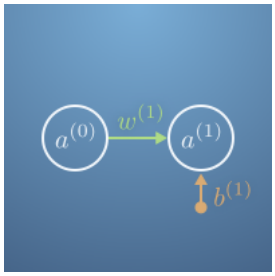
. Recuerde del video la estructura de una de las redes neuronales más simples, **1 / 1 punto**



Aquí solo hay dos neuronas (o nodos), y están unidas por un solo borde.

La *activación* de neuronas en la capa final, (1), está determinada por la activación de neuronas en la capa anterior, (0),

$a^{(1)} = s\,(\,w^{(1)}a^{(0)} + b^{(1)})$,

donde$en^{(1)}$es el *peso* de la conexión entre Neuron (0) y Neuron (1), y$b^{(1)}$es el sesgo de la Neurona (1). Estos están sujetos a la *función de activación* ,pagpara dar la activación de Neuron (1)

Nuestra pequeña red neuronal no podrá hacer mucho, es demasiado simple. Sin embargo, vale la pena ingresar algunos números para tener una idea de las partes.

Supongamos que queremos entrenar la red para dar una *función NOT* , es decir, si ingresa 1, devuelve 0, y si ingresa 0, devuelve 1.

Para simplificar, usemos,$\sigma\,(\,z\,) = \tanh\,(\,z\,)$, para nuestra función de activación, e inicializamos *aleatoriamente* nuestro peso y sesgo para$en^{(1)} = 1.3 y b^{(1)} = -0.1$.

Use el bloque de código a continuación para ver qué valores de salida devuelve inicialmente la red neuronal para los datos de entrenamiento.

```
# Primero establecemos el estado de la red
σ = np.tanh
w1 =  -5
b1 =  5

# Luego definimos la activación de la neurona.
def  a1(a0) :
  devuelve  σ(w1 * a0 + b1)

# ¡Finalmente, probemos la red!
# Reemplace x con 0 o 1 a continuación,
a1( 0 )

```

Ejecutar

Restablecer

¡No es muy bueno! Pero aún no está entrenado; experimente cambiando el peso y el sesgo y vea qué sucede.

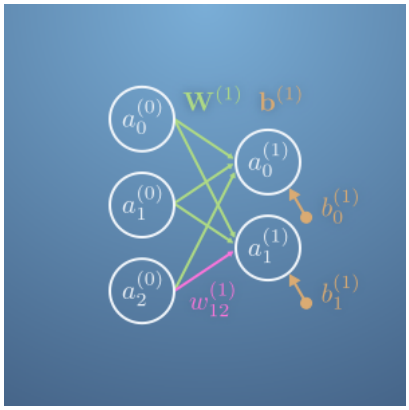
Elija el peso y el sesgo que dé el mejor resultado para una *función NOT* de todas las opciones presentadas.

- ○ $en^{(1)} = 0, b^{(1)} = 5$
- ○ $en^{(1)} = -3, b^{(1)} = 0$
- ○ $en^{(1)} = 10, b^{(1)} = 0$
- ◉ $en^{(1)} = -5, b^{(1)} = 5$
- ○ $en^{(1)} = 3, b^{(1)} = 1$

Correcto

Esta es la mejor de todas las opciones presentadas. Podemos hacerlo mejor con (por ejemplo) $en^{(1)} = -10, b^{(1)} = 10$.

2. Ampliemos nuestra red simple para incluir más neuronas. **1 / 1 punto**



Ahora tenemos una notación ligeramente modificada. Las neuronas que están etiquetadas por su capa con un superíndice entre paréntesis, ahora también están etiquetadas con su número en esa capa como un subíndice, y forman vectores $\mathbf{a}^{(0)}$ y $\mathbf{a}^{(1)}$.

Los pesos ahora forman una matriz, $\mathbf{En}^{(1)}$, donde cada elemento, $en_{yo\,j}^{(1)}$, es el enlace entre la neurona j en la capa anterior y neurona i en la capa actual. Por ejemplo $en_{12}^{(1)}$ se destaca el enlace $a_2^{(0)}$ a $a_1^{(1)}$.

Los sesgos forman de manera similar un vector $\mathbf{b}^{(1)}$.

Podemos actualizar nuestra función de activación para dar,

$$\mathbf{a}^{(1)} = s\,(\,\mathbf{W}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)}),$$

donde todas las cantidades de interés se han *actualizado* a su forma vectorial y matricial y pag actúa sobre cada elemento del vector de suma ponderada resultante por separado.

Para una red con pesos, $\mathbf{En}^{(1)} = \begin{bmatrix} -2 & 4 & -1 \\ 6 & 0 & -3 \end{bmatrix}$ y sesgo $\mathbf{b} = \begin{bmatrix} 0.1 \\ -2.5 \end{bmatrix}$,

calcular la salida, $\mathbf{a}^{(1)}$, dado un vector de entrada,

$$\mathbf{a}^{(0)} = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$$

Puede hacer este cálculo a mano (con 2 decimales) o escribiendo código python. Ingrese su respuesta en el bloque de código a continuación.

(Si elige codificar, recuerde que puede usar el operador @ en Python para operar una matriz en un vector).

```
# Primero configure la red.
sigma = np.tanh
W = np.arreglo([[ -2 ,  4 ,  -1 ],[ 6 ,  0 ,  -3 ]])
b = np.matriz([ 0.1 ,  -2.5 ])

# Definir nuestro vector de entrada
x = np.matriz([ 0.3 ,  0.4 ,  0.1 ])

# Calcular los valores a mano,
# y reemplace a1_0 y a1_1 aquí (a 2 decimales )
# (¡O si te sientes aventurero, encuentra los valores con el código!)
a1 = np.matriz([ 0.76159416 ,  -0.76159416 ])

```

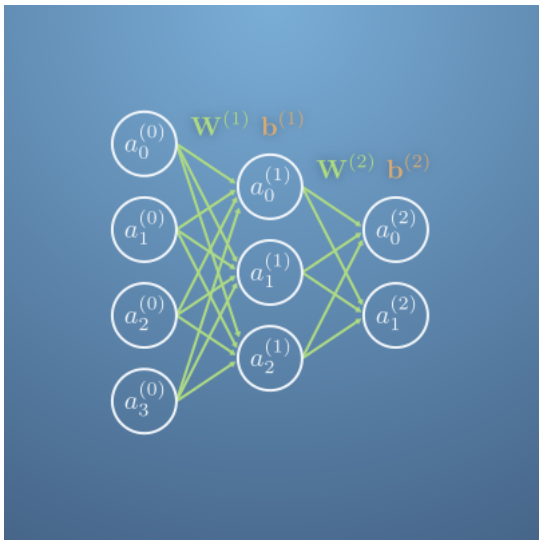
Ejecutar

Restablecer

Correcto

Bien hecho. ¿Por qué no intentarlo de nuevo usando el otro método?

3. Ahora veamos una red con una *capa oculta* . **1 / 1 punto**



Aquí, los datos se ingresan en la capa (0), esto activa las neuronas en la capa (1), que se convierten en las entradas para las neuronas en la capa (2).

(Hemos dejado de dibujar explícitamente los sesgos aquí).

¿Cuáles de las siguientes afirmaciones son verdaderas?

- [ ] Ninguna de las otras declaraciones.
- [ ] Esta red neuronal tiene 9 sesgos.
- [ ] Esta red siempre puede ser reemplazada por otra con la misma cantidad de neuronas de entrada y salida, pero sin capas ocultas.
- [x] Esta red neuronal tiene 5 sesgos.

Correcto
En general hay tantos sesgos como neuronas de salida y ocultas.

- [x] El número de pesos en una capa es el producto de las neuronas de entrada y salida a esa capa.

Correcto
Esto nos da 12 pesos en la primera capa y 6 pesos en la segunda.

- [ ] El número de pesos en una capa es la suma de las neuronas de entrada y salida de esa capa más 1.

4. ¿Cuáles de las siguientes afirmaciones sobre la red neuronal de la pregunta anterior son verdaderas? **1 / 1 punto**

- [x] $\mathbf{a}^{(2)} = s\,(\,\mathbf{W}^{(2)} s\,(\,\mathbf{W}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})$,

Correcto
De esta forma, se muestra la función completa de la red neuronal, con cada capa encadenada. Esto es lo mismo que,

$\mathbf{a}^{(2)} = s\,(\,\mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)})$

$\mathbf{a}^{(1)} = s\,(\,\mathbf{W}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)})$.

pero en una sola declaración.

- [ ] $\mathbf{a}^{(2)} = s\,(\,\mathbf{W}^{(2)}\mathbf{En}^{(1)}\mathbf{a}^{(0)} + \mathbf{En}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)})$

- [ ] Ninguna de las otras declaraciones.

- [ ] $\mathbf{a}^{(2)} = s\,(\,\mathbf{W}^{(1)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)})$,