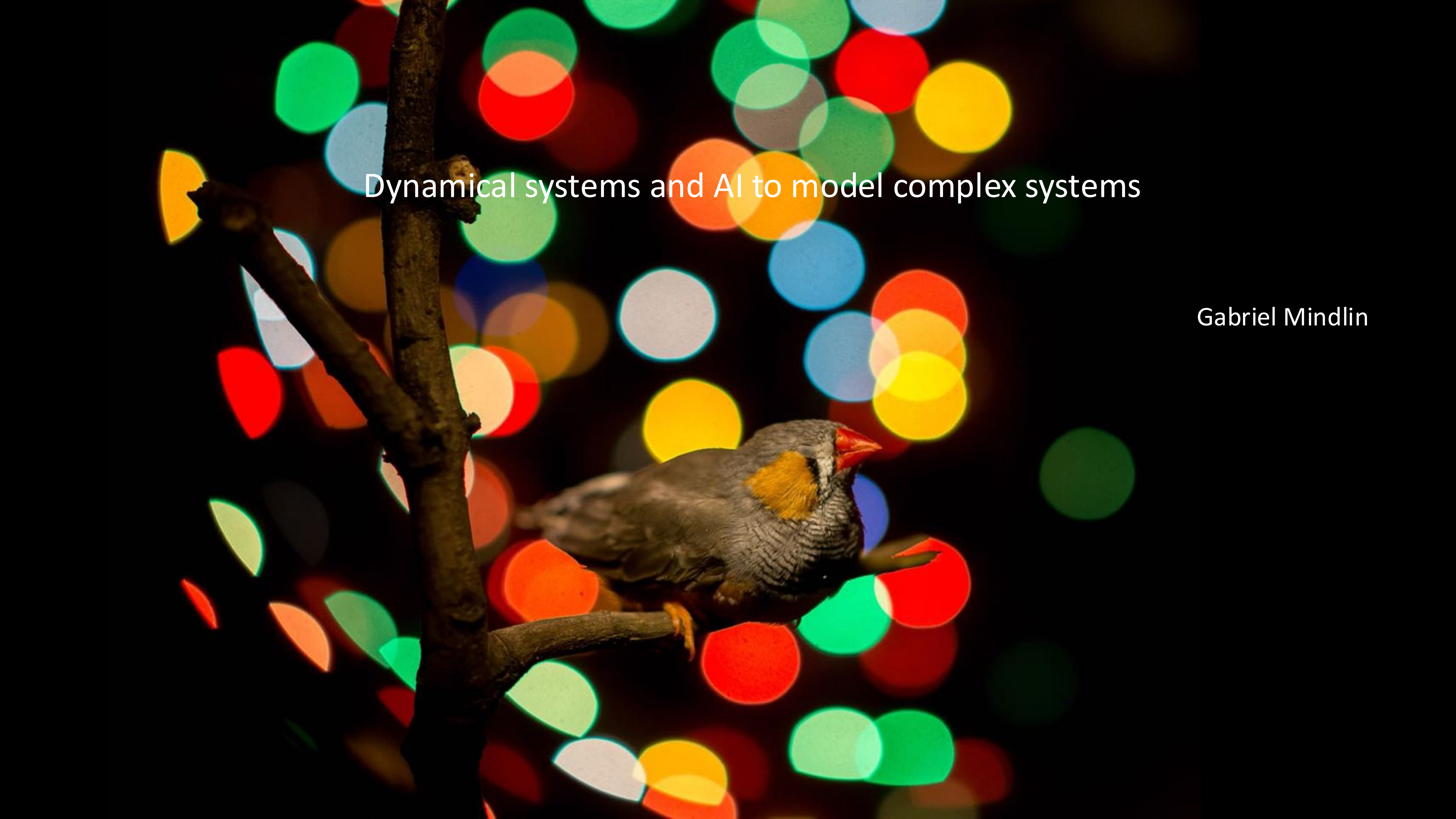
Dynamical systems and AI to model complex systems

Gabriel Mindlin



The rationale for the course:

One of the crucial intellectual decisions facing young scientists today is determining the balance between **interpretable** and **data-driven** approaches when addressing a scientific problem.

Interpretable science

Variables have meaning

The relationship between the Variables are interpretable mechanisms

Data driven science

A complex ”non interpretable” model (as a neural network) is trained with examples, and the model does not share with us the rationale behind its success

Dynamical systems
(in the framework of this course)

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(x) \qquad \boldsymbol{x} \in R^n$$

$$\boldsymbol{x}(t = 0) = \boldsymbol{x_0}$$

Under smoothness conditions for $\boldsymbol{f(x)}$ and ${}^{df}/_{dx}$ in a neighborhood of $\boldsymbol{x_0}$, the solution of the system exists and is unique. This means that each point (state) has a unique future (in geometrical terms, no auto-intersection of trajectories.

Dado un campo vector, que es el resultado de traducir **reglas en el lenguaje de las matematicas**,

$$\frac{dx}{dt} = f(x, \mu) \qquad x(0) = x_0$$

Encontramos el **flujo**

$$\phi(0, x) = x_0$$

$$\phi(t_2, \phi(t_1, x)) = \phi(t_2 + t_1, x)$$

Asi se predice con sistemas dinamicos

$$\phi(dt, x_0) = x_0 + f(x_0, \mu)\ \mathrm{dt}$$

Con la regla, computo el flujo
(i.e., el conjunto de soluciones)

La IA, es un paradigma diferente,
Un nuevo modo de predecir

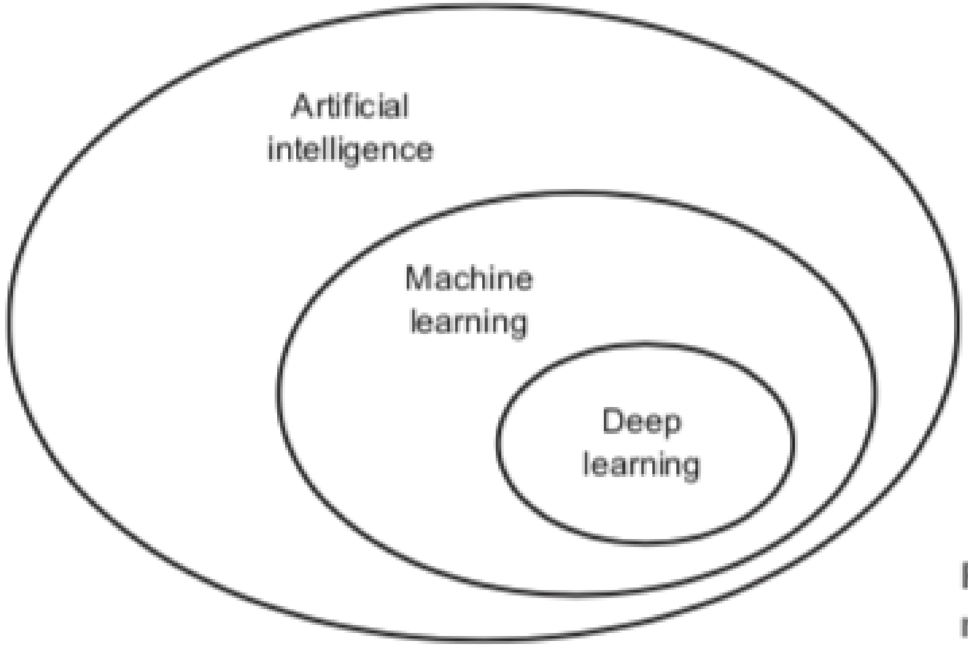


Figure 1.1 Artificial intelligence, machine learning, and deep learning

Sistemas dinamicos,
y tambien la vieja IA

answers

Desde machine learning
(“nueva” IA)

Data

Old answers

New answers

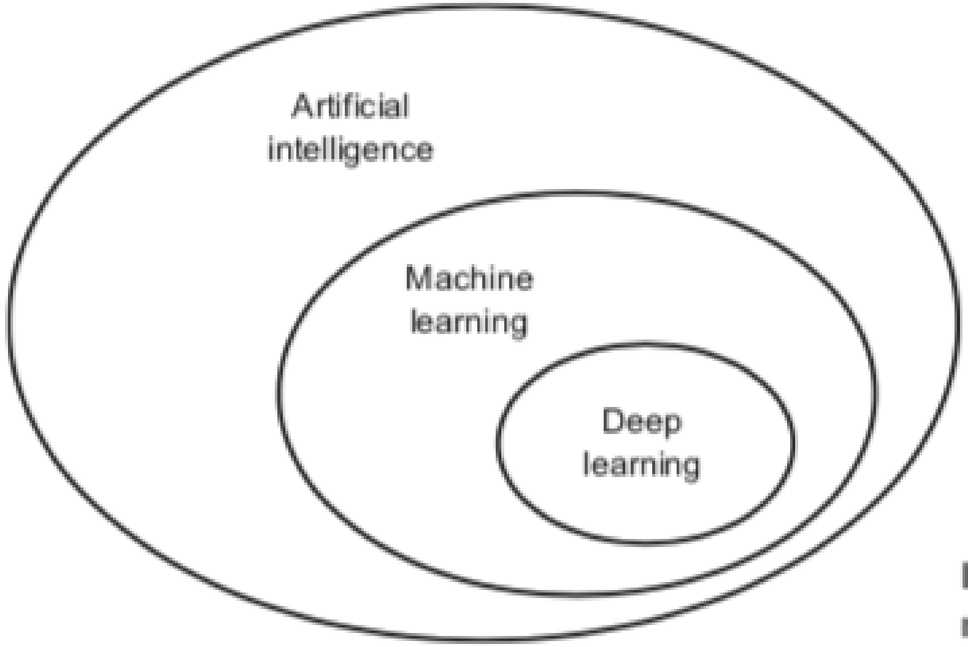


Figure 1.1 Artificial intelligence, machine learning, and deep learning

El campo vector
(las reglas)

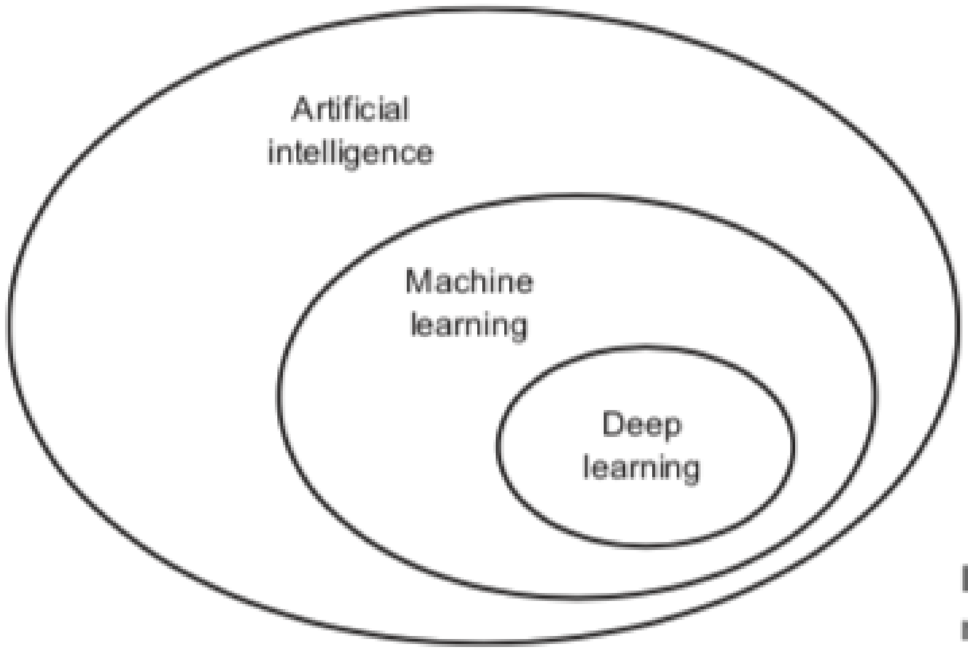
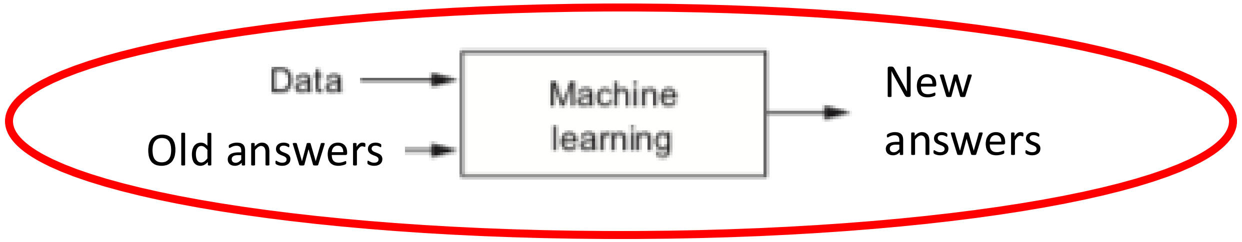


Figure 1.1 Artificial intelligence, machine learning, and deep learning

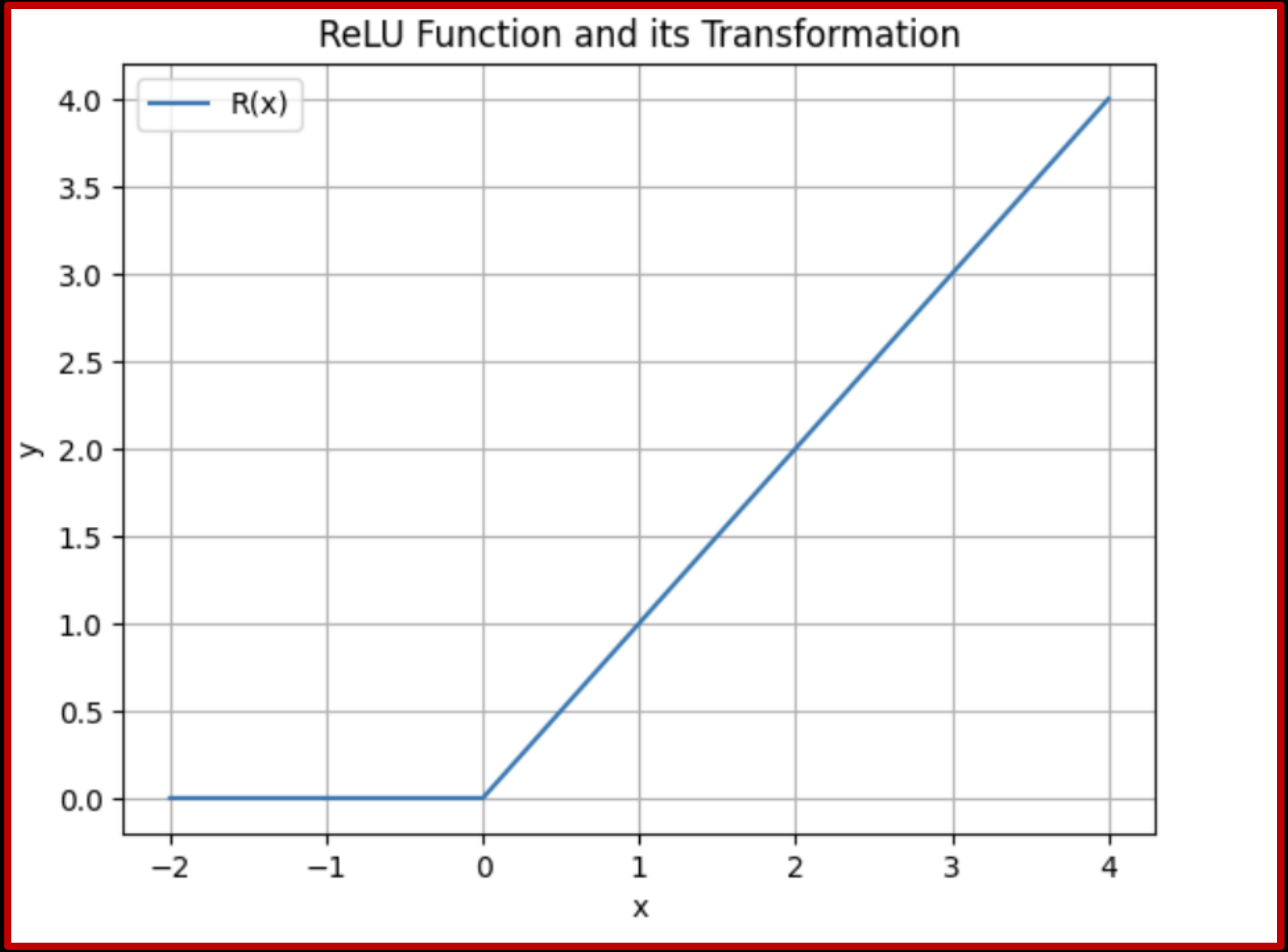


Hay que ***construir funciones*** complicadas y sofisticadas. Como?

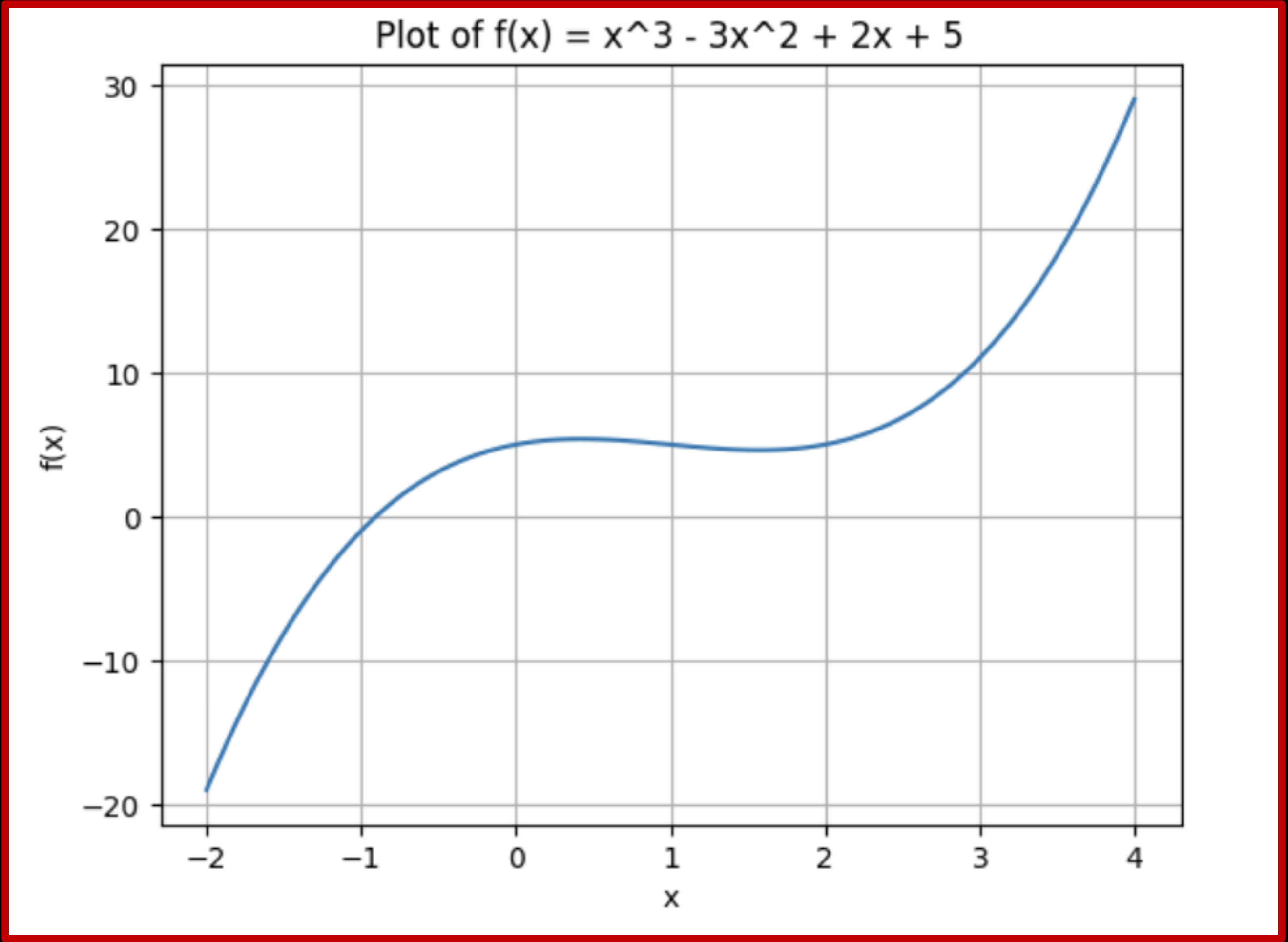
¿Como se “construyen” estas funciones?

Teorema de la aproximacion universal

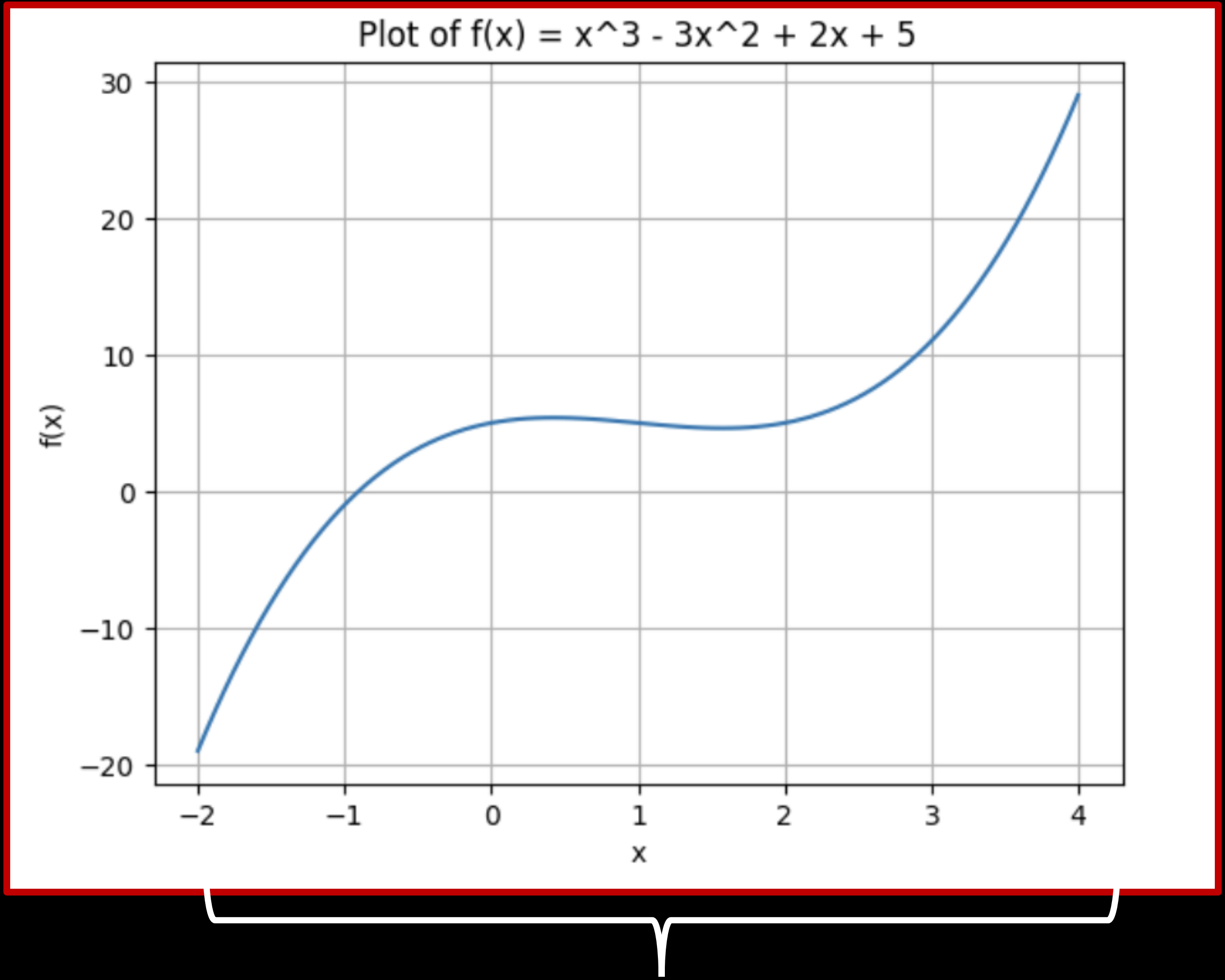
Una red neuronal feedforward, con una capa oculta constituida por un numero finito de neuronas, puede aproximar a cualquier funcion continua en un *rango finito del input*, con cualquier grado de precision, si la funcion de activacion es no lineal, acotada y continua si la funcion de activacion es no lineal, acotada (en el intervalo de interes) y continua



Intentemos aproximar esta funcion

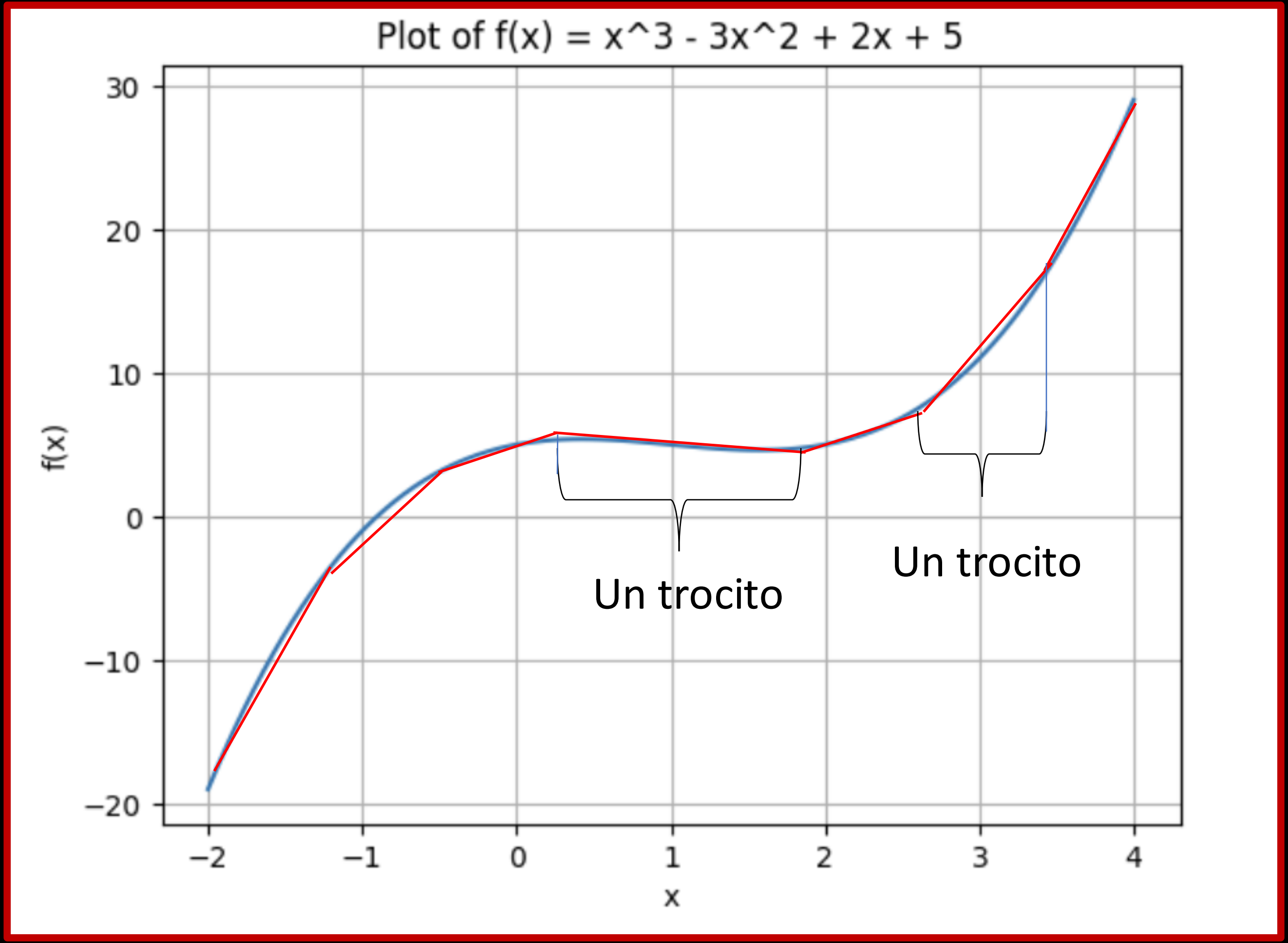


Intentemos aproximar esta funcion



rango finito del que habla el teorema

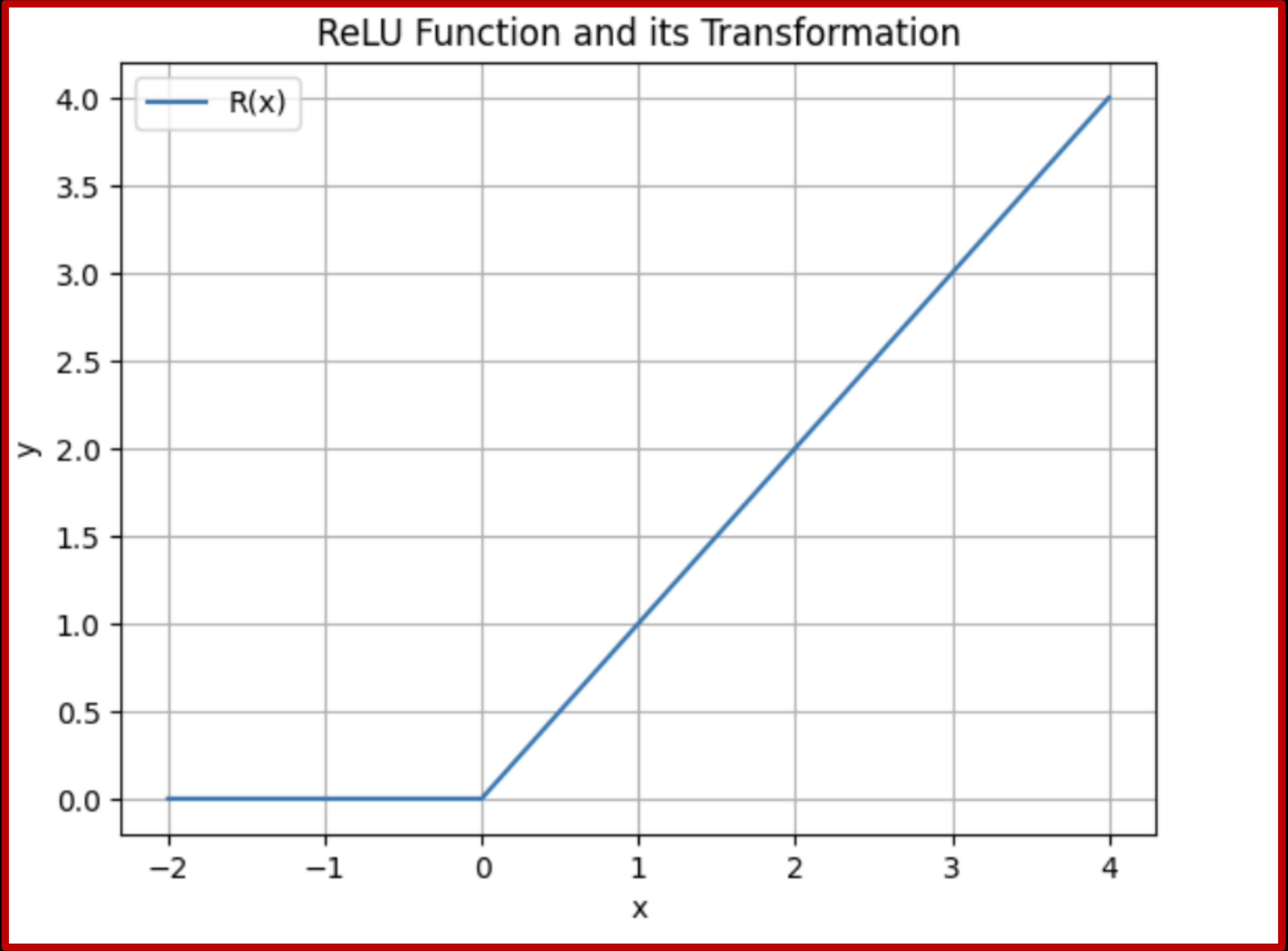
Un modo intuitivamente amigable de aproximar la funcion es pensar en un conjunto finito de rectas, con pendientes cercanas a las tangentes en algun conjunto de puntos del dominio.



Como generar esas rectas?
(con funciones en todo el rango!)

Vamos a introducir la funcion ReLu (rectified unit),
Y a trabajar con ella para poder lograr el ajuste deseado

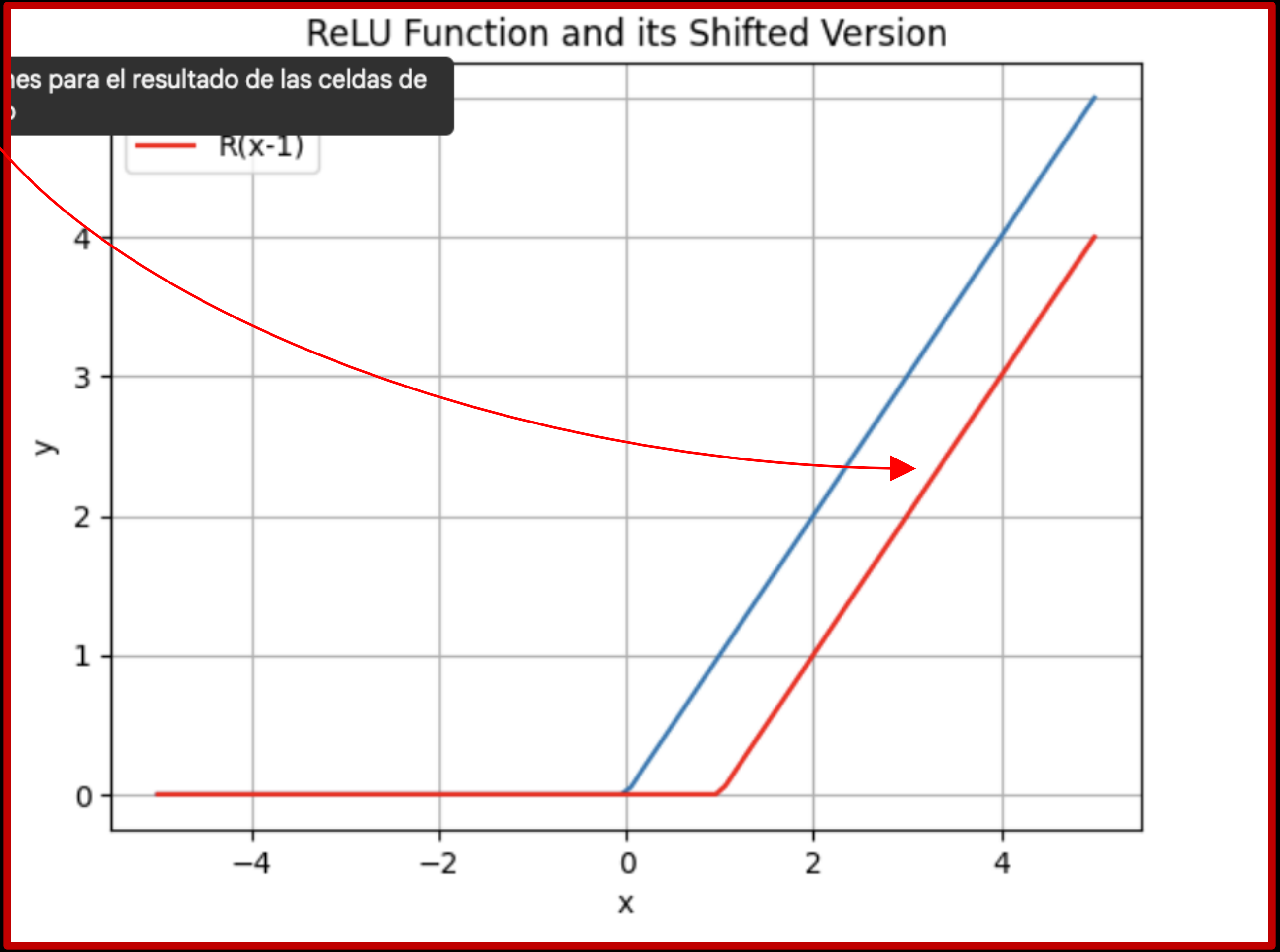
R(x)=max(0,x)



$R(x) = \max(0, x)$

En *x=0* tenemos el “quiebre”

Como construimos otra funcion, a partir de esta, con el “quiebre” en *x=1* ?



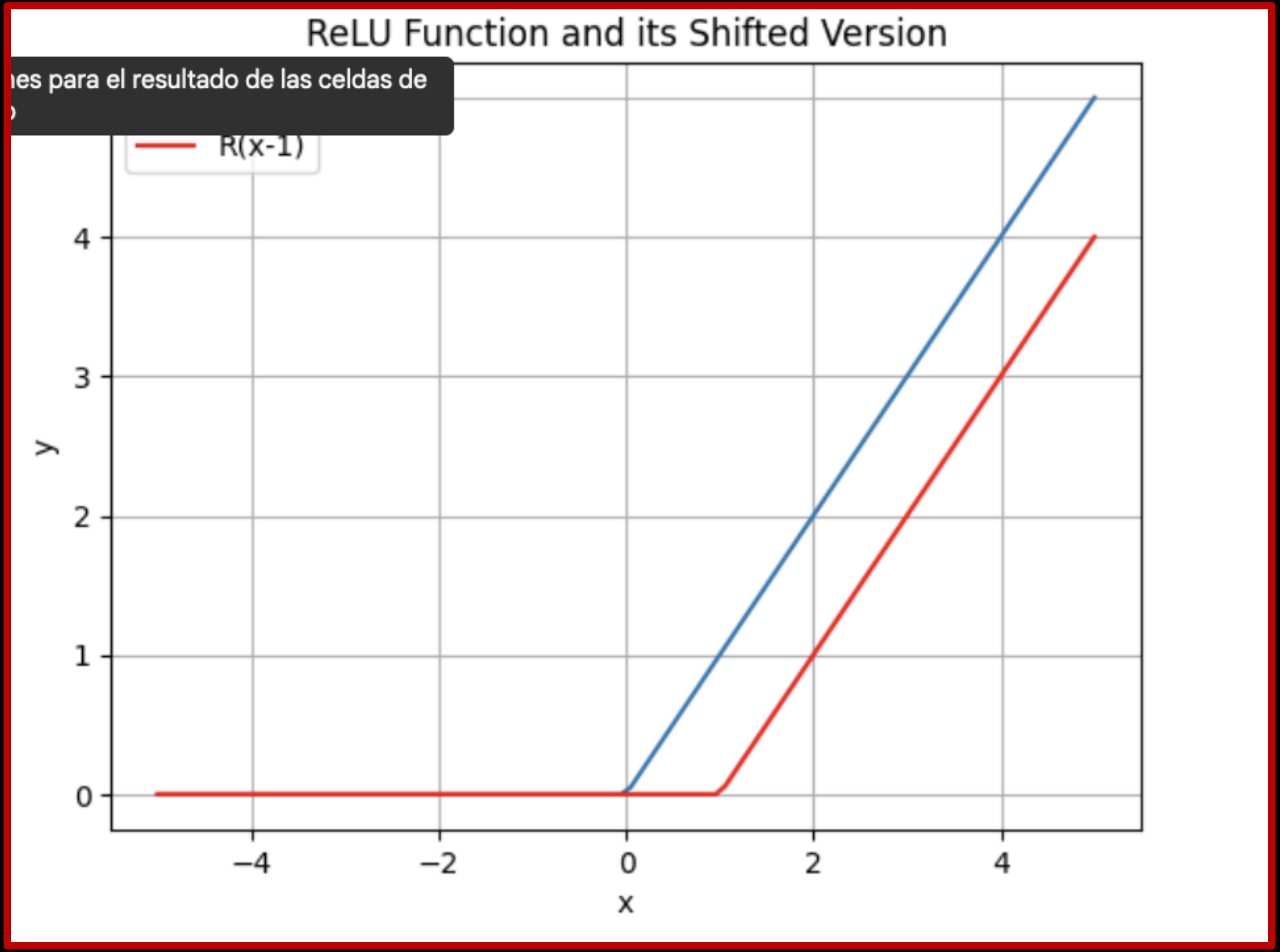
$R(x) = \max(0, x)$

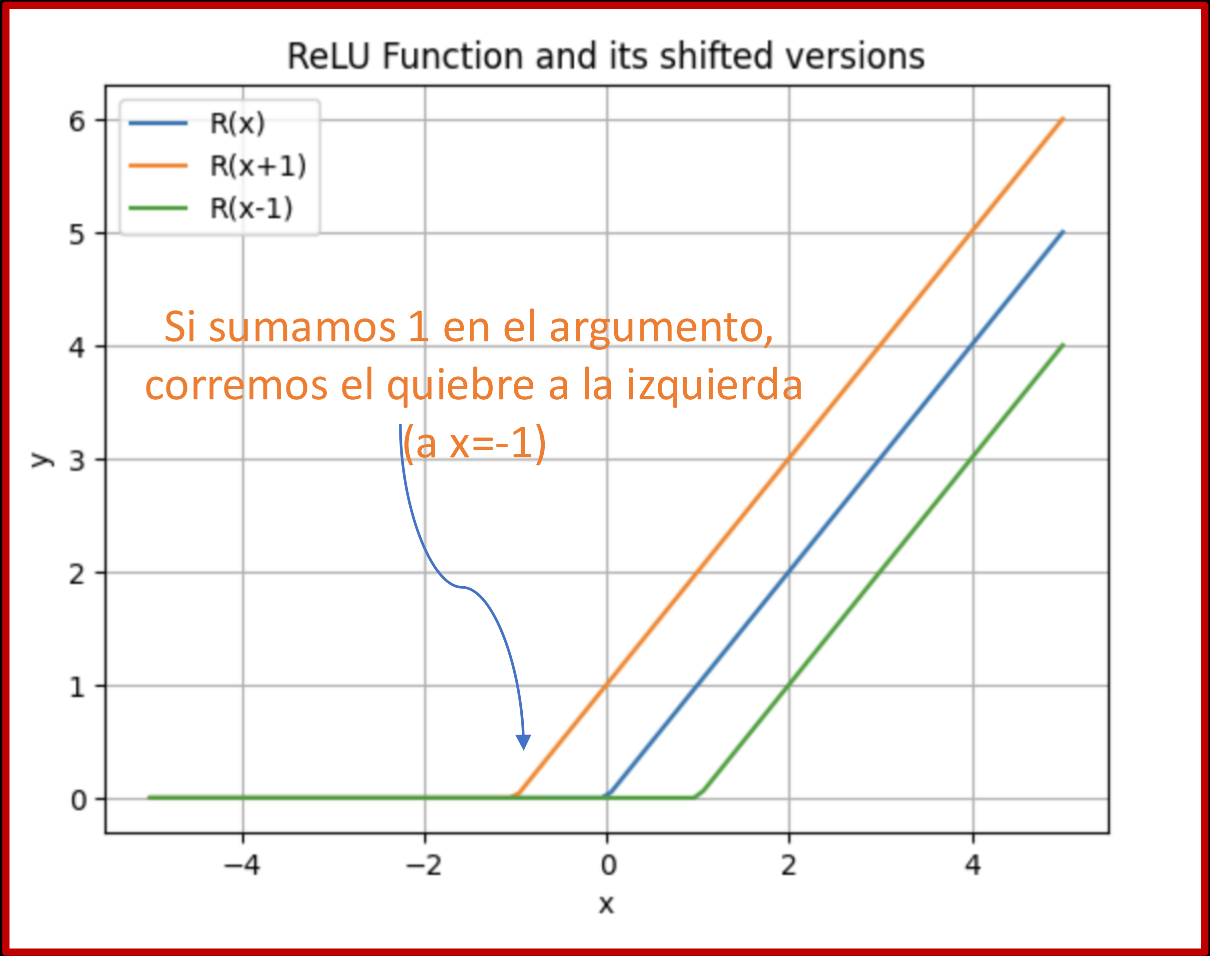
En *x=0* tenemos el “quiebre”

Construyamos otra funcion, con el “quiebre” en *x=1*

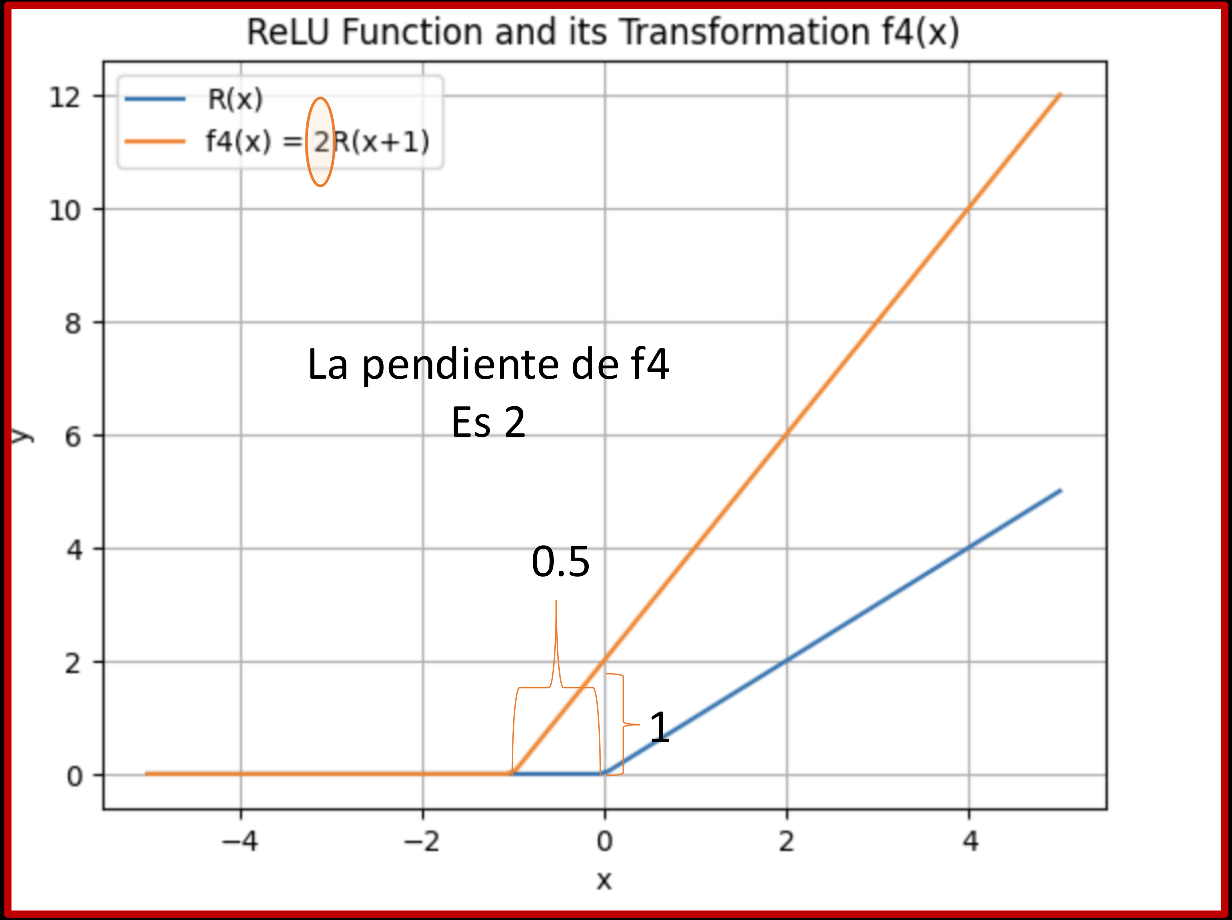
$f_1 = R(x-1)$

En *x-1=0* tenemos el “quiebre”

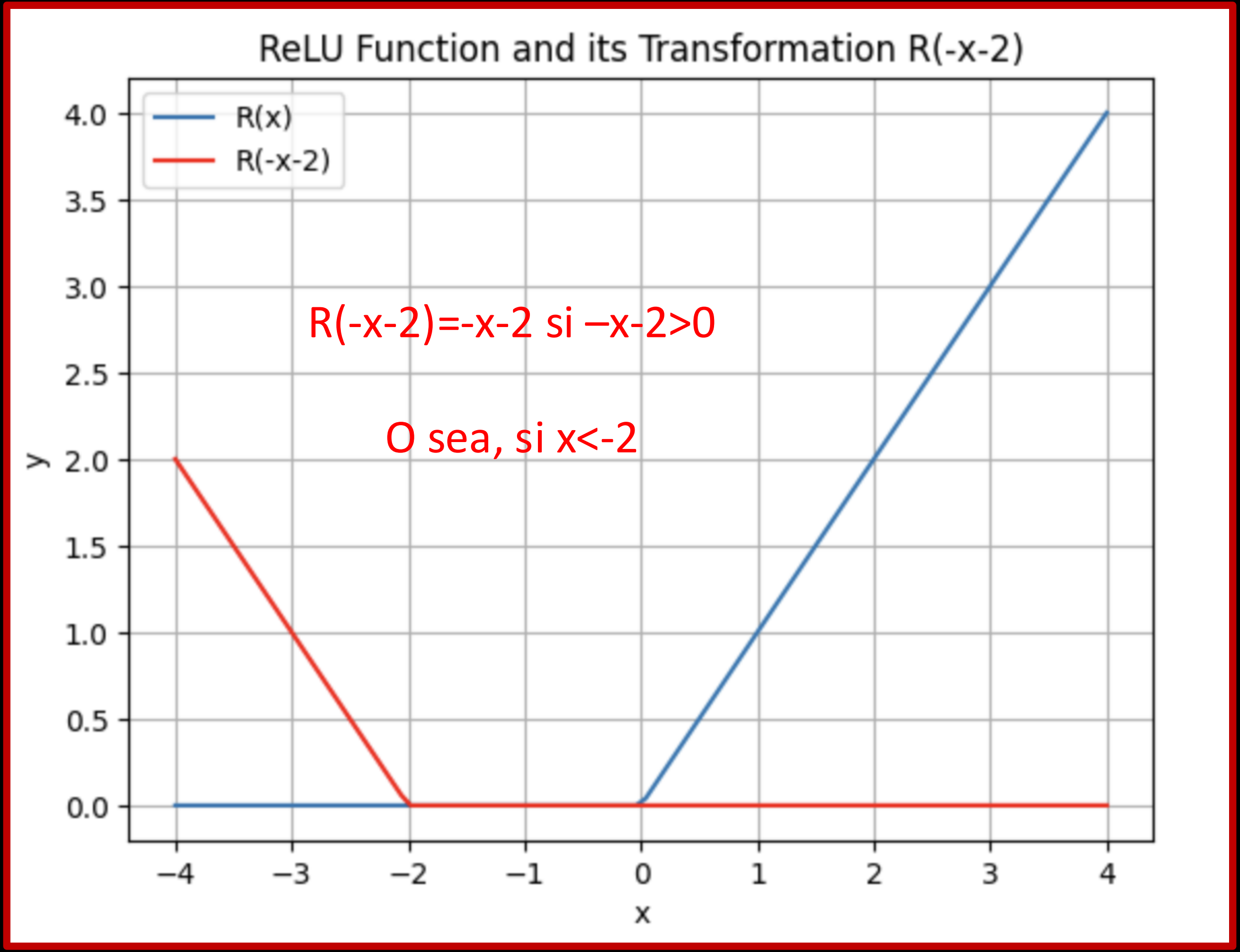



ReLU Function and its shifted versions
R(x)
R(x+1)
R(x-1)
Si sumamos 1 en el argumento, corremos el quiebre a la izquierda (a x=-1)
y
x
6
5
4
3
2
1
0
−4
−2
0
2
4

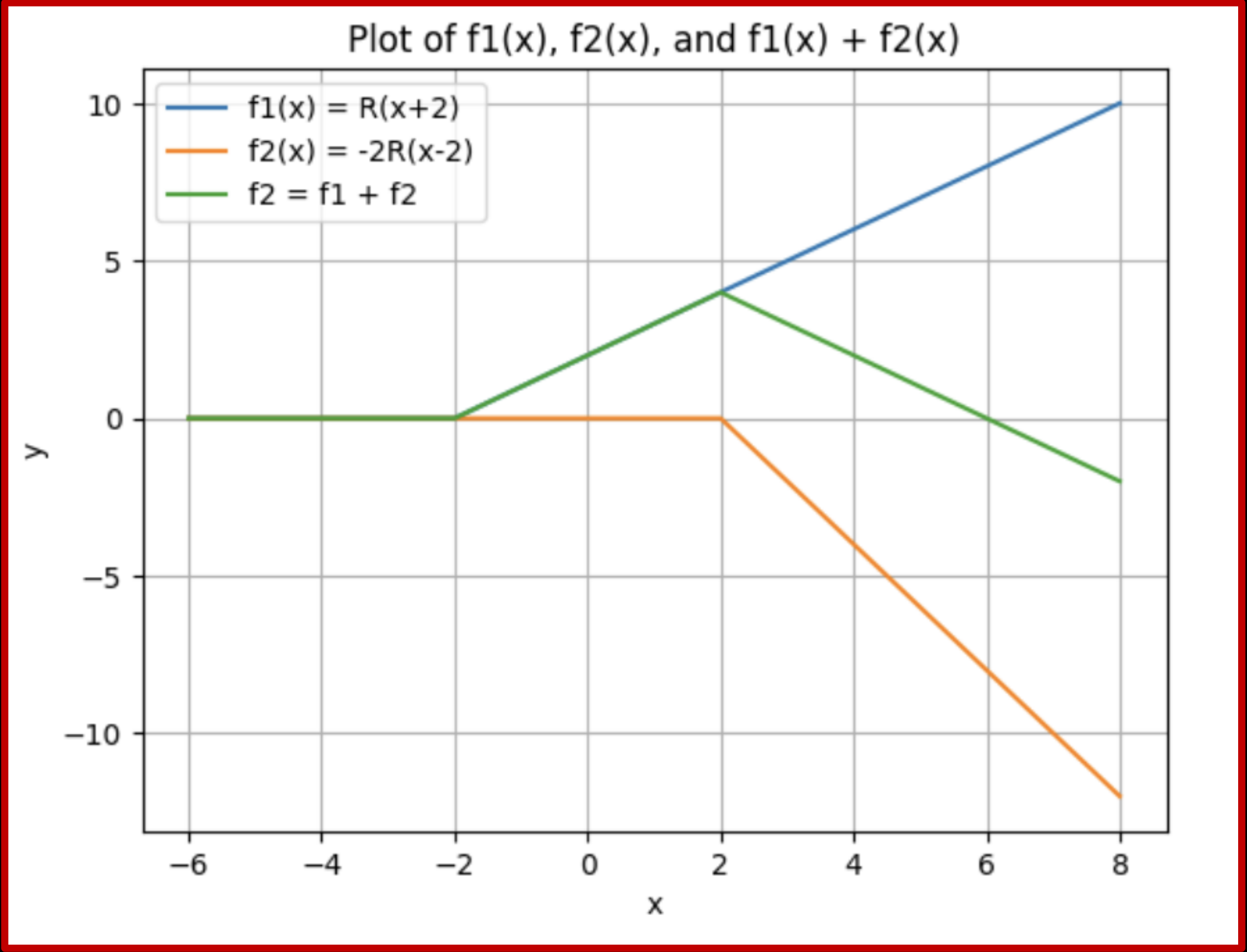
Tambien podemos cambiar la pendiente de las Funciones, multiplicando a la ReLu por un factor



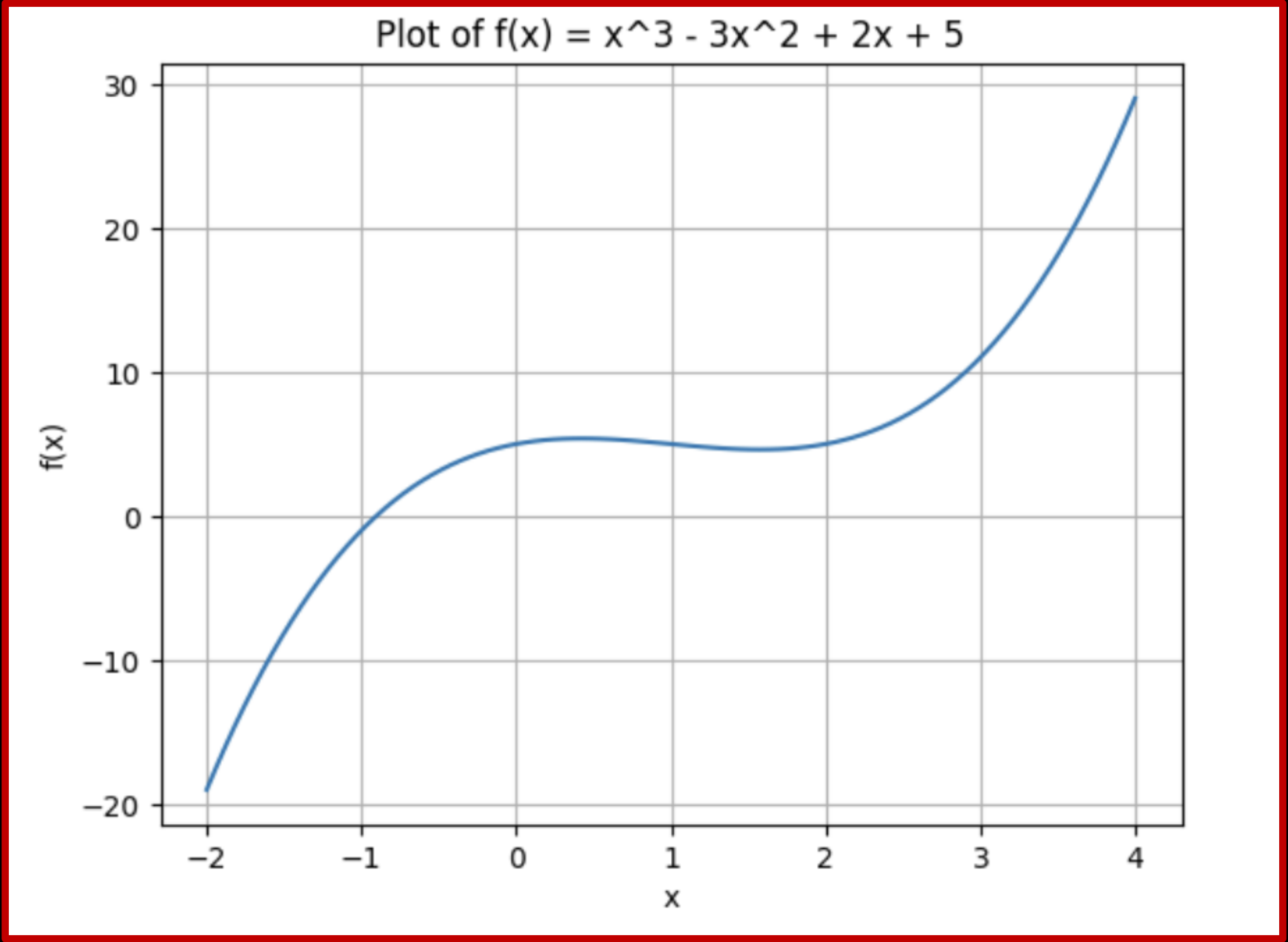
Cambiando el signo de la variable en el argumento,
Que la funcion se planche en 0 despues del quiebre.



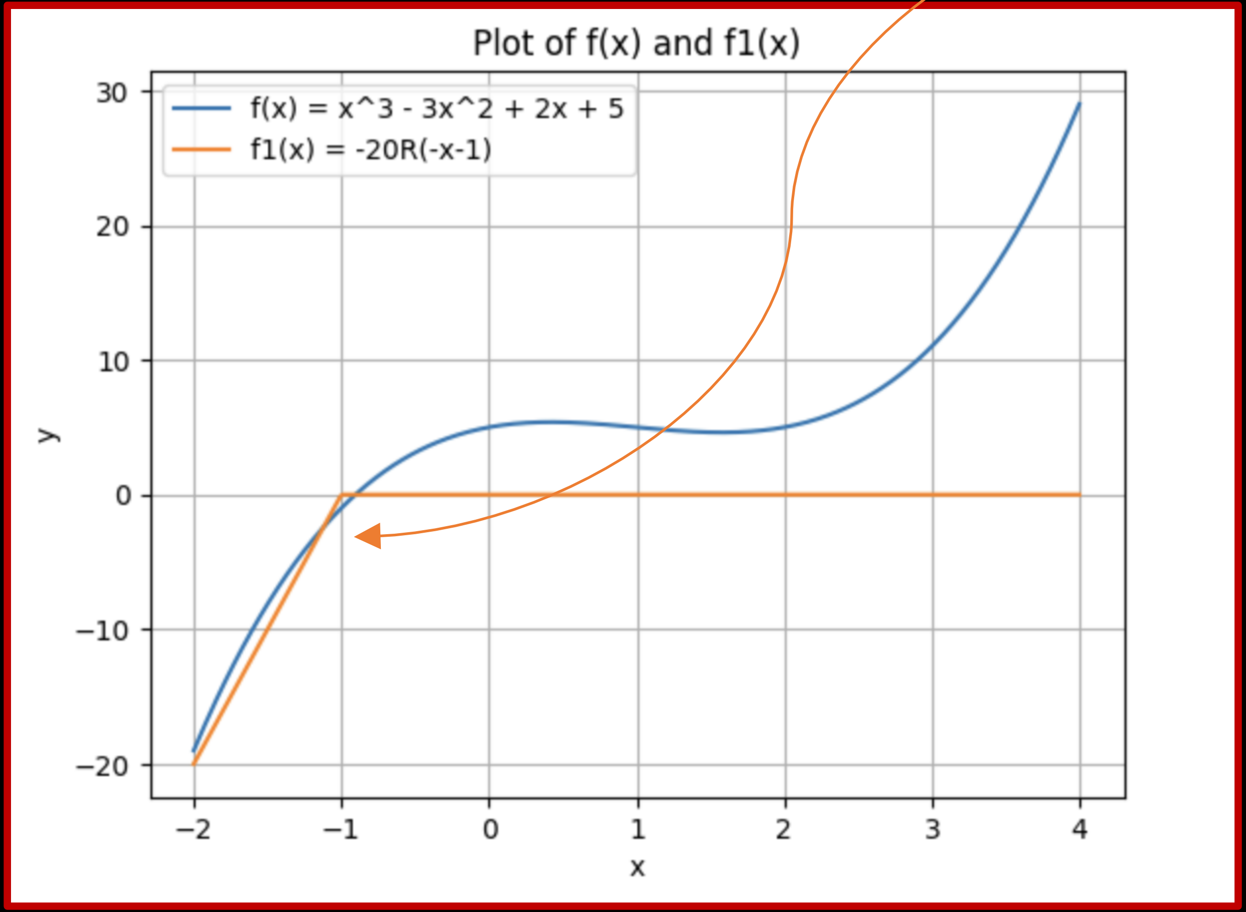
Sumando dos funciones con quiebres en dos puntos, armo una funcion de tres tramos



Intentemos aproximar esta funcion... con Relus



En este rango !



Empezamos con una funcion
Que se planche a partir de
Si x>-1, o sea, cuando x+1>0

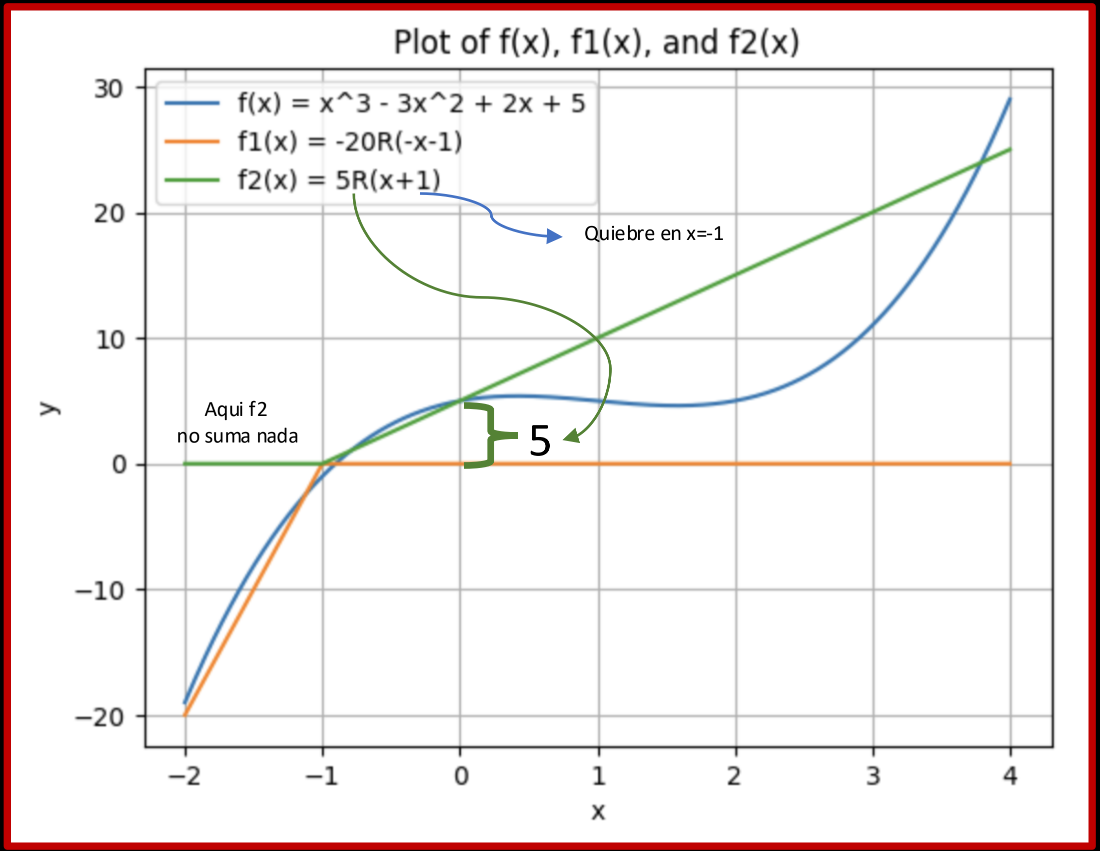
Pero como la ReLu se hace
Cero si el argumento es negativo

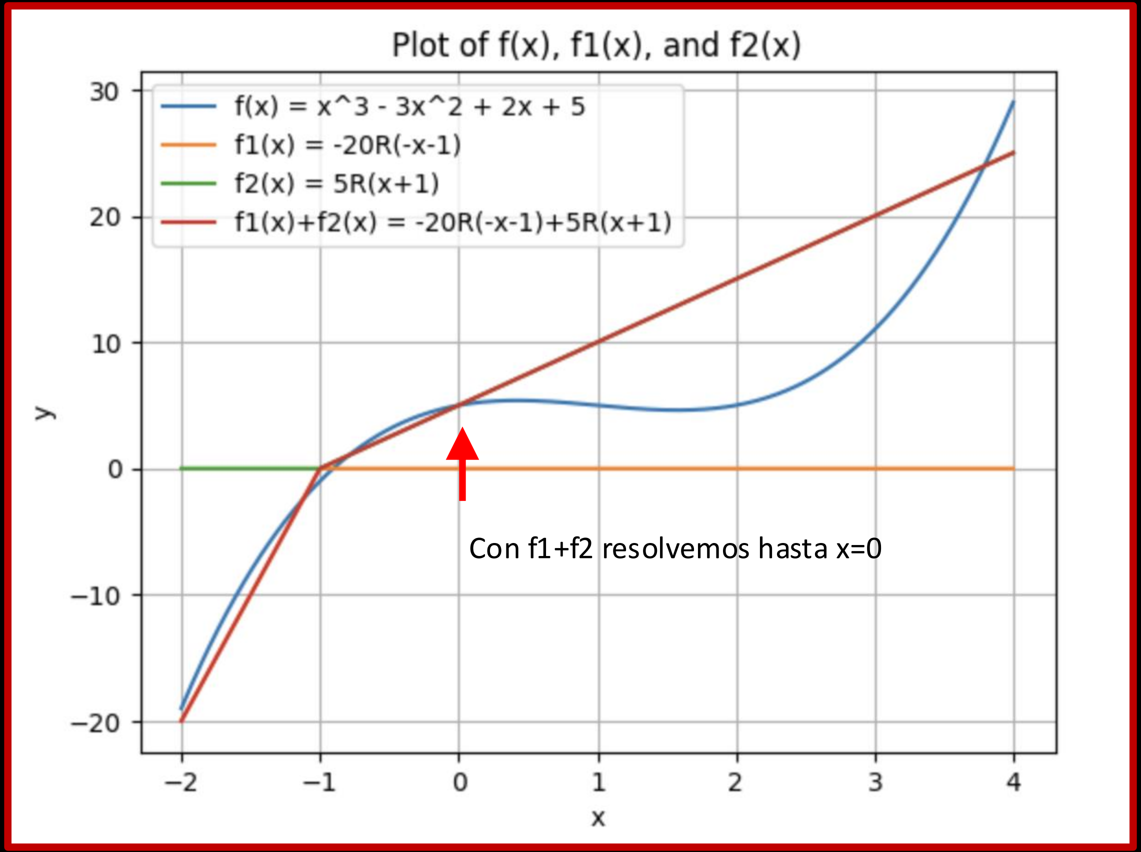
El argumento de ReLu
Debe ser negativo si x+1>0

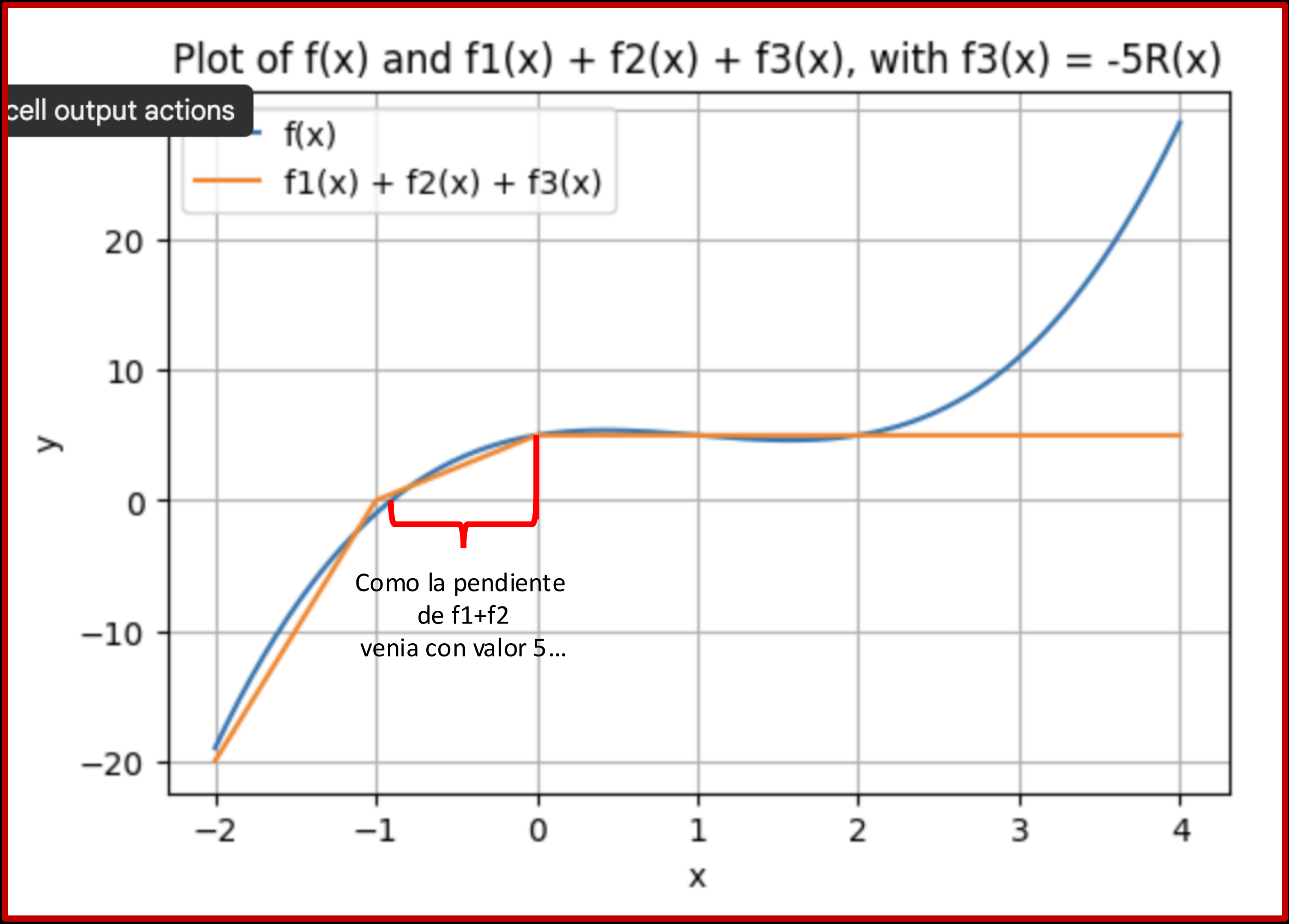
F= alfa ReLu(-(x+1))

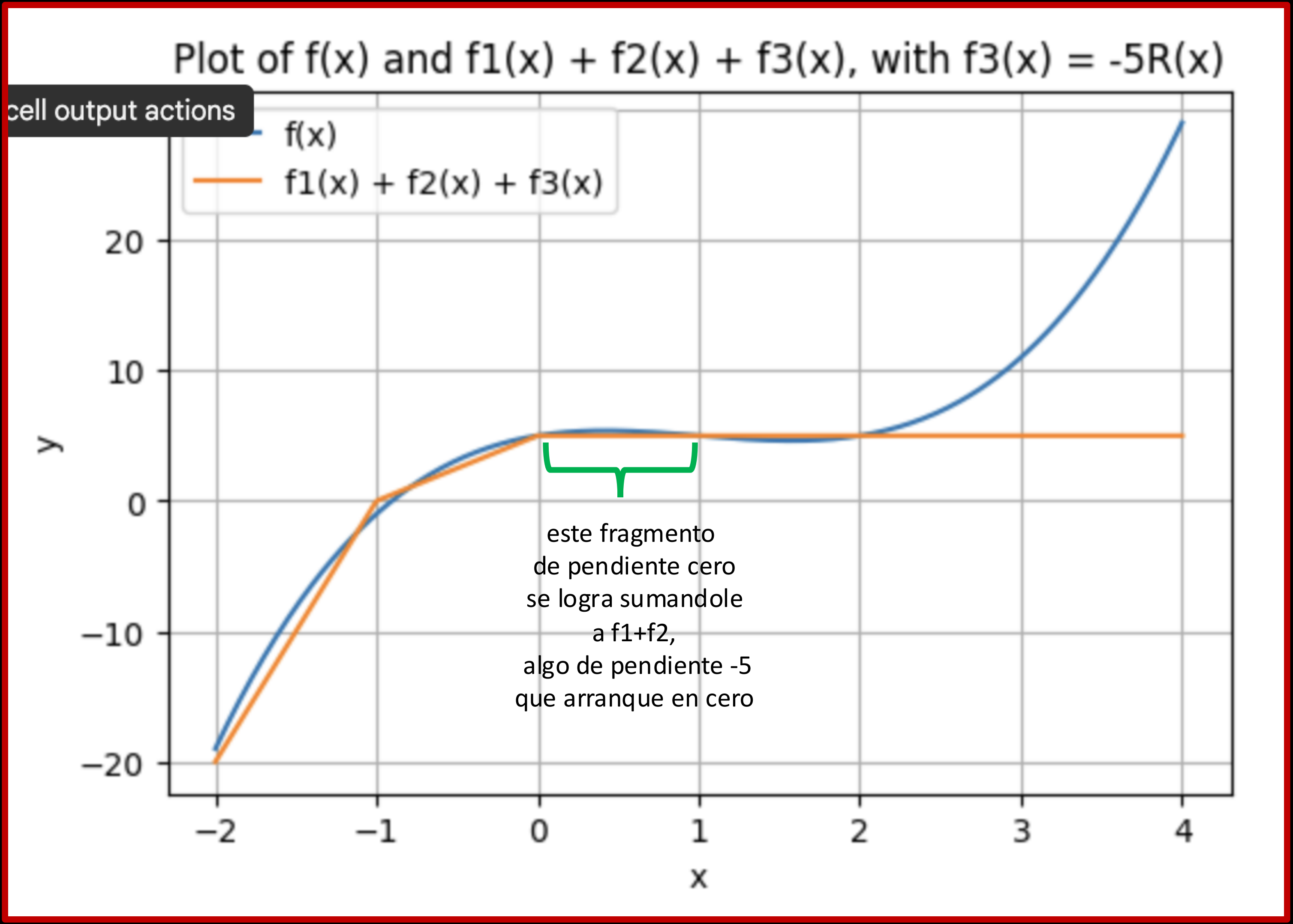
Y como quiero que la pendiente
Sea -20,

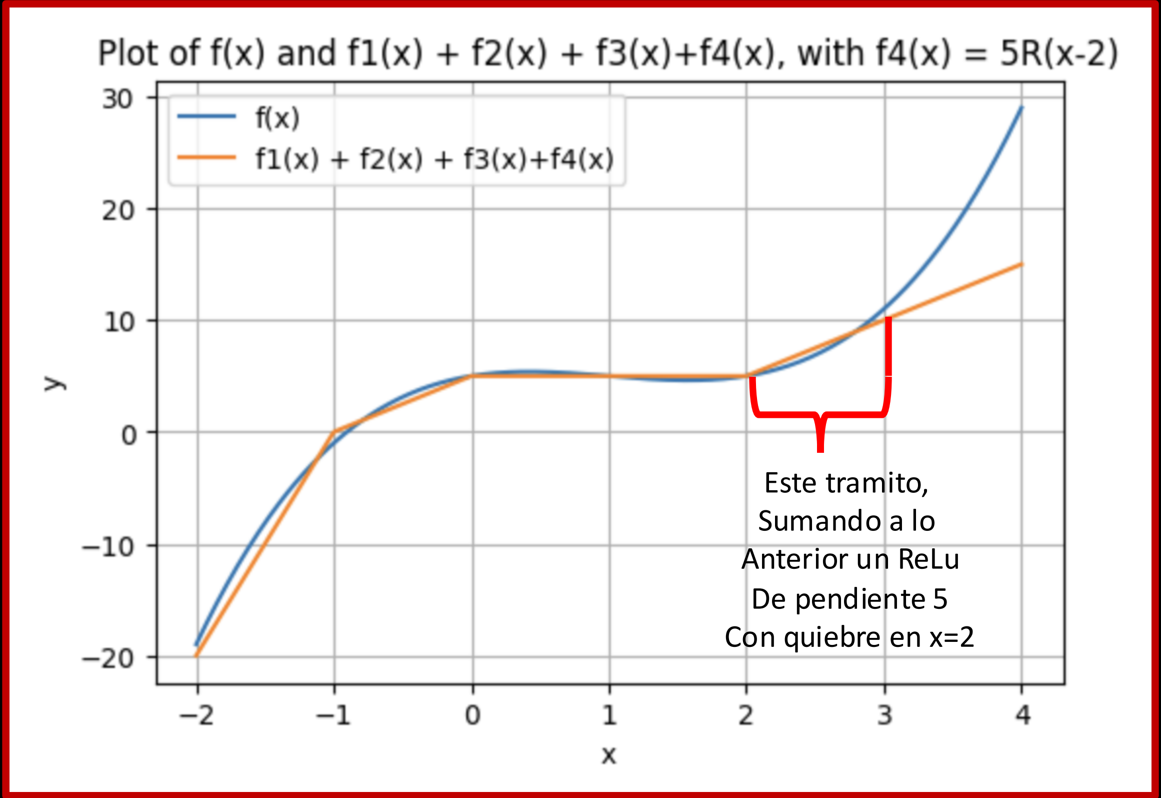
f1(x)=-20ReLu(-x-1)

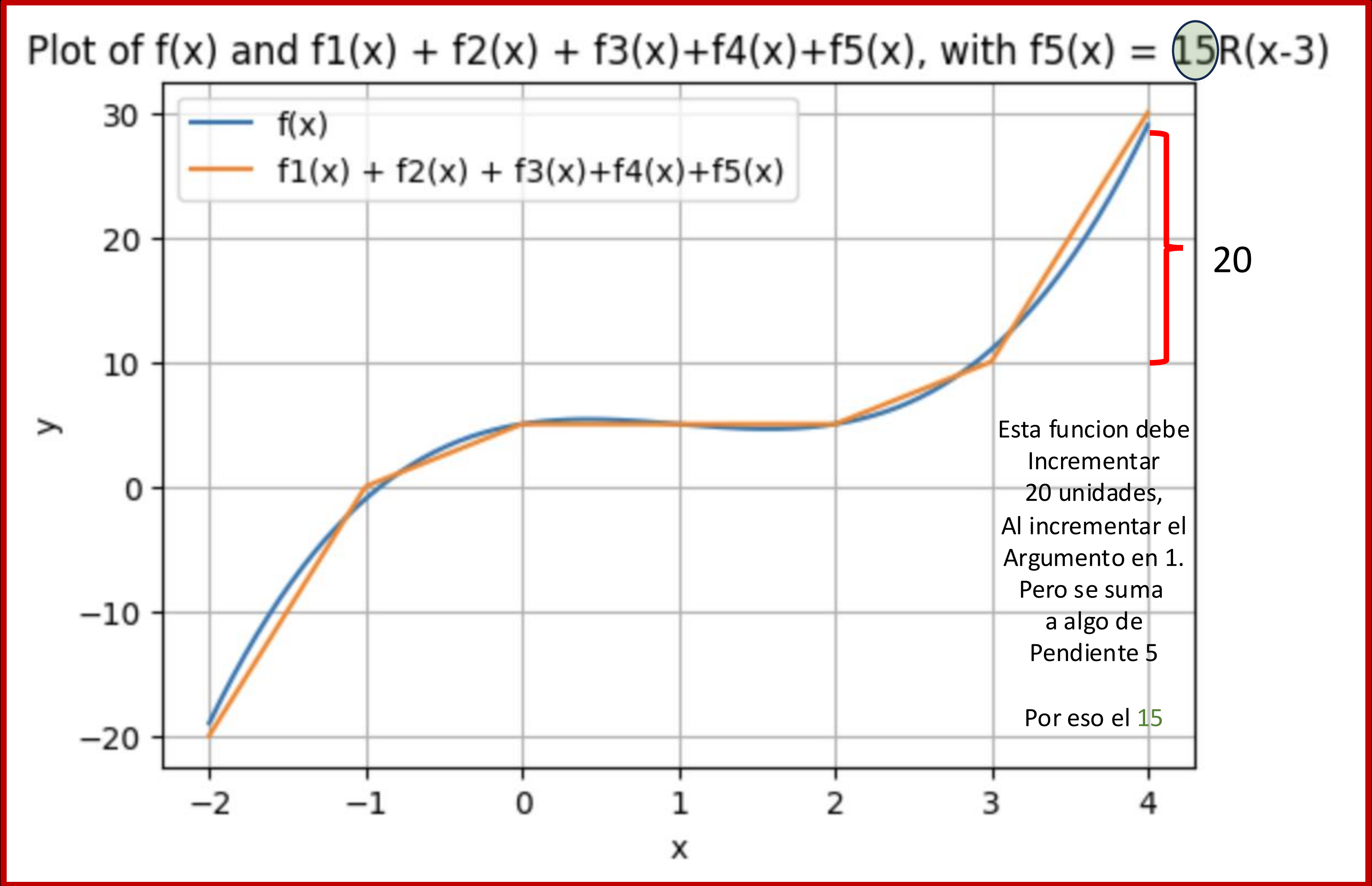

Plot of f(x), f1(x), and f2(x)
f(x) = x^3 - 3x^2 + 2x + 5
f1(x) = -20R(-x-1)
f2(x) = 5R(x+1)
Quiebre en x=-1
Aqui f2
no suma nada
5
y
x


Plot of f(x), f1(x), and f2(x)
f(x) = x^3 - 3x^2 + 2x + 5
f1(x) = -20R(-x-1)
f2(x) = 5R(x+1)
f1(x)+f2(x) = -20R(-x-1)+5R(x+1)
Con f1+f2 resolvemos hasta x=0
y
x


Plot of f(x) and f1(x) + f2(x) + f3(x), with f3(x) = -5R(x)
cell output actions
f(x)
f1(x) + f2(x) + f3(x)
20
10
0
−10
−20
y
−2
−1
0
1
2
3
4
x
Como la pendiente
de f1+f2
venia con valor 5…

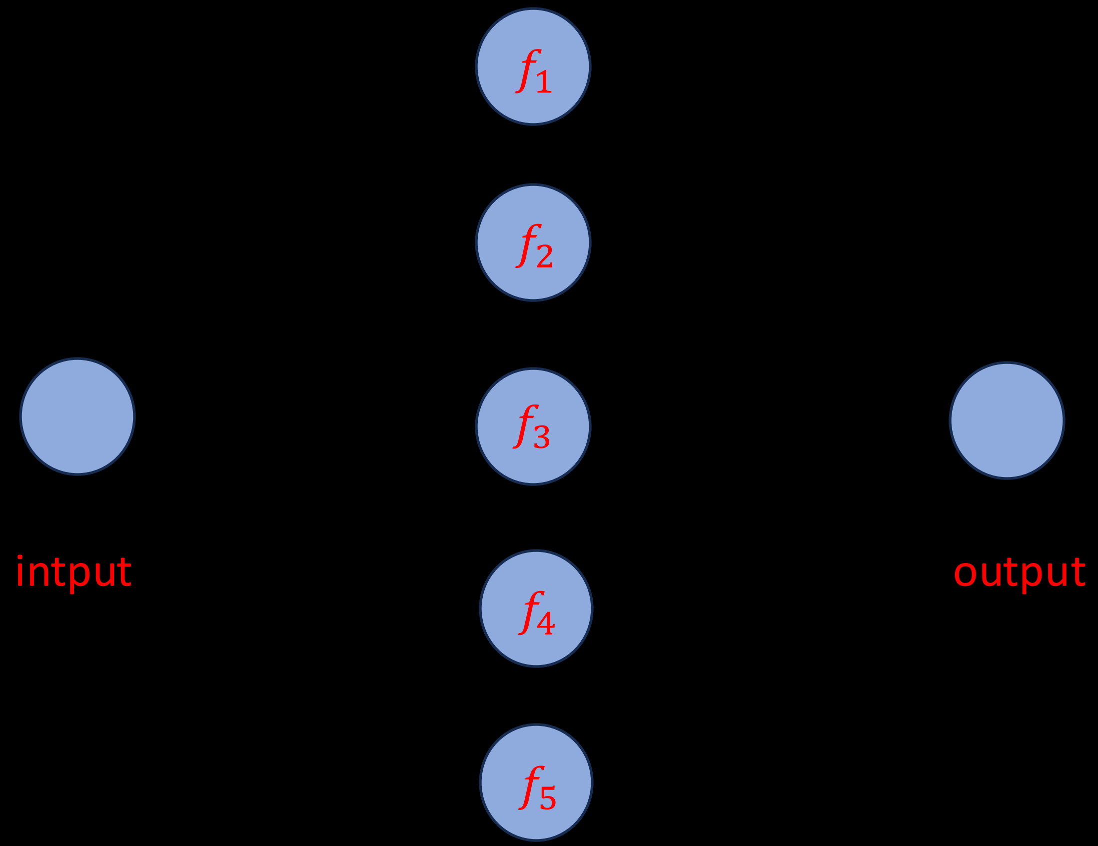

Plot of f(x) and f1(x) + f2(x) + f3(x), with f3(x) = -5R(x)
cell output actions
f(x)
f1(x) + f2(x) + f3(x)
20
10
0
−10
−20
y
−2
−1
0
1
2
3
4
x
este fragmento
de pendiente cero
se logra sumandole
a f1+f2,
algo de pendiente -5
que arranque en cero


Plot of f(x) and f1(x) + f2(x) + f3(x)+f4(x), with f4(x) = 5R(x-2)
f(x)
f1(x) + f2(x) + f3(x)+f4(x)
30
20
10
0
−10
−20
y
−2
−1
0
1
2
3
4
x
Este tramito,
Sumando a lo
Anterior un ReLu
De pendiente 5
Con quiebre en x=2

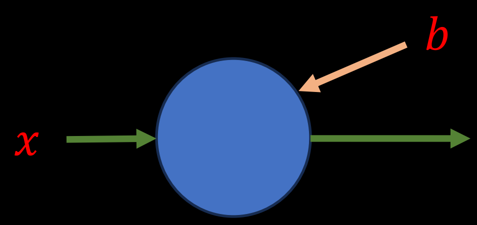

Plot of f(x) and f1(x) + f2(x) + f3(x)+f4(x)+f5(x), with f5(x) = 15R(x-3)
f(x)
f1(x) + f2(x) + f3(x)+f4(x)+f5(x)
30
20
10
0
−10
−20
y
−2
−1
0
1
2
3
4
x
20
Esta funcion debe
Incrementar
20 unidades,
Al incrementar el
Argumento en 1.
Pero se suma
a algo de
Pendiente 5
Por eso el 15

Ahora bien: como representar a estas funciones como una red neuronal?

Ahora bien: como representar a estas funciones como una red neuronal?

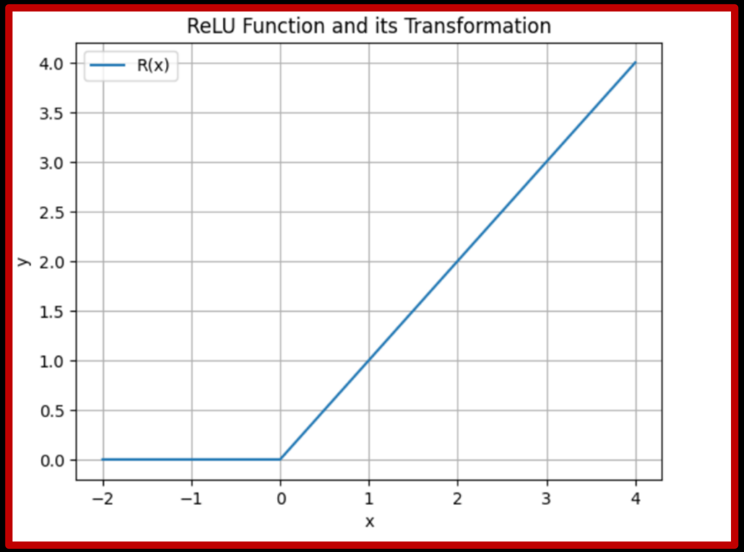


Donde cada unidad se rige por estas reglas:

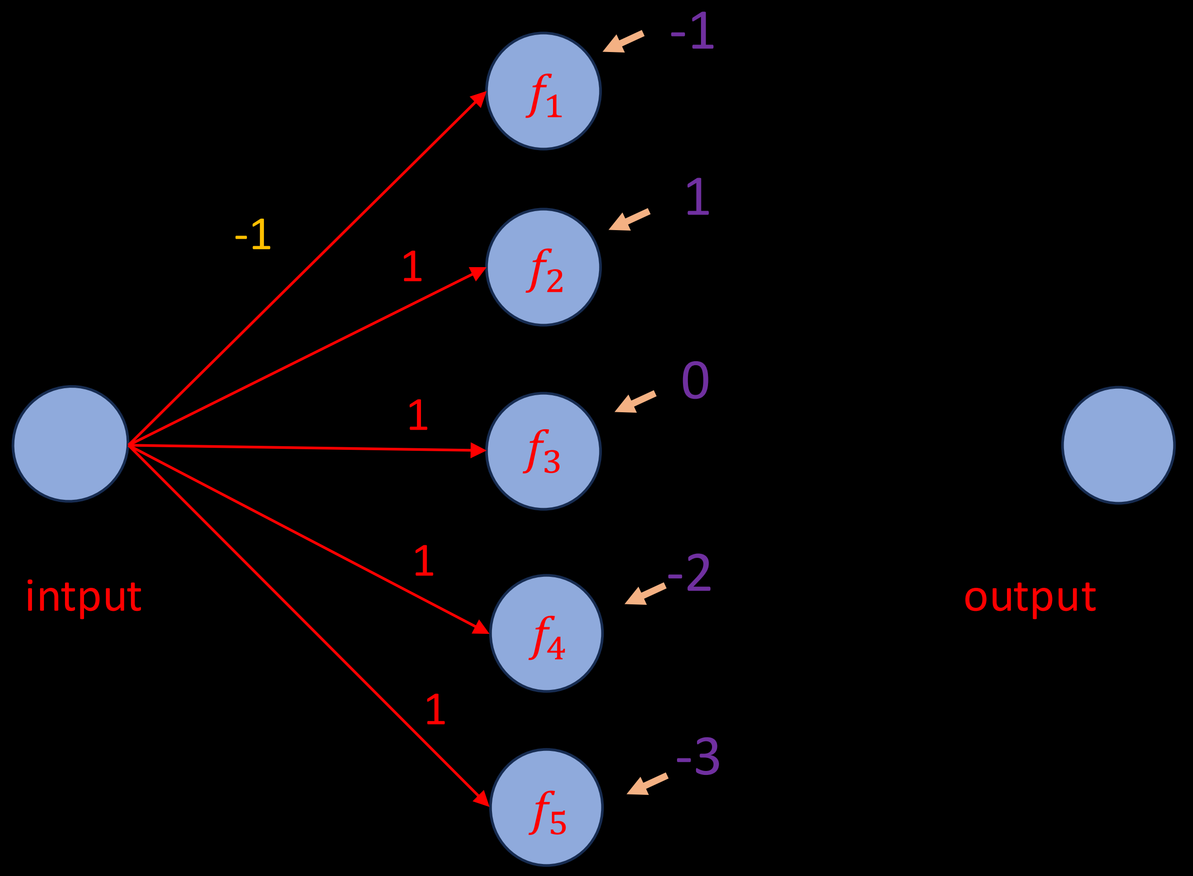


$output(x) = R(x - b))$

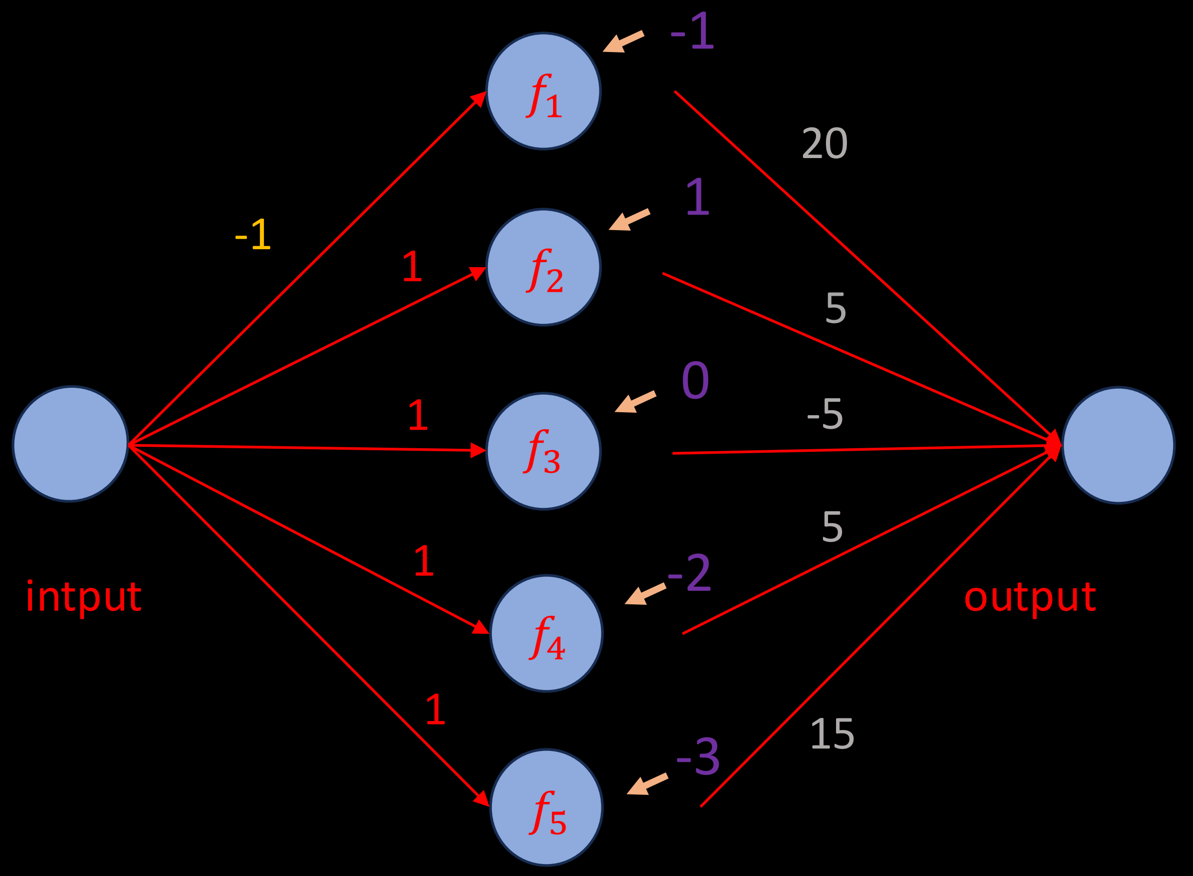
$R(x) = \max(0, x)$, b *se conoce como "bias"*)

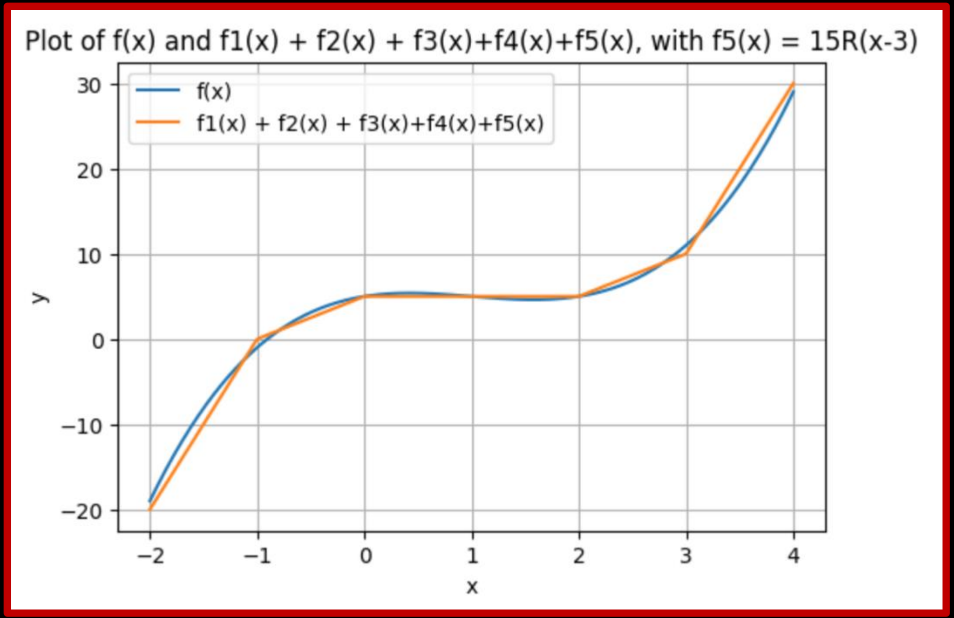


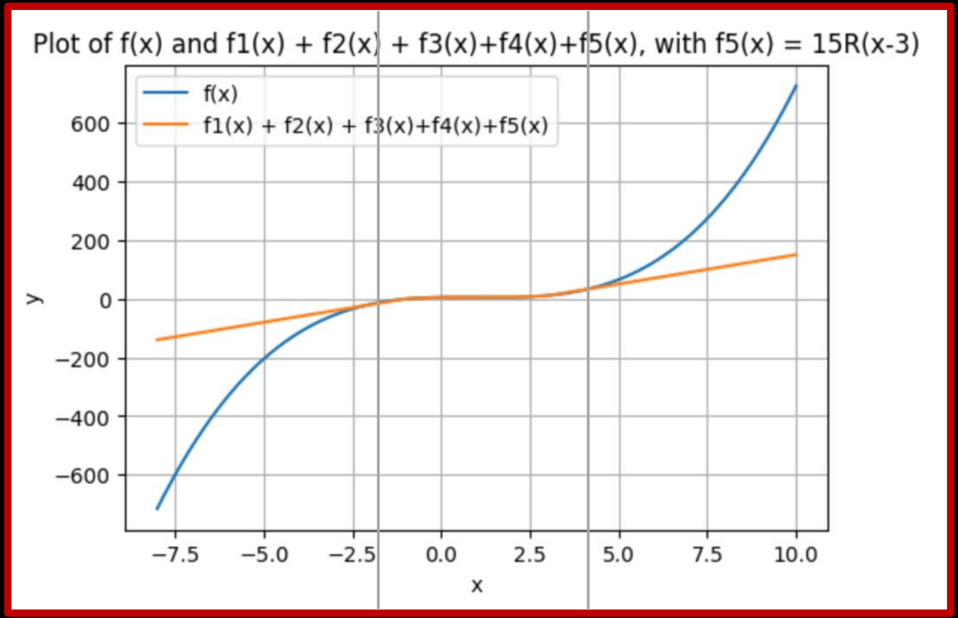
$$Output(x) = 20\,R(-x-1) + 5\,R(x+1) - 5R(x) + 5\,R(x-2) + 15\,R(x-3)$$



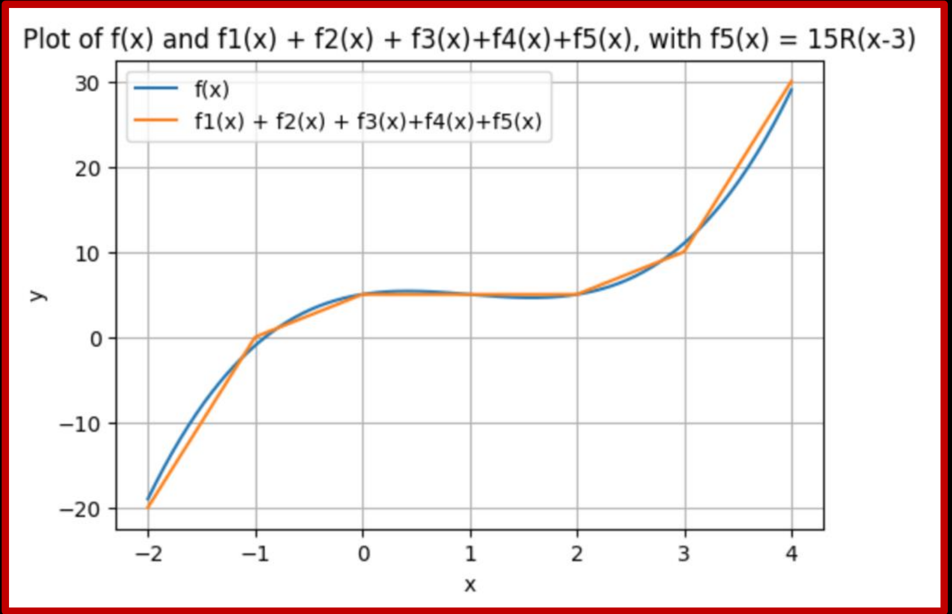
$$Output(x) = 20\,R(-x-1) + 5\,R(x+1) - 5R(x) + 5\,R(x-2) + 15\,R(x-3)$$

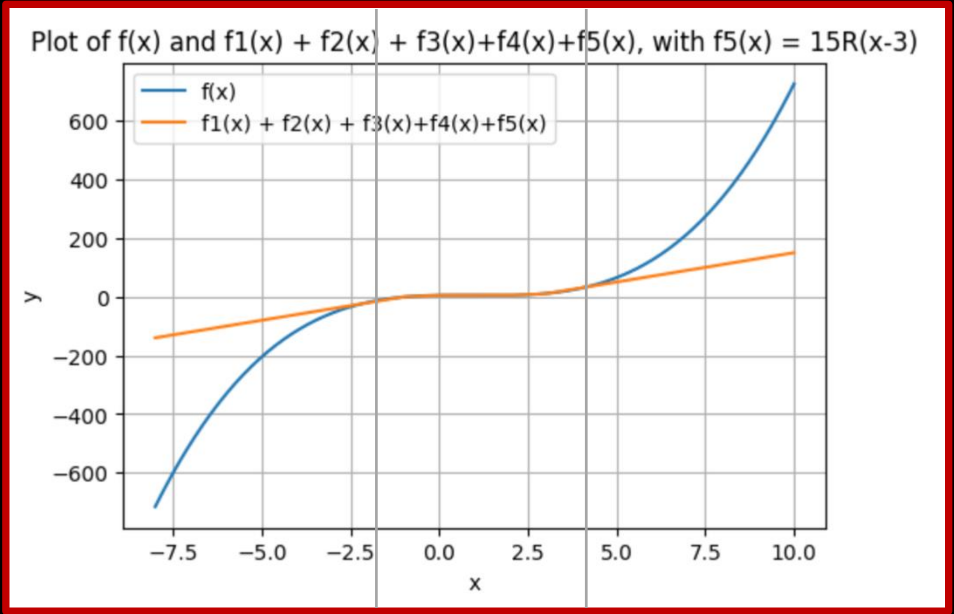






Buena aproximacion
en un rango dado!





Buena aproximacion
en un rango dado!

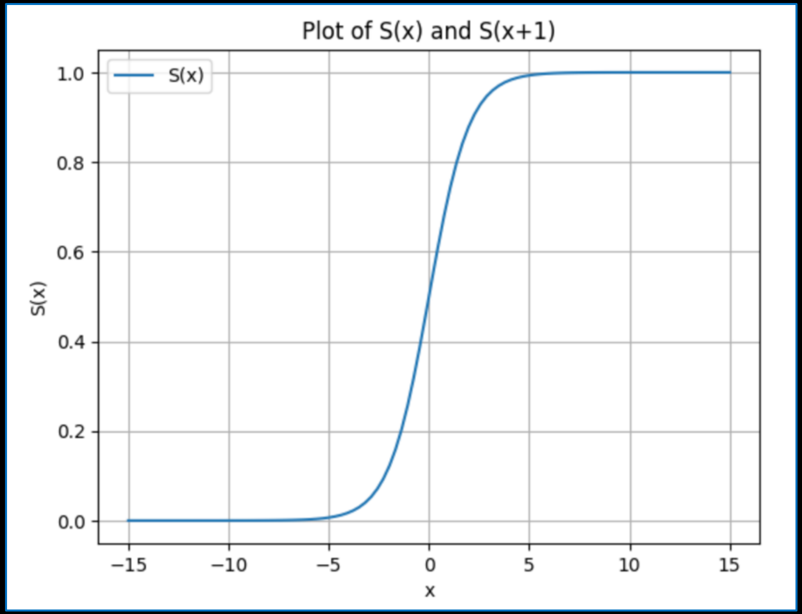
Y por supuesto, si las R(x) hubiesen sido lineales… la combinacion hubiera sido… otra recta

Teorema de aproximacion universal

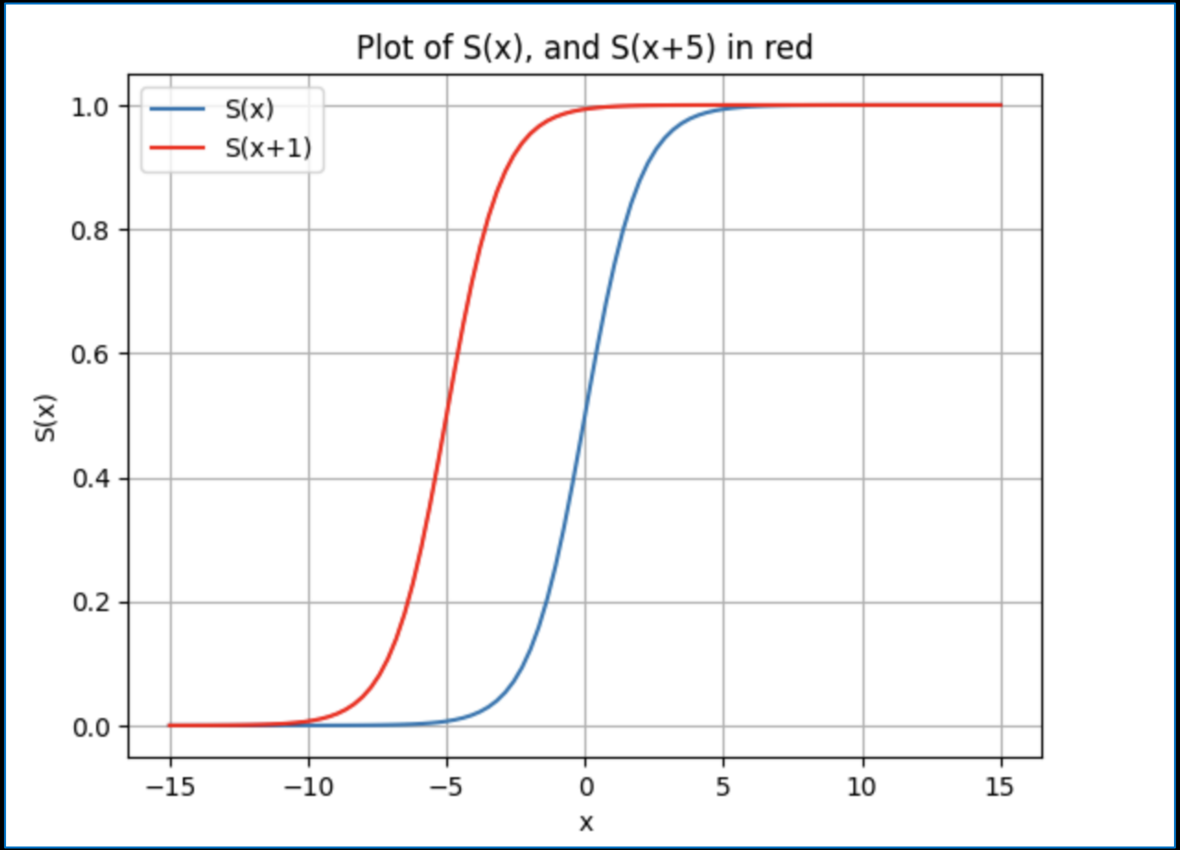
Una red neuronal feedforward, con una capa oculta constituida por un numero finito de neuronas, puede aproximar a cualquier funcion continua en un rango finito del input, con cualquier grado de precision, si la funcion de activacion es no lineal, acotada y continua

Con otras funciones no lineales
puede argumentarse de modo similar.

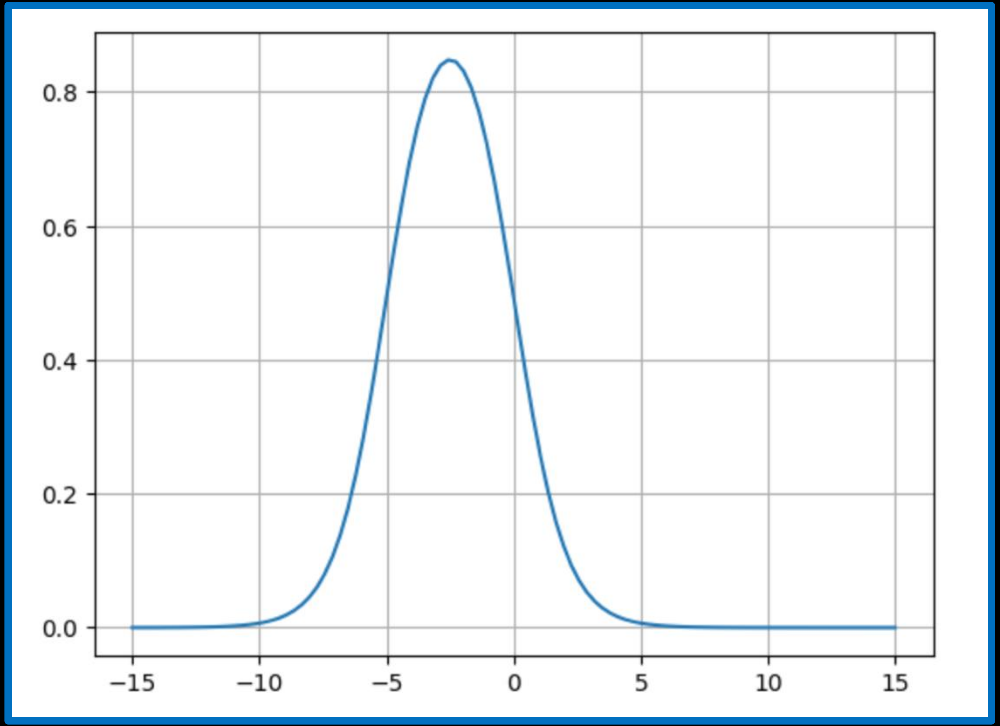
$$S(x) = \frac{1}{1 + e^{-x}}$$



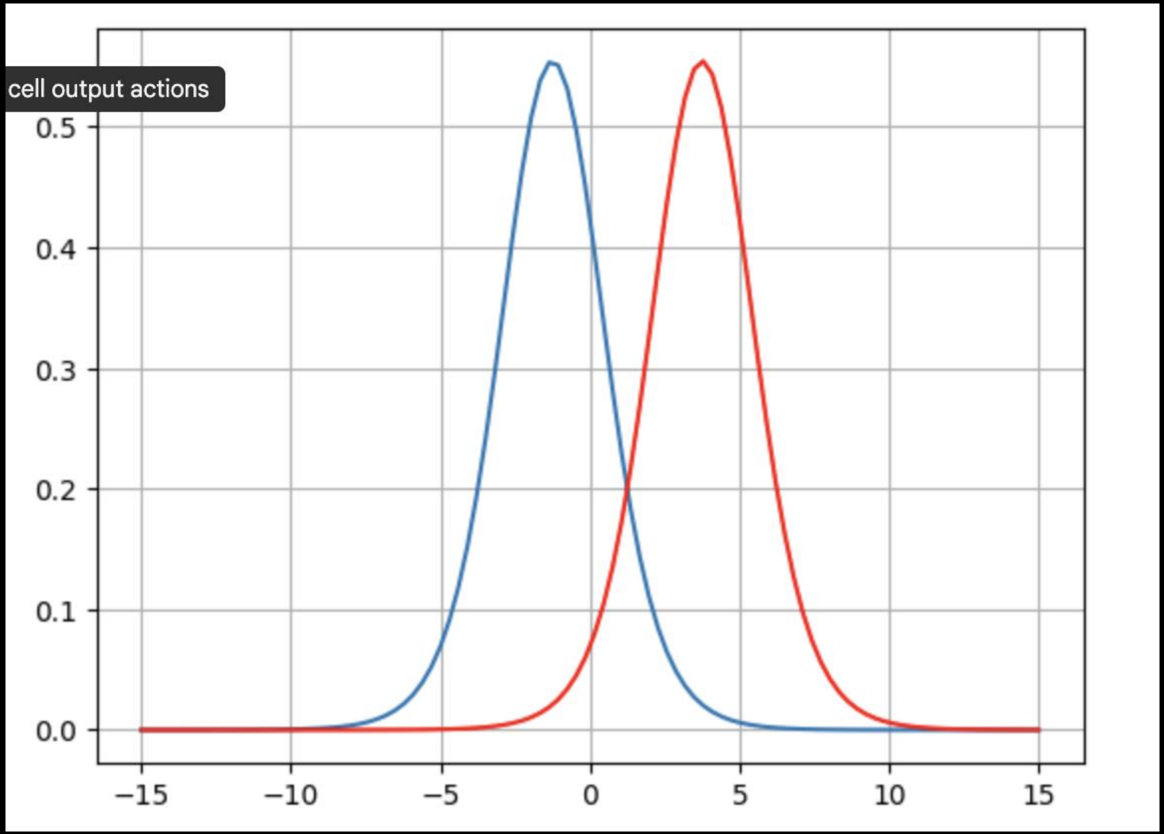
$$S(x+5) = \frac{1}{1+e^{-(x+5)}} \qquad S(x) = \frac{1}{1+e^{-x}}$$



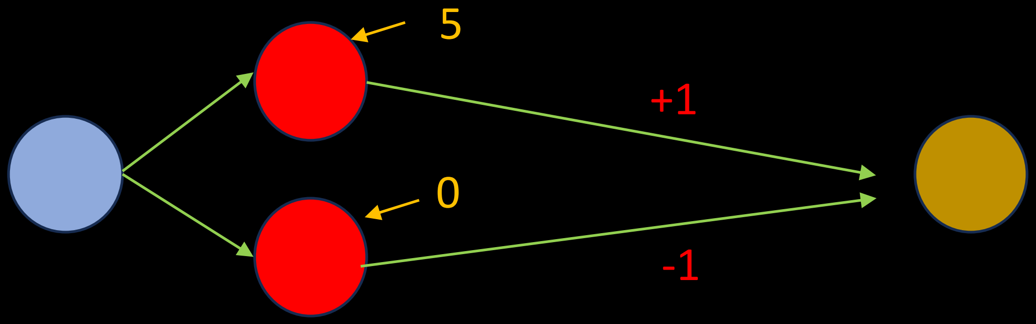
$$f(x) = S(x+5) - S(x)$$

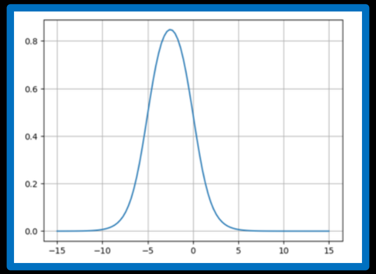


$f(x) = S(x + 2.5) - S(x)$ $g(x) = S(x - 2.5) - S(x - 5)$



$$S(x+5) = \frac{1}{1+e^{-(x+5)}}$$





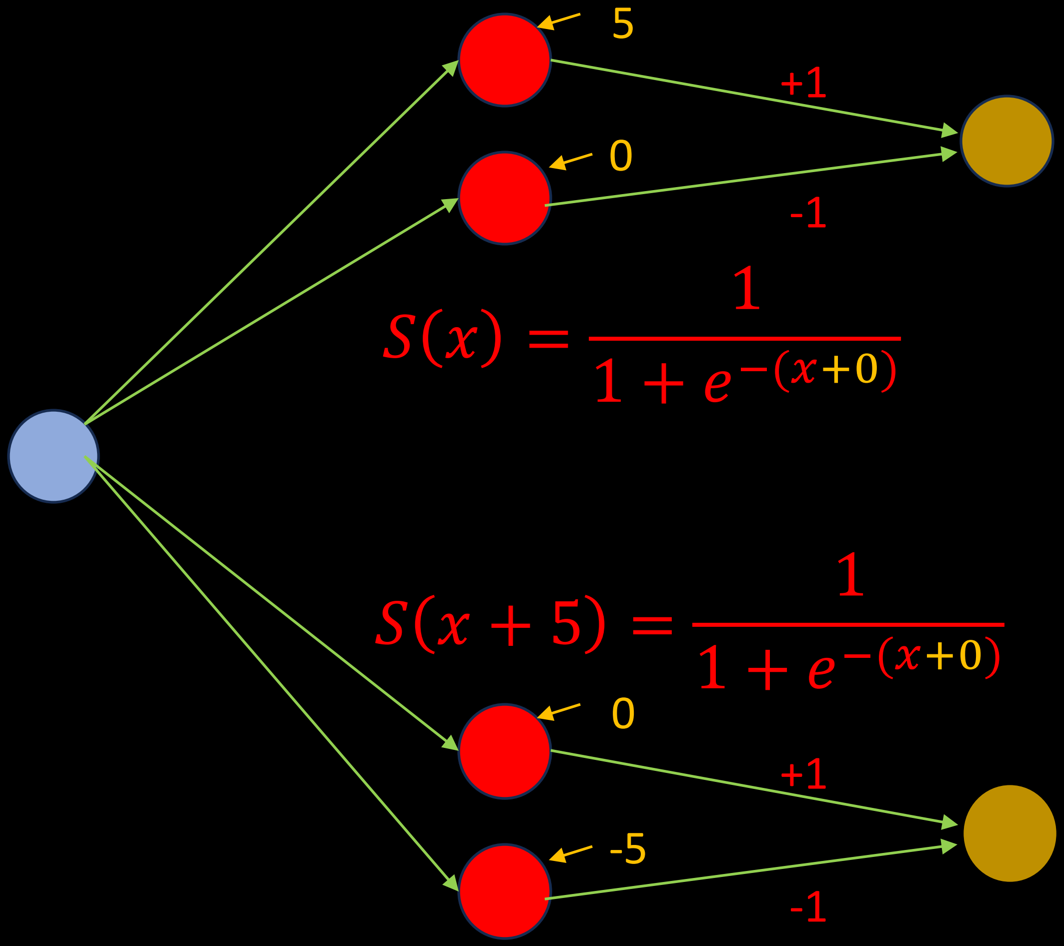
$$S(x) = \frac{1}{1+e^{-(x+0)}}$$

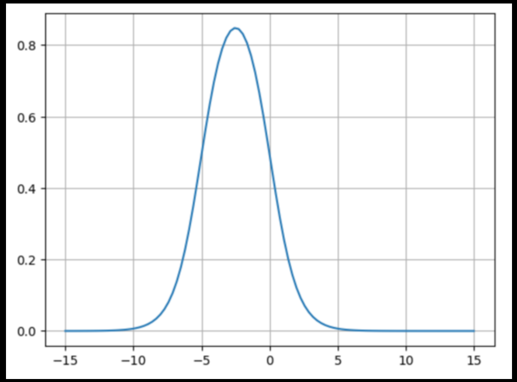
$$S(x+5) = \frac{1}{1+e^{-(x+5)}}$$

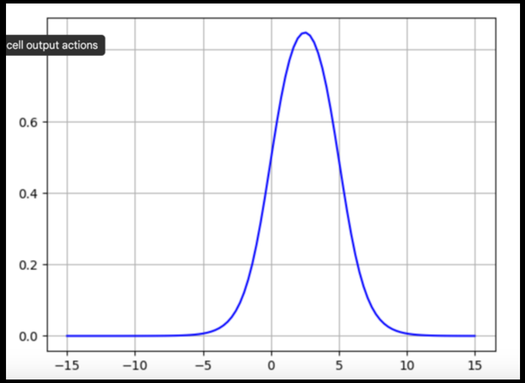
$$S(x) = \frac{1}{1+e^{-(x+0)}}$$

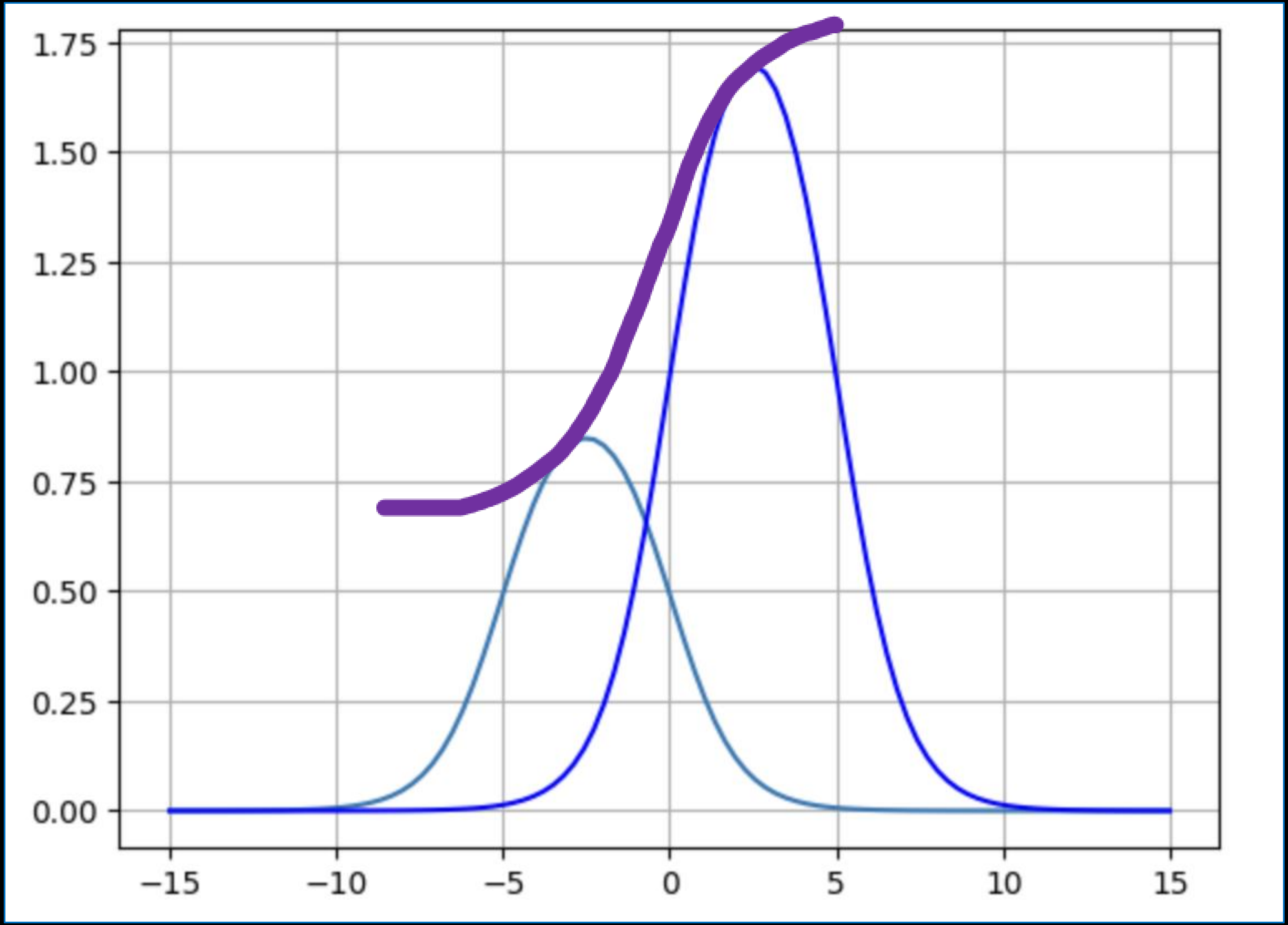
$$S(x+5) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x) = \frac{1}{1+e^{-(x-5)}}$$

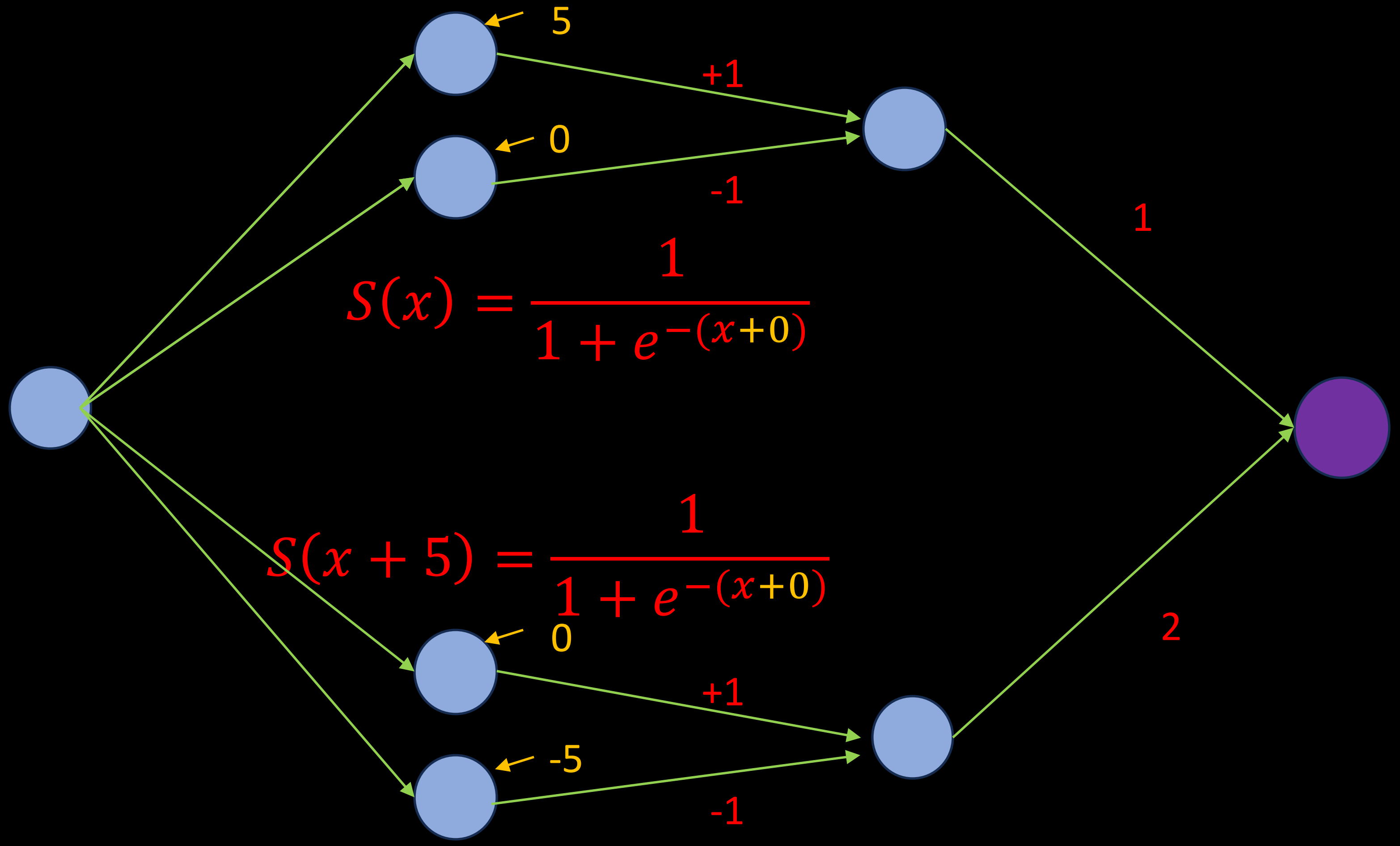







1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−15
−10
−5
0
5
10
15

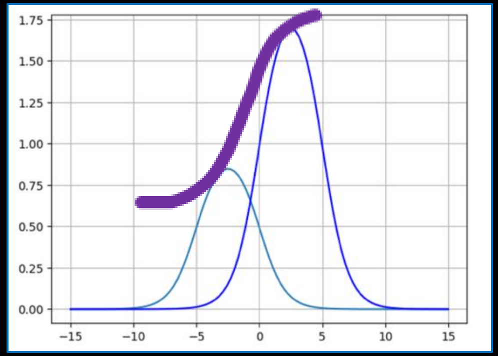
$$S(x+5) = \frac{1}{1+e^{-(x+5)}}$$



$$S(x) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x+5) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x) = \frac{1}{1+e^{-(x-5)}}$$

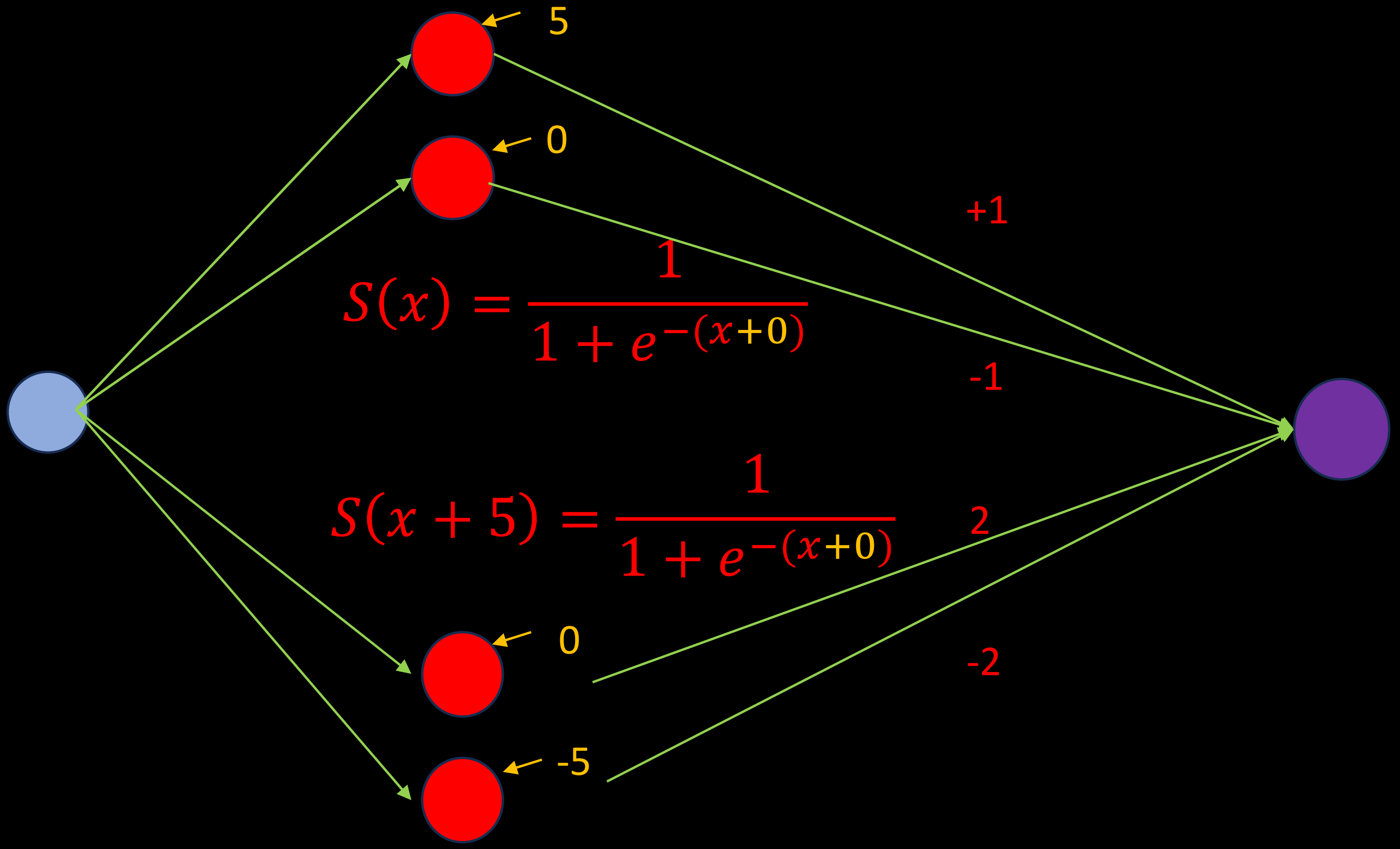


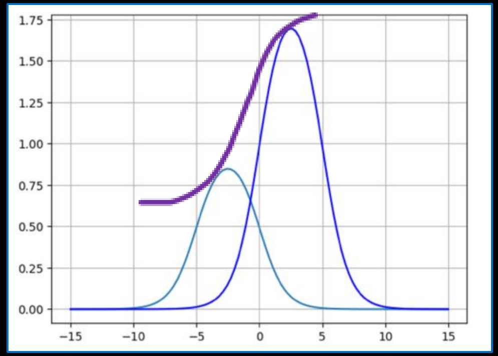
$$S(x+5) = \frac{1}{1+e^{-(x+5)}}$$

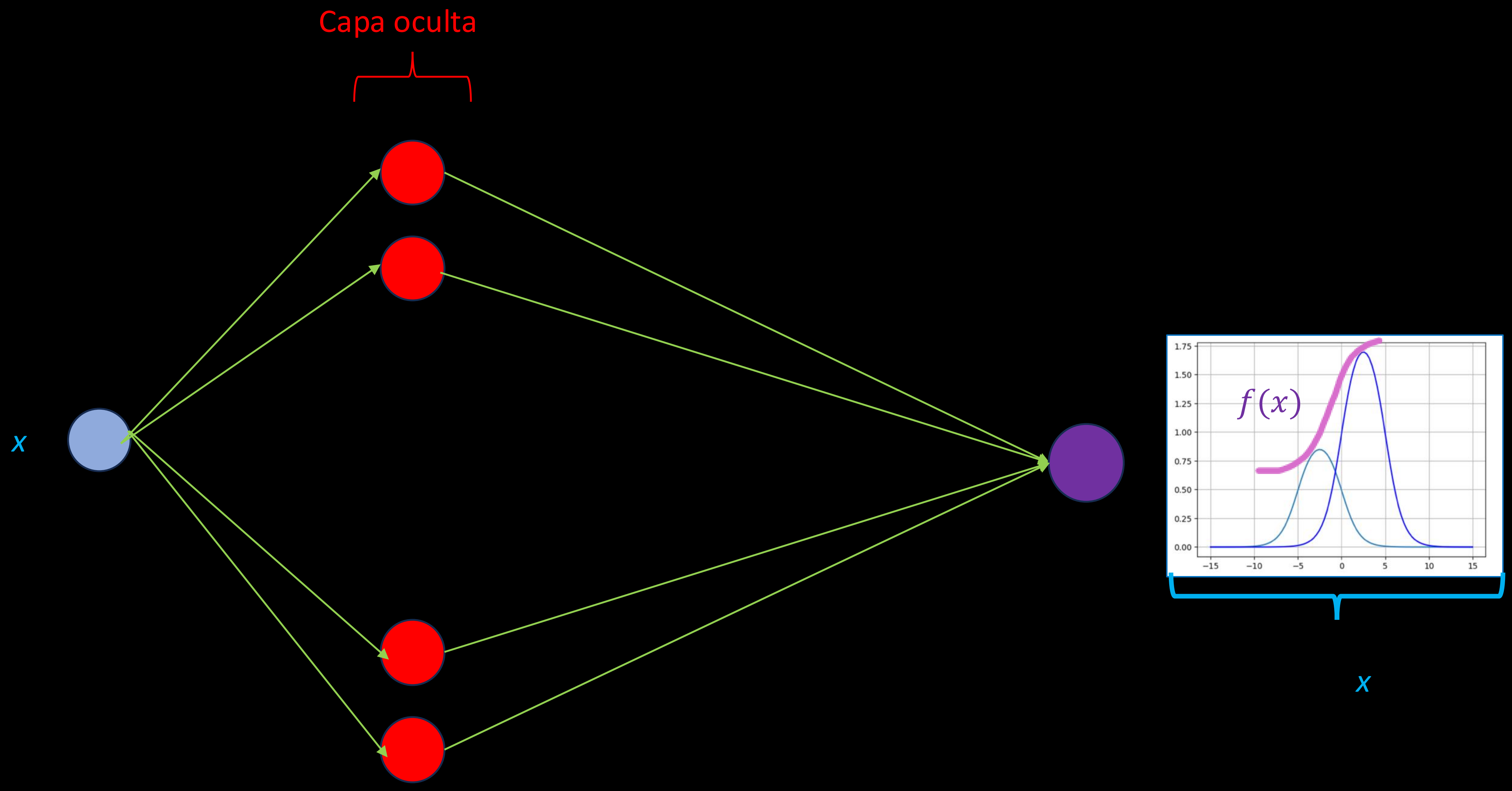
$$S(x) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x+5) = \frac{1}{1+e^{-(x+0)}}$$

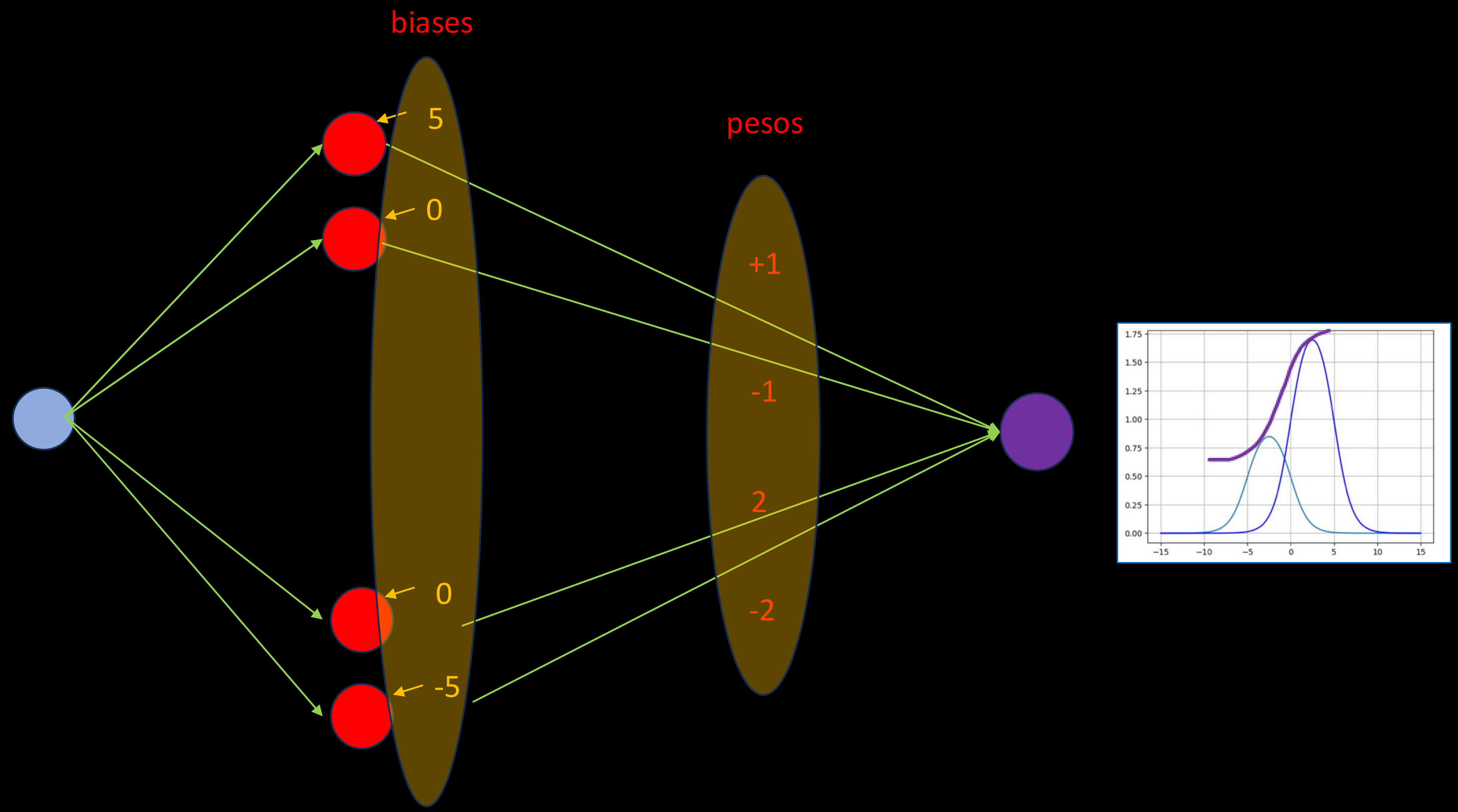
$$S(x) = \frac{1}{1+e^{-(x-5)}}$$

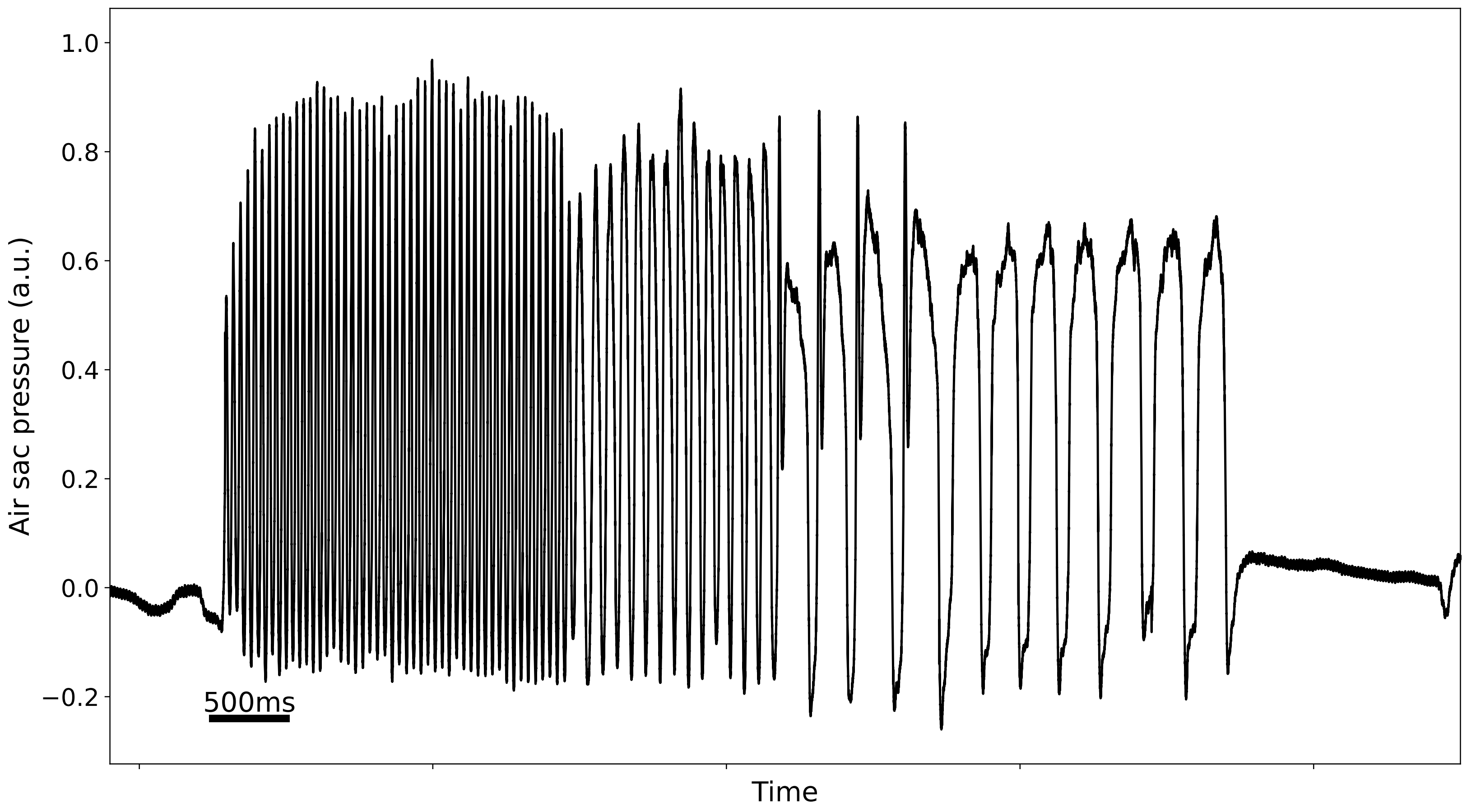


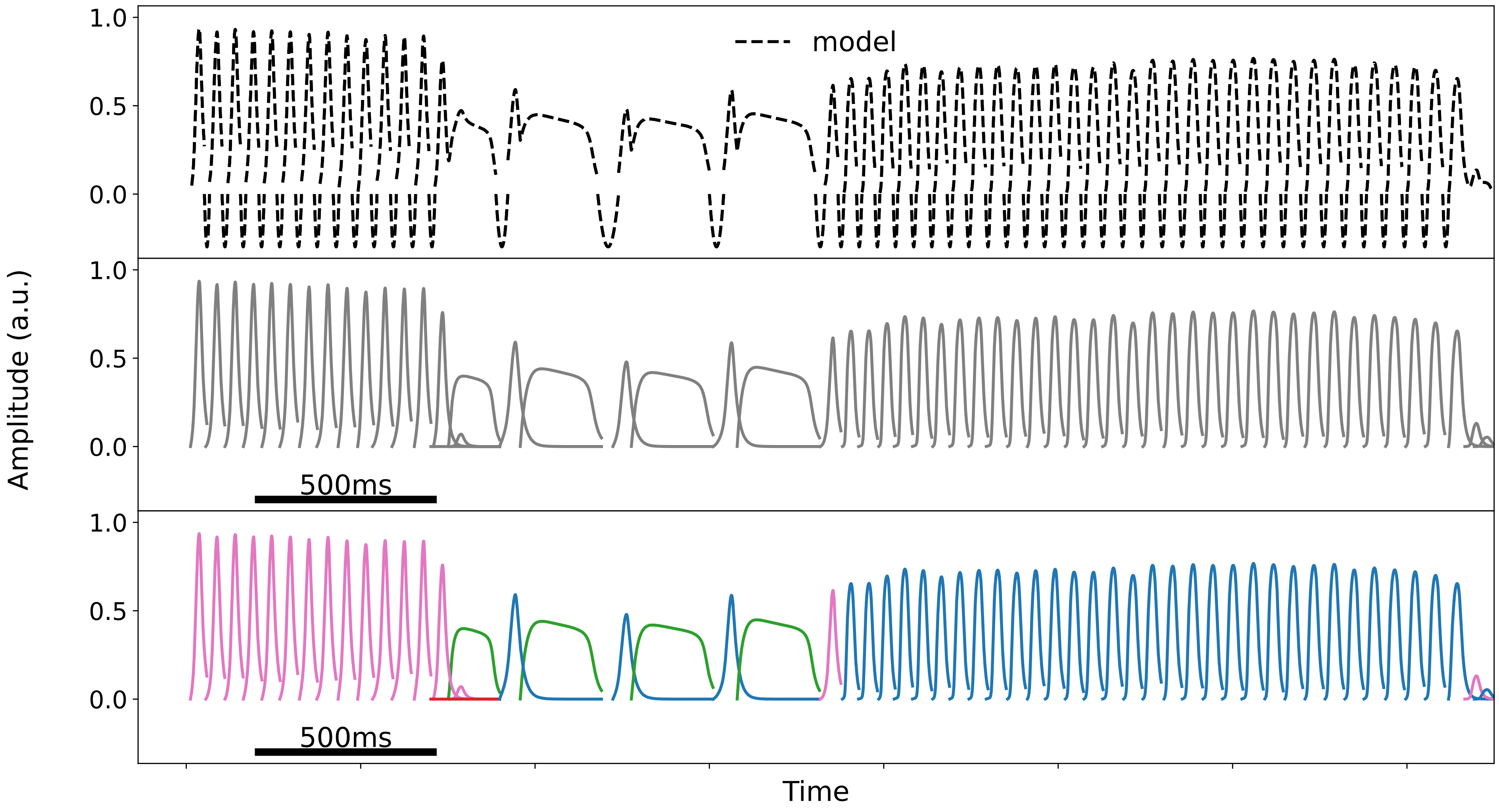



Capa oculta
x
f(x)
x

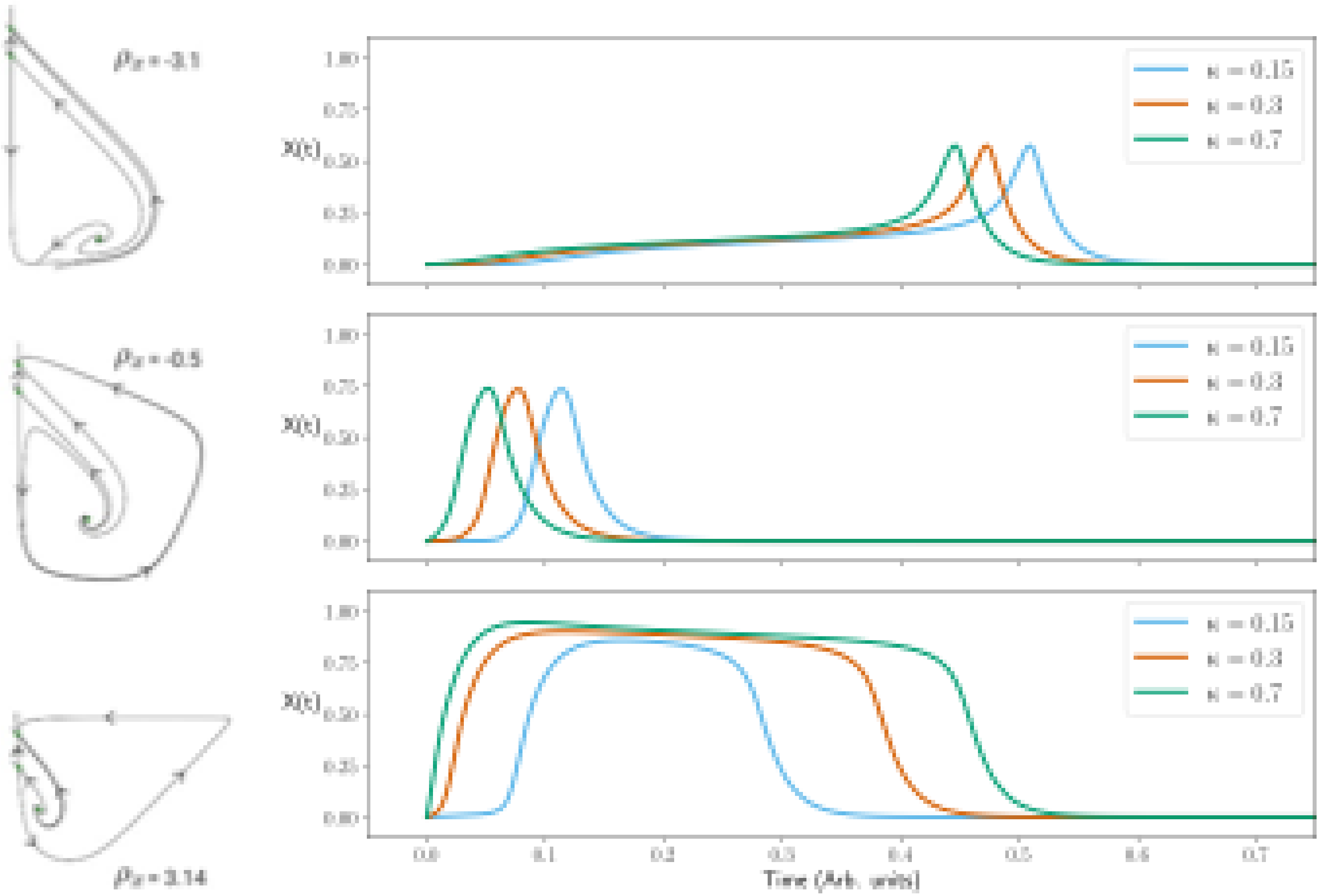
Que se aprende cuando se aprende?


biases
5
0
0
-5
pesos
+1
-1
2
-2
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−15
−10
−5
0
5
10
15


Air sac pressure (a.u.)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
500ms
Time

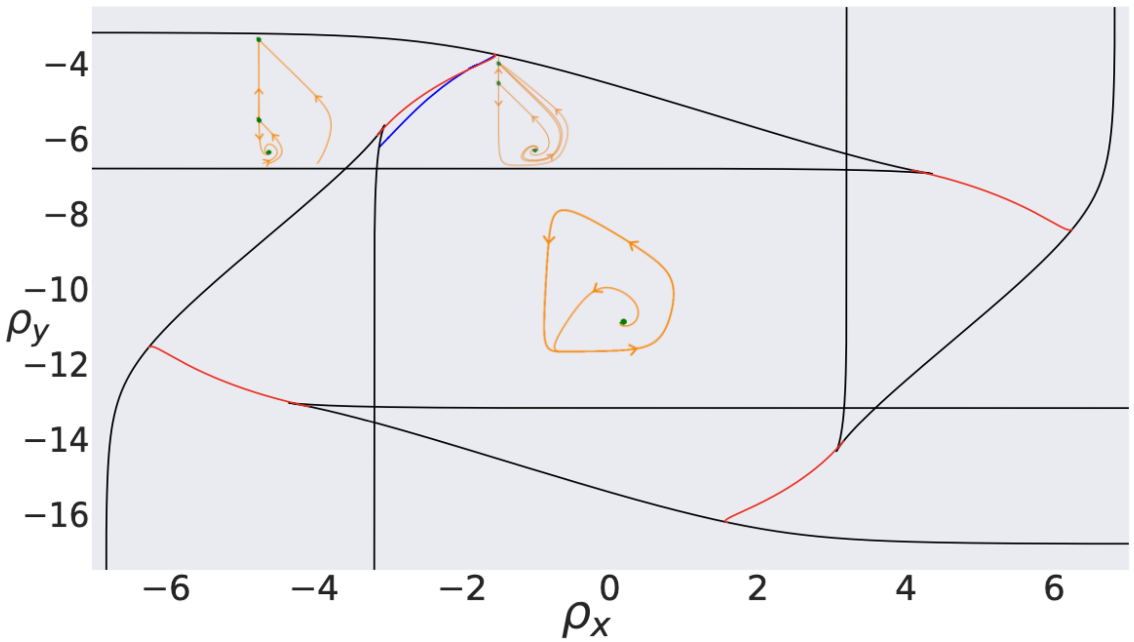


model
1.0
0.5
0.0
Amplitude (a.u.)
500ms
500ms
Time

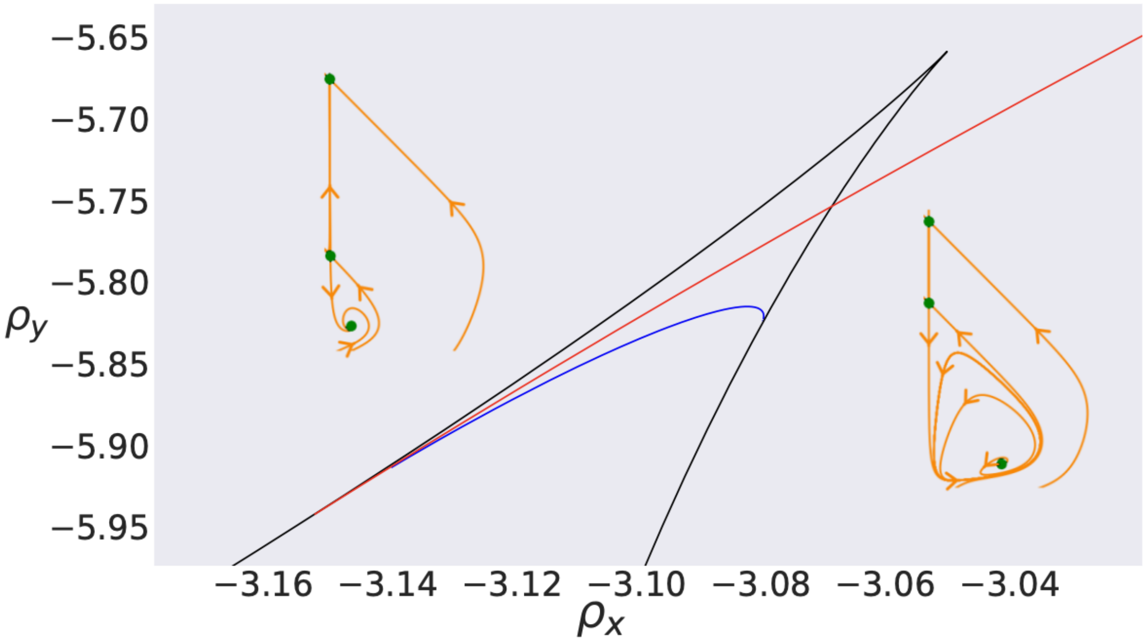


$$\dot{x} = \mu_j \left(-x + S\left(\rho_{x_{ij}} + ax - by \right) \right) \quad 1.1$$

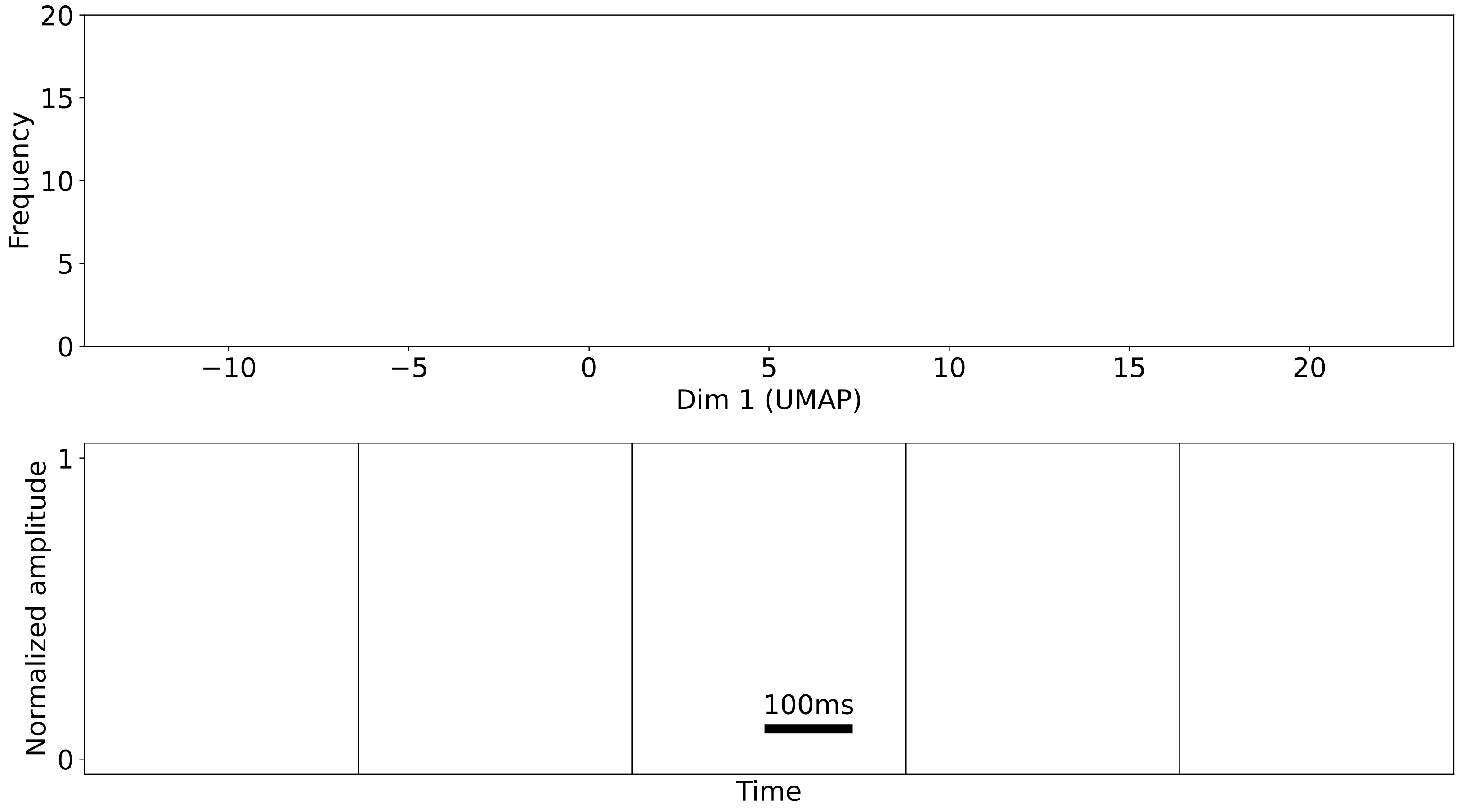
$$\dot{y} = \mu_j \left(-y + S(\rho_y + cx - dy) \right) + \kappa_{ij} \delta(t - T_{ij}), \quad 1.2$$

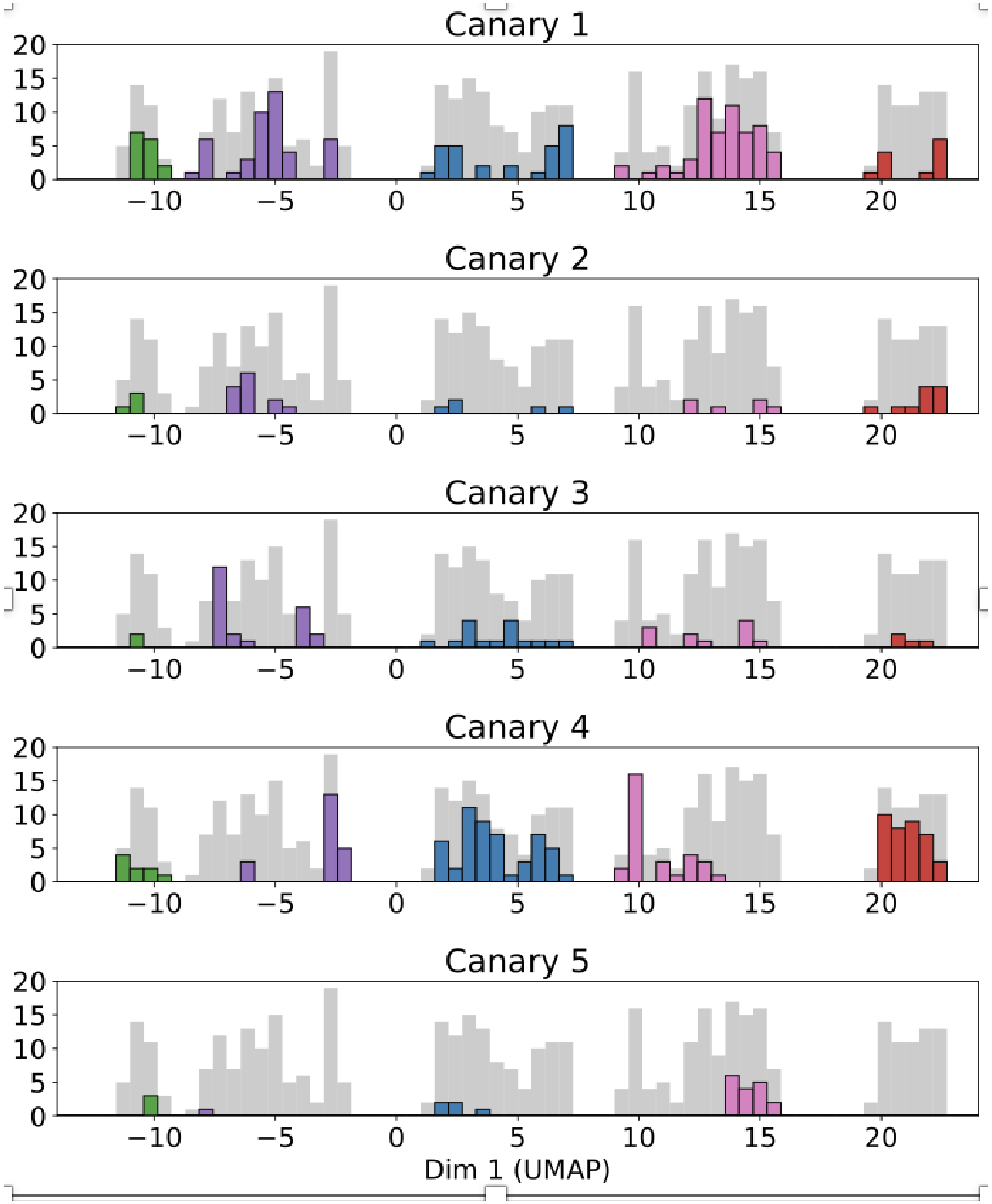


−4
−6
−8
−10
−12
−14
−16
ρ_y
−6
−4
−2
0
2
4
6
ρ_x



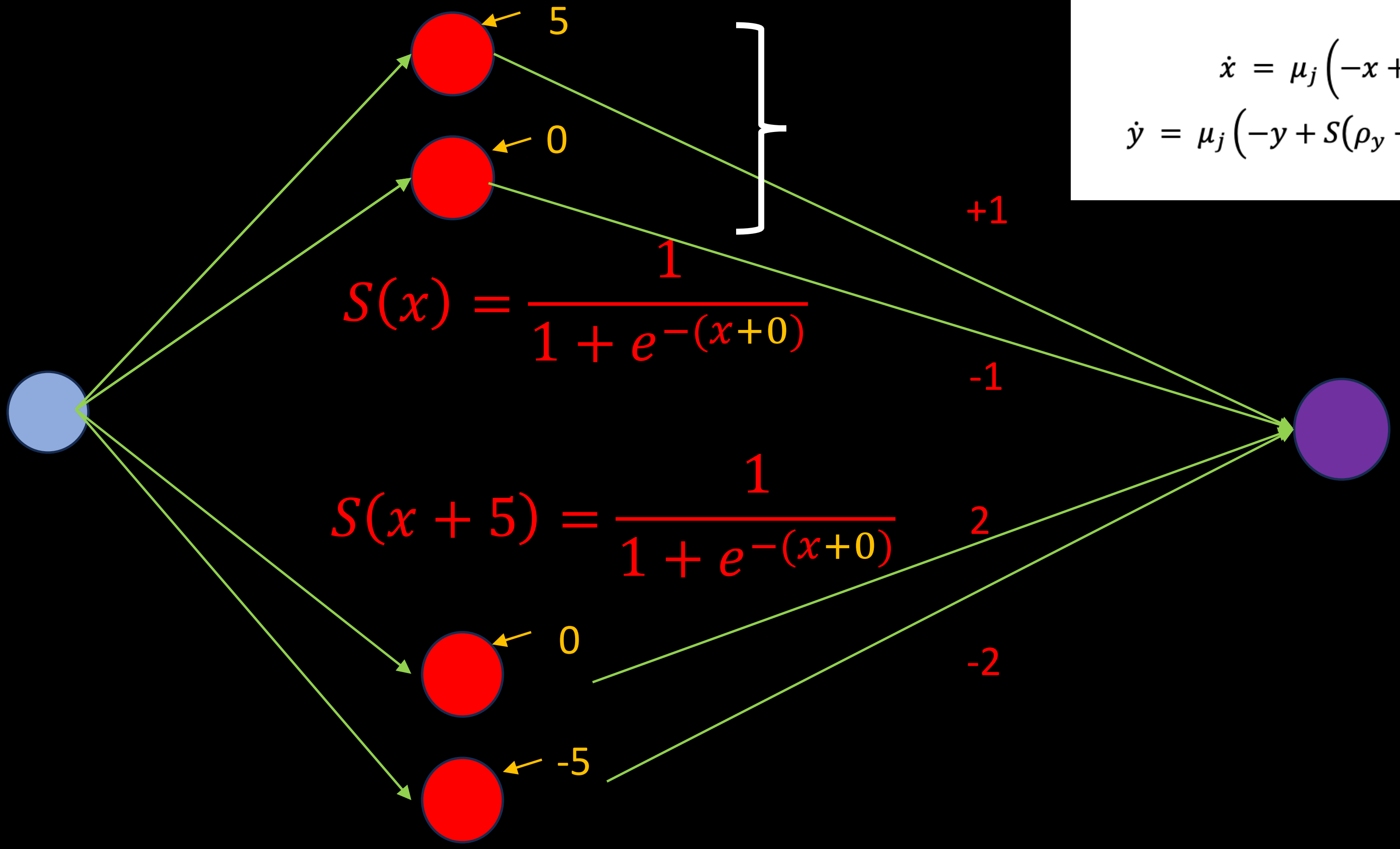
−5.65
−5.70
−5.75
−5.80
−5.85
−5.90
−5.95
ρ_y
−3.16
−3.14
−3.12
−3.10
−3.08
−3.06
−3.04
ρ_x


20
15
10
5
0
Frequency
−10
−5
0
5
10
15
20
Dim 1 (UMAP)
1
0
Normalized amplitude
100ms
Time


Canary 1
Canary 2
Canary 3
Canary 4
Canary 5
20
15
10
5
0
−10
−5
0
5
10
15
20
Dim 1 (UMAP)

Sugestiva similitud

$$S(x+5) = \frac{1}{1+e^{-(x+5)}}$$



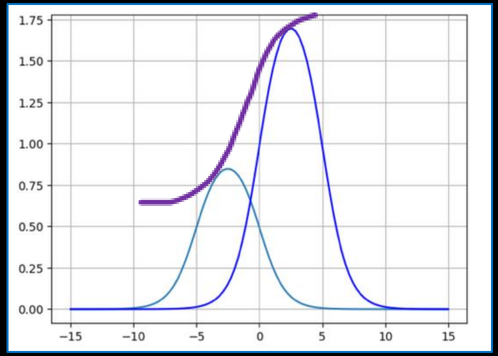
$$S(x) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x+5) = \frac{1}{1+e^{-(x+0)}}$$

$$S(x) = \frac{1}{1+e^{-(x-5)}}$$

$$\dot{x} = \mu_j\left(-x + S\left(\rho_{x_{ij}} + ax - by\right)\right) \quad 1.1$$

$$\dot{y} = \mu_j\left(-y + S(\rho_y + cx - dy)\right) + \kappa_{ij}\delta(t - T_{ij}), 1.2$$



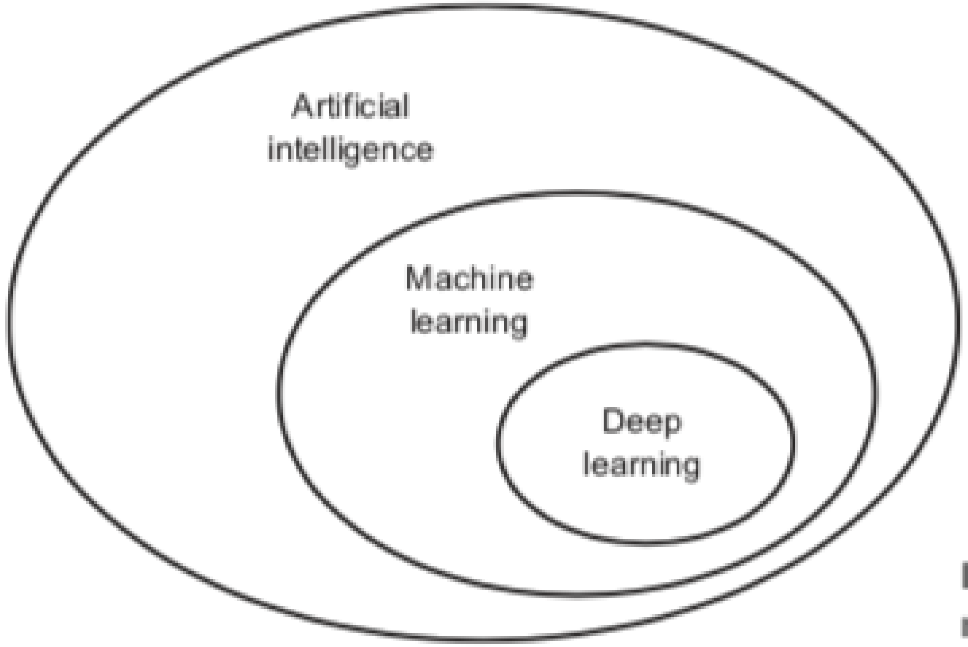
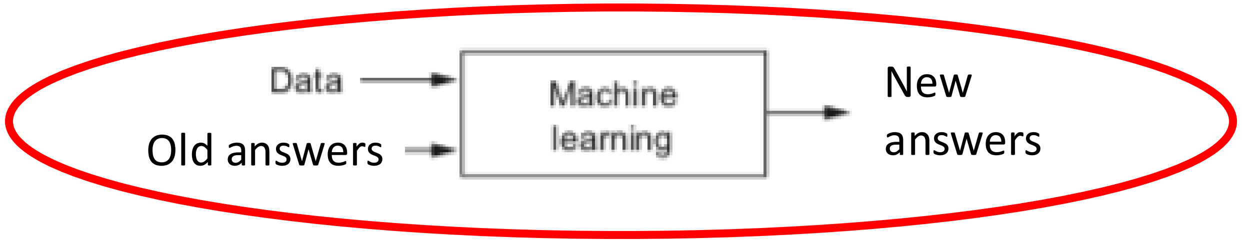


Figure 1.1 Artificial intelligence, machine learning, and deep learning



Hay que construir funciones complicadas y sofisticadas. Como?

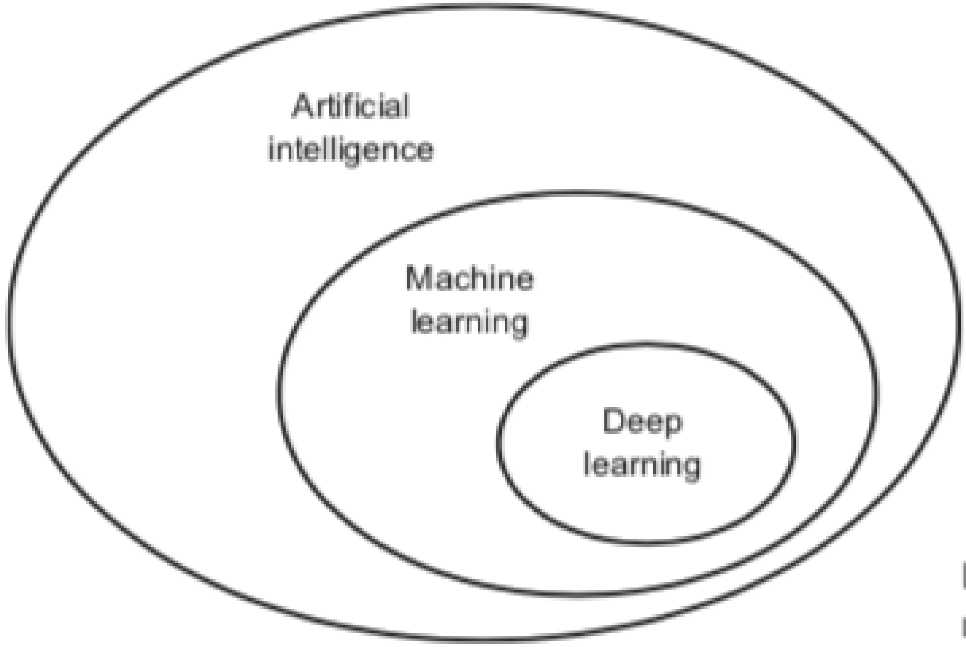
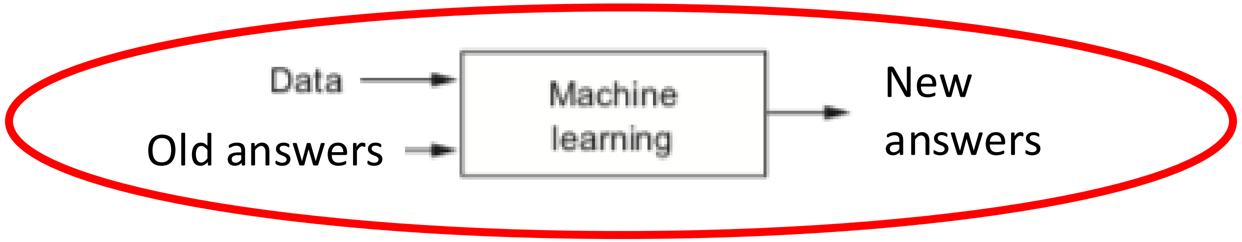


Figure 1.1 Artificial intelligence, machine learning, and deep learning



Como?
Con ejemplos

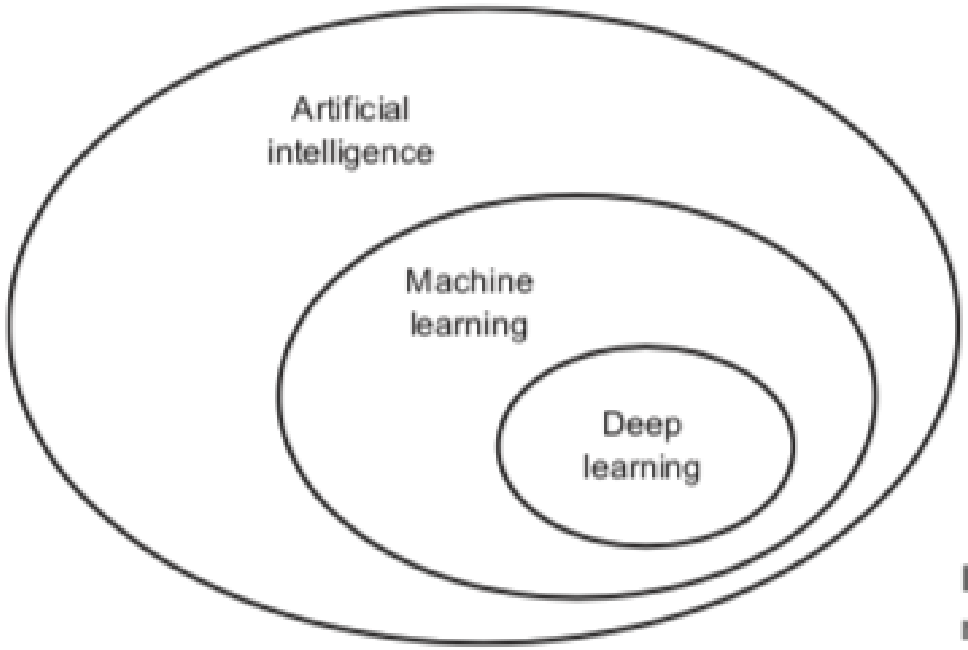
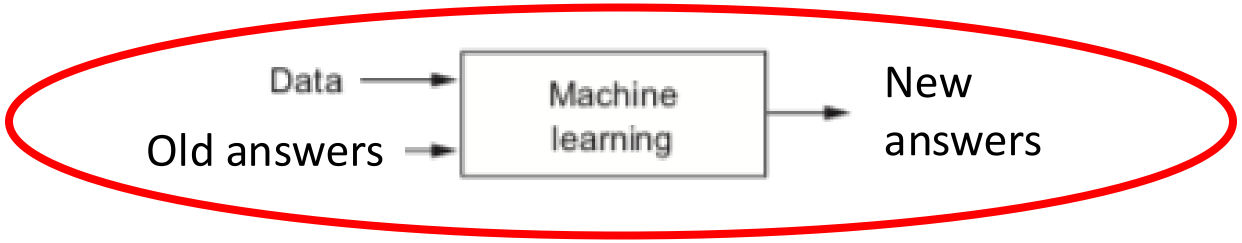


Figure 1.1 Artificial intelligence, machine learning, and deep learning

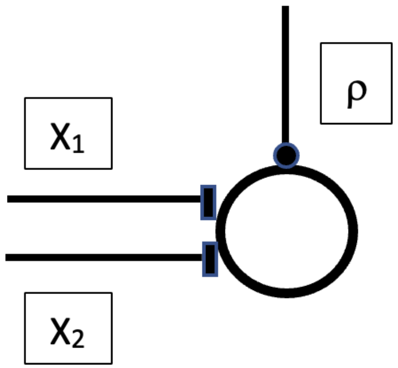


Como?

Y que hacemos con los ejemplos?

Aunque la historia fue distinta…
La idea era reproducir puertas logicas

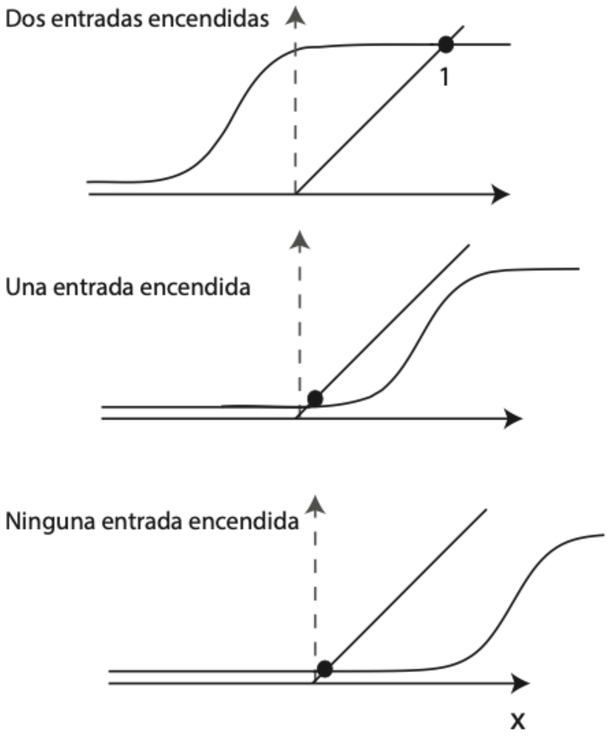
1. **McCulloch y Pitts:** armar puertas lógicas, y mostrar que con neuronas se puede computar.

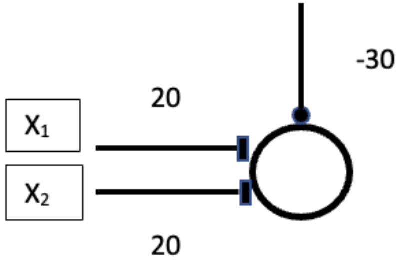


$$\frac{dx}{dt} = -x + S\left(\rho + \sum_{1}^{n} c_j x_j\right)$$

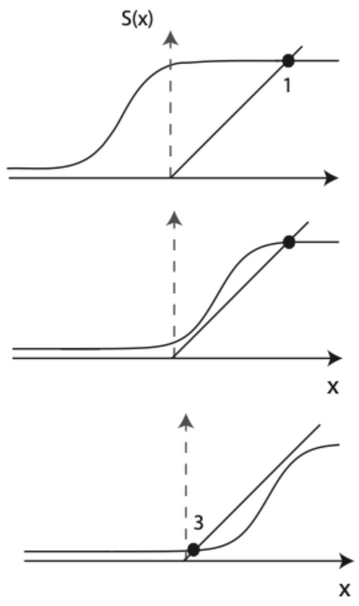
$$output = S\left(\rho + \sum_{1}^{n} c_j x_j\right).$$

Se puede, por ejemplo, armar una puerta lógica "**AND**", que dará un estado "encendido" cuando ambas entradas estén encendidas, si la entrada ρ es lo suficientemente negativa.

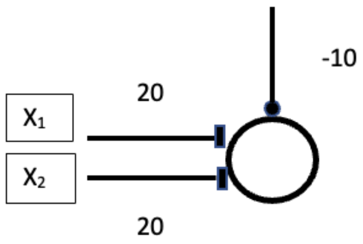




Si quisiéramos, en cambio, realizar una puerta "**OR inclusiva**", podríamos arreglar que el umbral fuera mas pequeño, de modo que con una sola entrada encendida, ya tuviésemos una salida encendida:

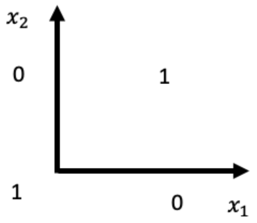


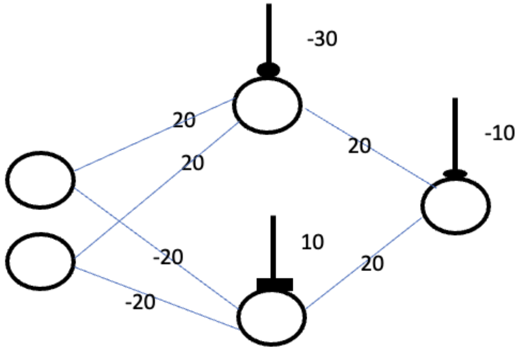
Es la figura vemos como basta con una entrada positiva para que ya la salida sea positiva.



<u>Finalmente, una bastante sutil: x1 XNOR x2.</u>
Esta red da 1 si ambas entradas son uno, o ambas son cero.

Si lo pensamos como un mecanismo de regresión logística, es bastante sutil:





x_1	x_2	a_1 (AND)	a_2 (Nx1 and Nx2)	Output (x1 or x2)
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Este ejemplo, además de introducir una puerta lógica, pone en evidencia como con mas capas, se logran funciones no triviales de las entradas.

2. **Rosenblatt:** el perceptron.

Con la misma matemática, pero una concepción radicalmente diferente (y mucho mas cercana al actual proceso de Deep learning), Rosenblatt propuso el perceptron: una "red" de un elemento con una capa de entradas x_i, que apropiadamente sumadas, entran a una unidad no lineal (output tipo sigmoidea):

$$out(t) = S(\sum_{i=1}^{n} W_i \, x_i)$$

Lo interesante es que no diseñamos, como en el caso de las puertas lógicas, los pesos para tener una salida dada, **si no que sometemos a la red a un aprendizaje**, cuya regla es:

$$\delta w_i = \alpha \, \delta x_i$$

$$\delta = salida - maestro$$

(0 si es correcto, y +1 o -1 si no lo es)