Redes neuronales

Data Mining

Ester Vidaña Vila

laSalle ENG
Universitat Ramon Llull

Redes Neuronales

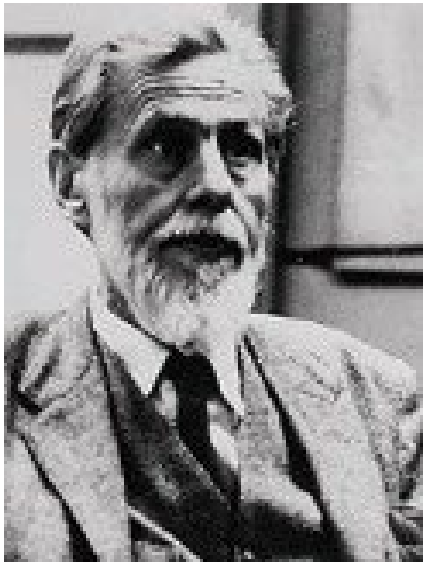
Las redes neuronales como modelo computacional existen desde mediados del siglo pasado, pero no ha sido hasta hace unos pocos años que estamos haciendo un gran uso de ellas.

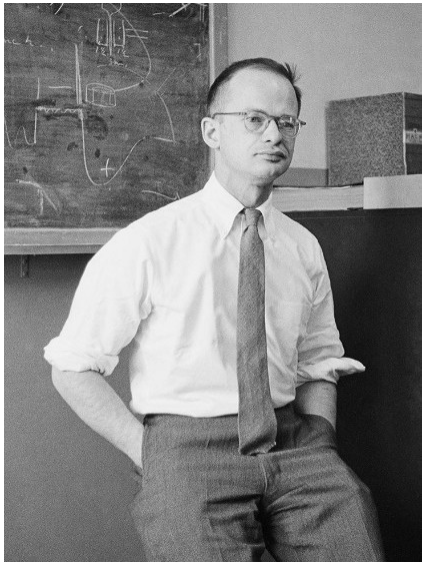
Historia

En los años 40, el neurofisiólogo Warren McCulloch acogió en su casa a Walter Pitts, un genio autodidacta que aprendió lógica y matemática por su cuenta, además de varios idiomas.

La imaginación de McCulloch y sus conocimientos de fisiología, medicina y psiquiatría se fundieron con la cultura lógico-matemática de Pitts.

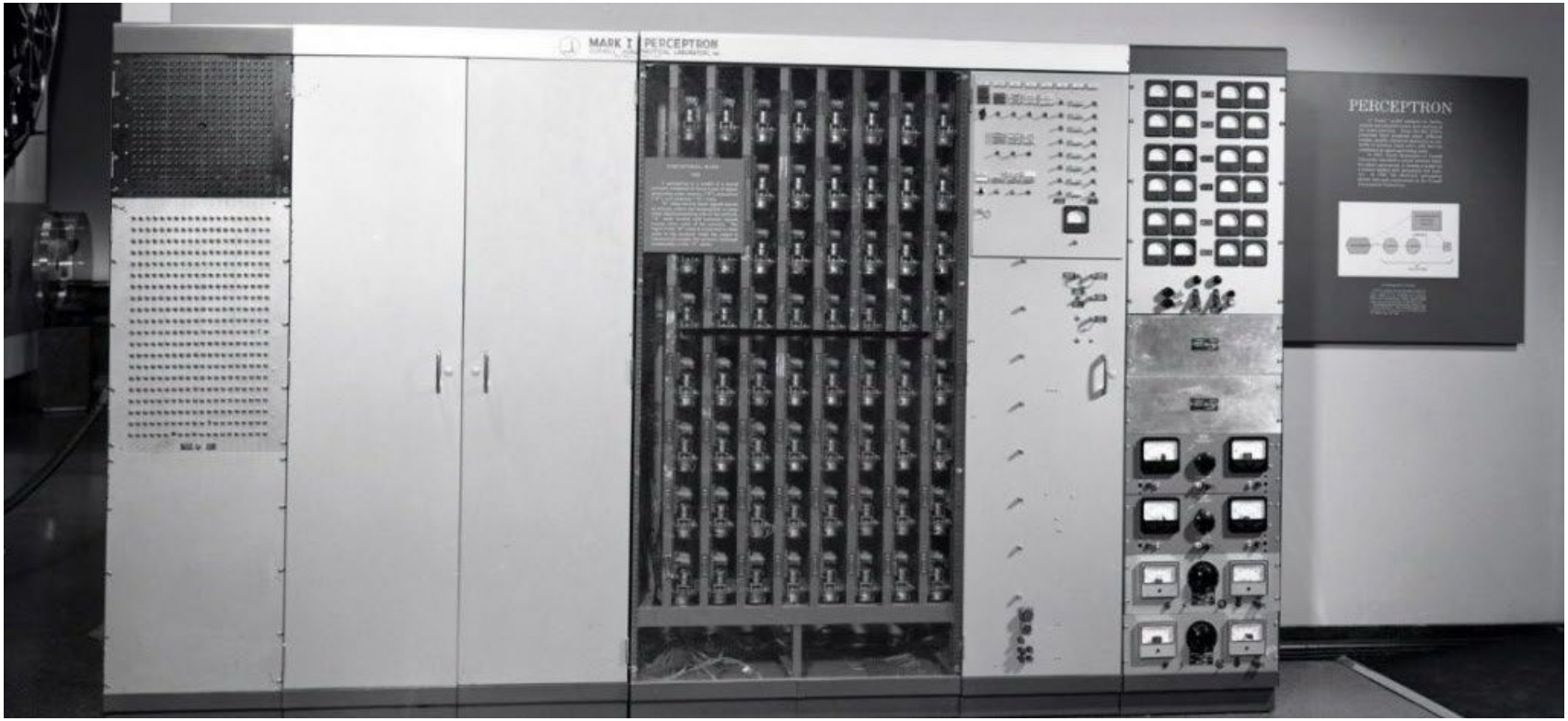
Juntos desarrollaron la primera explicación lógico-matemática del cerebro en 1943.





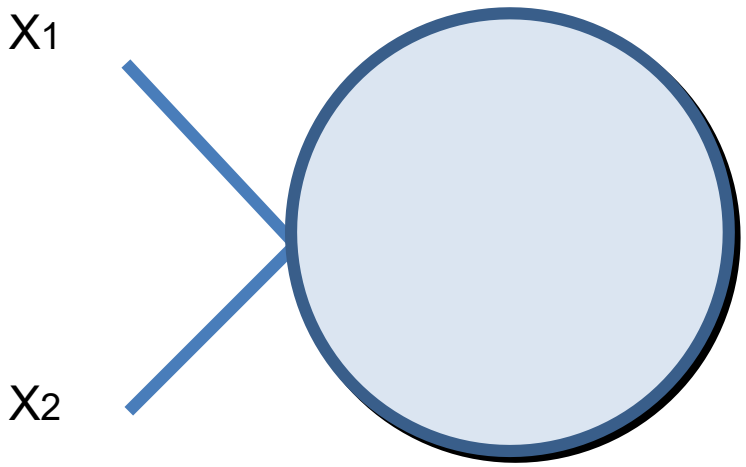
Historia

Frank Rosenblatt en 1957, desarrolló en el *Aeronautical Laboratory* de la Cornell University el "**Perceptron**" (Mark 1), dispositivo electrónico construido basándose en **principios biológicos** y que era capaz de **aprender** a partir de datos.



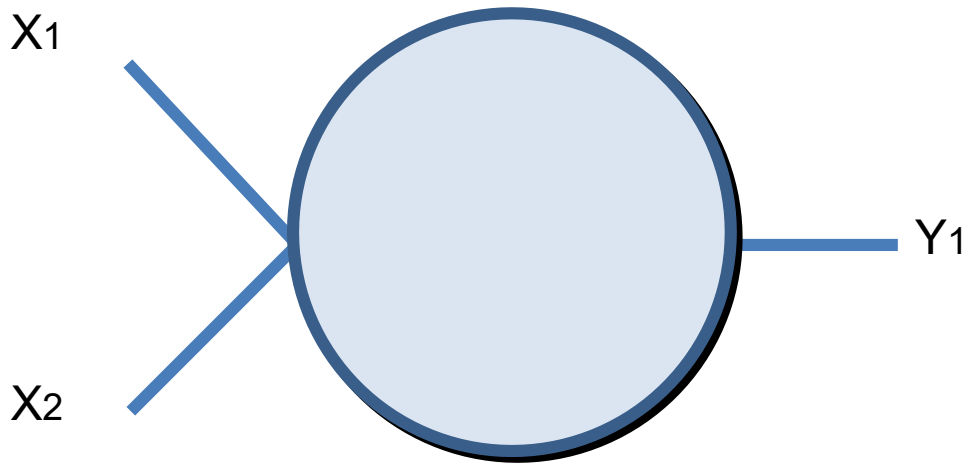
Neurona

Las neuronas tienen conexiones de entrada, por donde reciben los valores de entrada.



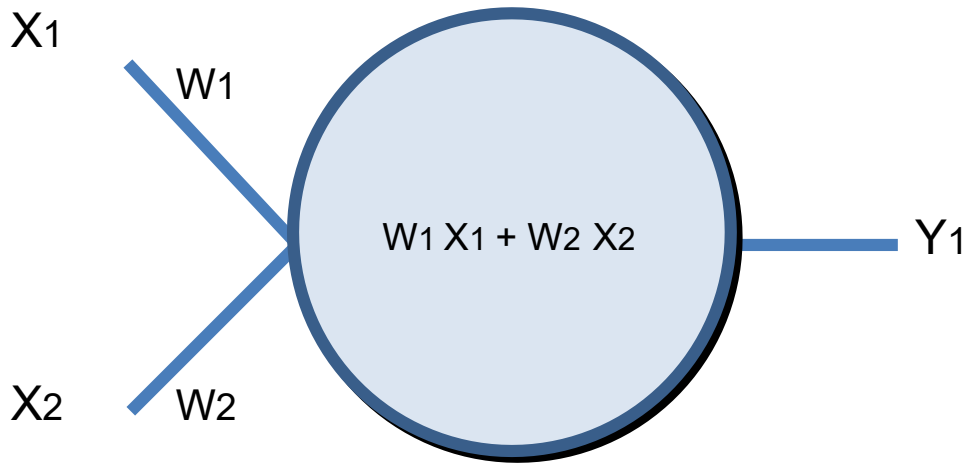
Neurona

Una vez recibidos los valores de entrada, la neurona realizará un cálculo interno y nos dará un valor de salida.



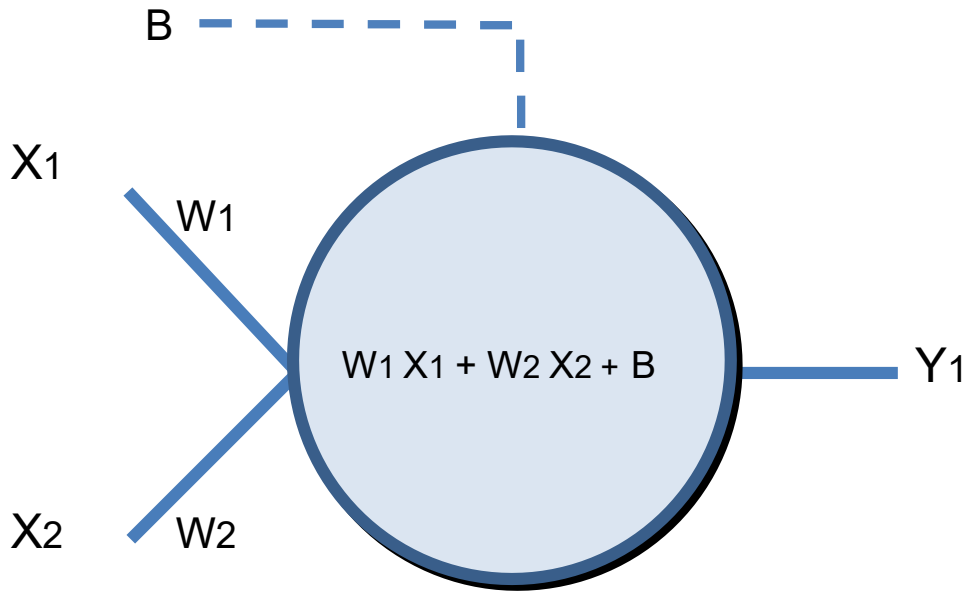
Neurona

El cálculo que realiza la neurona es una suma ponderada:



Neurona

La función de la neurona, también tiene un término independiente llamado **Bias** (sesgo).



$$W_1 X_1 + W_2 X_2 + B = Y_1$$

Neurona: ejemplo

“Si el día está soleado y es fin de semana, vamos a la playa. En cualquier otro caso no vamos a la playa."

Lo que estamos esperando es una salida binaria:

Soleado	Fin de semana	Ir a la playa
NO	NO	0
SI	NO	0
NO	SI	0
SI	SI	1

Neurona: ejemplo

El inconveniente que encontraremos es que actualmente nuestra neurona funciona como un modelo de regresión lineal.

Por lo tanto, nos da como resultado un número continuo en vez de uno binario.

Soleado	Fin de semana	Ir a la playa
0	0	0
1	0	0
0	1	0
1	1	1

Neurona: ejemplo

Para binarizar los datos tendremos que definir un umbral.

Aquí es donde nuestro valor de **Bias** nos es de utilidad:

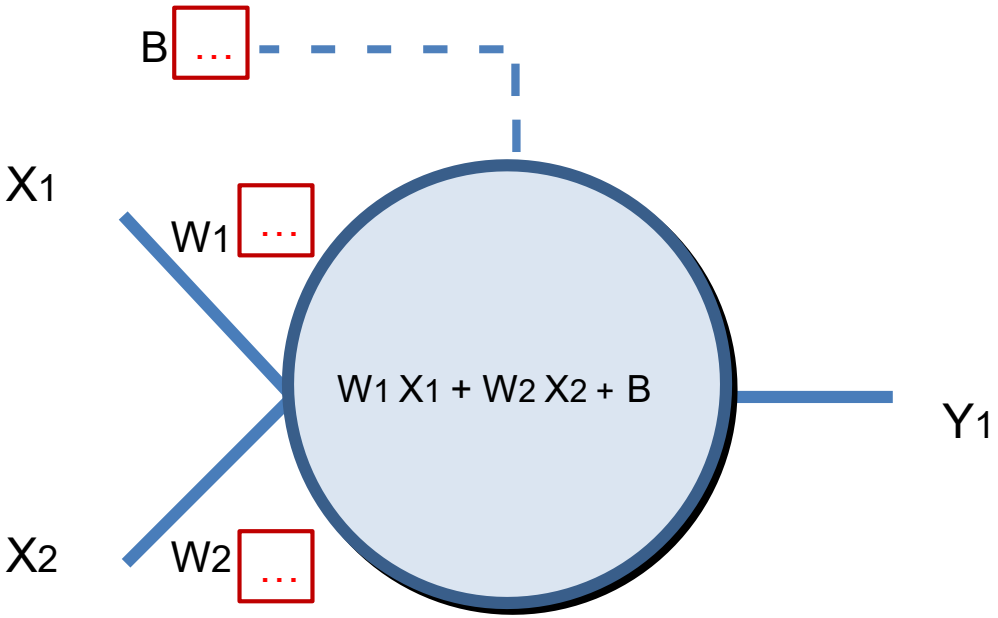
Podemos emplear el **Bias como el negativo del umbral**.

$$\text{Umbral} = - \text{Bias}$$

$$W_1 X_1 + W_2 X_2 + B \leq (\text{Umbral}) \rightarrow \text{Representa } Y = 0$$

$$W_1 X_1 + W_2 X_2 + B > (\text{Umbral}) \rightarrow \text{Representa } Y = 1$$

Neurona: ejemplo



Soleado	Fin de semana	Ir a la playa
NO	NO	…
SI	NO	…
NO	SI	…
SI	SI	…

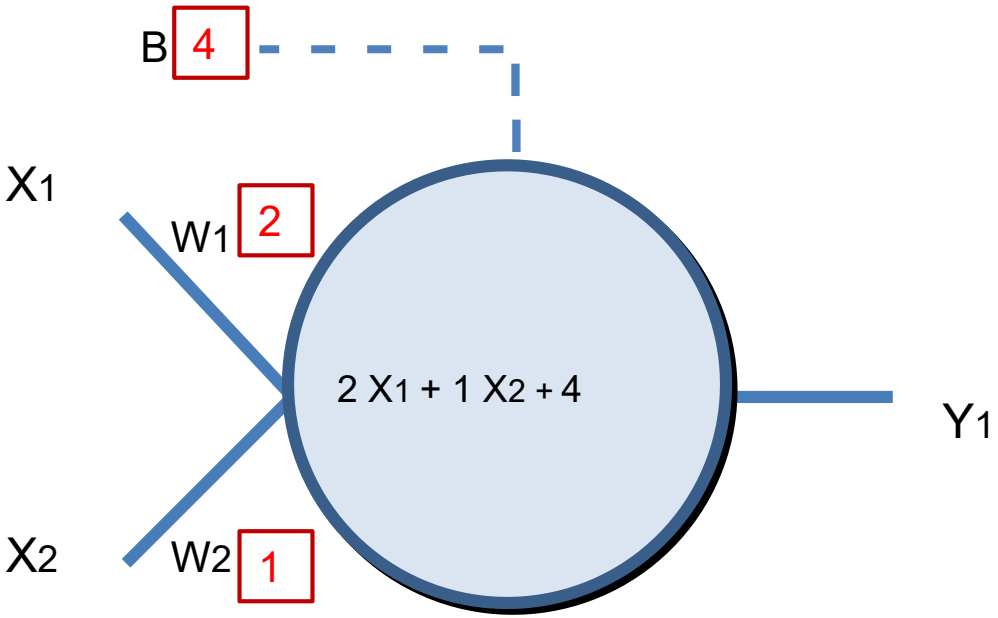
Soleado = X_1

Fin de semana = X_2

Si el resultado es ≤ (Umbral) → Representa un 0

Si el resultado es > (Umbral) → Representa un 1

Neurona: ejemplo

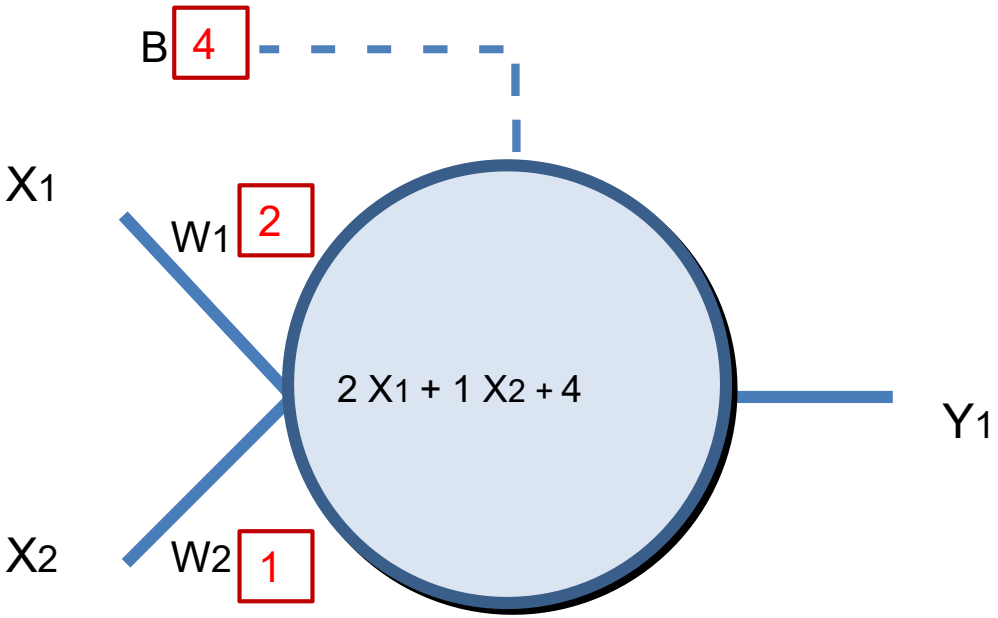


Soleado	Fin de semana	Ir a la playa
NO	NO	
SI	NO	
NO	SI	
SI	SI	

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo



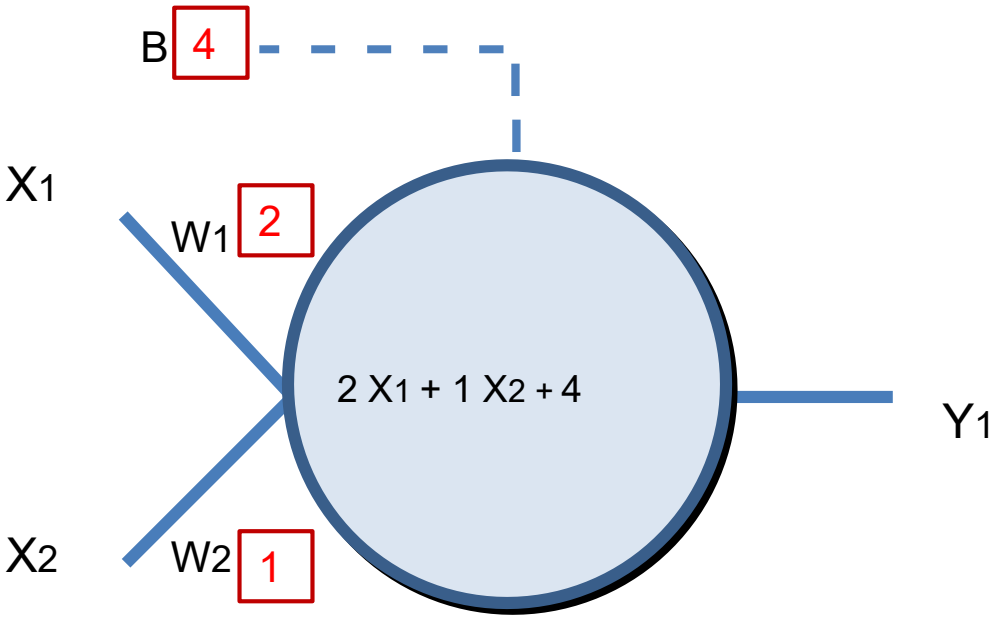
Soleado	Fin de semana	Ir a la playa
0	0	4
1	0	
0	1	
1	1	

(2 x 0) + (1 x 0) + (4) = 4

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo



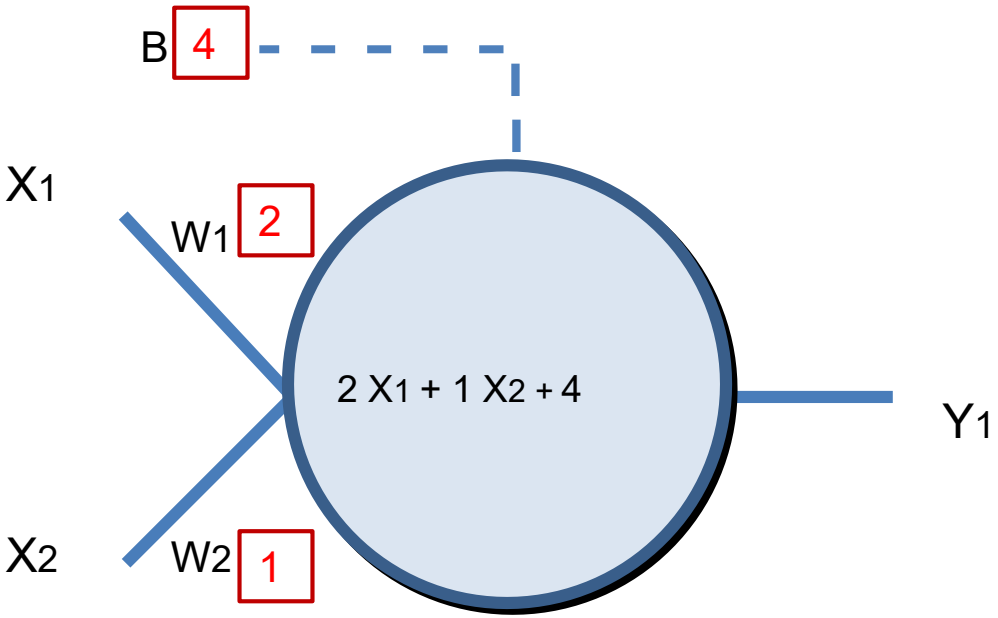
Soleado	Fin de semana	Ir a la playa
0	0	4
1	0	6
0	1	
1	1	

(2 x 1) + (1 x 0) + (4) = 6

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo



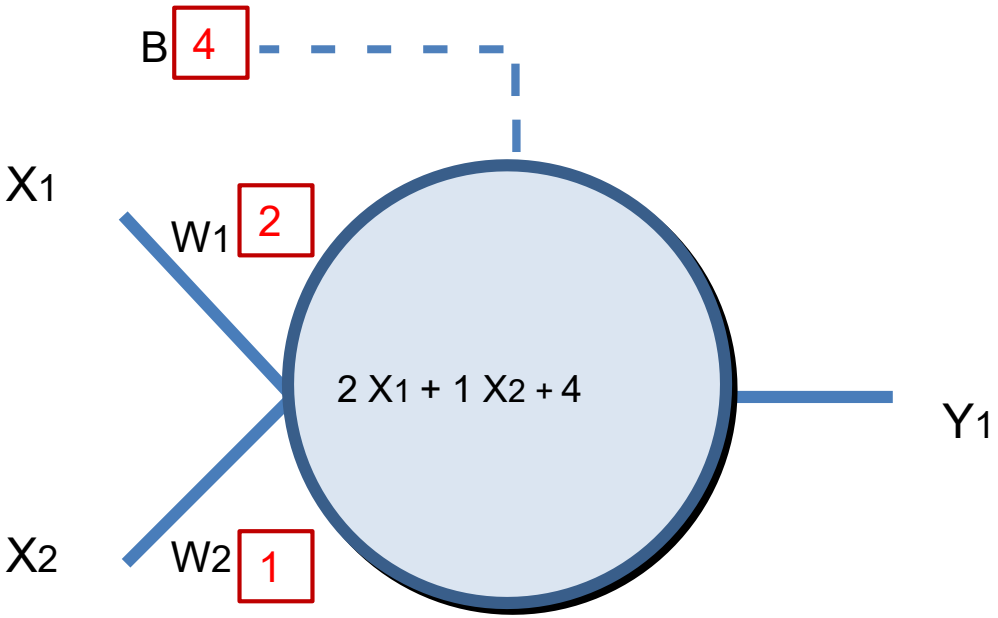
Soleado	Fin de semana	Ir a la playa
0	0	4
1	0	6
0	1	5
1	1	

(2 x 0) + (1 x 1) + (4) = 5

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo



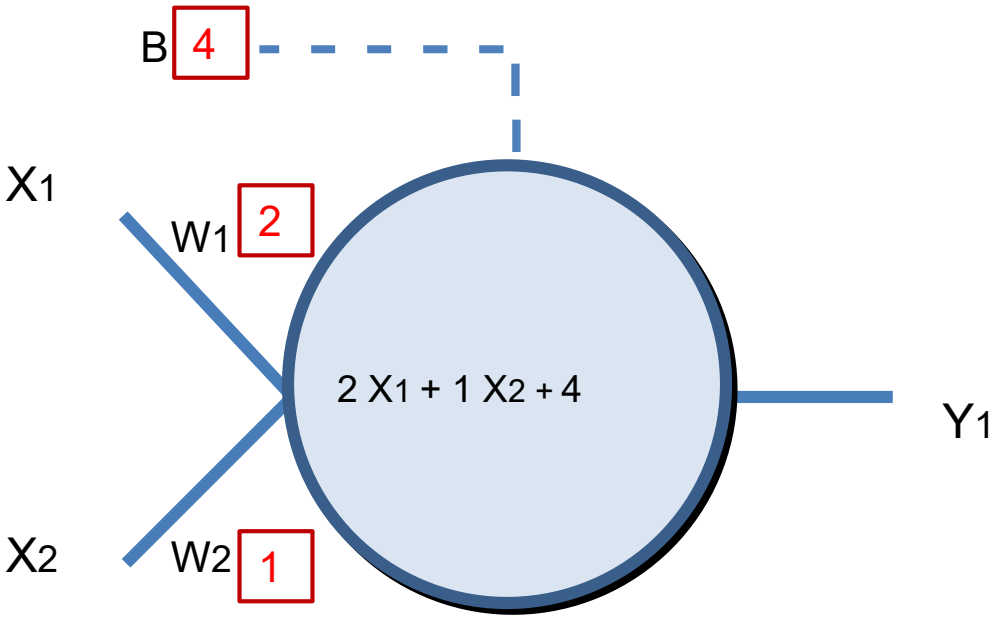
Soleado	Fin de semana	Ir a la playa
0	0	4
1	0	6
0	1	5
1	1	7

(2 x 1) + (1 x 1) + (4) = 7

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo



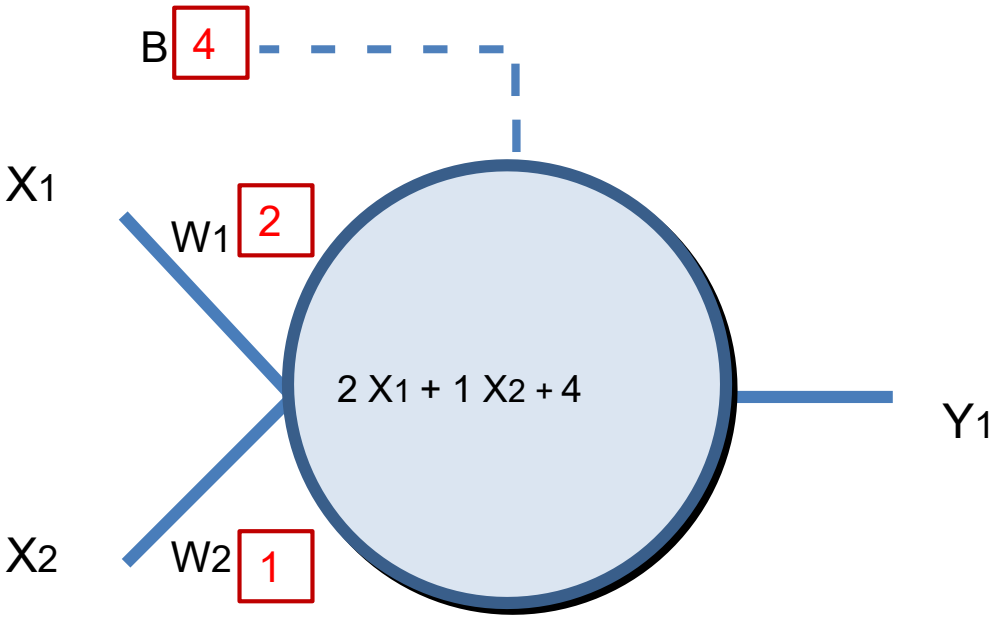
Soleado	Fin de semana	Ir a la playa
0	0	1
1	0	1
0	1	1
1	1	1

Ahora aplicamos el umbral

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo

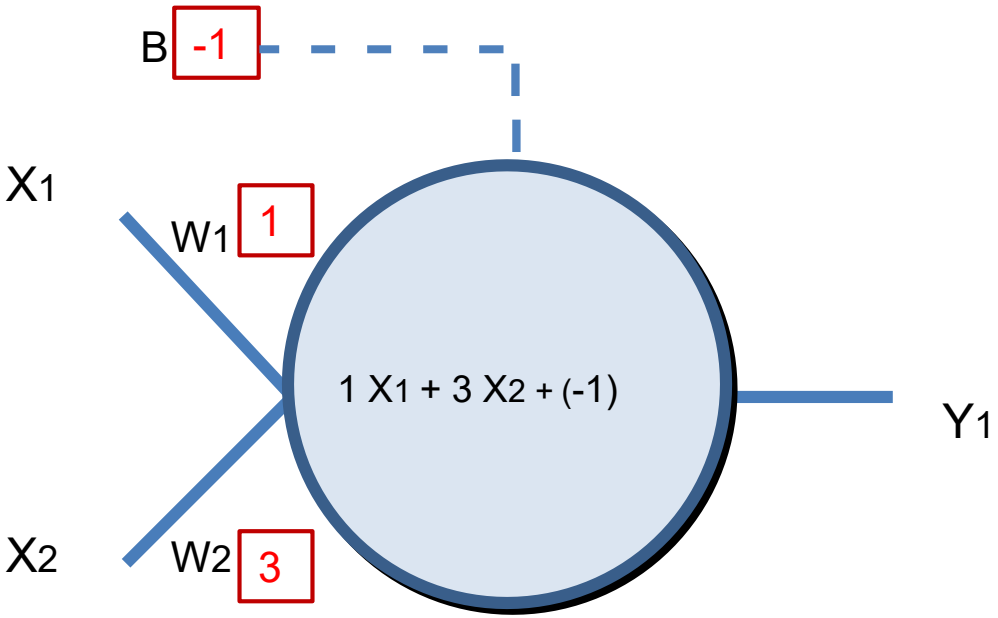


Soleado	Fin de semana	Ir a la playa
0	0	1
1	0	1
0	1	1
1	1	1

Si el resultado es ≤ - 4 → Representa un 0

Si el resultado es > - 4 → Representa un 1

Neurona: ejemplo

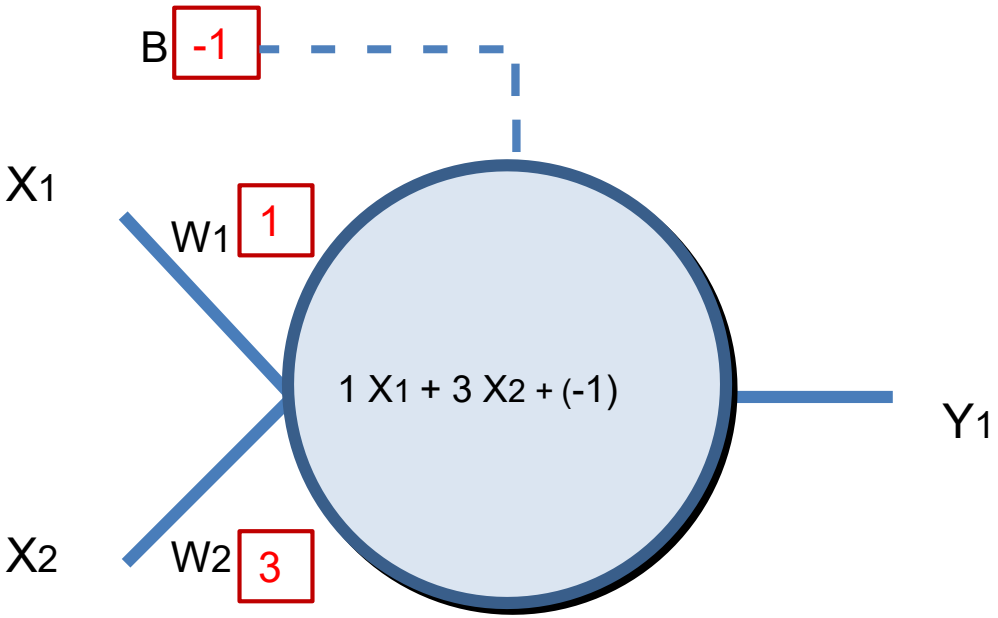


Soleado	Fin de semana	Ir a la playa
NO	NO	-1
SI	NO	0
NO	SI	8
SI	SI	9

Si el resultado es ≤ 1 → Representa un 0

Si el resultado es > 1 → Representa un 1

Neurona: ejemplo

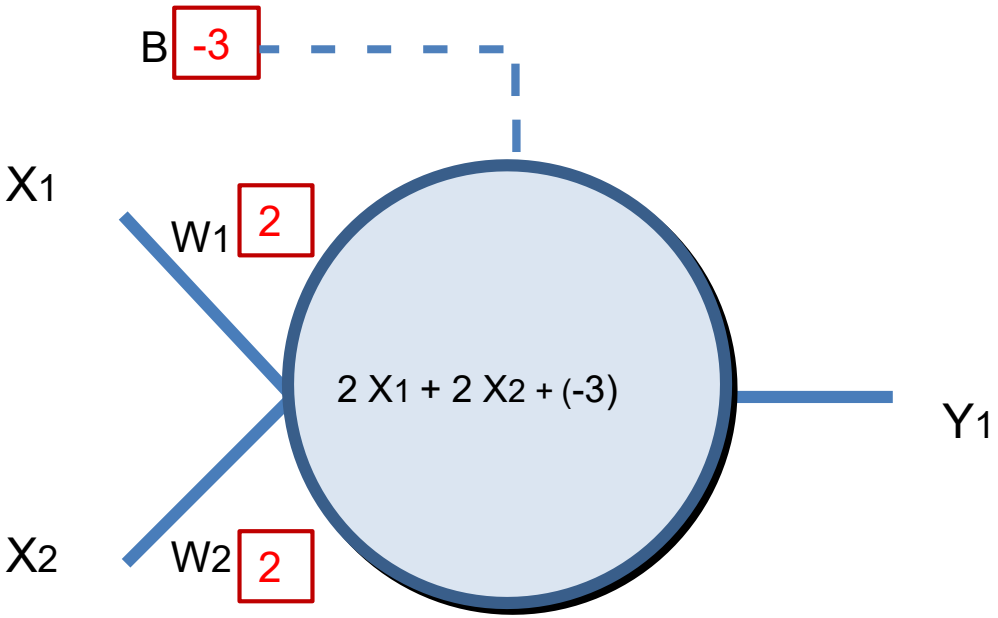


Soleado	Fin de semana	Ir a la playa
NO	NO	0
SI	NO	0
NO	SI	1
SI	SI	1

Si el resultado es ≤ 1 → Representa un 0

Si el resultado es > 1 → Representa un 1

Neurona: ejemplo

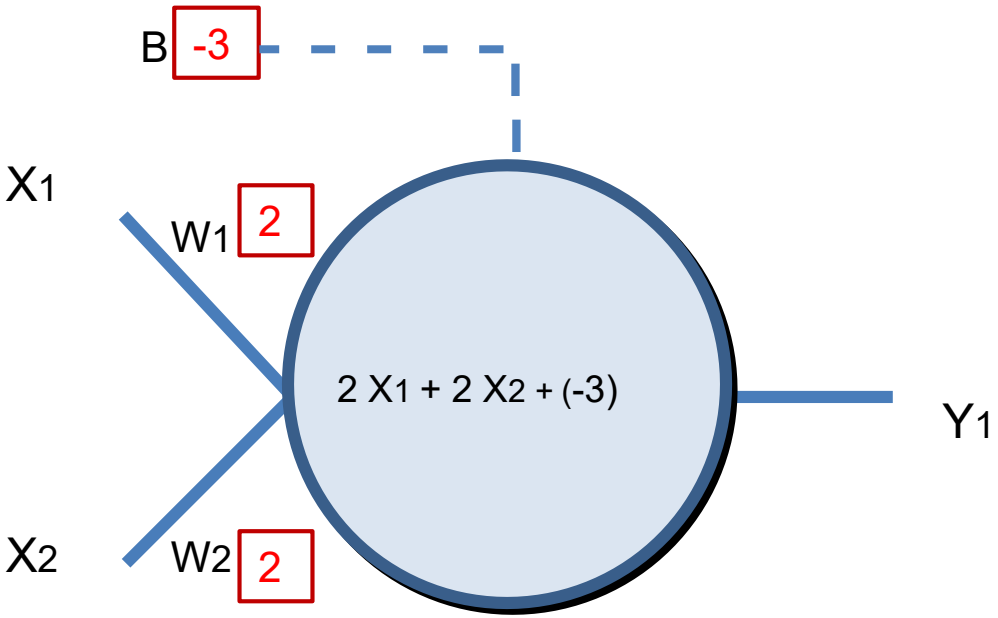


Soleado	Fin de semana	Ir a la playa
NO	NO	-3
SI	NO	-1
NO	SI	-1
SI	SI	5

Si el resultado es ≤ 3 → Representa un 0

Si el resultado es > 3 → Representa un 1

Neurona: ejemplo



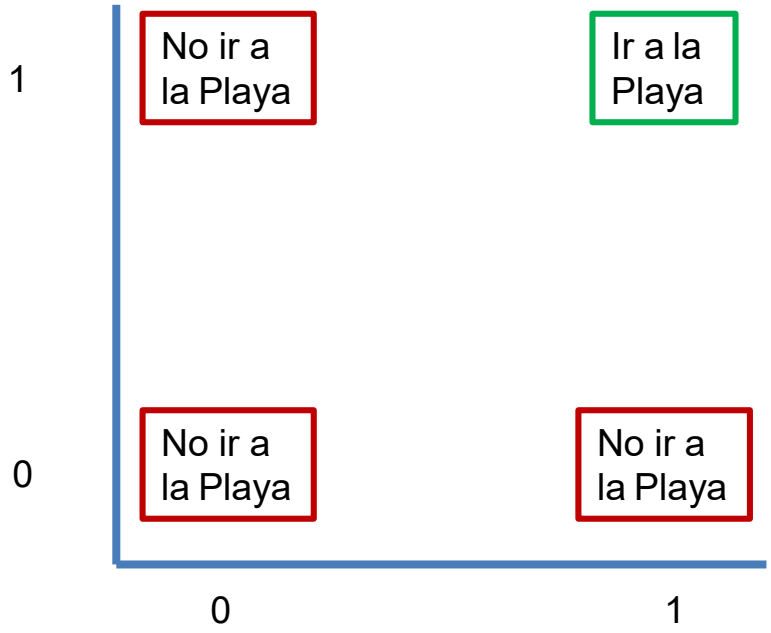
Soleado	Fin de semana	Ir a la playa
NO	NO	0
SI	NO	0
NO	SI	0
SI	SI	1

Si el resultado es ≤ 3 → Representa un 0

Si el resultado es > 3 → Representa un 1

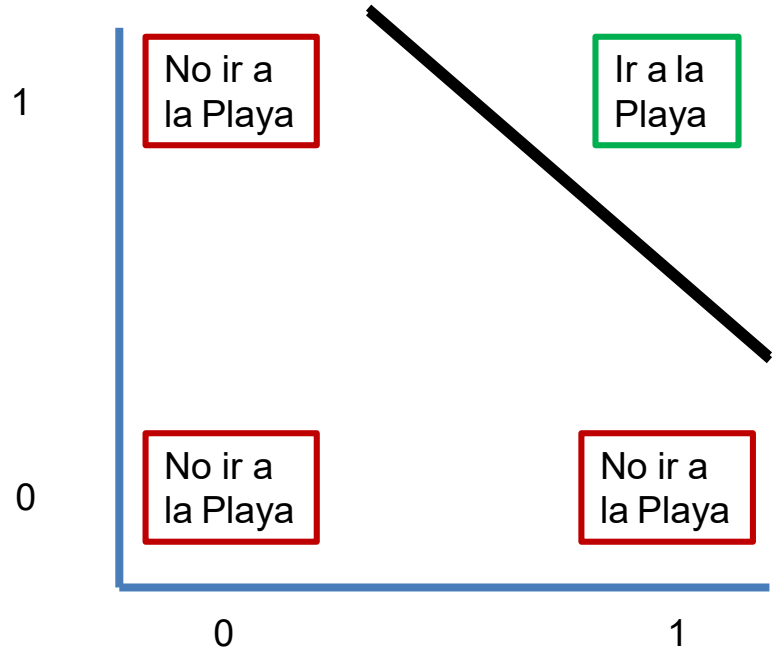
Neurona: ejemplo

También podemos representar nuestro problema como:



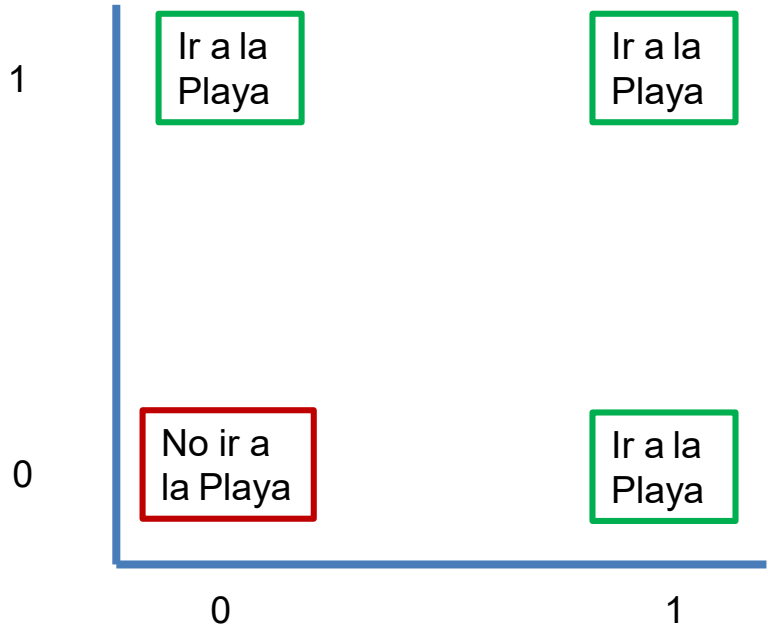
Neurona: ejemplo

También podemos representar nuestro problema como:



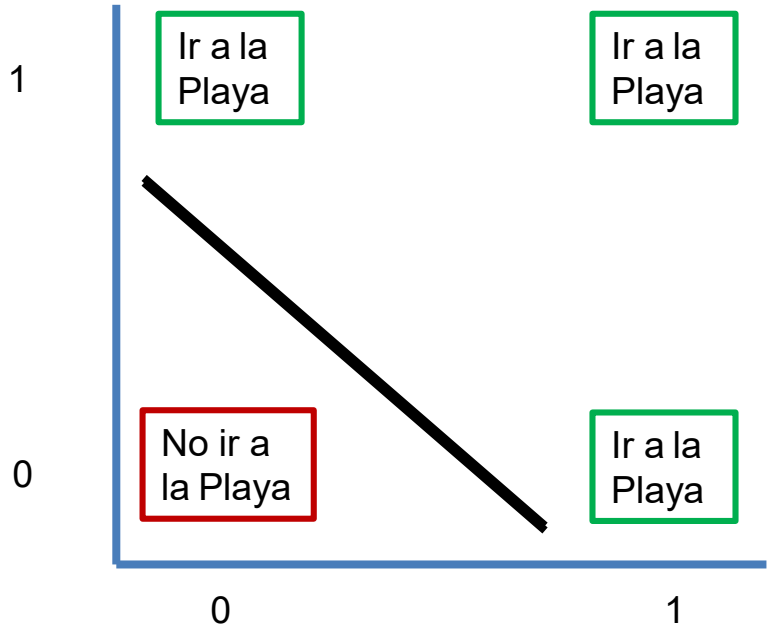
Neurona: ejemplo

Pero existen otros problemas….



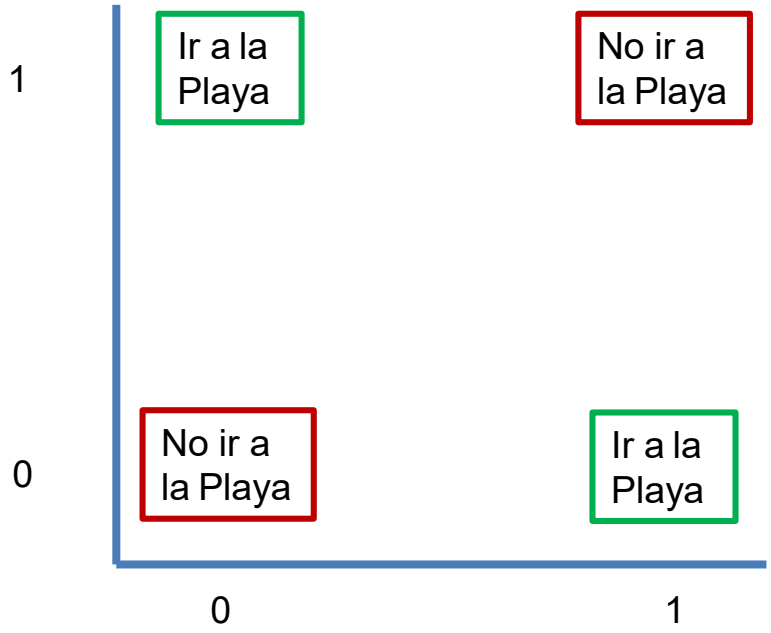
Neurona: ejemplo

Pero existen otros problemas….

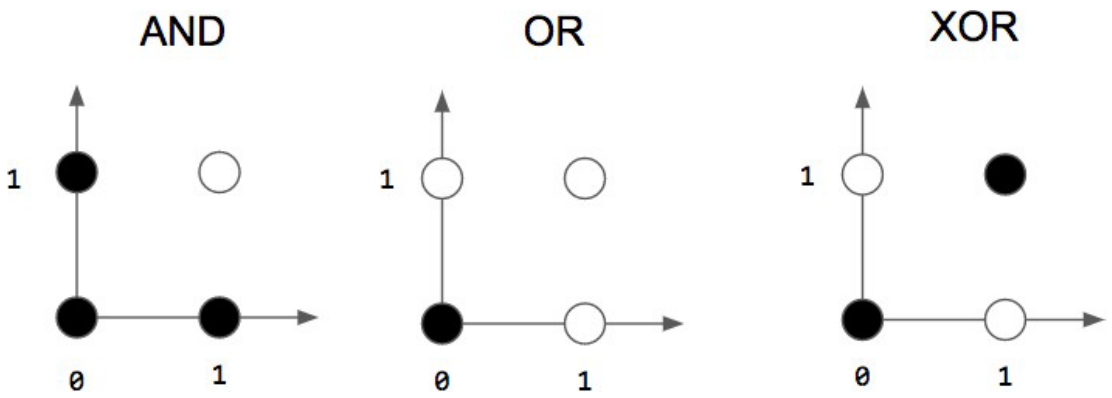


Neurona: ejemplo

En este caso encontramos la limitación del Perceptrón:



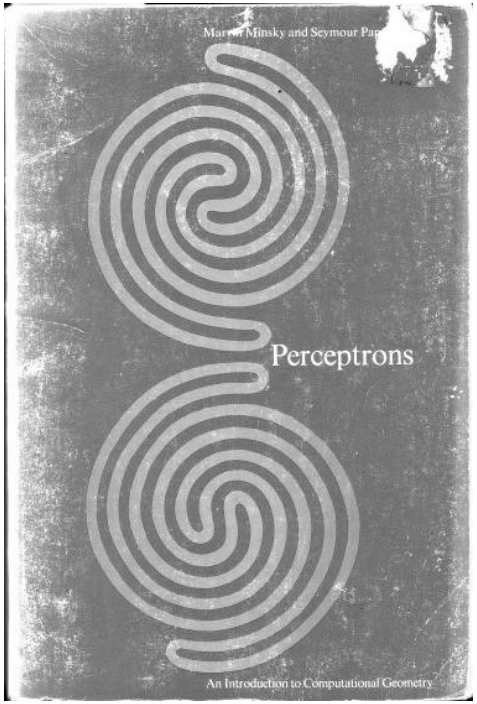
Limitaciones del perceptrón



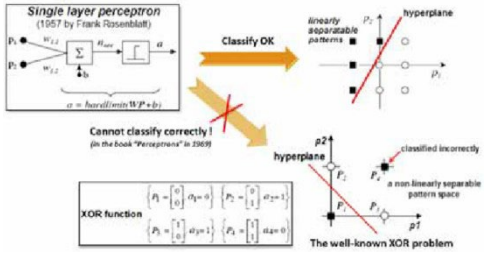
Puertas Lógicas

Limitaciones del perceptrón

Invierno de la Inteligencia Artificial (1969 – 1986)

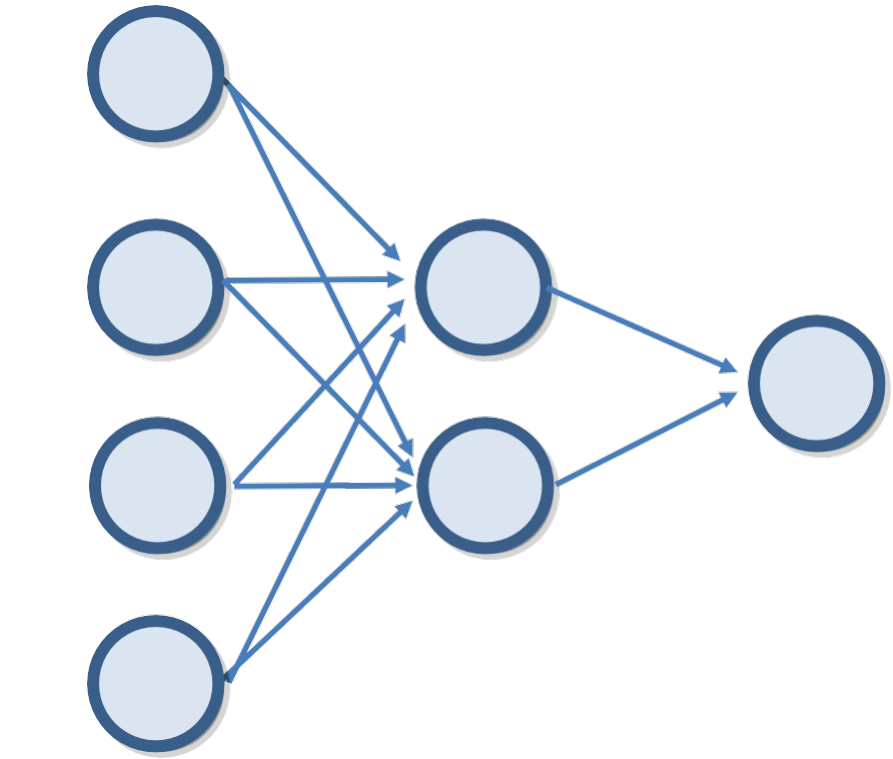


Marvin Minsky y Seymour Papert



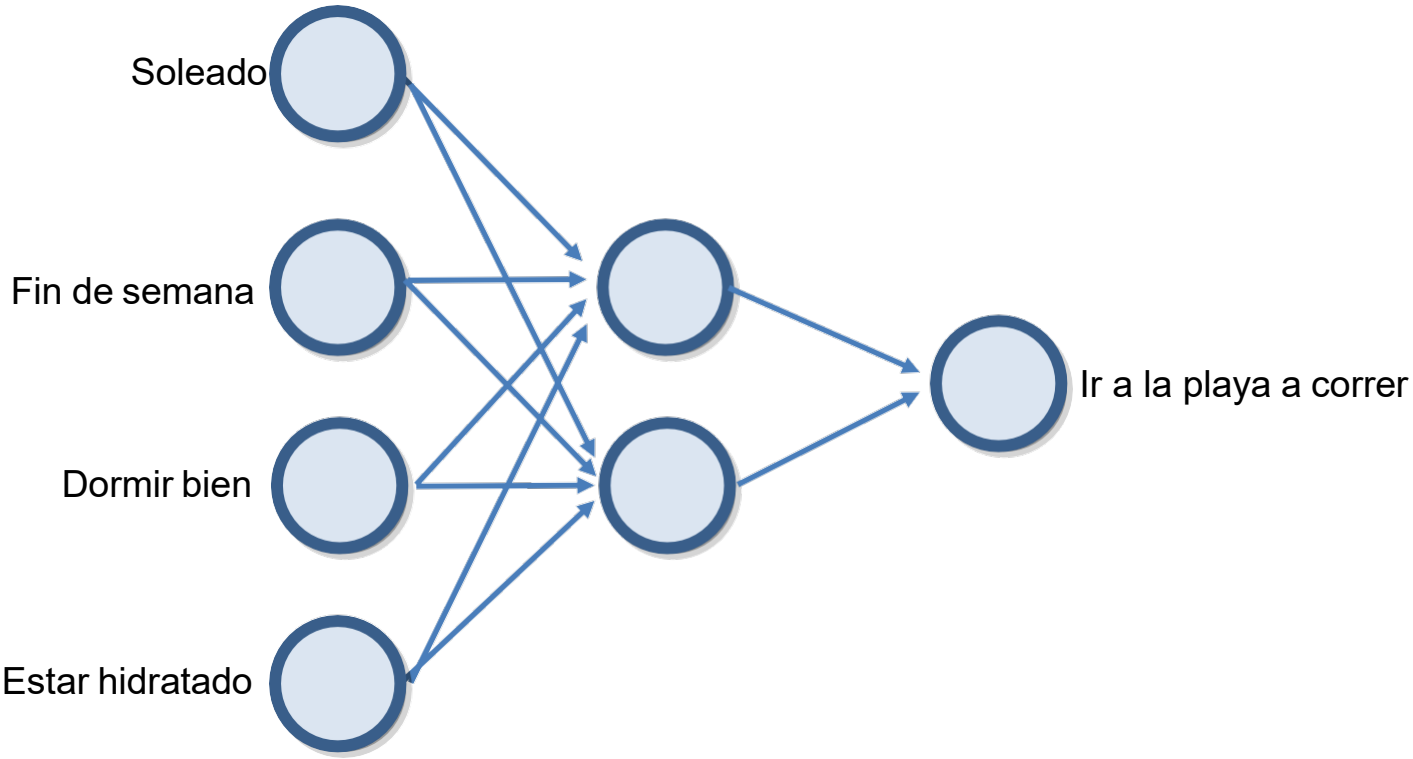
Arquitectura básica de una red neuronal

Al momento de construir una red neuronal, nos encontramos con tres partes que la definen:



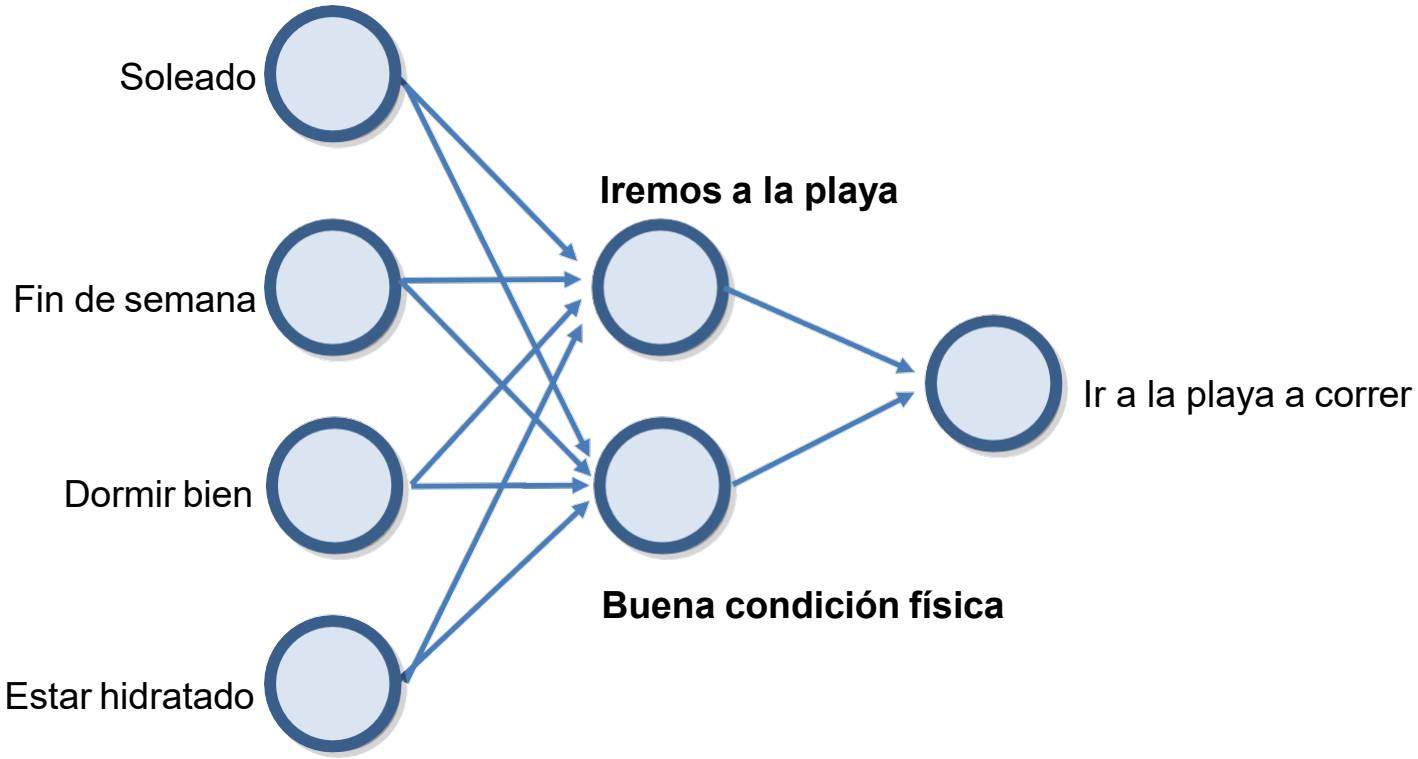
Arquitectura básica de una red neuronal: ejemplo

Al utilizar más neuronas obtenemos lo que se conoce como “Conocimiento Jeraquizado”:



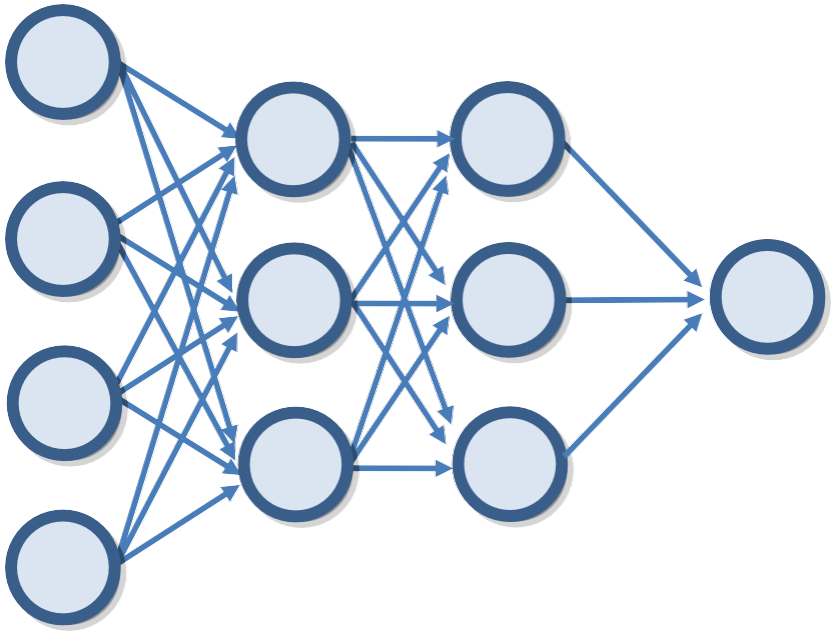
Arquitectura básica de una red neuronal: ejemplo

Al utilizar más neuronas obtenemos lo que se conoce como “Conocimiento Jeraquizado”



Deep learning

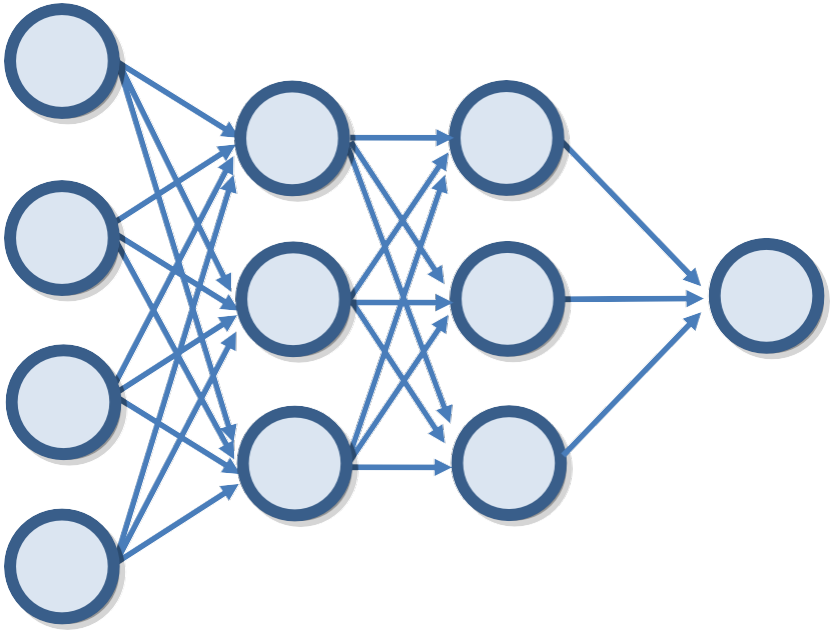
Cuantas más capas, más complejo será el conocimiento.



Deep learning

Para crear una arquitectura básica de Deep Learning, conectaremos neuronas secuencialmente.

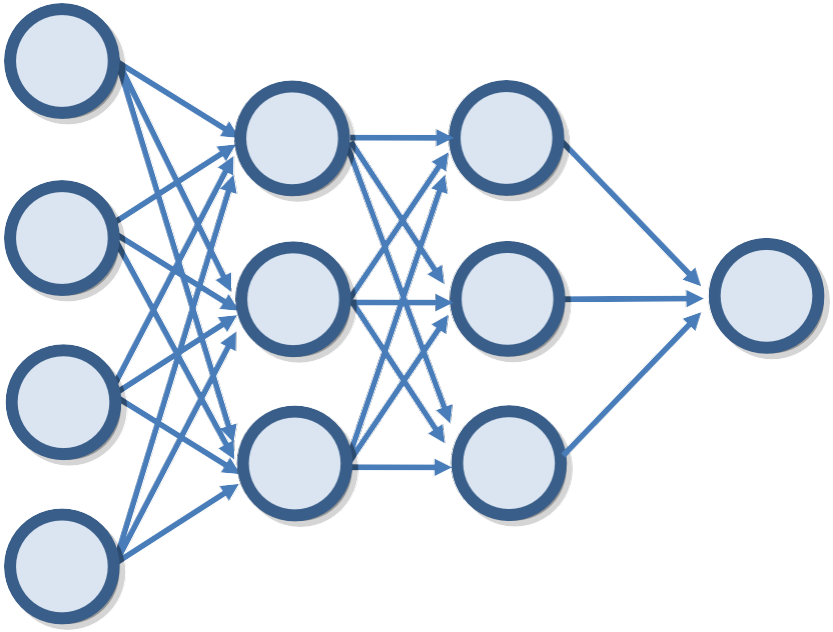
Hay que tener en cuenta que nuestra red de momento esta realizando regresiones lineales.



Deep learning

El problema que encontraremos es que concatenando regresiones lineales, obtendremos otra regresión lineal, es decir otra línea recta.

Por lo que tendremos el mismo problema que con una neurona.

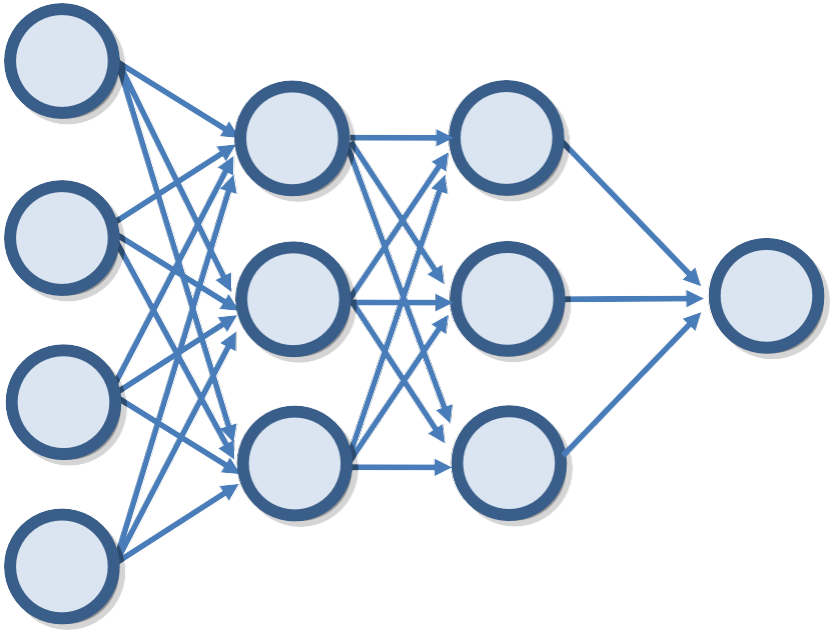


Funciones de activación

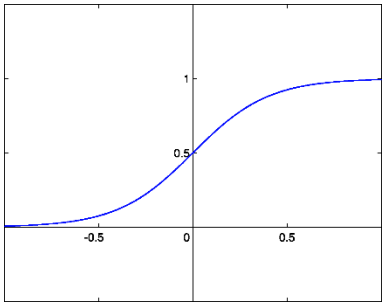
Para solucionar este problema, existen las **funciones de activación f(x)**.

Estas funciones manipulan las neuronas de la capa oculta para obtener una **salida no lineal**.

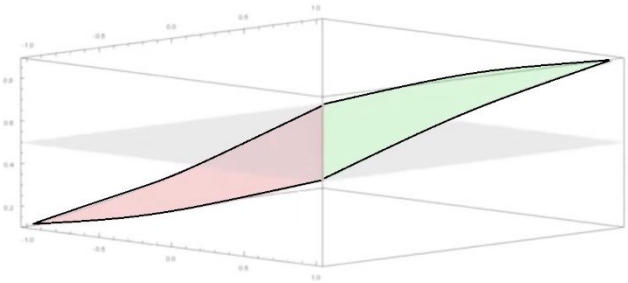
Es decir, simplemente aplican una función a la salida de cada neurona.

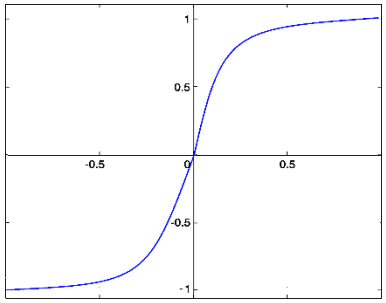


Funciones de activación: tipos

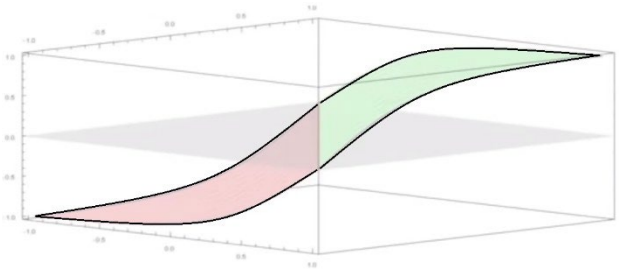


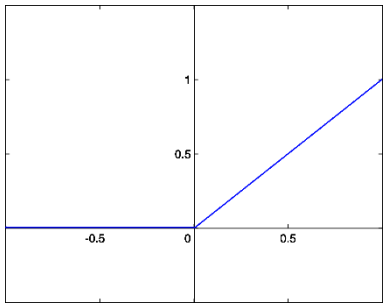
Sigmoide



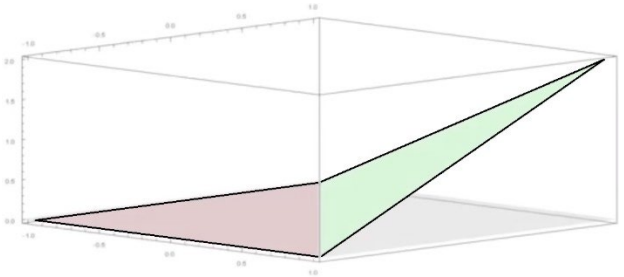


Tangente hiperbólica





Unidad lineal rectificada (ReLU)

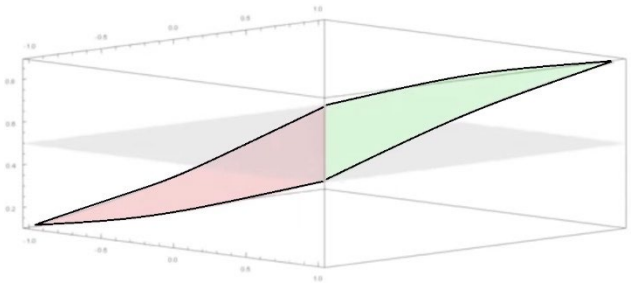


Funciones de activación: ejemplo visual

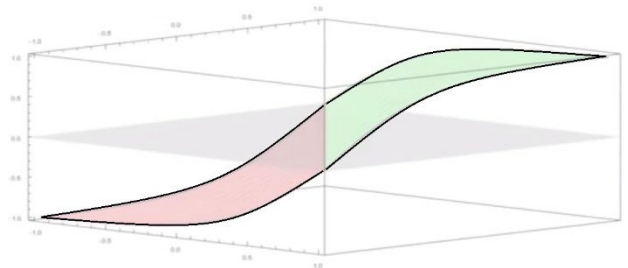
Para escribir las funciones y visualizarlas en 3D podemos usar el siguiente recurso:

https://al-roomi.org/3DPlot/

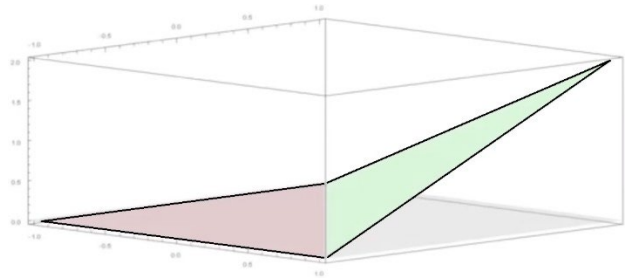
Sigmoide



Tangente hiperbólica



Unidad lineal rectificada (ReLU)



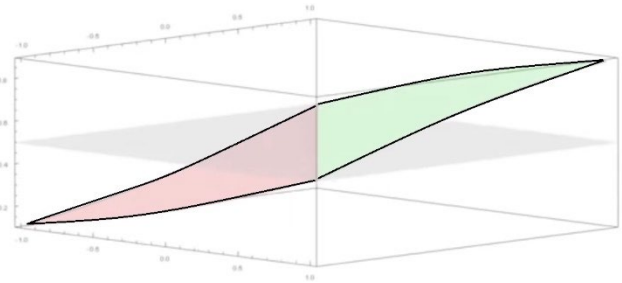
Funciones de activación: ejemplo visual

Para escribir las funciones y visualizarlas en 3D podemos usar el siguiente recurso:

https://al-roomi.org/3DPlot/

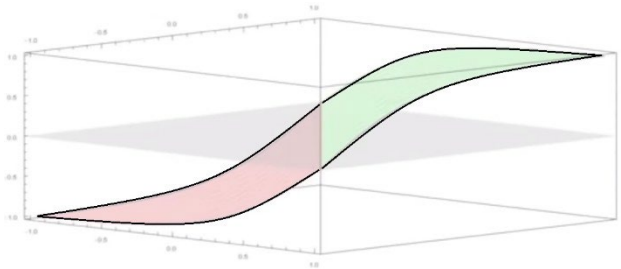
Sigmoide

1 / (1 + E^(-x))



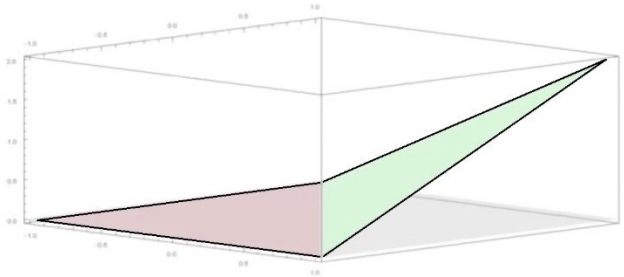
Tangente hiperbólica

(E^x - E^(-x)) / (E^x + E^(-x))

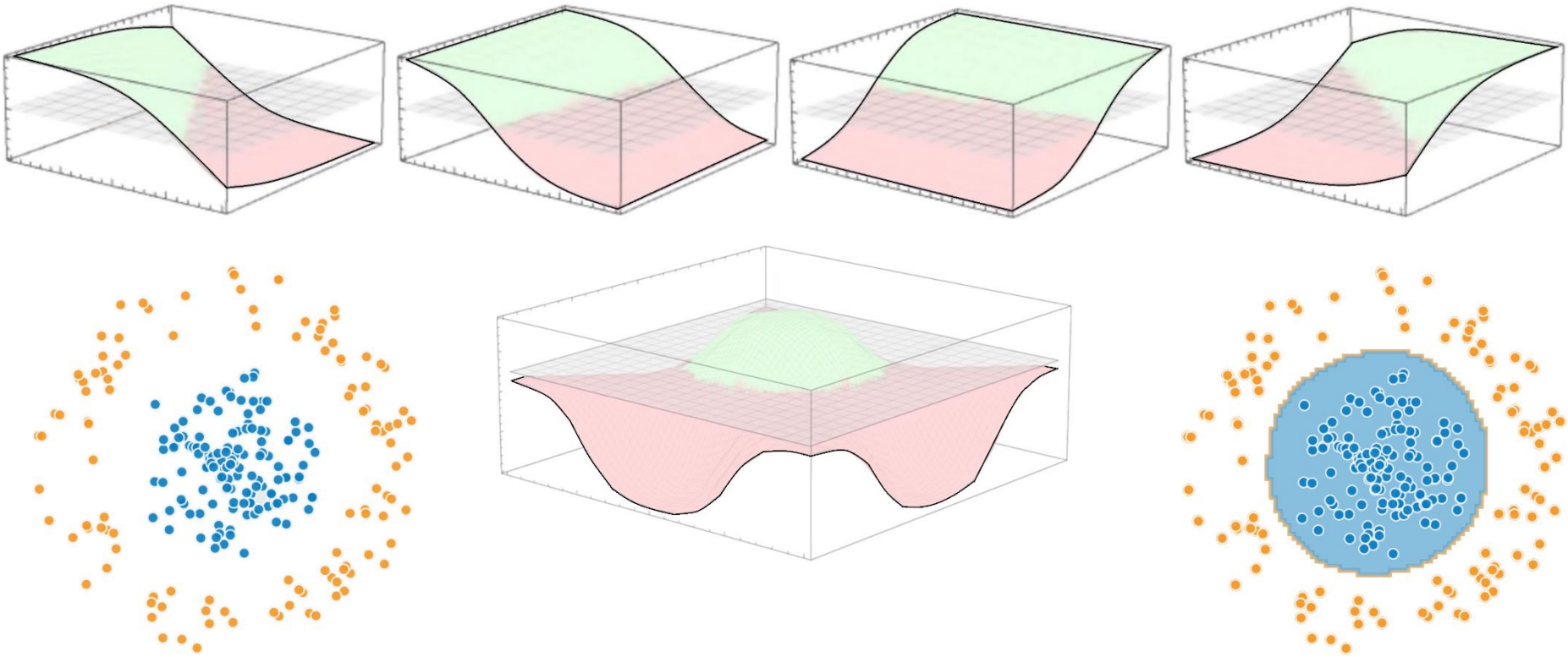


Unidad lineal rectificada (ReLU)

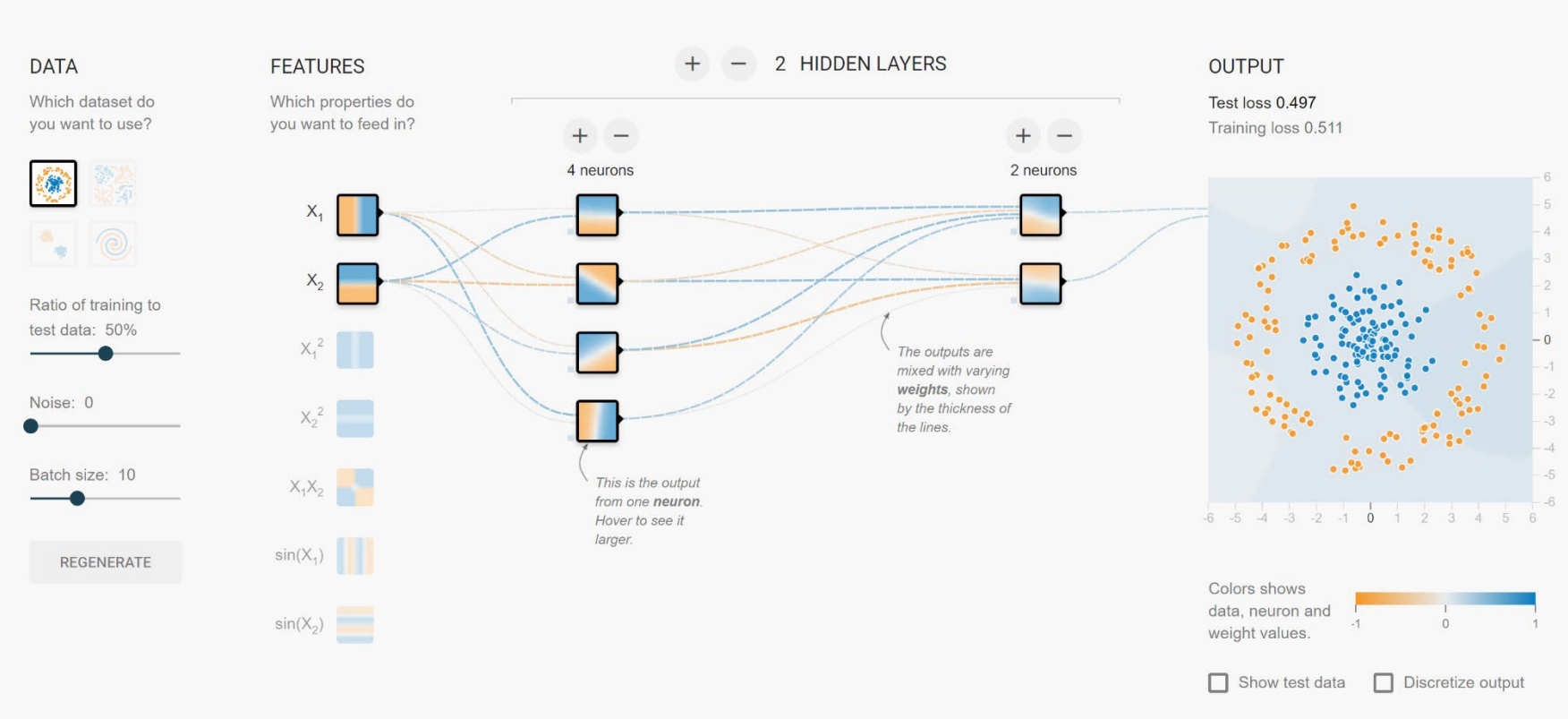
max (0,x)



Funciones de activación



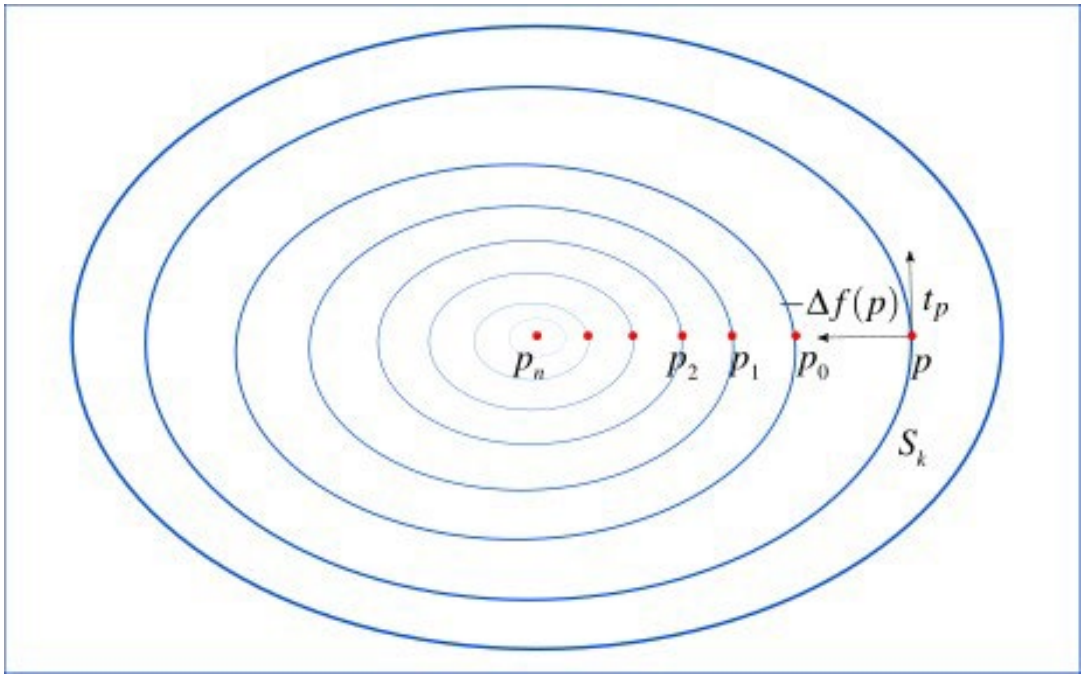
Playground



bit.ly/tf-playground-full

Gradient Descent

Para endender qué hace el algoritmo de Gradient Descent, realizaremos un repaso visual de los conceptos matemáticos que nos ayudaran a su comprensión.

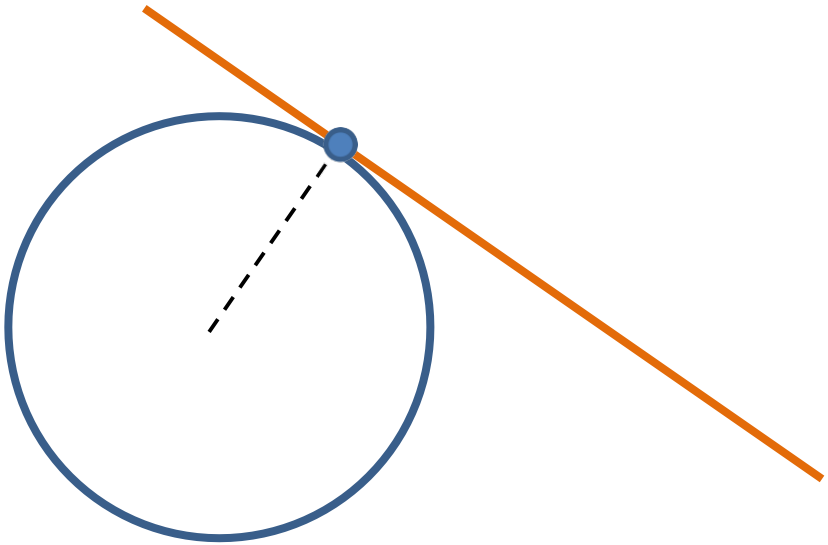


Gradient Descent

Recta Tangente:

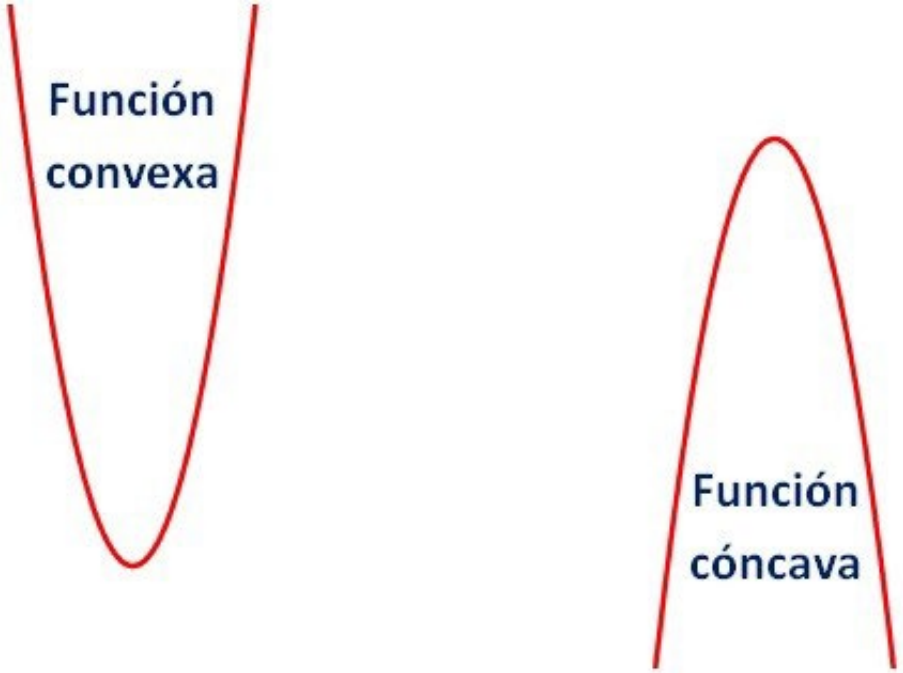
Una tangente (latín. tangere “tocar") es una línea que toca una curva en exactamente **un punto**.

Por ejemplo, una línea tangente a un círculo es perpendicular al radio dibujado al punto de tangencia.



Gradient Descent

Pero... ¿Cómo podemos calcular la tangente de cualquier punto de estas funciones?

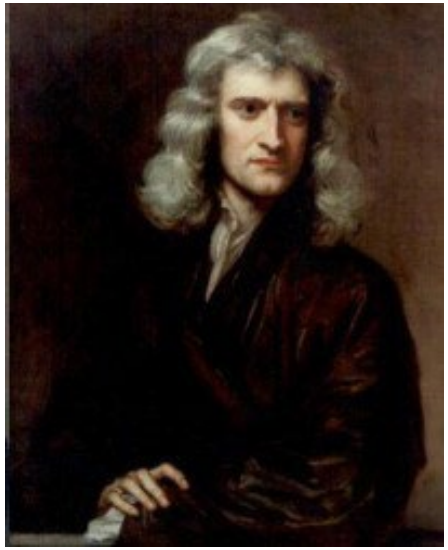


Gradient Descent

Cálculo de la derivada

La derivada de una función unos indica la pendiente de una función en ese punto.

https://goo.gl/wz7S76



Isaac Newton y Gottfried Leibniz

Gradient Descent

Extremos de una función:

El localizar valores extremos es el objetivo básico de la optimización matemática.

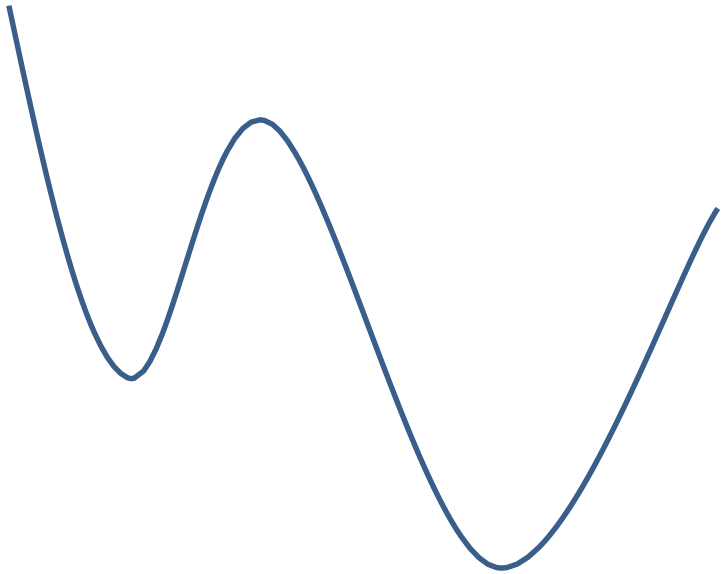
Por tanto si igualamos a cero la derivada de la función, obtendremos el punto (o los puntos) donde la pendiente es nula (un máximo o un mínimo).

$$f'(x) = 0$$

Gradient Descent

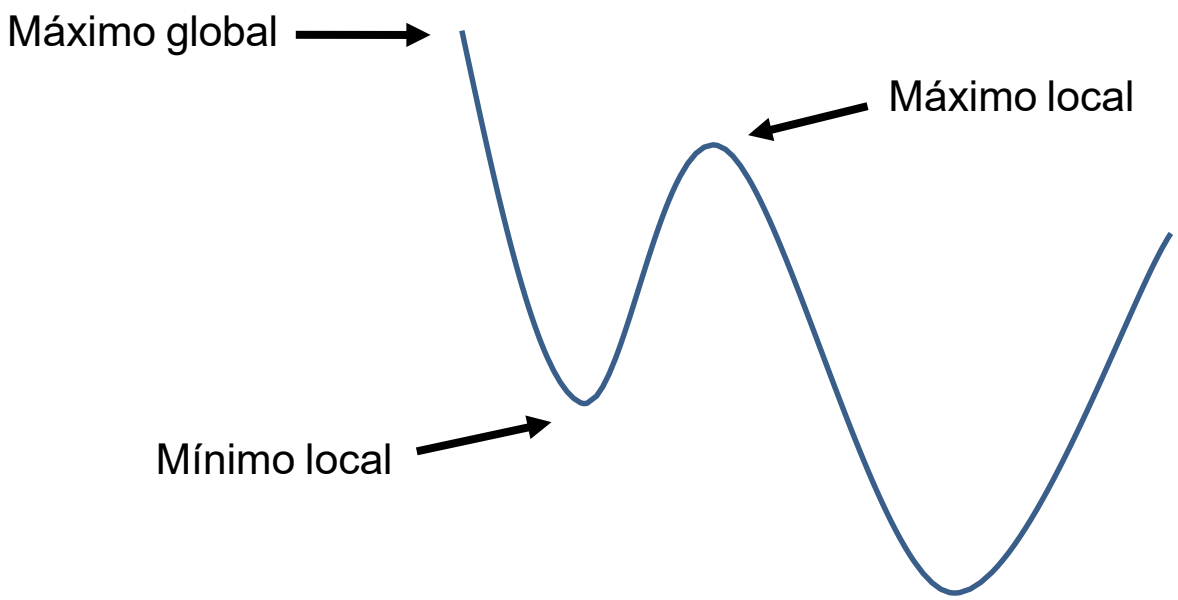
Funciones no convexas

Presentan una anatomía más compleja y, por lo tanto, un problema por resolver:



Gradient Descent

Extremos de una función no convexa:

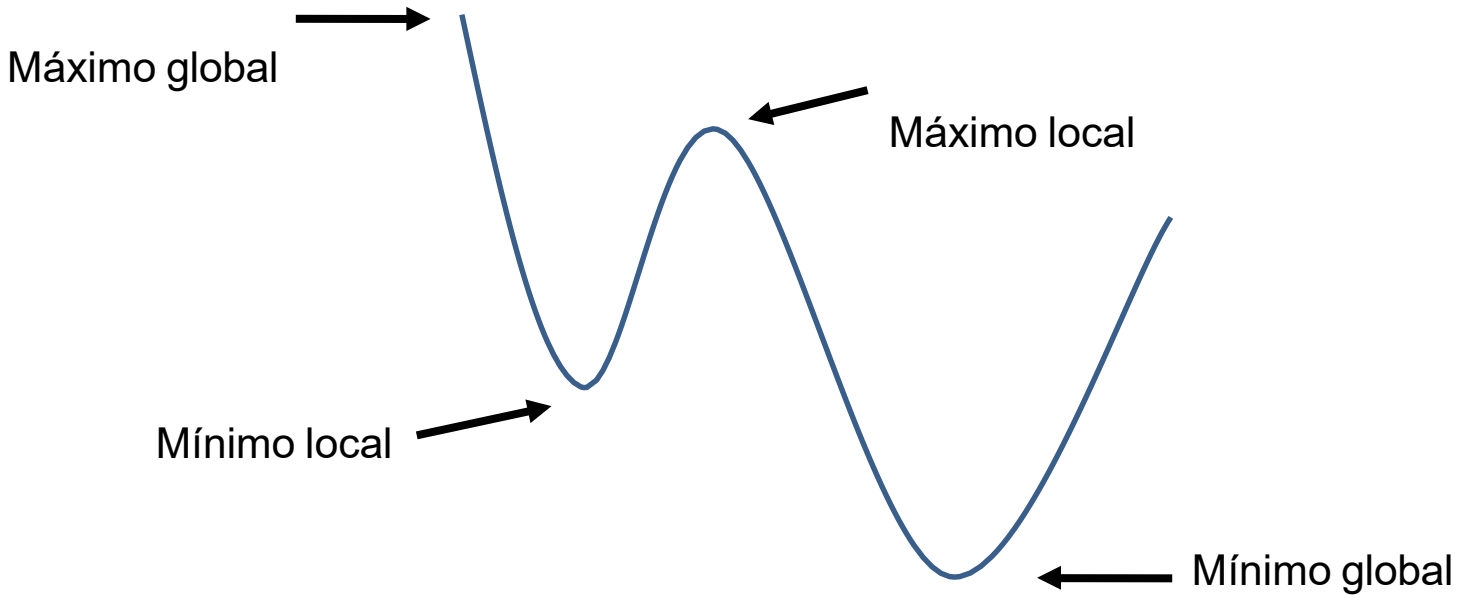


Gradient Descent

Extremos de una función no convexa:

Nuestro problema es que ahora podemos tener más de un punto donde la pendiente es cero.

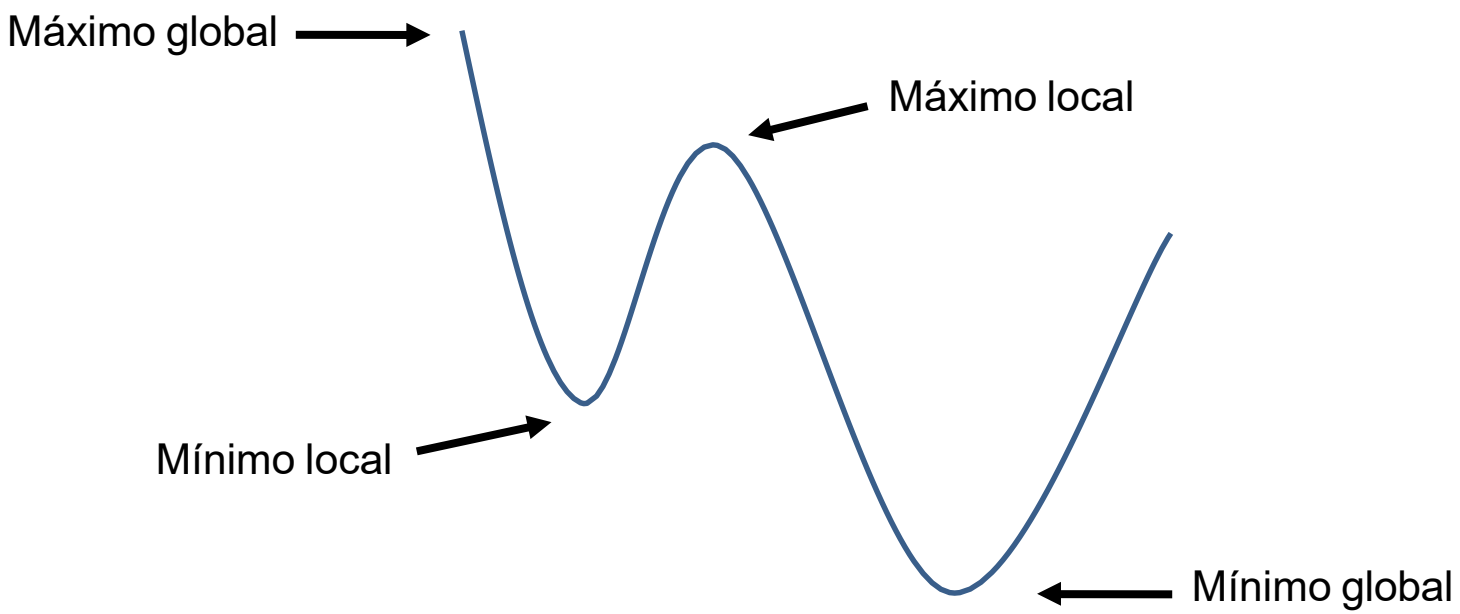
Por ejemplo máximos locales y puntos de inflexión (aka punto de silla)



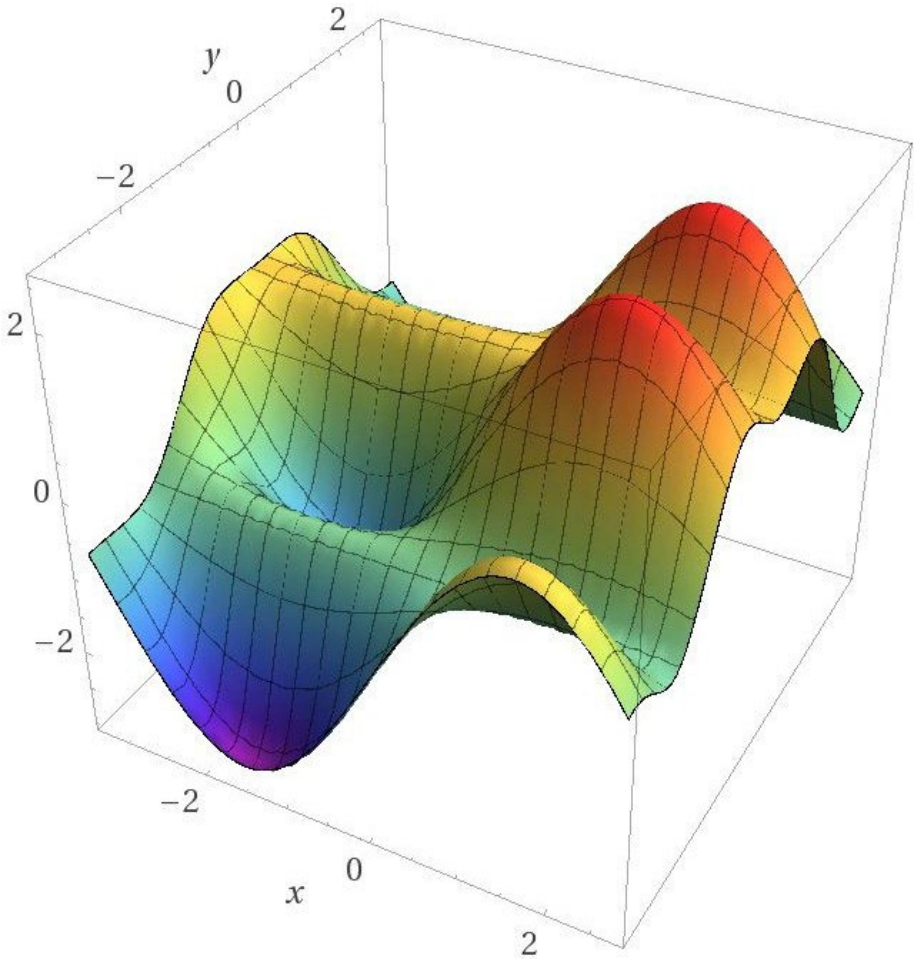
Gradient Descent

Extremos de una función:

¿Cómo podemos resolver este problema?



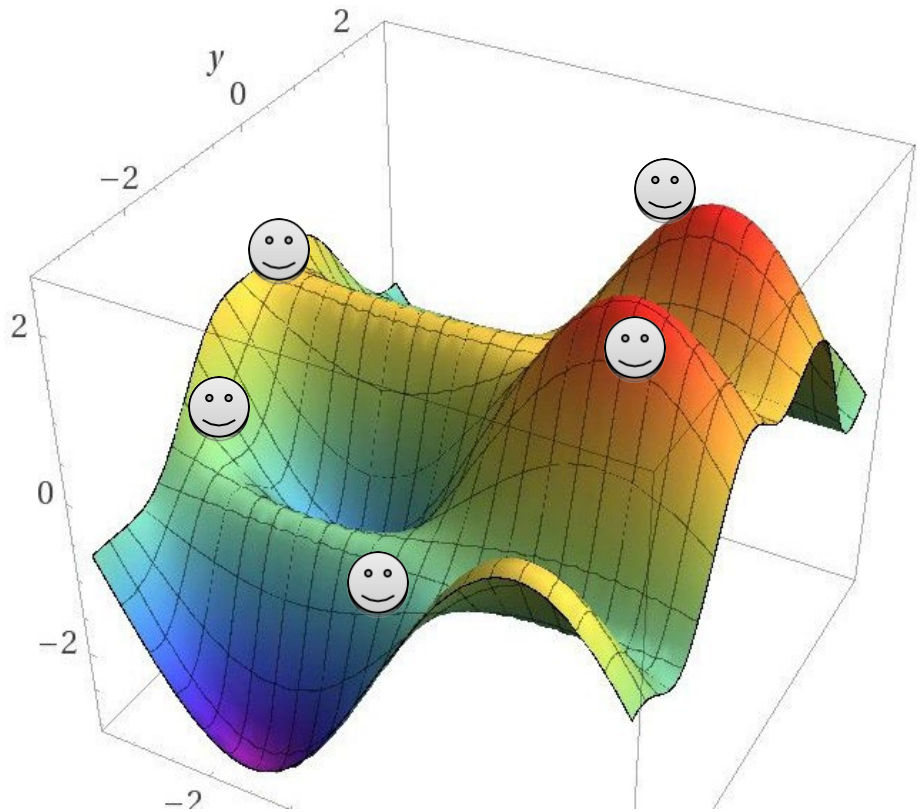
Gradient Descent



Gradient Descent

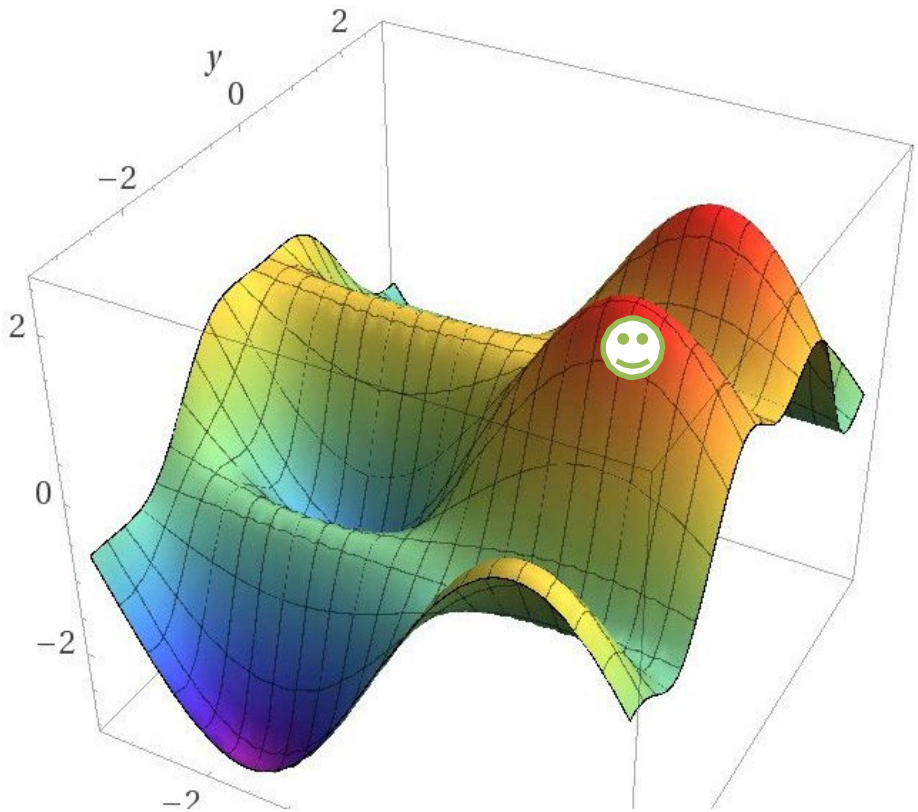
Gradient Descent

Tened en cuenta que al comienzo del entrenamiento los parámetros serán inicializados de manera aleatoria, por lo que nuestra posición inicial en el mapa también lo será.



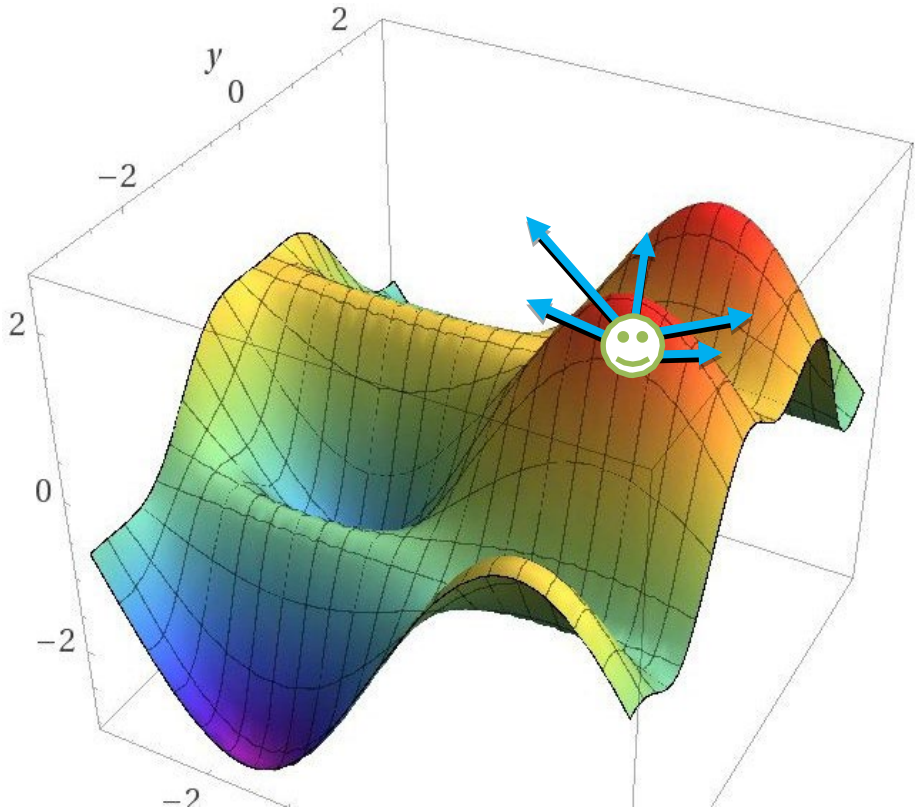
Gradient Descent

Tened en cuenta que al comienzo del entrenamiento los parámetros serán inicializados de manera aleatoria, por lo que nuestra posición inicial en el mapa también lo será.



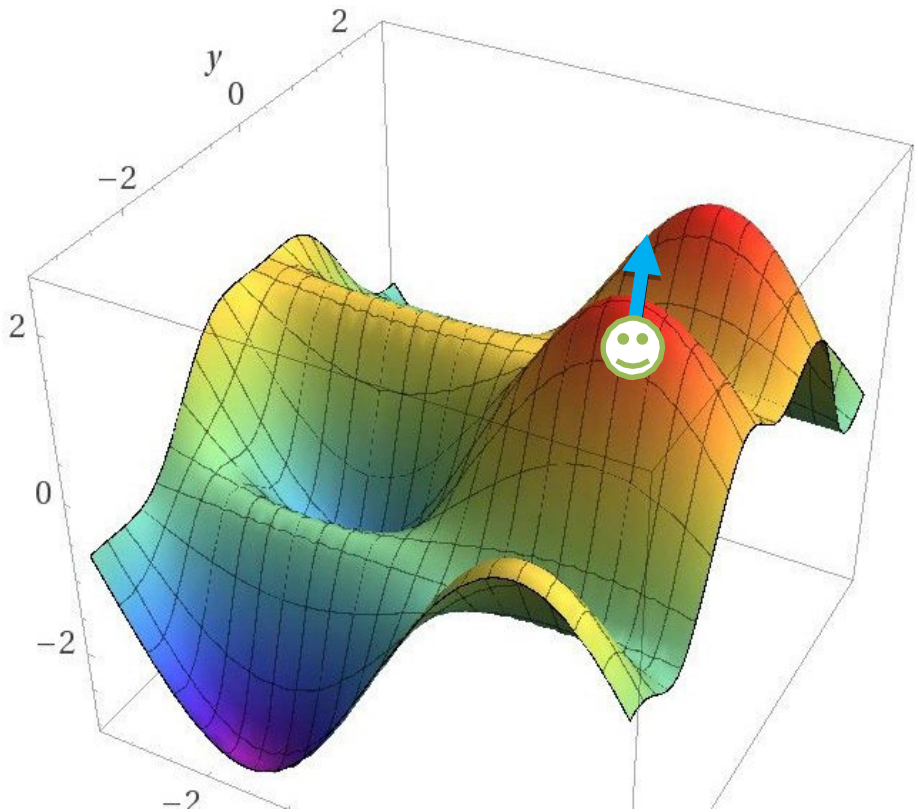
Gradient Descent

Luego se realizaran calculos de derivadas parciales, lo que nos dará los valores que indicaran la inclinación en el eje, en distintas direcciones.



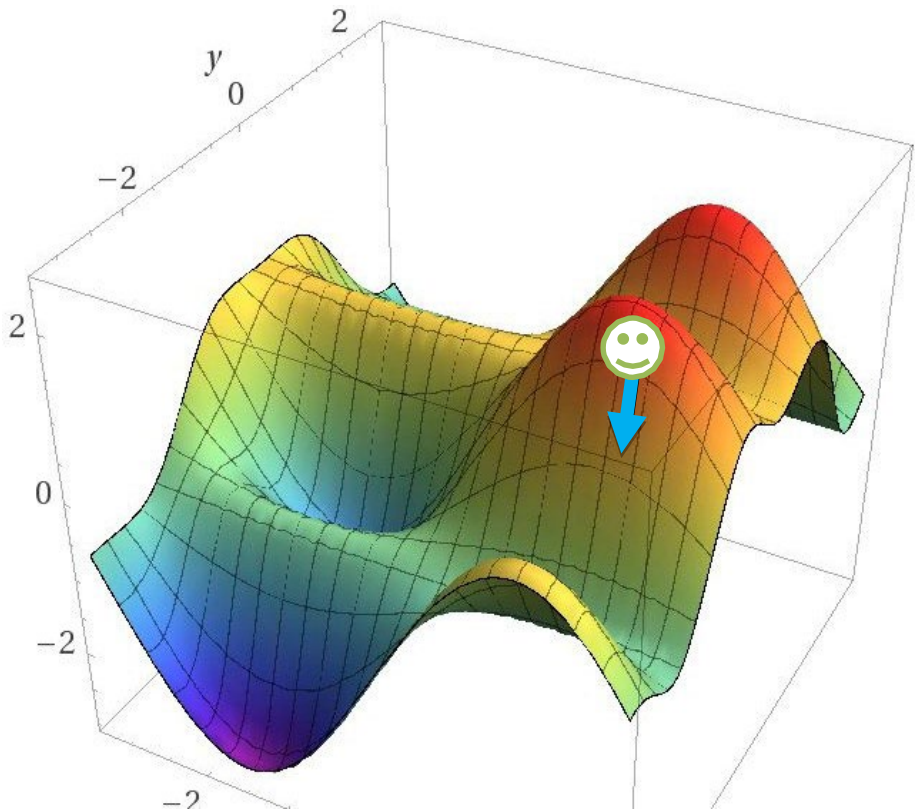
Gradient Descent

Con ello se obtiene un vector que nos señala cuál es la dirección donde la pendiente tiene más inclinación (**mayor error**). Este vector es conocido como Gradiente.



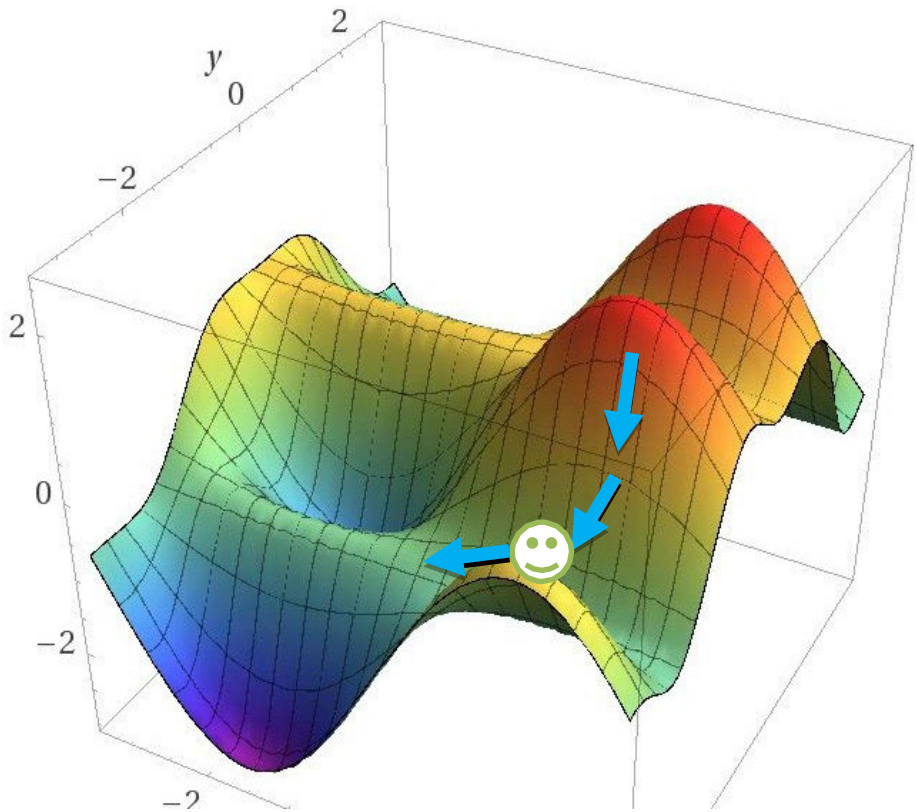
Gradient Descent

Por ello, emplearemos el opuesto del gradiente para poder descender y, de esta manera, se actualizaran nuestros parámetros. A continuación volveremos a repetir el proceso.



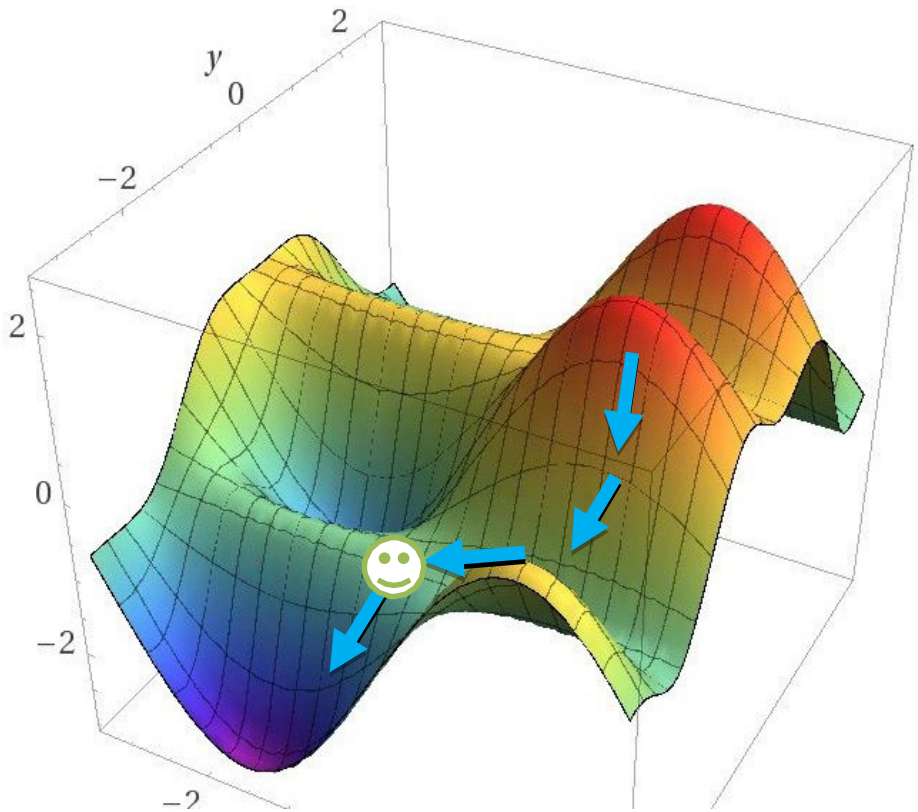
Gradient Descent

Iteraremos hasta llegar a el punto más bajo de coste.



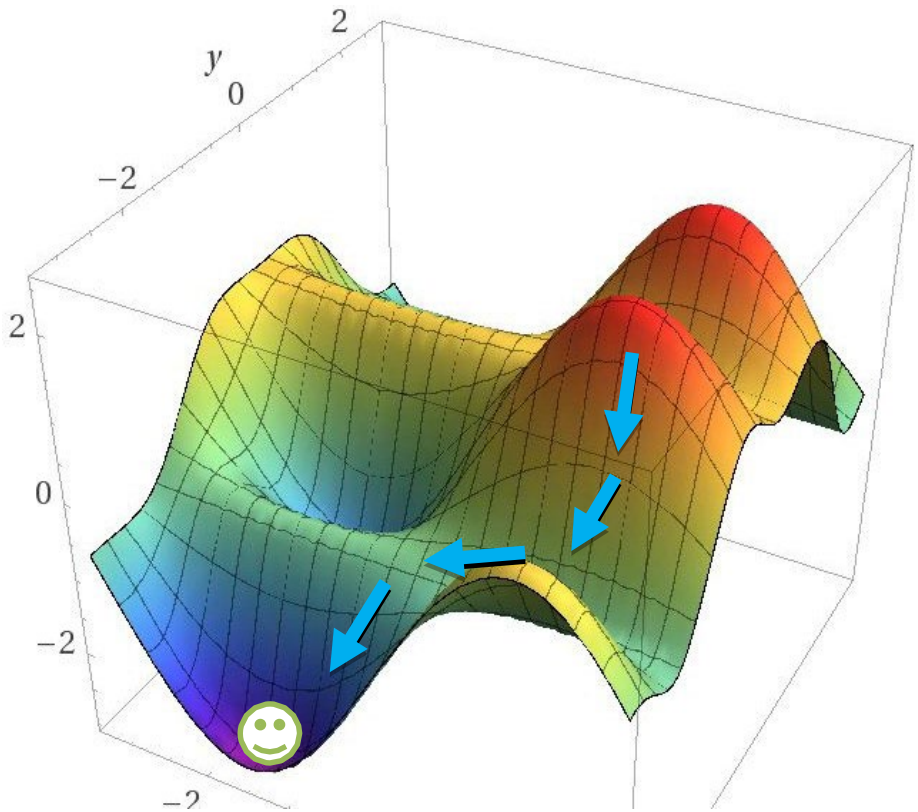
Gradient Descent

Iteraremos hasta llegar a el punto más bajo de coste.



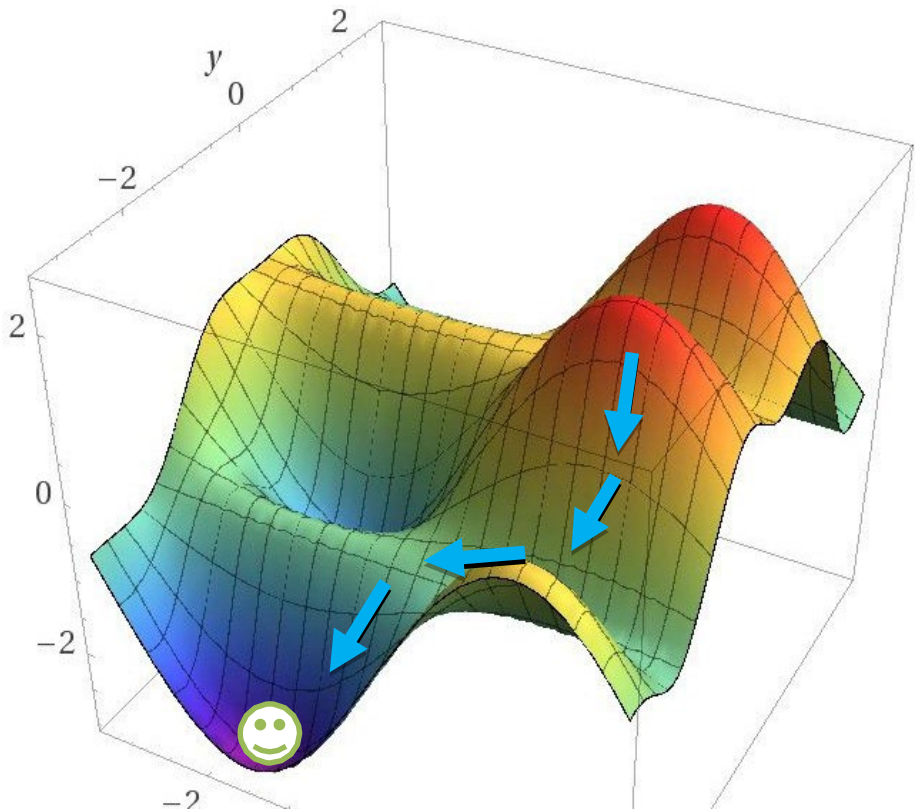
Gradient Descent

Iteraremos hasta llegar a el punto más bajo de coste.



Gradient Descent

¿Falta algo?



Gradient Descent

¿Falta algo?

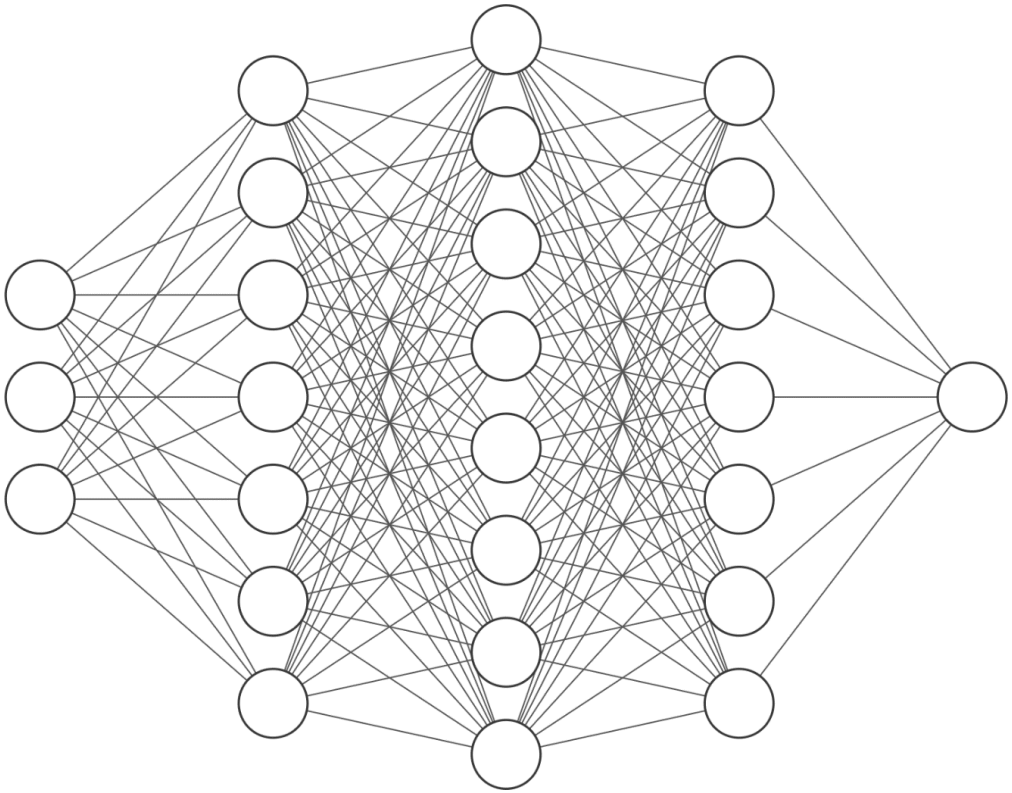
Learning Rate: durante cada iteración, el algoritmo de descenso de gradientes multiplica la tasa de aprendizaje por el gradiente. El producto resultante se denomina paso de gradiente.

La tasa de aprendizaje es un hiperparámetro fundamental.

bit.ly/learning-rate

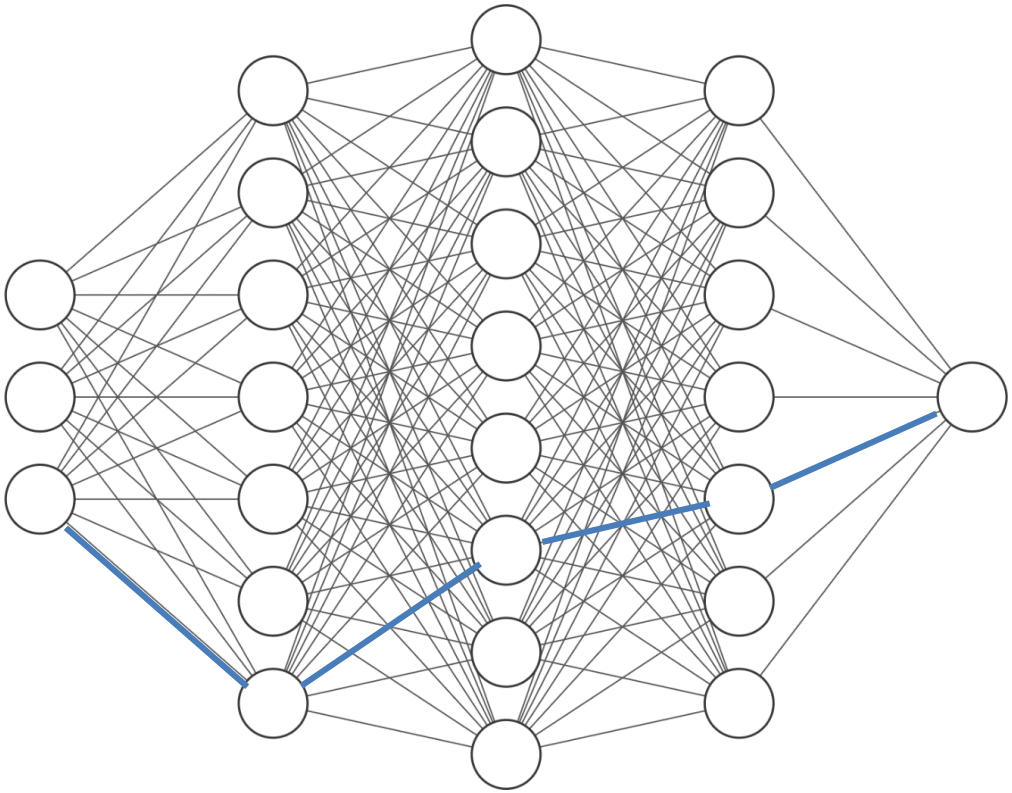
Backpropagation

Si bien tenemos funciones de activación y funciones del coste del error, resulta que calcular el coste de una red neuronal es muy costoso e ineficiente.



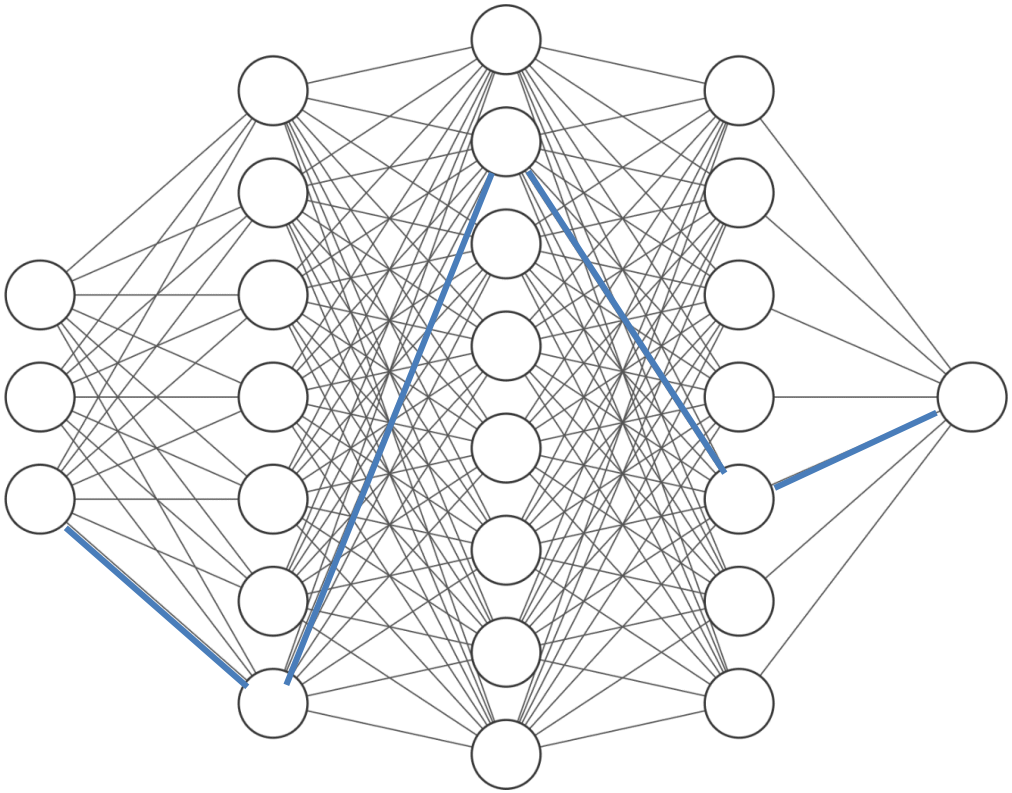
Backpropagation

Cada vez que queremos calcular el efecto del cambio de un parámetro, este parámetro se puede ver afectado por el camino que toma.



Backpropagation

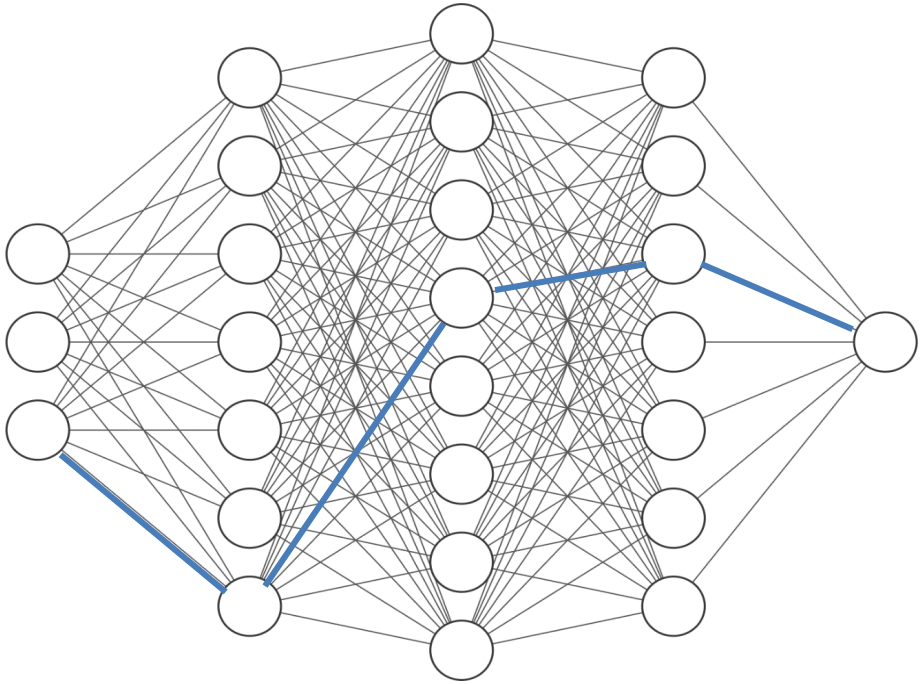
Cada vez que queremos calcular el efecto del cambio de un parámetro, este parámetro se puede ver afectado por el camino que toma.



Backpropagation

Cada vez que queremos calcular el efecto del cambio de un parámetro, este parámetro se puede ver afectado por el camino que toma.

Además del hecho de que cada neurona aplica un función, también se suma la problemática de que cada función dispone de parámetros dificultando más el cálculo.



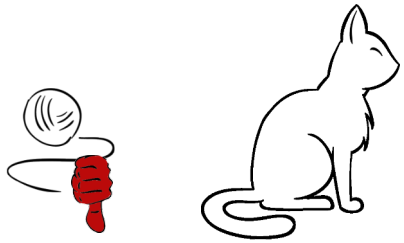
Coste
(Cálculo del error)

A este problema de lo conoce como “Chain of Responsibility”.

Backpropagation

Para entenderlo mejor, veremos un ejemplo.

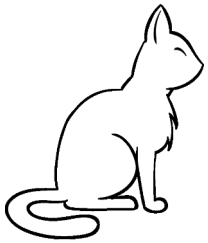
Hoy he decidido comprarle un nuevo juguete a mi gato, pero al llegar a casa y dárselo, este se ha dado la vuelta y lo ha rechazado. ¿Por qué? ¿Qué ha pasado?



Backpropagation

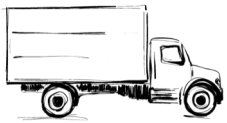
Este fuerte mensaje de error, nos hace surgir preguntas de este tipo:

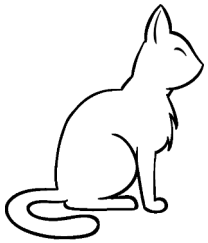
¿Me habré equivocado al elegir la tienda donde lo compré?



Backpropagation

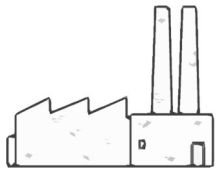
¿Pasó algo en el transporte del juguete hacia la tienda?

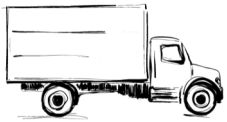


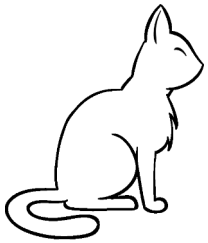


Backpropagation

¿Hubo algún tipo de problema durante su fabricación?

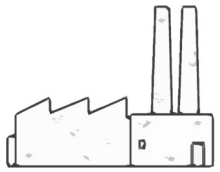


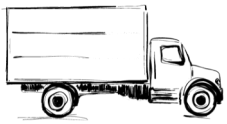


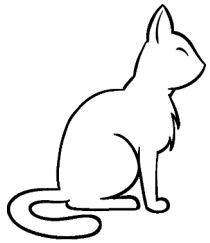


Backpropagation

¿O bien, el problema viene de las materias primas utilizadas para su fabricación?

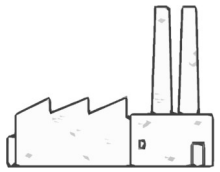


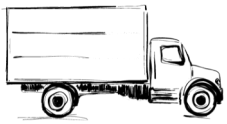


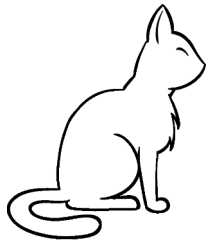


Backpropagation

Todas esas preguntas sirven para saber cuál es la responsabilidad de cada uno de los componentes de nuestra red en el coste final.

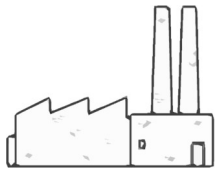


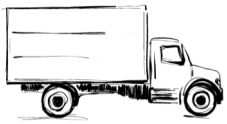


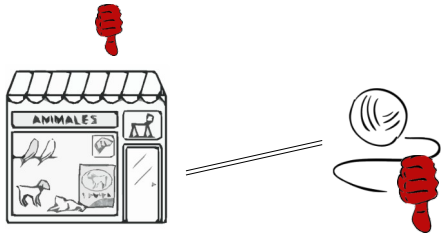


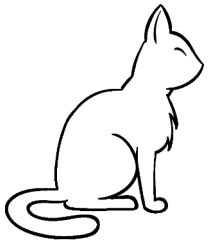
Backpropagation

Tenemos que ir hacía atrás y calcular el error dentro de cada uno de los componentes de nuestra red.



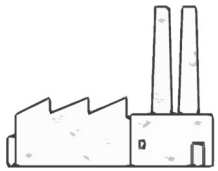


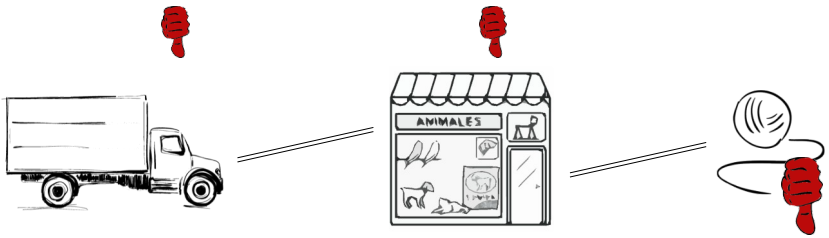


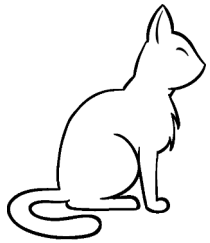


Backpropagation

Tenemos que ir hacía atrás y calcular el error dentro de cada uno de los componentes de nuestra red.

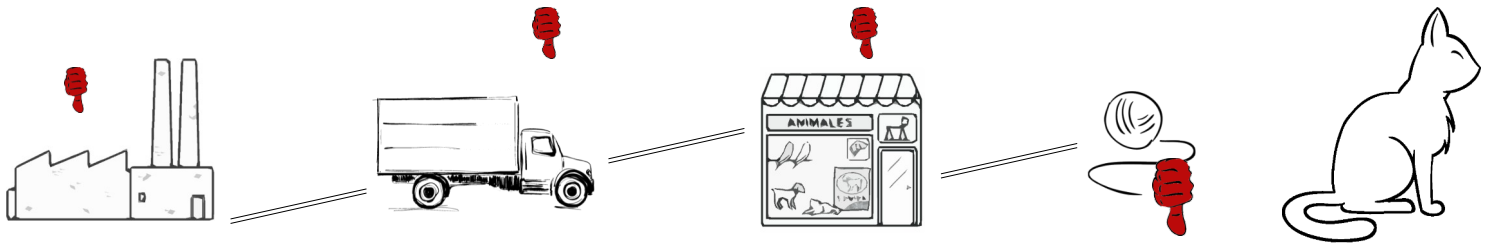






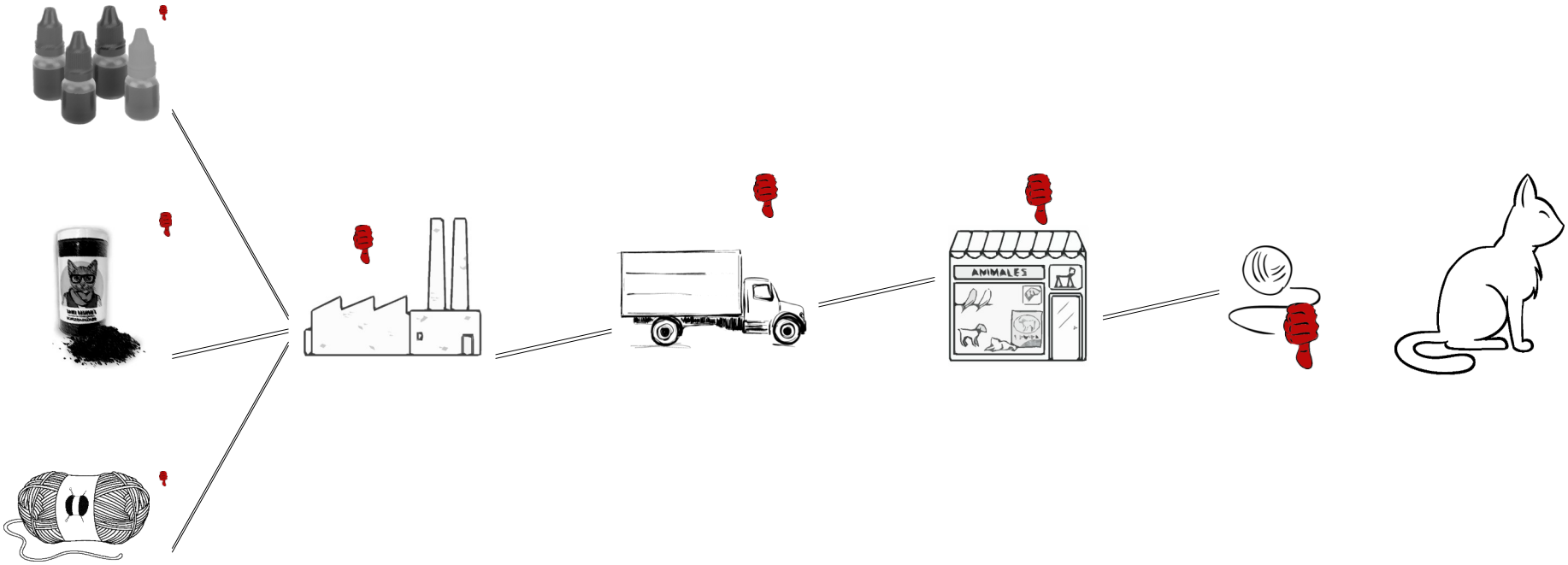
Backpropagation

Tenemos que ir hacía atrás y calcular el error dentro de cada uno de los componentes de nuestra red.



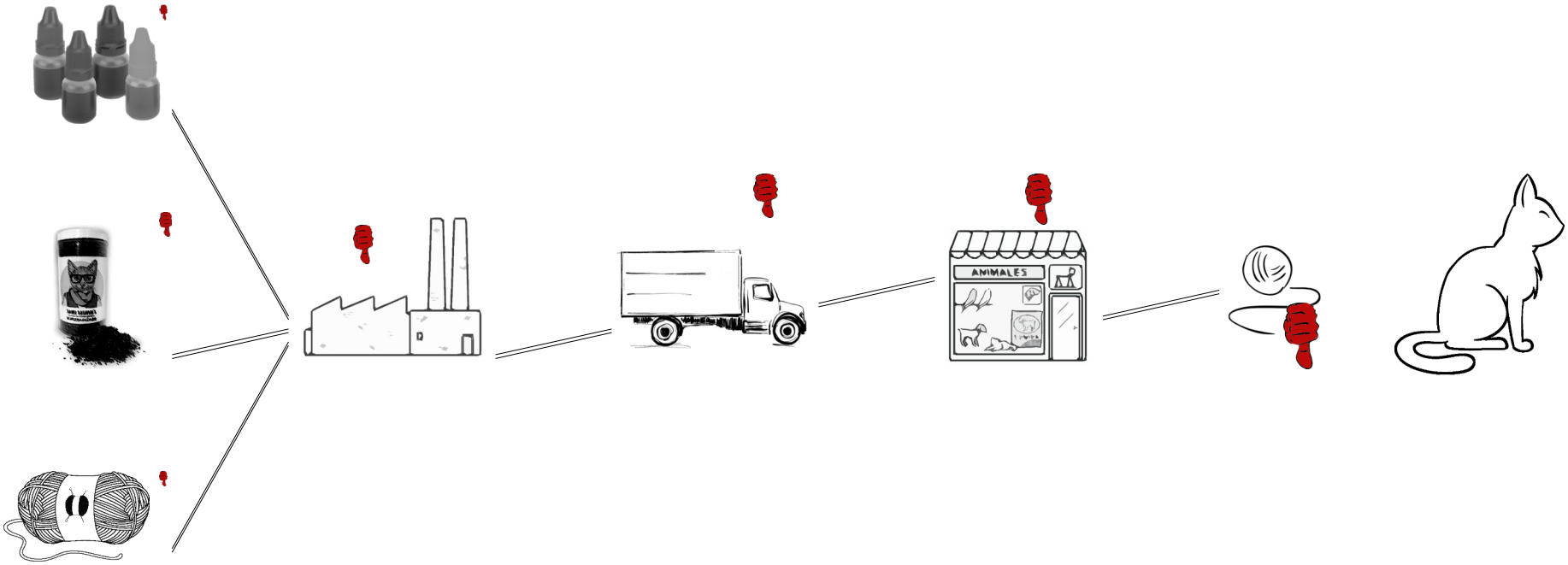
Backpropagation

Tenemos que ir hacía atrás y calcular el error dentro de cada uno de los componentes de nuestra red para poder reajustar nuestros parámetros y así minimizar el error.



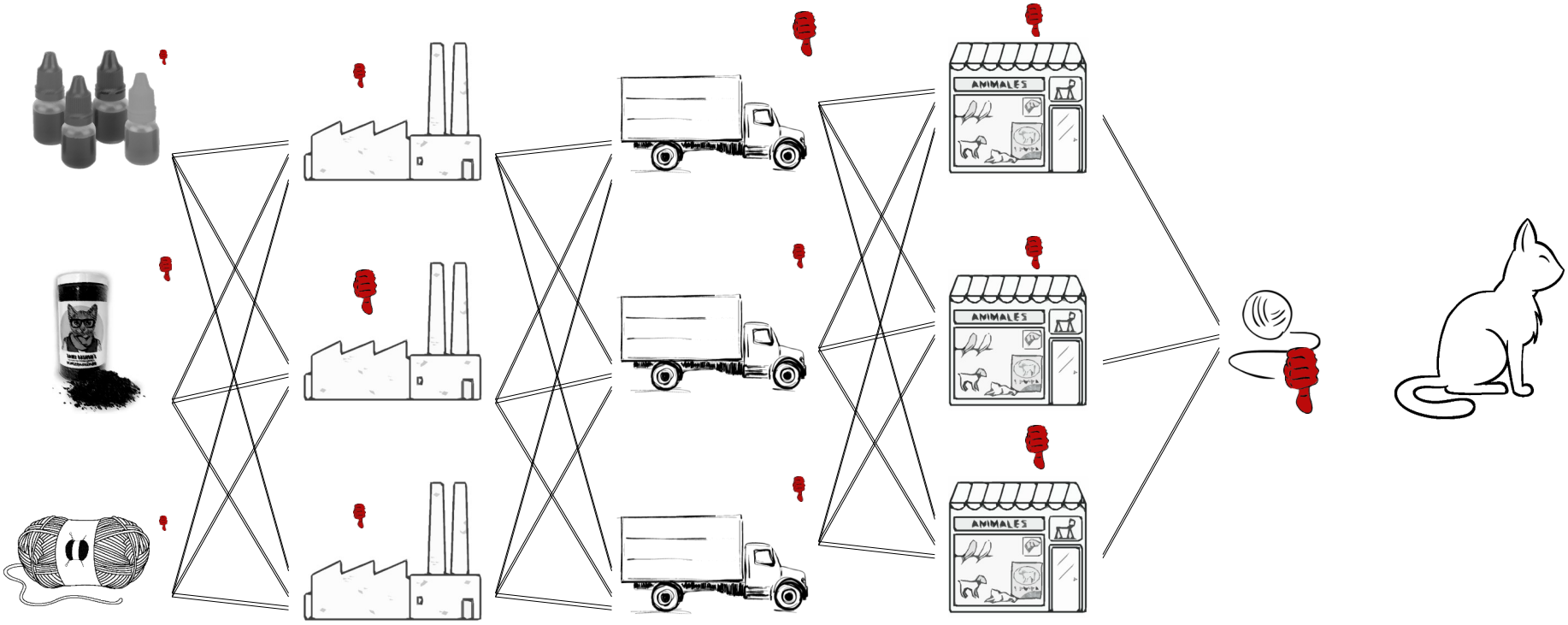
Backpropagation

Esto es lo que nos permite el algoritmo de Backpropagation.



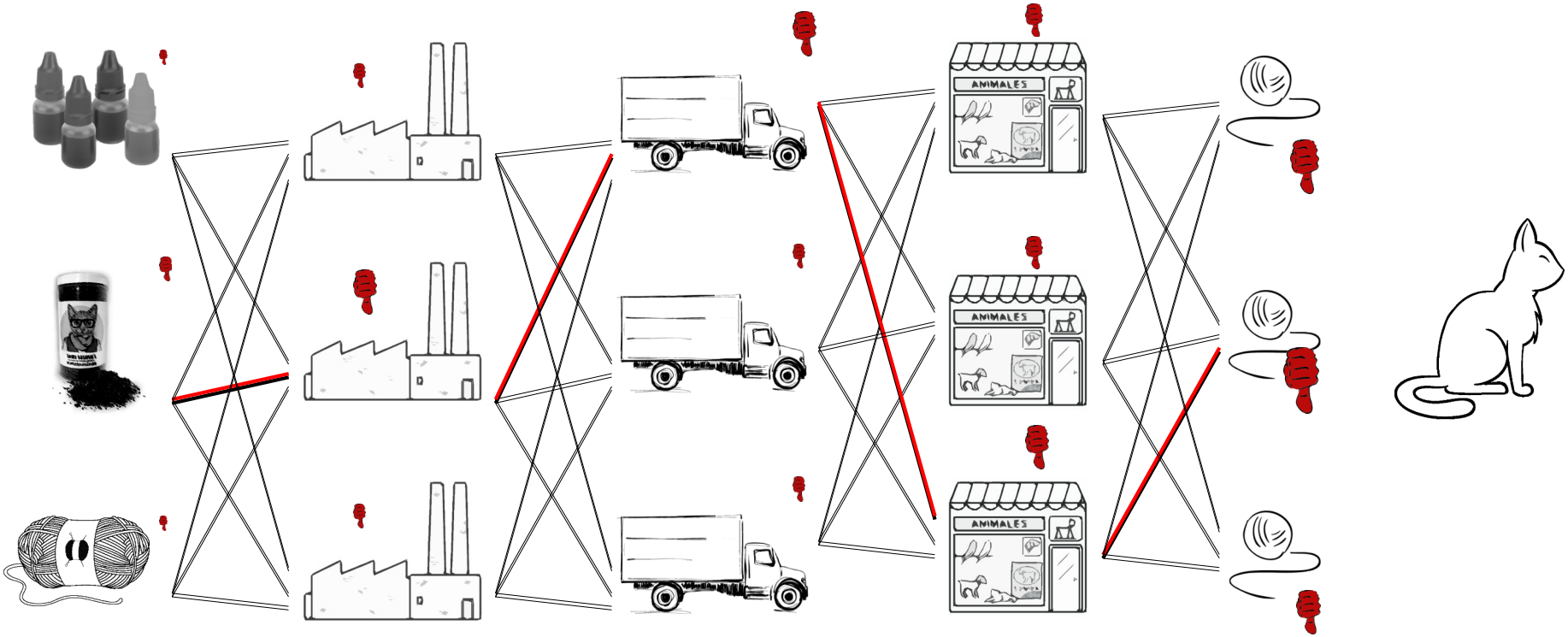
Backpropagation

Se trata de aplicar de forma recursiva capa tras capa, moviendo el error hacia atrás hasta llegar a la primera neurona y así saber cuáles fueron los errores cometidos en cada neurona de nuestra red.



Backpropagation

Tras saber los errores cometidos se puede aplicar el Gradient Descent para minimizar el error de cada neurona.



Backpropagation

Otro ejemplo que puede aclarar más la necesidad de usar el backpropagation es cualquier sistema jerárquico.

Una mala gestión de cualquier punto de una cadena de mando puede afectar al ingreso final de una empresa, en este caso es imperativo que haya reportes por parte de cada capa de mando para así saber quién hizo el qué y cuál es el peso de estas decisiones en el resultado final.

Así, se podran modificar el o los puntos afectados y reajustar la cadena de mandos.

Back Propagation = retropropagación de errores

O también conocido como el teorema de las derivadas de funciones compuestas.

Gradient descent: ejemplo completo de backpropagation

- *Ejemplo visual*:

https://hmkcode.com/ai/backpropagation-step-by-step/

Resumen

En 1969, Minsky y Papert, demuestran que el perceptrón simple no puede resolver problemas no lineales (por ejemplo, XOR).

La combinación de varios perceptrones simples podría resolver ciertos problemas no lineales pero no existía un mecanismo automático para adaptar los pesos de la capa oculta.

Rumelhart y otros autores, en 1986, presentan la "Regla Delta Generalizada" para adaptar los pesos propagando los errores hacia atrás, es decir, propagar los errores hacia las capas ocultas inferiores.

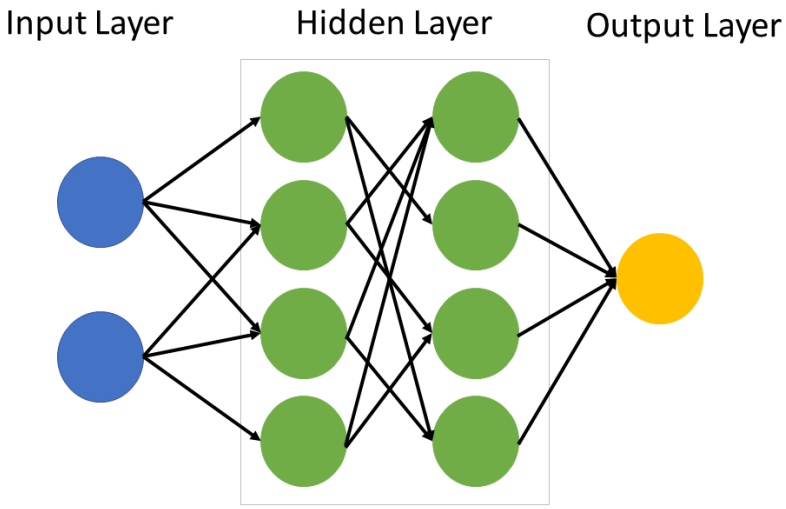
De esta forma se consigue trabajar con múltiples capas y con funciones de activación no lineales. Así, se demuestra que el perceptrón multicapa es un aproximador universal.

Un perceptrón multicapa puede aproximar relaciones no lineales entre los datos de entrada y salida. Esta red se ha convertido en una de las arquitecturas más utilizadas en el momento.

Ejercicio

Lo siguiente que haremos será un ejemplo de implementación de un perceptrón multicapa (multilayer perceptron).

Llevaremos a prueba la implementación para resolver dos tipos de problemas: uno de clasificación y otro de regresión.



Artificial Neural Networks