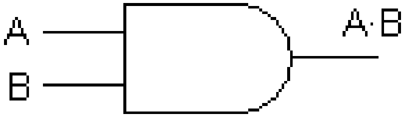
f(x)

- 1 si x >= 0
- 0 para todo otro valor

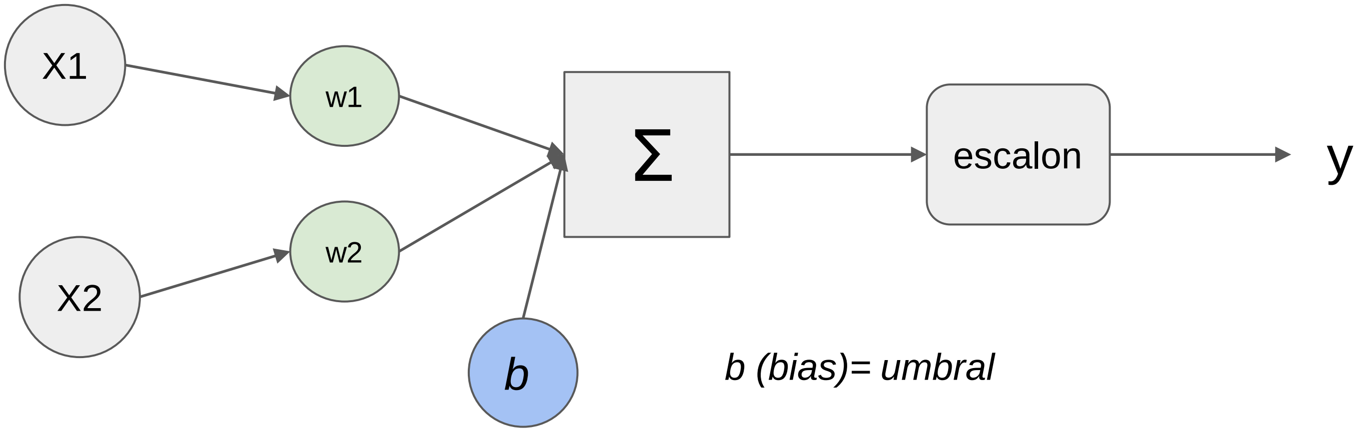
	Función	Rango	Gráfica
Identidad	$y = x$	[-∞, +∞]	
Escalón	$y = sign(x)$ $y = H(x)$	{-1, +1} {0, +1}	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	[-1, +1]	
Sigmoidea	$y = \dfrac{1}{1+e^{-x}}$ $y = tgh(x)$	[0, +1] [-1, +1]	
Gaussiana	$y = Ae^{-Bx^2}$	[0,+1]	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	[-1,+1]	

Perceptrón simple - AND

COMPUERTA AND		
A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1



Perceptrón simple - AND



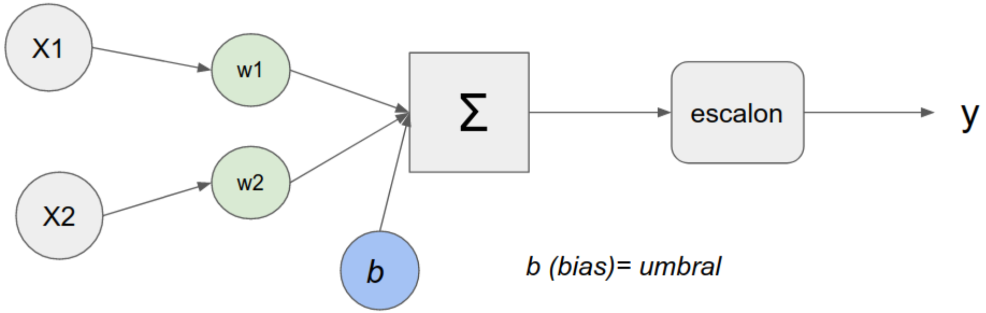
b (bias)= umbral

X1 * w1 + x2*w2 + b >   0 ⇨ y = 1

X1 * w1 + x2*w2 + b <= 0 ⇨ y = 0

Perceptrón simple - AND

X1	X2	y
0	0	0
0	1	0
1	0	0
1	1	1



$X1 * w1 + x2*w2 + b >= 1 \Rightarrow y = 1$

$X1 * w1 + x2*w2 + b < 1 \Rightarrow y = 0$

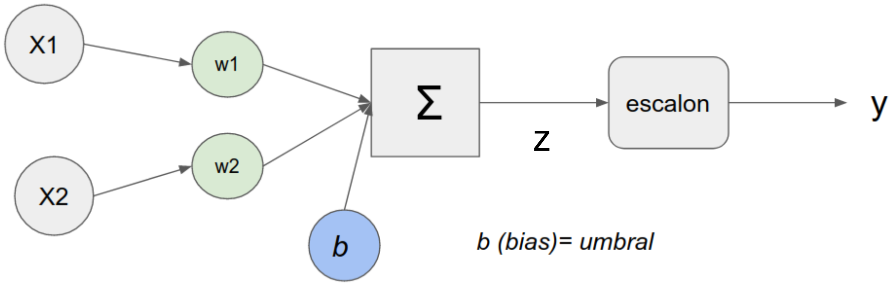
Perceptrón simple - Conjunto etiquetado

Datos de entrenamiento

X1	X2	y
0	0	0
0	1	0
1	0	0
1	1	1

Etiquetas. Valores esperados de salida

Perceptrón simple - AND



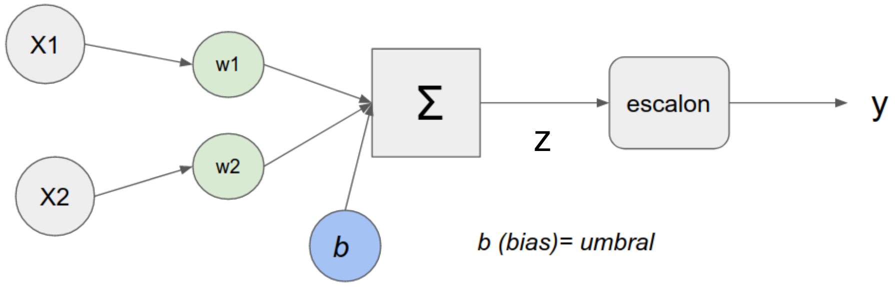
X1	X2	x1*w1+x2*w2 + b	z	y
0	0	b	<0	0
0	1	w2 + b	<0	0
1	0	w1 + b	<0	0
1	1	w1 + w2 + b	>=0	1

¿w_1, w_2 y b?

Aleatoria

- W1 = 0.3
- W2 = 0.2
- b = -1

Perceptrón simple - AND

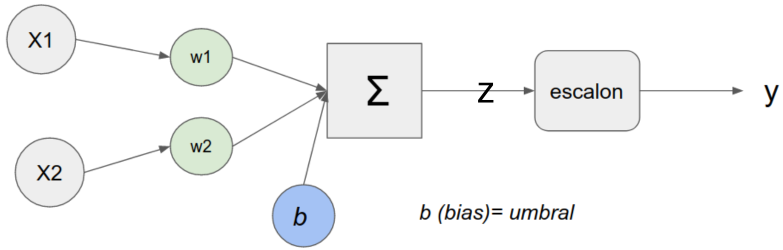


- W1 = 0.3
- W2 = 0.2
- b   = -1

X1	X2	x1*w1+x2*w2 + b	z	y
0	0	b	<0	0
0	1	w2 + b	<0	0
1	0	w1 + b	<0	0
1	1	w1 + w2 + b	>=0	1

Perceptrón simple - Primera Iteración

- W1 = 0.3
- W2 = 0.2
- b = -1



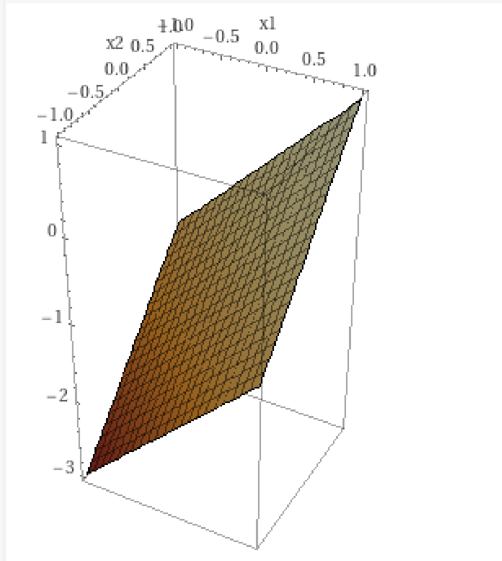
X1	X2	x1*w1+x2*w2 + b	z	y’	y	Error (|y-y’|)
0	0	-1	-1	0	0	0
0	1	w2 + b	-0.8	0	0	0
1	0	w1 + b	-0.7	0	0	0
1	1	w1 + w2 + b	-0.5	0	1	1

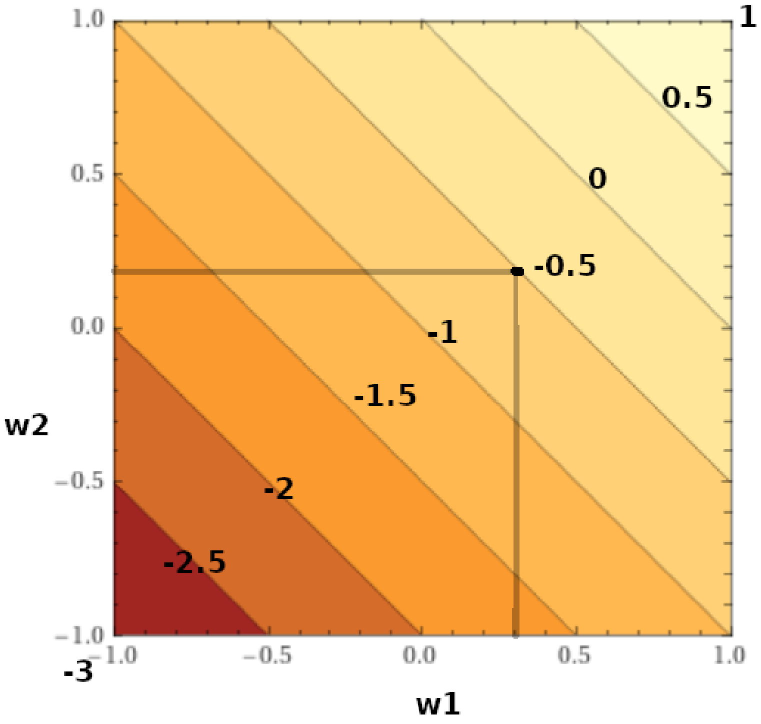
Función del error

- W1 = 0.3
- W2 = 0.2
- b = -1

Escalon(w1 + w2 -1) - 1 = error

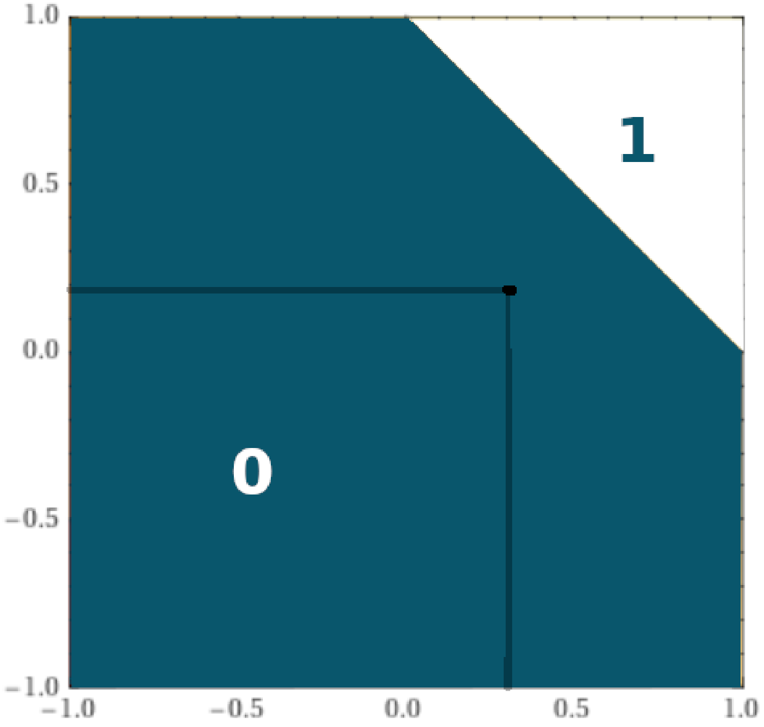
Si { w1 + w2 -1 < 0 => 0
w1 + w2 -1 <= 0 => }





Función del error: Escalón

- W1 = 0.3
- W2 = 0.2
- b   = -1

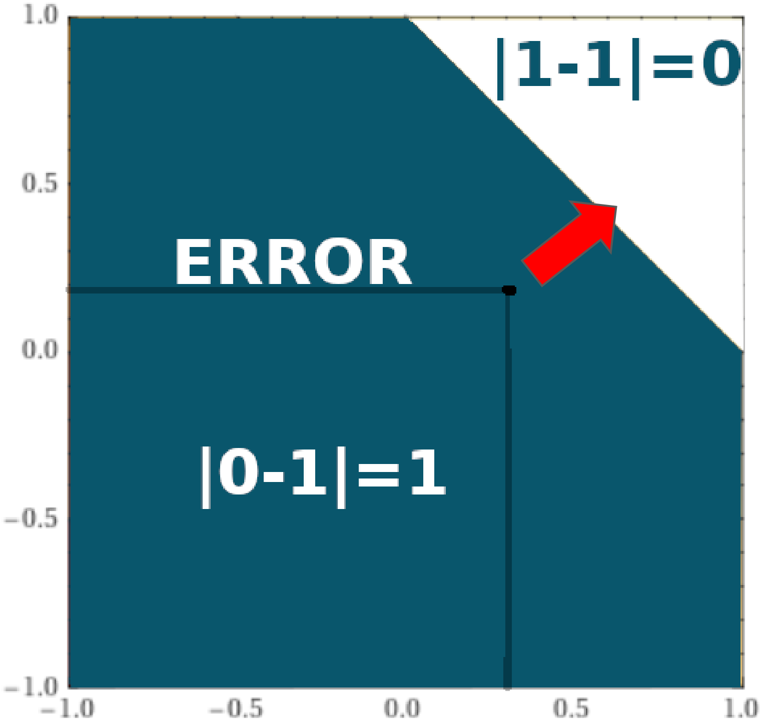


Escalon(w1 + w2 -1) - 1 = error

Si { w1 + w2 -1 < 0   => 0
w1 + w2 -1 <= 0 => 1 }

Función del error:

- W1 = 0.3
- W2 = 0.2
- b = -1



Dirección de decrecimiento

Error:

Si $\left\{ \begin{array}{l} w1 + w2 -1 < 0 \;\Rightarrow |0 -1| = 0 \\ w1 + w2 -1 <= 0 \;\Rightarrow |1-1| \end{array} \right\}$

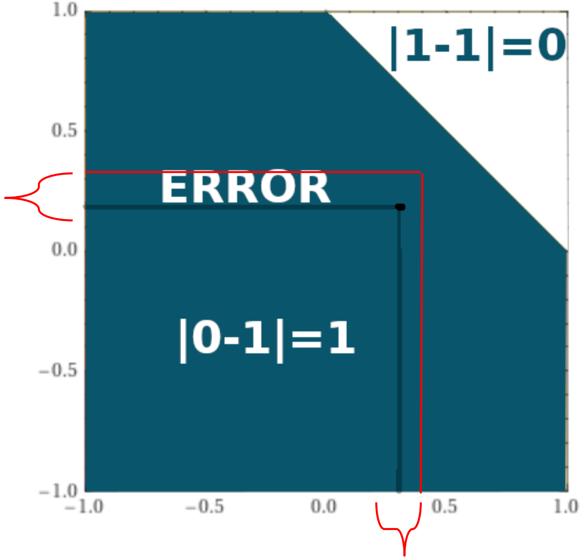
= 1

Actualización de los pesos

α = tasa de crecimiento.
Valor entre 0 y 1.
Suele ser 0.5

α = 0.2

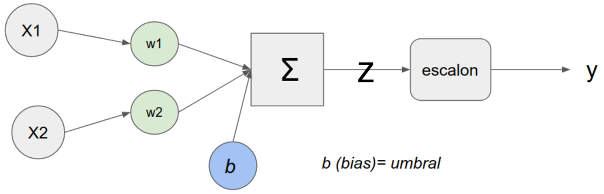
- W1 = 0.3 + 0.2 * error
- W2 = 0.2 + 0.2 * error
- b = -1



A medida que se acerque a cero, más pequeña será la actualización

Perceptrón simple - Segunda Iteración

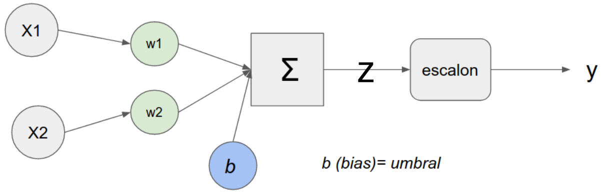
- W1 = 0.5
- W2 = 0.4
- b = -1



X1	X2	x1*w1+x2*w2 + b	z	y’	y	Error (|y-y’|)
0	0	-1	-1	0	0	0
0	1	w2 + b	-0.6	0	0	0
1	0	w1 + b	-0.5	0	0	0
1	1	w1 + w2 + b	-0.1	0	1	1

Perceptrón simple - Tercera Iteración

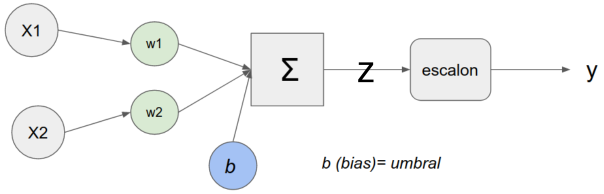
- W1 = 0.5 + 0.2 = 0.7
- W2 = 0.4 + 0.2 = 0.6
- b = -1



X1	X2	x1*w1+x2*w2 + b	z	y’	y	Error (|y-y’|)
0	0	-1	-1	0	0	0
0	1	w2 + b	-0.4	0	0	0
1	0	w1 + b	-0.3	0	0	0
1	1	w1 + w2 + b	0.3	1	1	0

Perceptrón simple - AND en producción

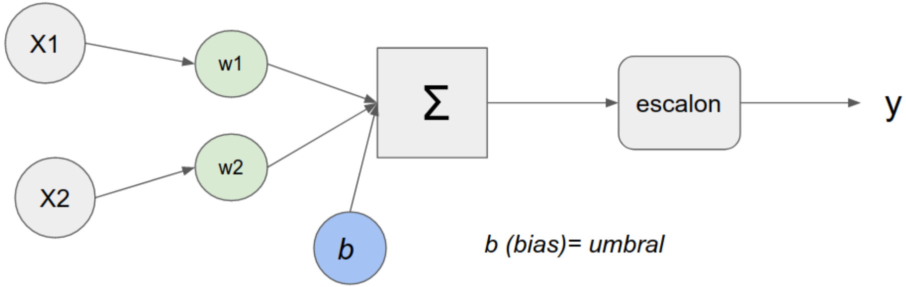
- W1 = 0.7
- W2 = 0.6
- b = -1

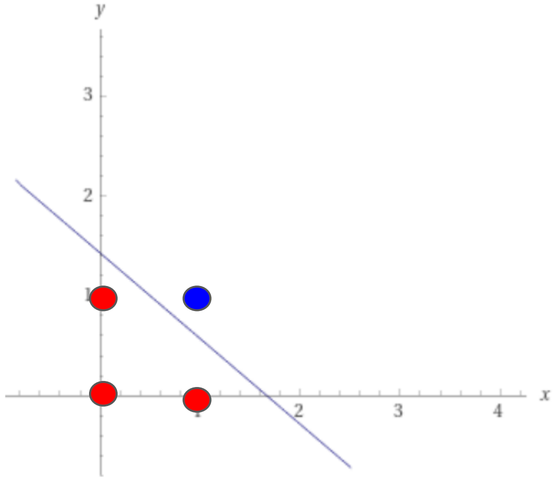


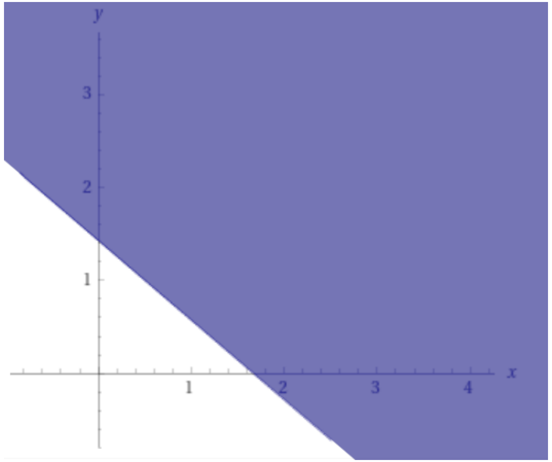
X1	X2	z	AND= y'
0.5	0.5	-0.35	0
0.5	1	-0,05	0
0.9	0.9	0.17	1

Perceptrón simple - Red entrenada - AND

- W1 = 0.7
- W2 = 0.6
- b   = -1
- x1*0.7+x2*0.6 -1







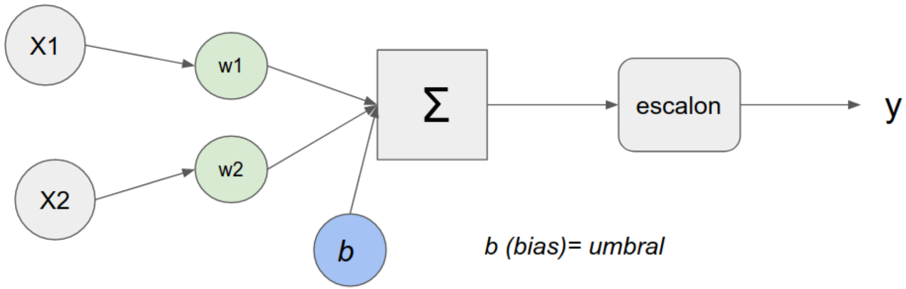
Perceptrón simple - Almacenamiento

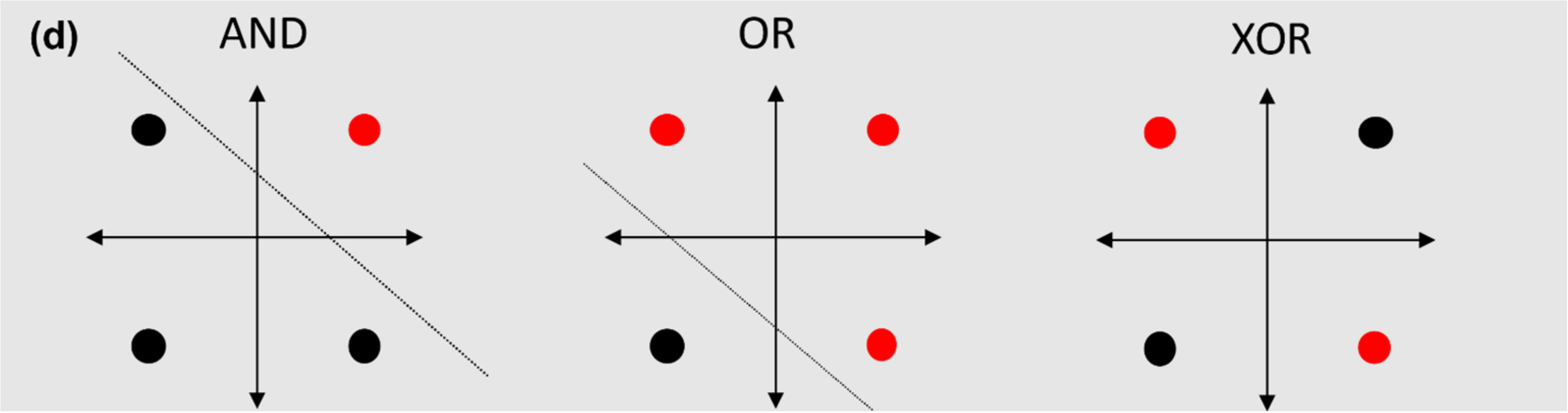
$$\begin{bmatrix} X1 & X2 & 1 \end{bmatrix} \boxed{\begin{bmatrix} W1 \\ W2 \\ b \end{bmatrix}} = x1 * w1 + x2 * w2 + b$$

Modelo entrenado = matriz de números flotantes

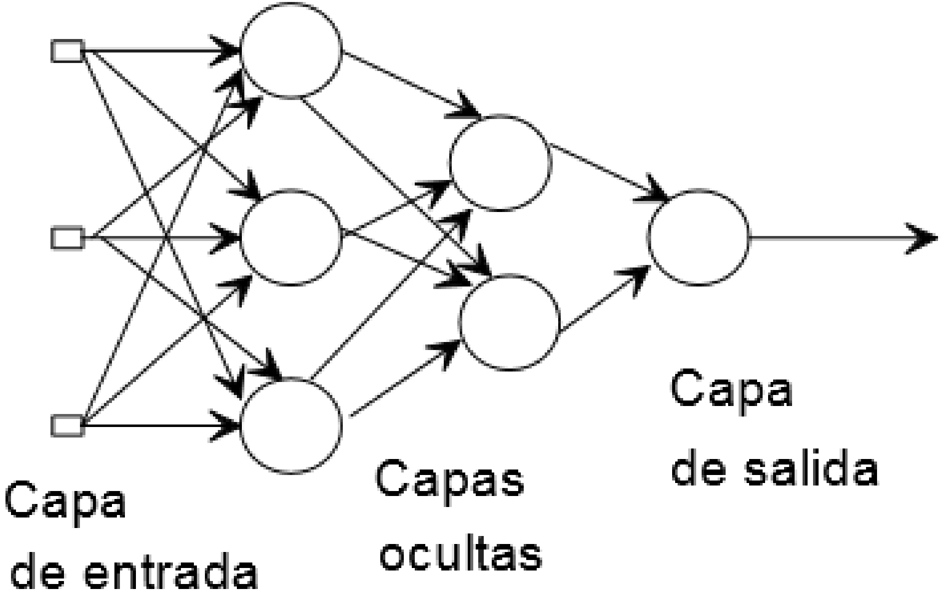
$$\begin{bmatrix} 0.7 \\ 0.6 \\ -1 \end{bmatrix}$$

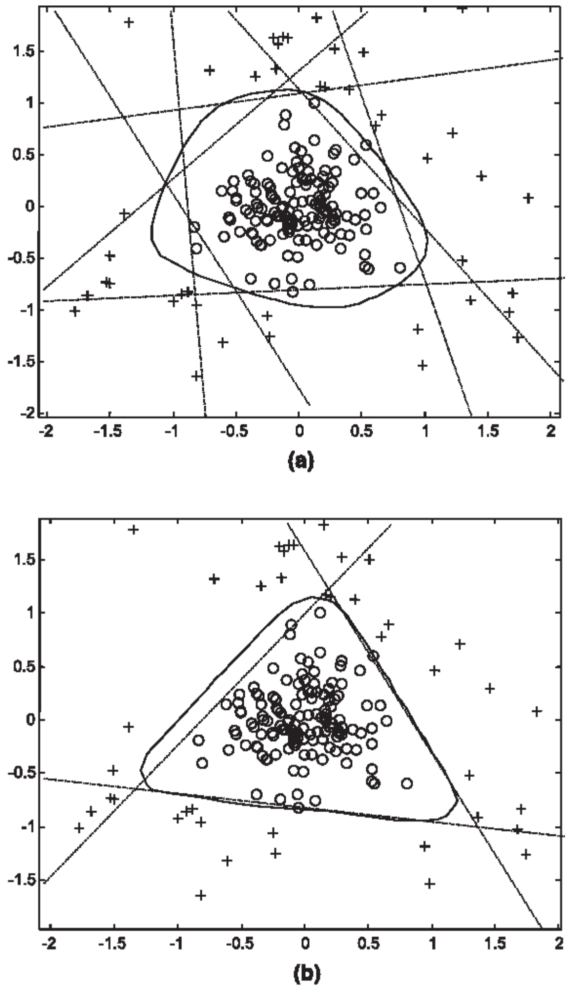
Perceptrón simple - Limitaciones





Perceptrón Multicapa



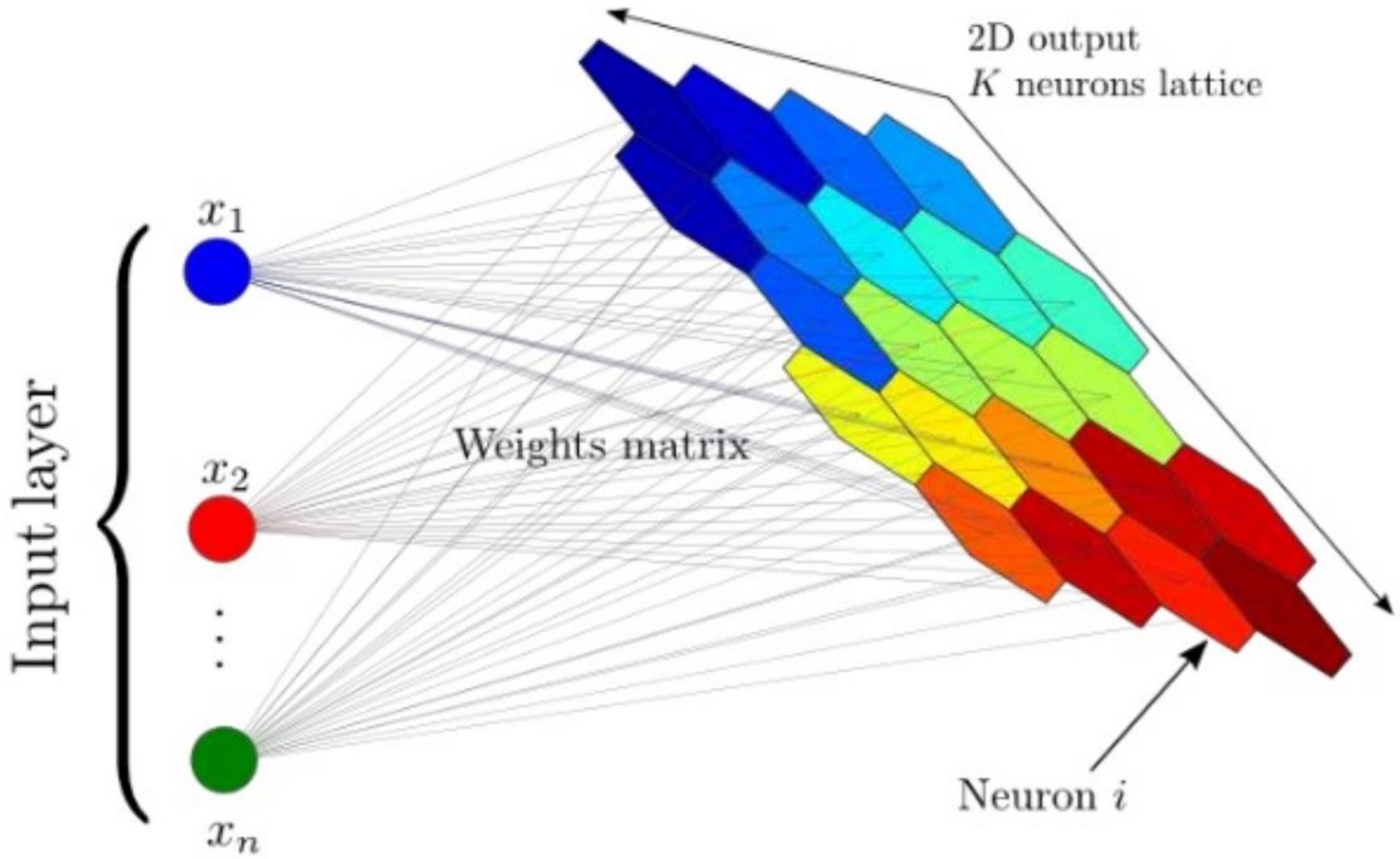


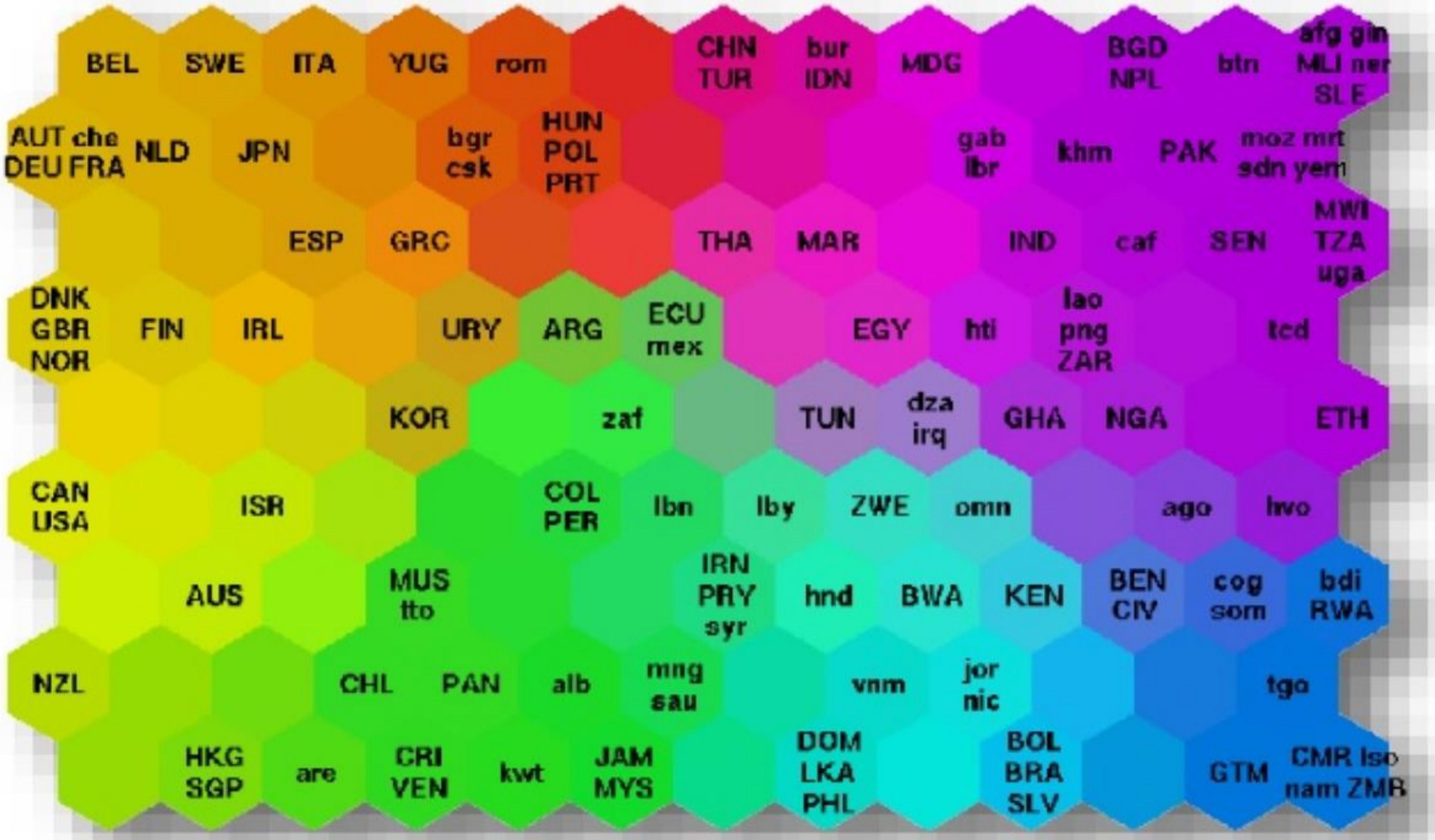
Entrenamiento

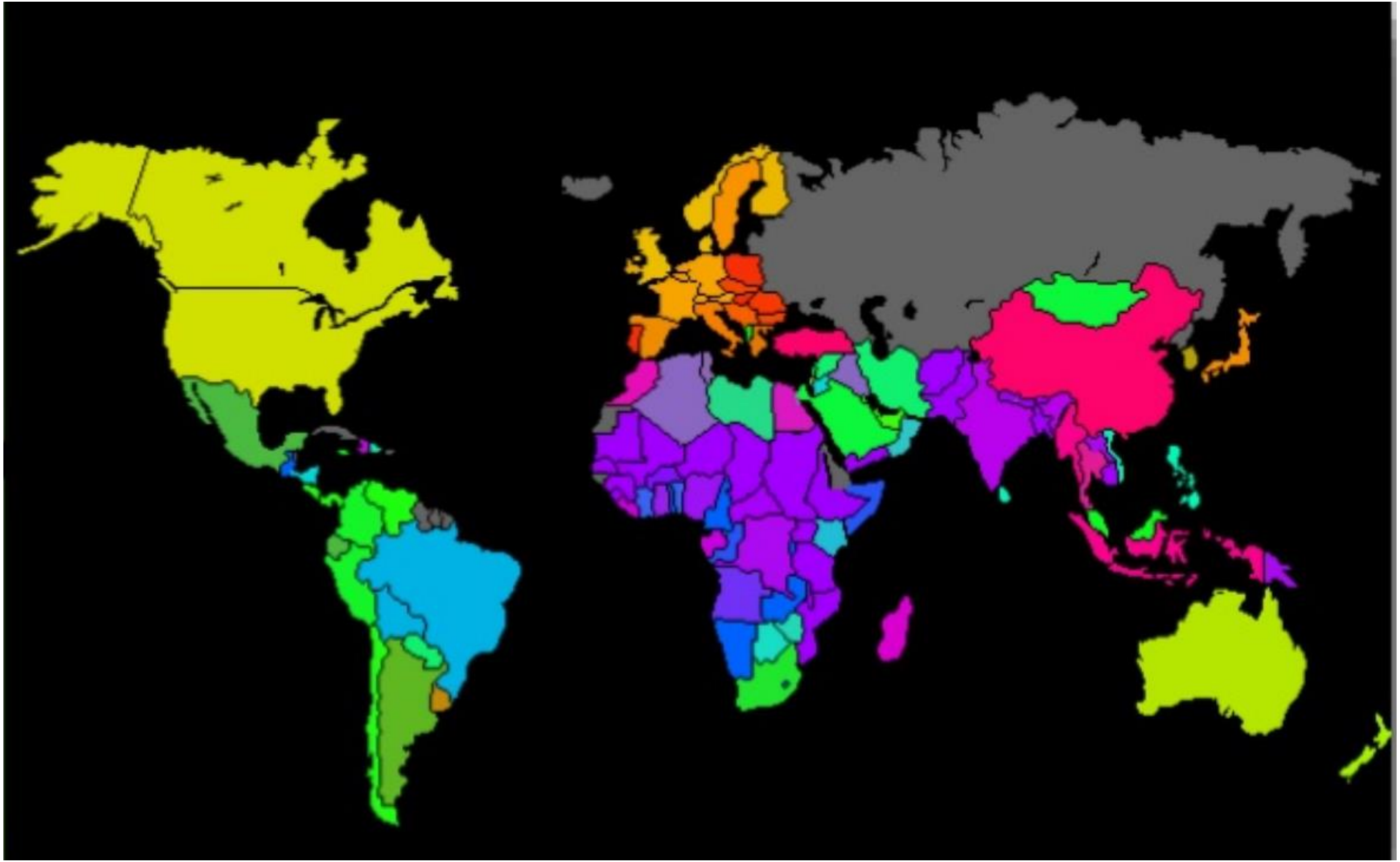
Backpropagation

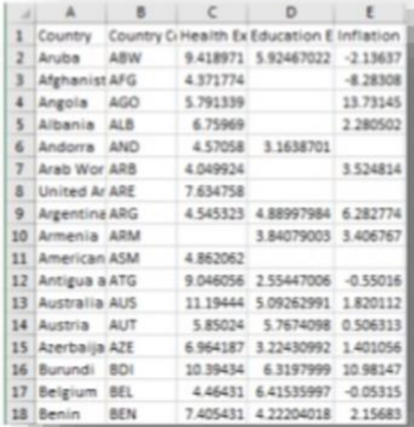
Redes SOM (Kohonen)

Teuvo Kohonen

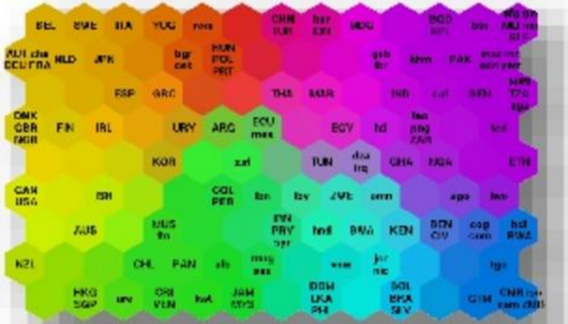

2D output
K neurons lattice
x_1
x_2
x_n
Input layer
Weights matrix
Neuron i

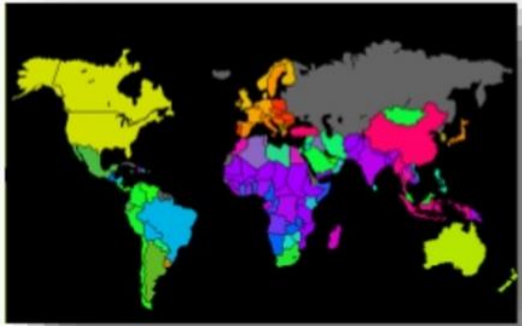

BEL
SWE
ITA
YUG
rom
CHN TUR
bur IDN
MDG
BGD NPL
btn
afg gin MLI ner SLE
AUT che DEU FRA
NLD
JPN
bgr csk
HUN POL PRT
gab lbr
khm
PAK
moz mrt sdn yem
ESP
GRC
THA
MAR
IND
caf
SEN
MWI TZA uga
DNK GBR NOR
FIN
IRL
URY
ARG
ECU mex
EGY
hti
lao png ZAR
tcd
KOR
zaf
TUN
dza irq
GHA
NGA
ETH
CAN USA
ISR
COL PER
lbn
lby
ZWE
omn
ago
hvo
AUS
MUS tto
IRN PRY syr
hnd
BWA
KEN
BEN CIV
cog som
bdi RWA
NZL
CHL
PAN
alb
mng sau
vnm
jor nic
tgo
HKG SGP
are
CRI VEN
kwt
JAM MYS
DOM LKA PHL
BOL BRA SLV
GTM
CMR lso nam ZMB

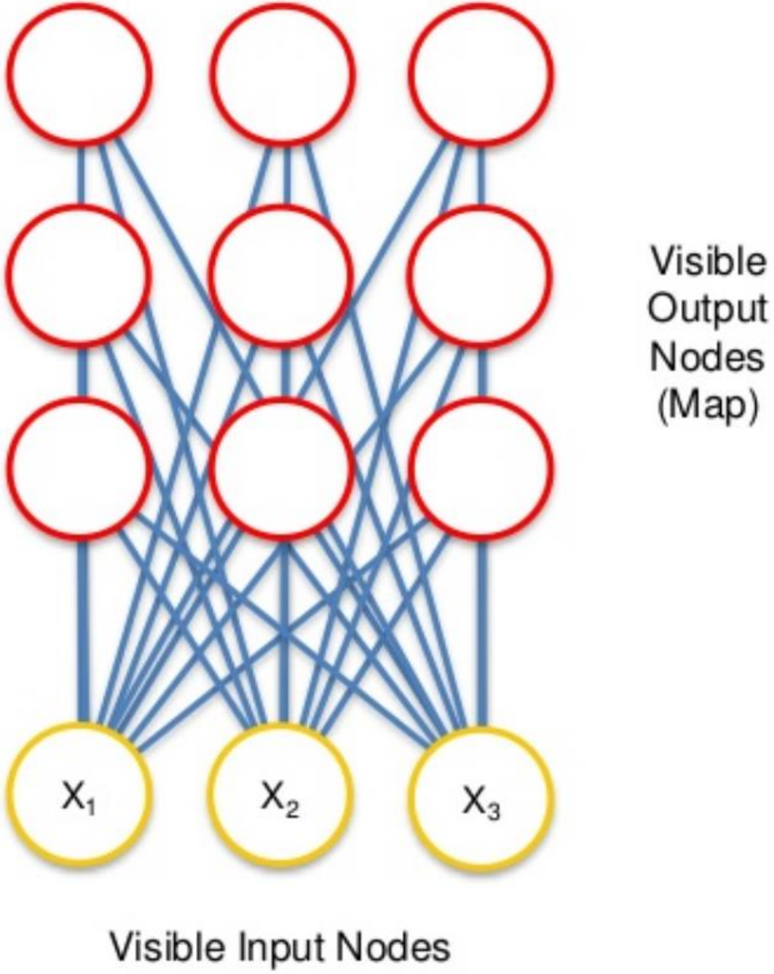


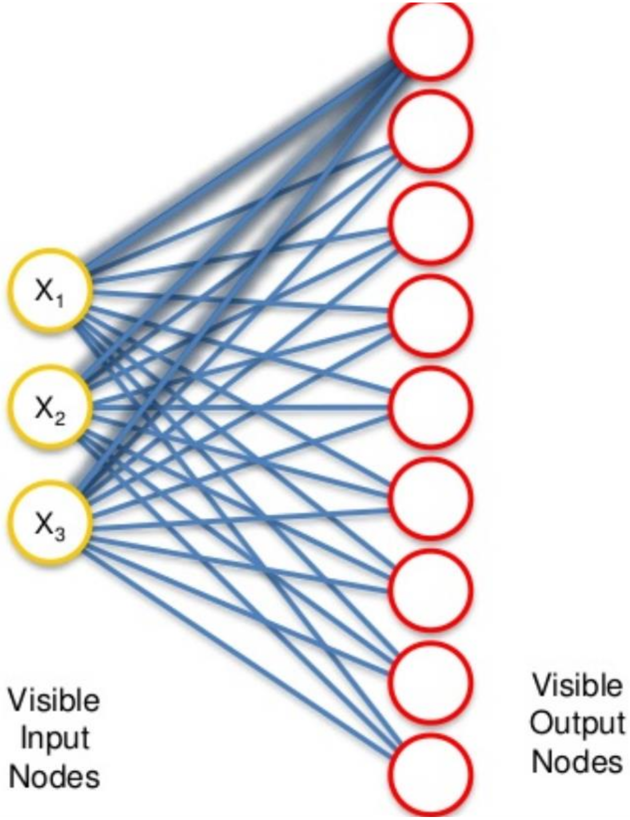


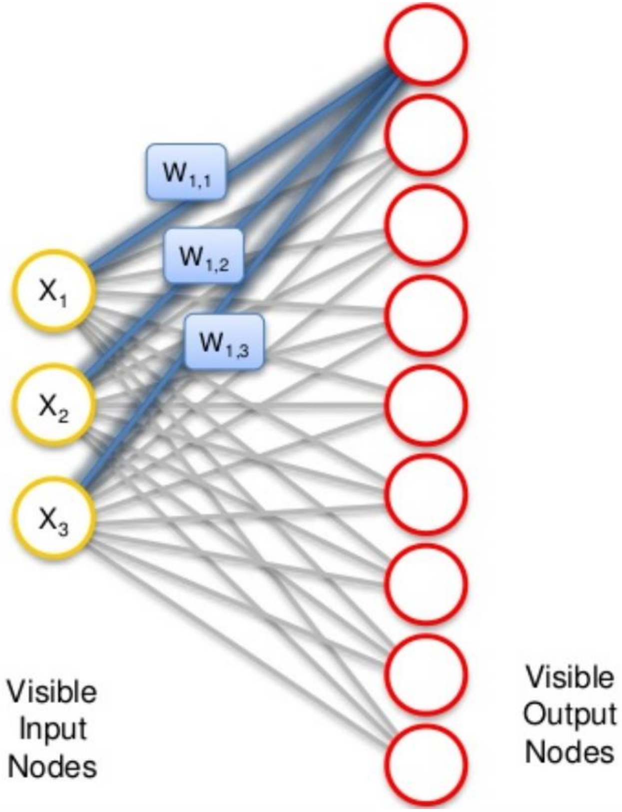
	A	B	C	D	E
1	Country	Country C	Health Ex	Education E	Inflation
2	Aruba	ABW	9.418971	5.92467022	-2.13637
3	Afghanist	AFG	4.371774		-8.28308
4	Angola	AGO	5.791339		13.73145
5	Albania	ALB	6.75969		2.280502
6	Andorra	AND	4.57058	3.1638701	
7	Arab Wor	ARB	4.049924		3.524814
8	United Ar	ARE	7.634758		
9	Argentina	ARG	4.545323	4.88997984	6.282774
10	Armenia	ARM		3.84079003	3.406767
11	American	ASM	4.862062		
12	Antigua a	ATG	9.046056	2.55447006	-0.55016
13	Australia	AUS	11.19444	5.09262991	1.820112
14	Austria	AUT	5.85024	5.7674098	0.506313
15	Azerbaija	AZE	6.964187	3.22430992	1.401056
16	Burundi	BDI	10.39434	6.3197999	10.98147
17	Belgium	BEL	4.46431	6.41535997	-0.05315
18	Benin	BEN	7.405431	4.22204018	2.15683

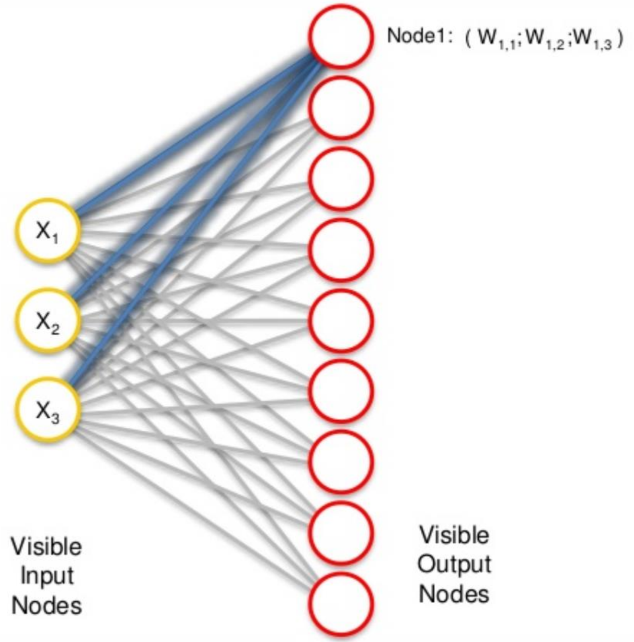


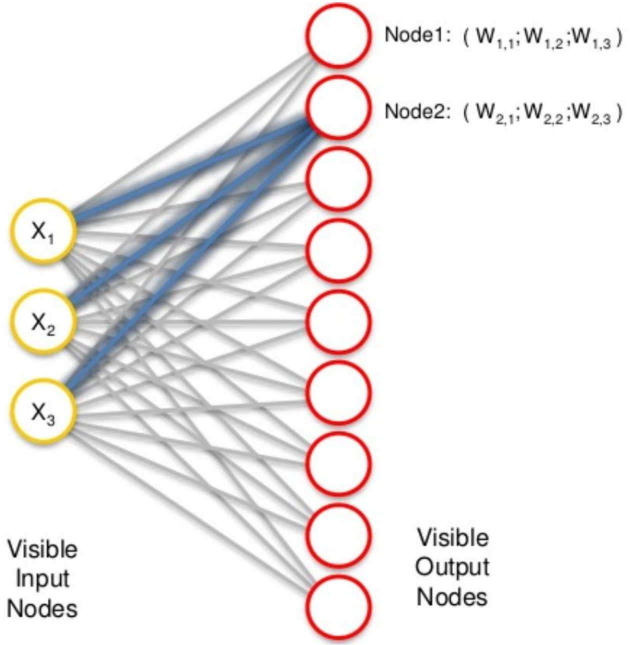


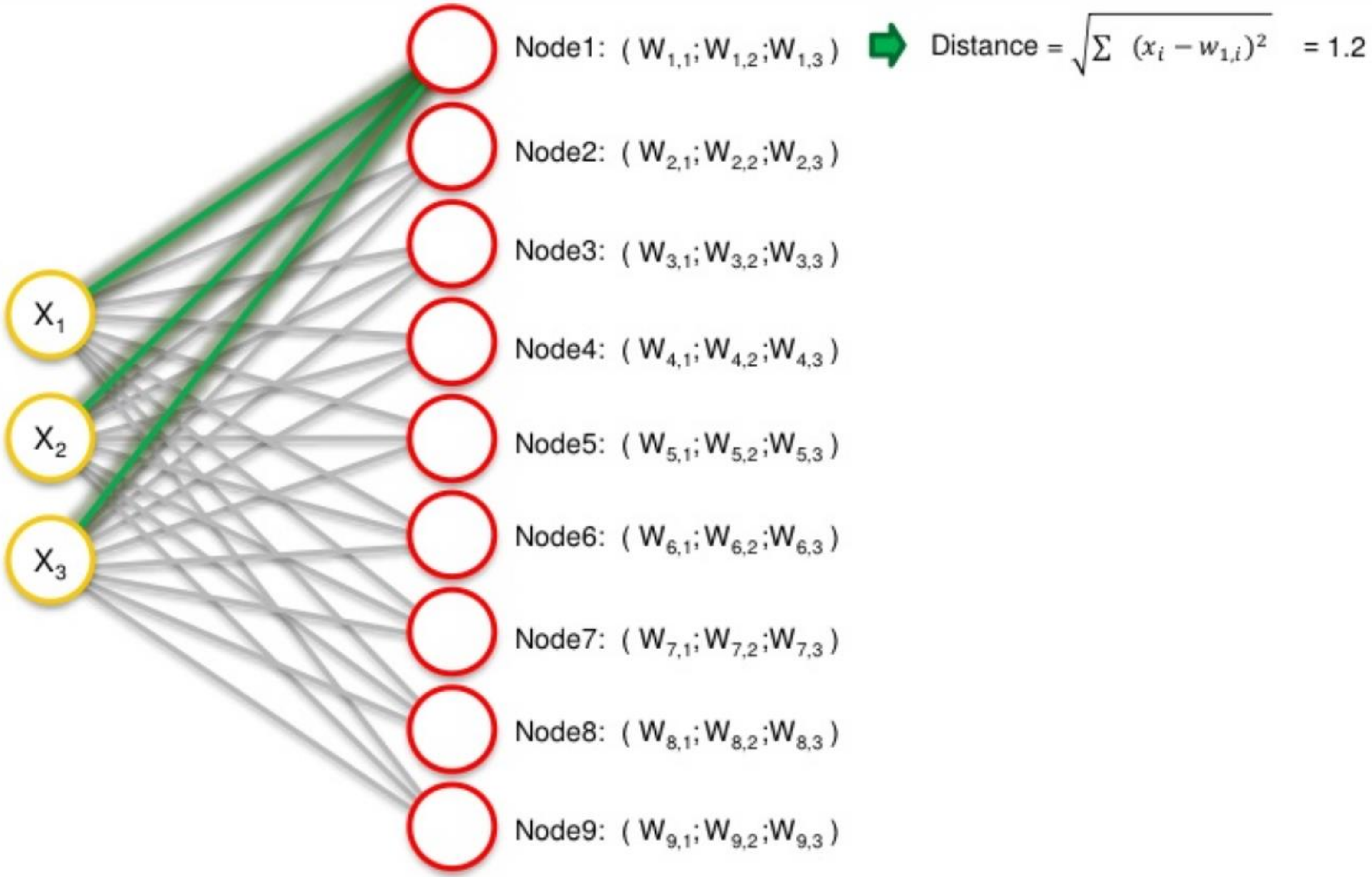

Visible
Output
Nodes
(Map)
X1
X2
X3
Visible Input Nodes


X1
X2
X3
Visible Input Nodes
Visible Output Nodes

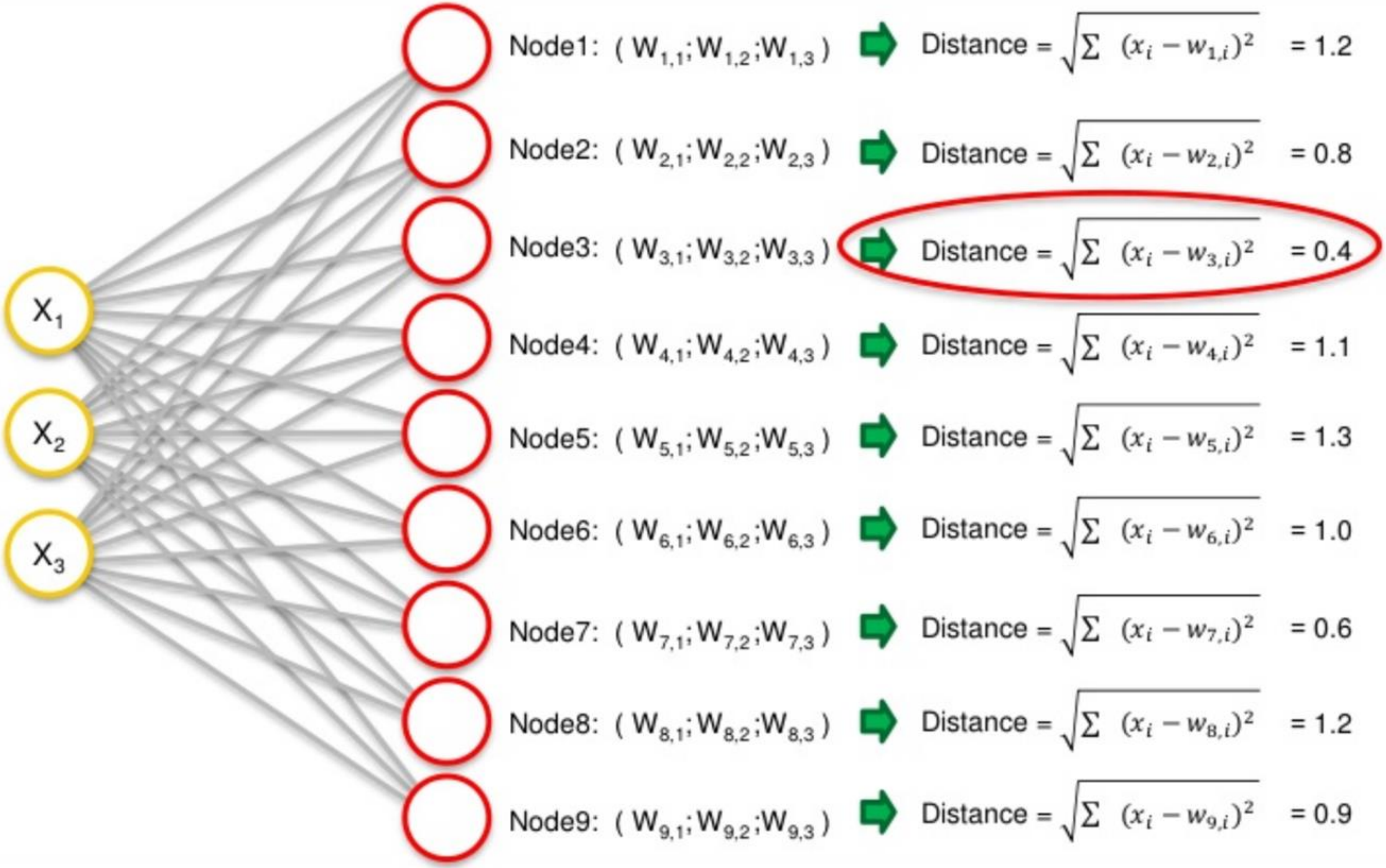

W1,1
W1,2
W1,3
X1
X2
X3
Visible Input Nodes
Visible Output Nodes


Node1: ($W_{1,1}$;$W_{1,2}$;$W_{1,3}$)
X_1
X_2
X_3
Visible Input Nodes
Visible Output Nodes

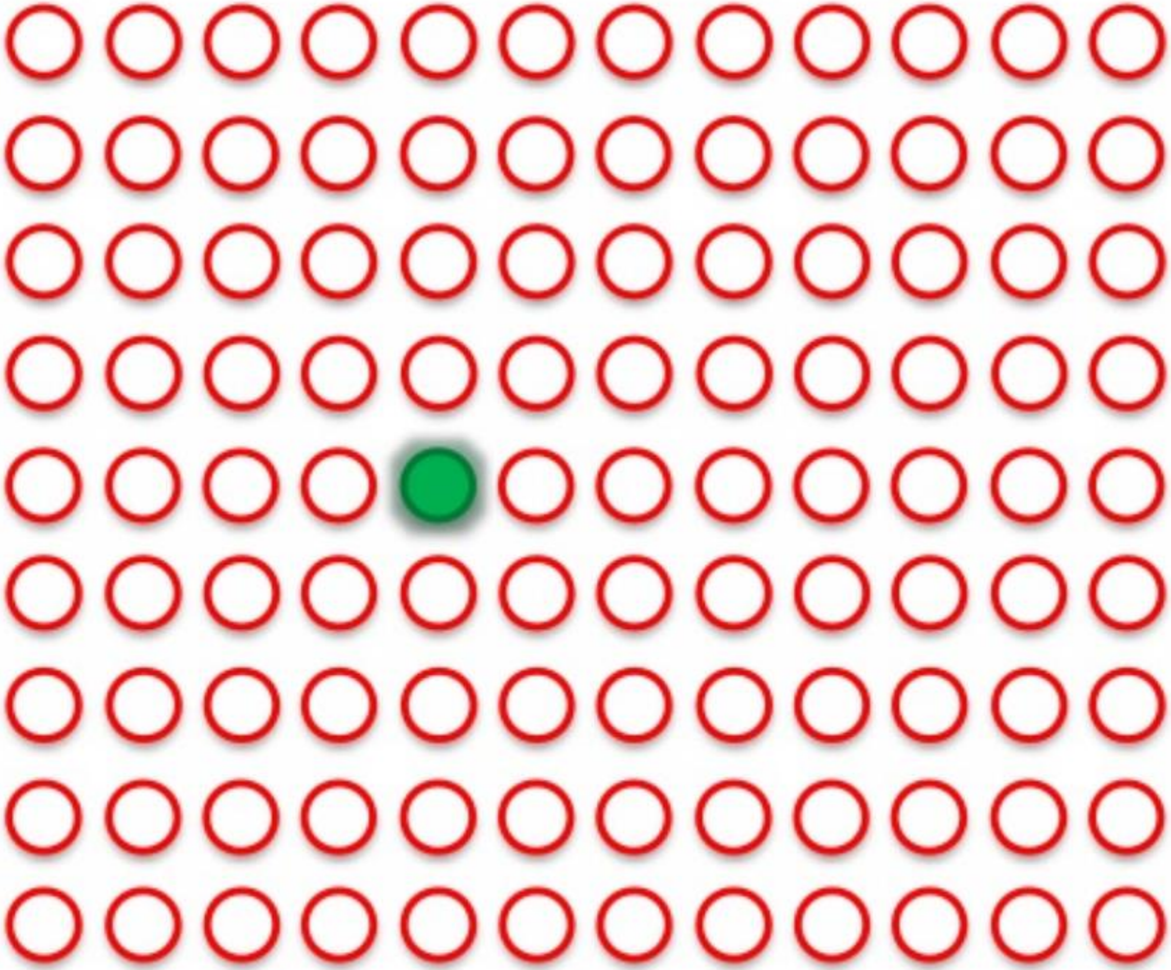

Node1: ($W_{1,1}$;$W_{1,2}$;$W_{1,3}$)
Node2: ($W_{2,1}$;$W_{2,2}$;$W_{2,3}$)
X_1
X_2
X_3
Visible Input Nodes
Visible Output Nodes

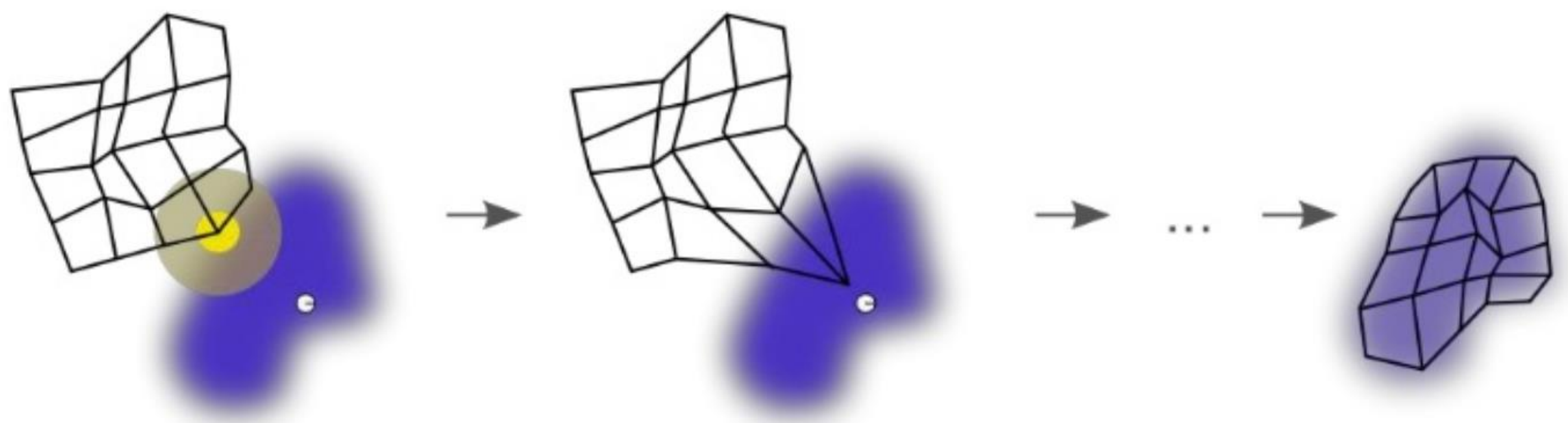


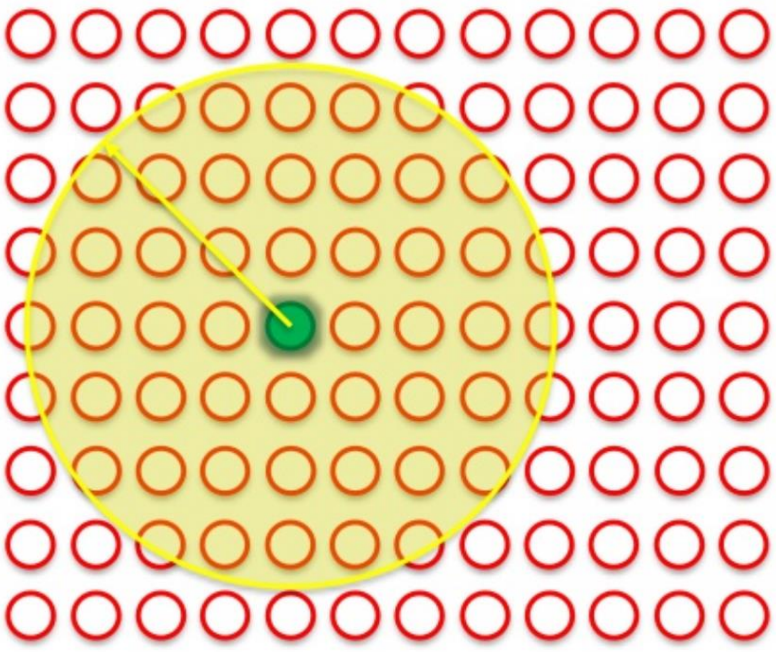
X_1
X_2
X_3
Node1: $(W_{1,1}; W_{1,2}; W_{1,3})$
Distance = $\sqrt{\sum (x_i - w_{1,i})^2}$ = 1.2
Node2: $(W_{2,1}; W_{2,2}; W_{2,3})$
Node3: $(W_{3,1}; W_{3,2}; W_{3,3})$
Node4: $(W_{4,1}; W_{4,2}; W_{4,3})$
Node5: $(W_{5,1}; W_{5,2}; W_{5,3})$
Node6: $(W_{6,1}; W_{6,2}; W_{6,3})$
Node7: $(W_{7,1}; W_{7,2}; W_{7,3})$
Node8: $(W_{8,1}; W_{8,2}; W_{8,3})$
Node9: $(W_{9,1}; W_{9,2}; W_{9,3})$

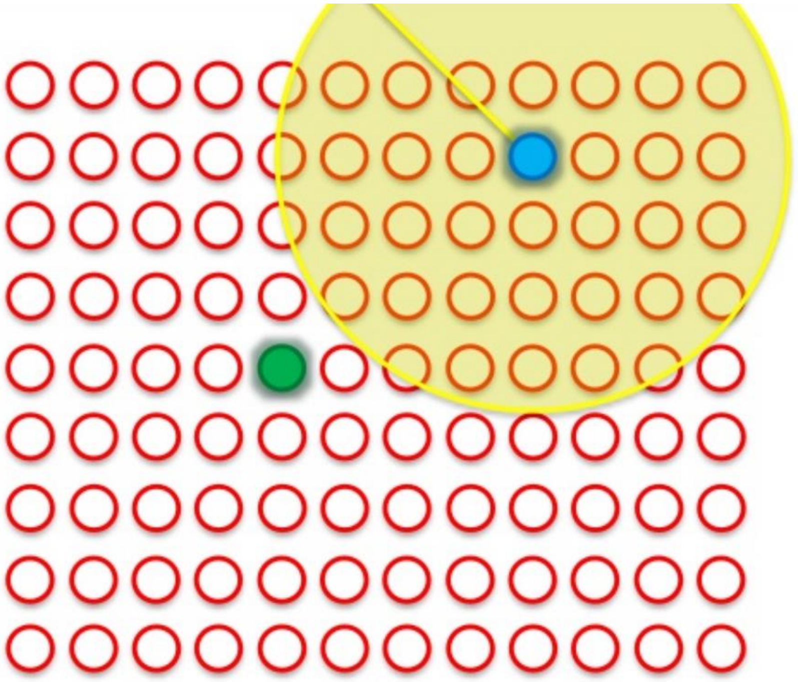


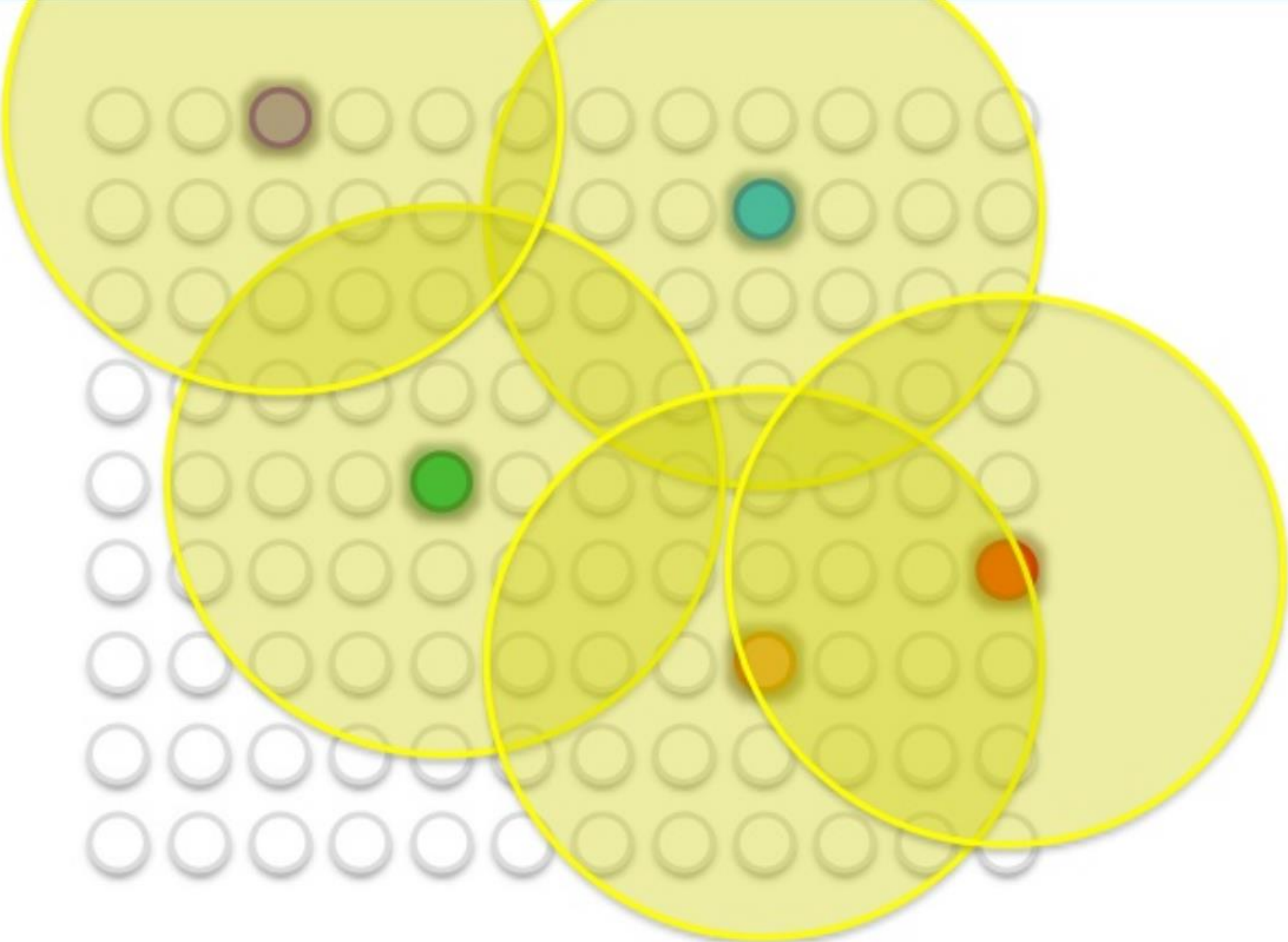
X_1
X_2
X_3
Node1: ($W_{1,1}$;$W_{1,2}$;$W_{1,3}$)
Distance = $\sqrt{\sum (x_i - w_{1,i})^2}$ = 1.2
Node2: ($W_{2,1}$;$W_{2,2}$;$W_{2,3}$)
Distance = $\sqrt{\sum (x_i - w_{2,i})^2}$ = 0.8
Node3: ($W_{3,1}$;$W_{3,2}$;$W_{3,3}$)
Distance = $\sqrt{\sum (x_i - w_{3,i})^2}$ = 0.4
Node4: ($W_{4,1}$;$W_{4,2}$;$W_{4,3}$)
Distance = $\sqrt{\sum (x_i - w_{4,i})^2}$ = 1.1
Node5: ($W_{5,1}$;$W_{5,2}$;$W_{5,3}$)
Distance = $\sqrt{\sum (x_i - w_{5,i})^2}$ = 1.3
Node6: ($W_{6,1}$;$W_{6,2}$;$W_{6,3}$)
Distance = $\sqrt{\sum (x_i - w_{6,i})^2}$ = 1.0
Node7: ($W_{7,1}$;$W_{7,2}$;$W_{7,3}$)
Distance = $\sqrt{\sum (x_i - w_{7,i})^2}$ = 0.6
Node8: ($W_{8,1}$;$W_{8,2}$;$W_{8,3}$)
Distance = $\sqrt{\sum (x_i - w_{8,i})^2}$ = 1.2
Node9: ($W_{9,1}$;$W_{9,2}$;$W_{9,3}$)
Distance = $\sqrt{\sum (x_i - w_{9,i})^2}$ = 0.9

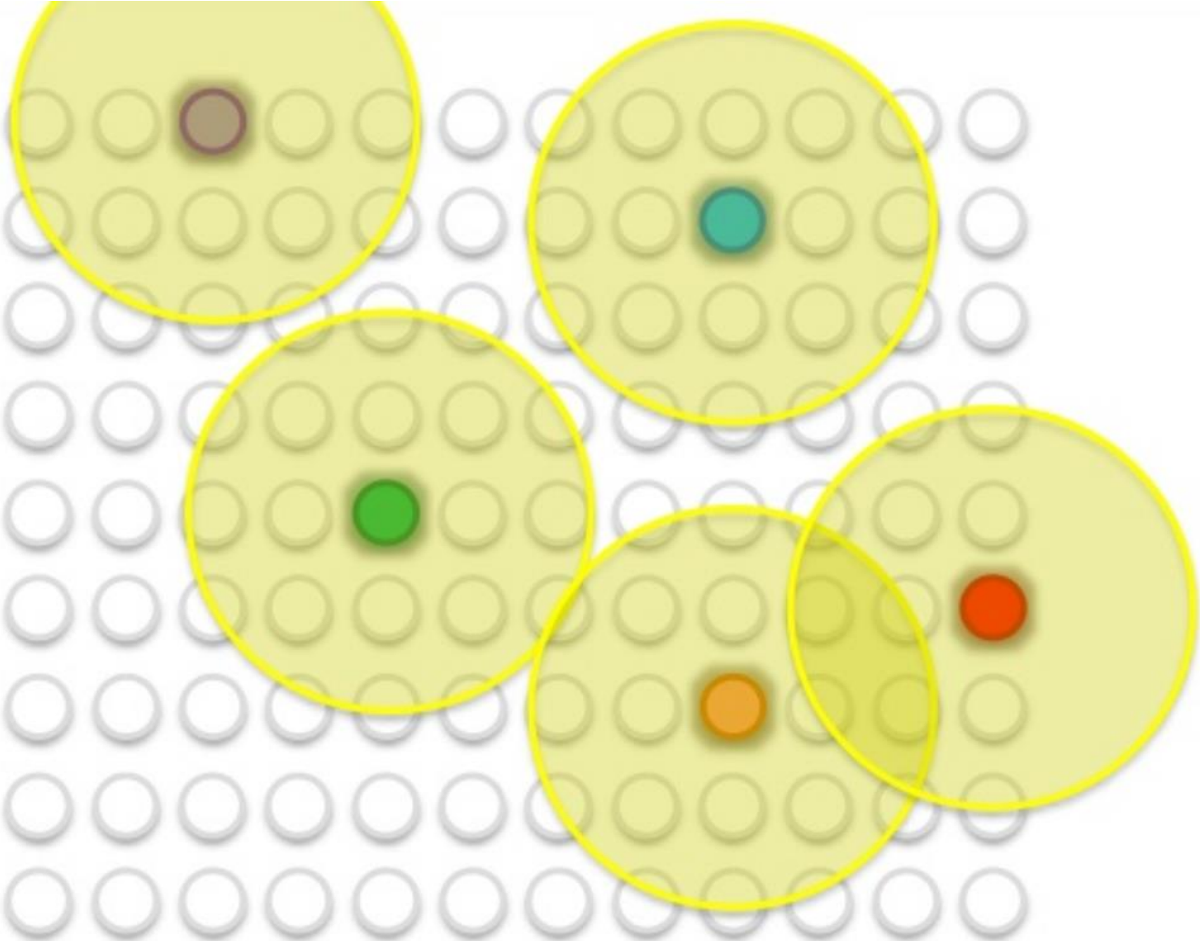


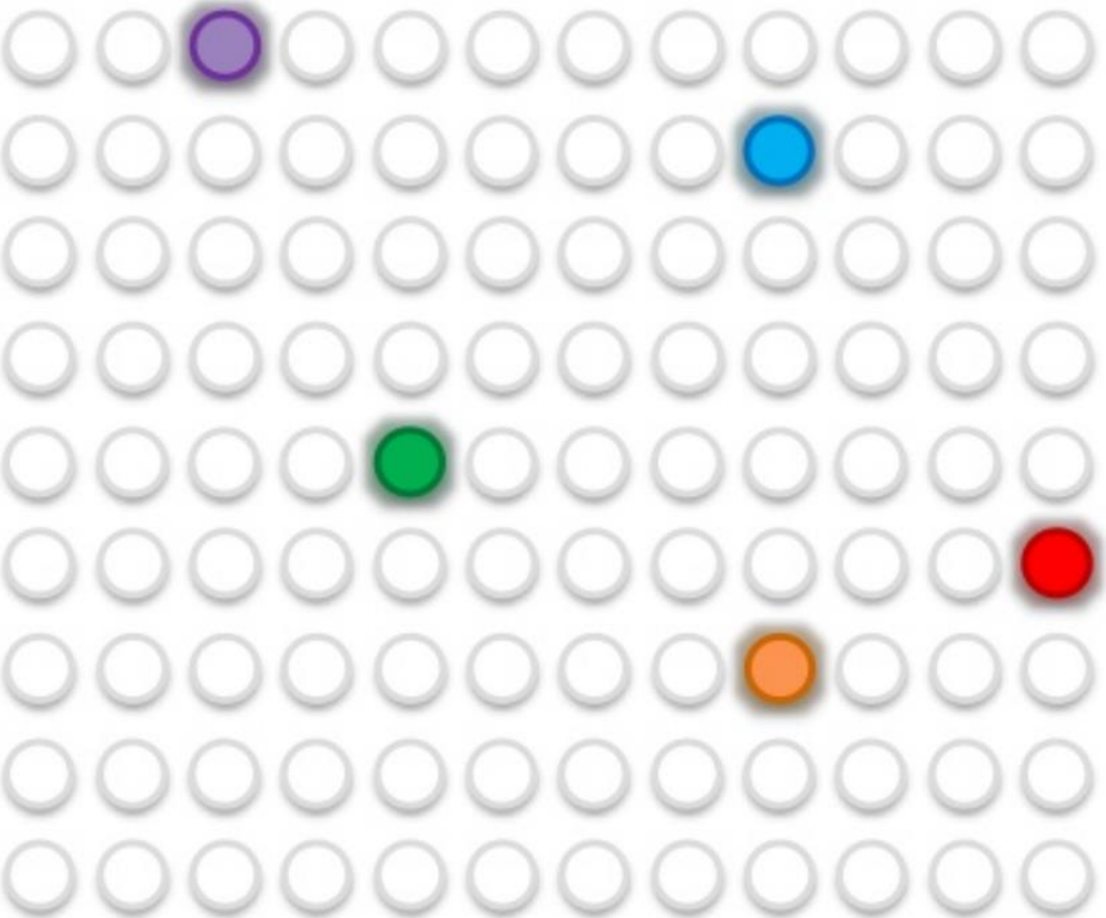


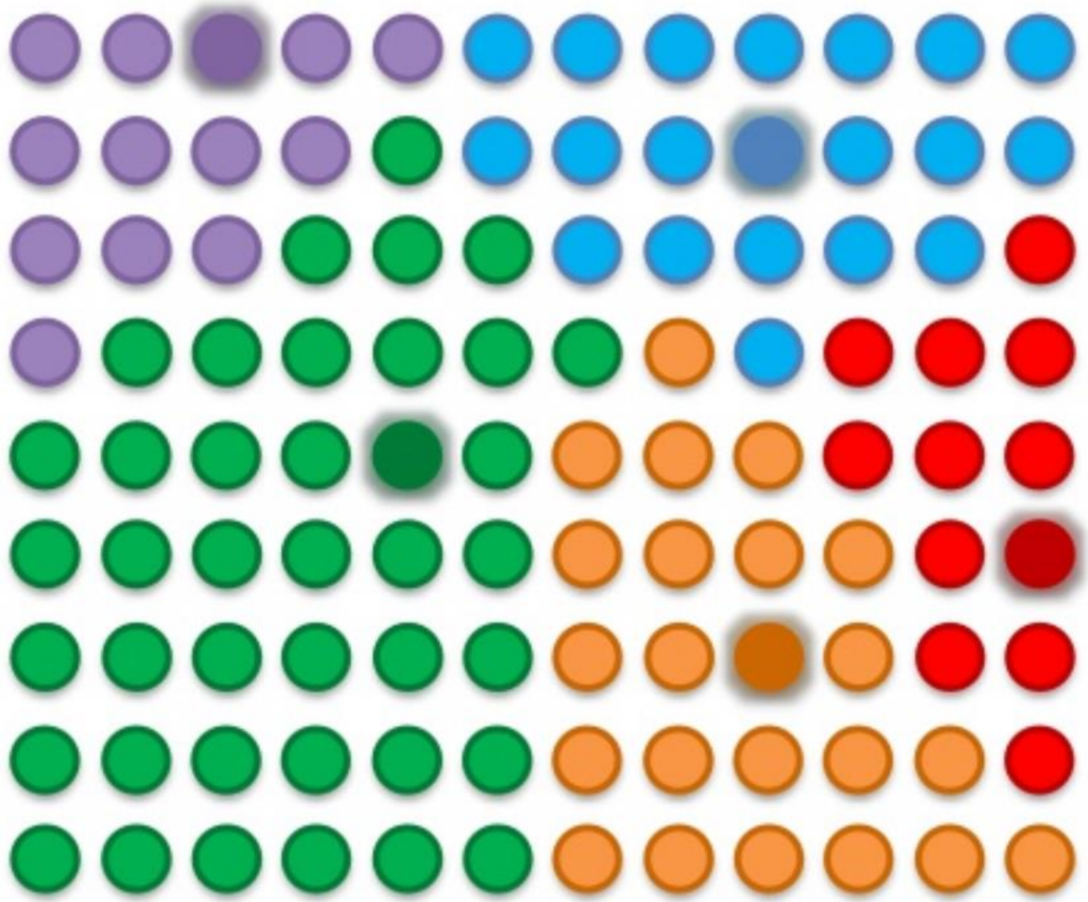












Referencias

- https://www.researchgate.net/publication/331641249_Tutorial_sobre_Redes_Neuronales_Artificiales_Los_Mapas_Autoorganizados_de_Kohonen

- https://www.slideshare.net/KirillEremenko/deep-learning-az-self-organizing-maps-som-module-4