{desafío}
latam_

Arquitectura mínima de una Red Neuronal _

Motivación

¿Por qué?

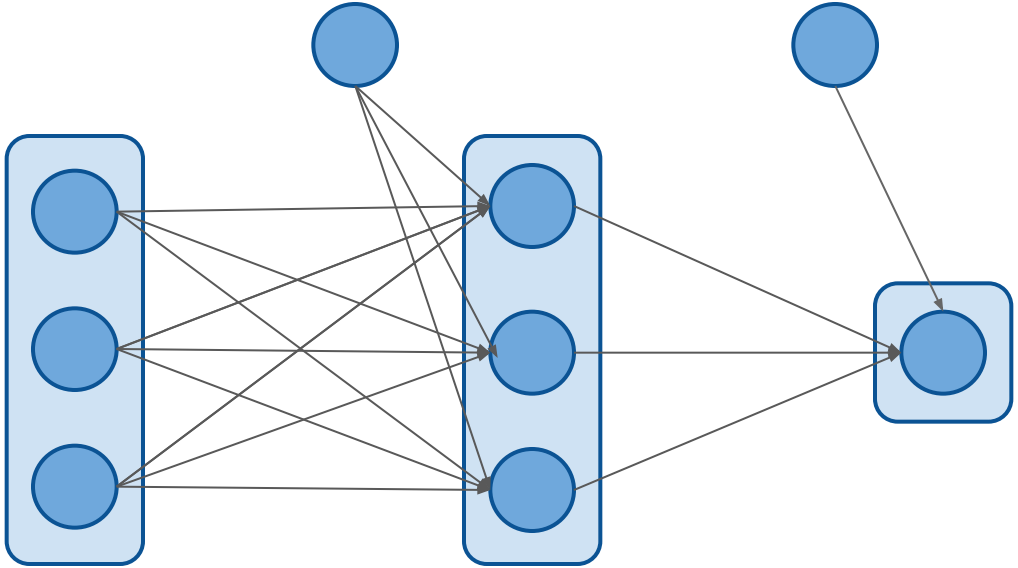
- Hasta el momento hemos trabajado con capas de ingreso y egreso.
- Mediante la inclusión de capas ocultas logramos flexibilizar la cantidad de atributos a inferir
- Ventajas de la implementación de capas ocultas:
  - Son aproximadores universales.
  - Permiten generar representaciones distribuídas sobre los datos.
- Esto se debe a que una capa puede contener múltiples neuronas que representen distintos atributos.

Elementos conformantes de una Red Neuronal

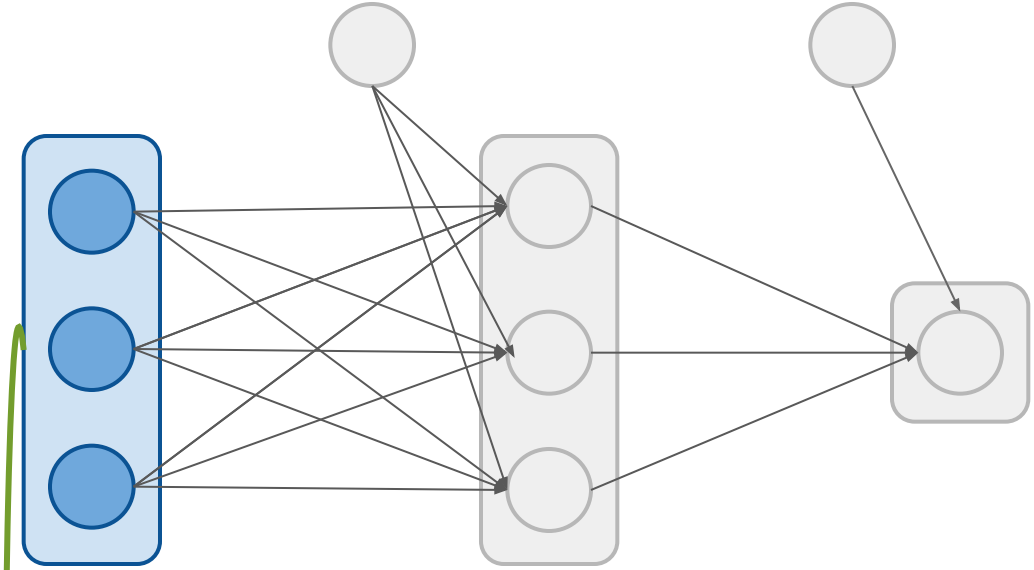
Terminología

- **Neuronas:** Elemento base de una red. Recibe un impulso y devuelve un output procesado por una función de activación
- **Capas:** Un ensamble de neuronas se conoce como capas. Éstas permiten definir cierta secuencialidad en el flujo.
- **Función de activación:** Permiten traducir el output de una neurona/capa.
- **Modelos:** Las capas y neuronas se ensamblan dentro de modelos, que definen el flujo de conexión.
- **Funciones de pérdida/optimización:** Definimos el proceso de optimización a nivel de modelo.

Elementos conformantes

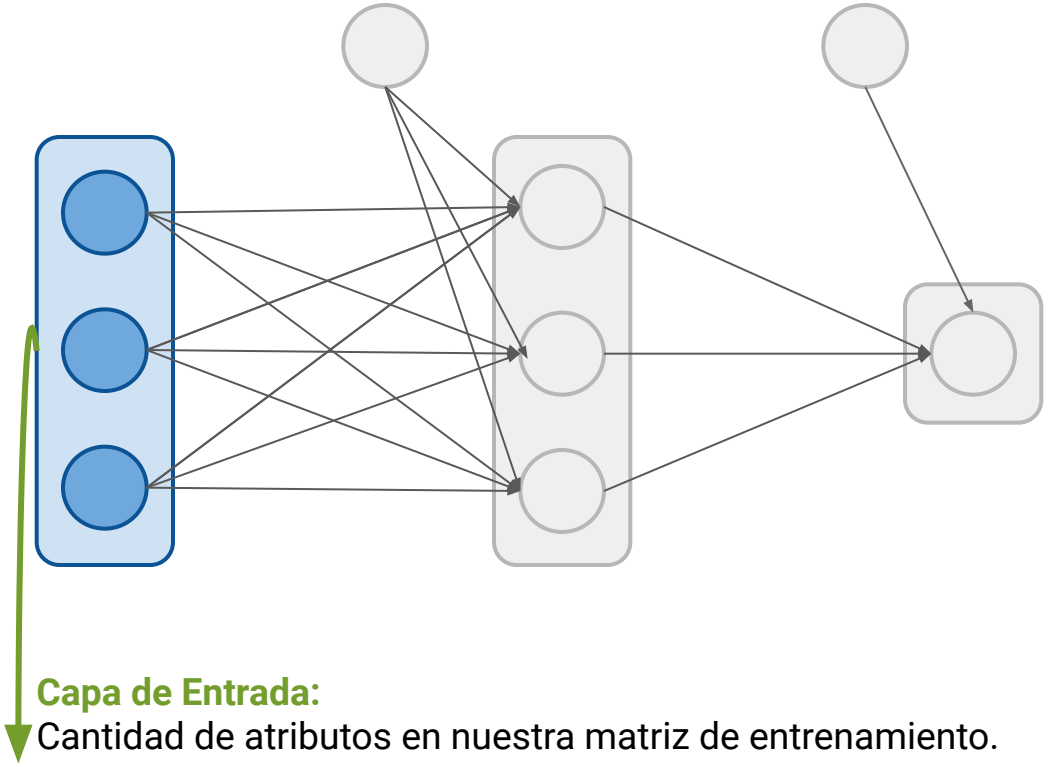


Elementos conformantes



Capa de Entrada:
Cantidad de atributos en nuestra matriz de entrenamiento.

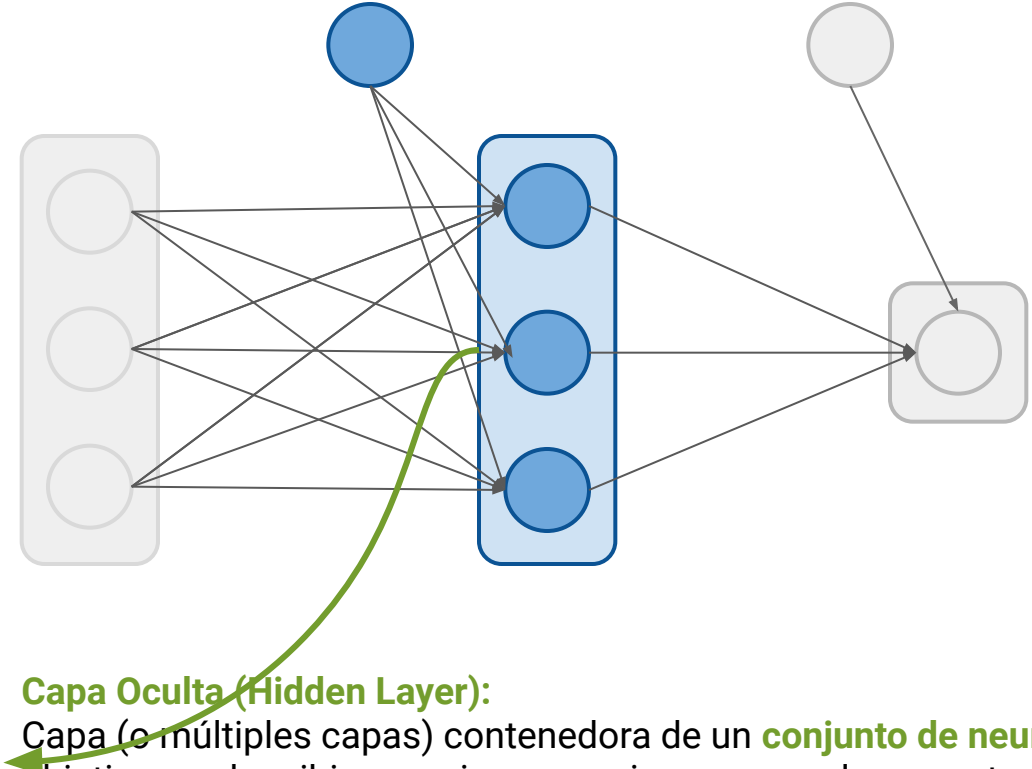
Elementos conformantes



Capa de Entrada:
Cantidad de atributos en nuestra matriz de entrenamiento.

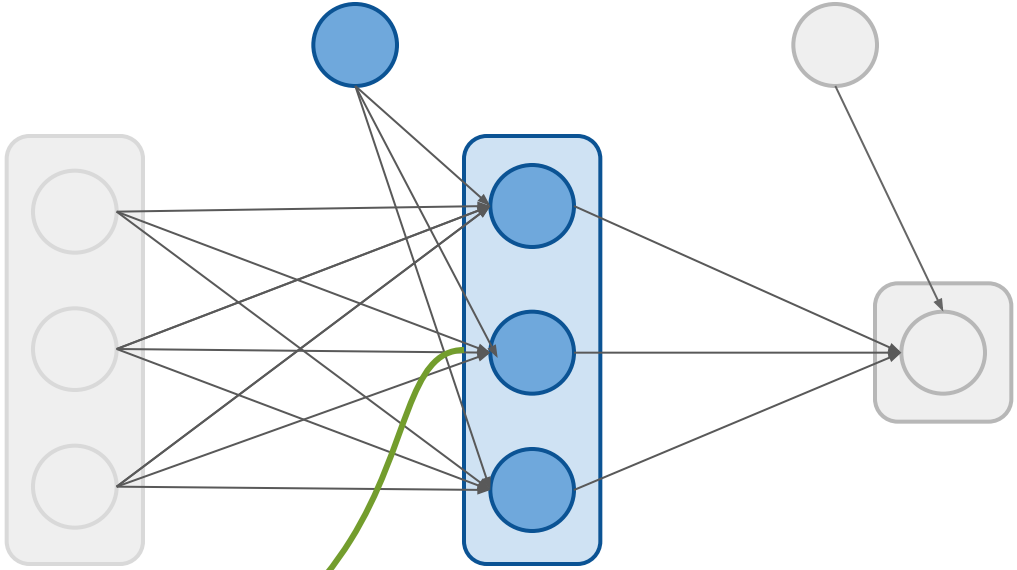
Por lo general declaramos tantas neuronas de entrada como atributos existan.

Elementos conformantes



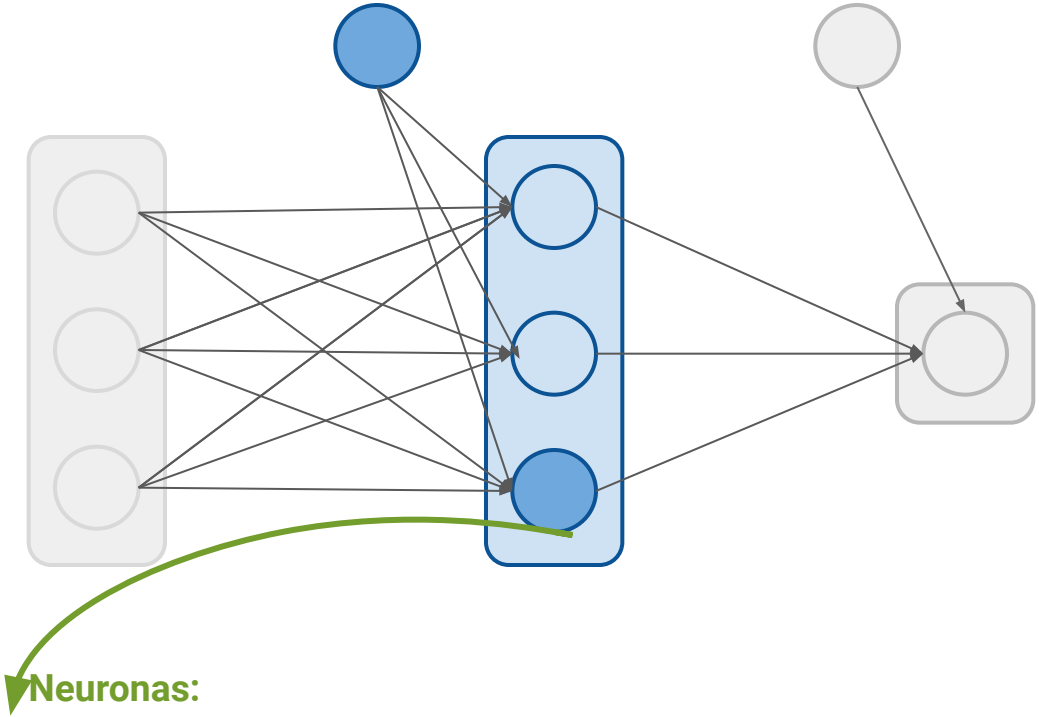
Capa Oculta (Hidden Layer):
Capa (o múltiples capas) contenedora de un **conjunto de neuronas**, cuyo objetivo es el recibir conexiones previas, procesarlas y posteriormente emitir un impulso que puede ser capturado por otra capa oculta o una capa de salida.

Elementos conformantes



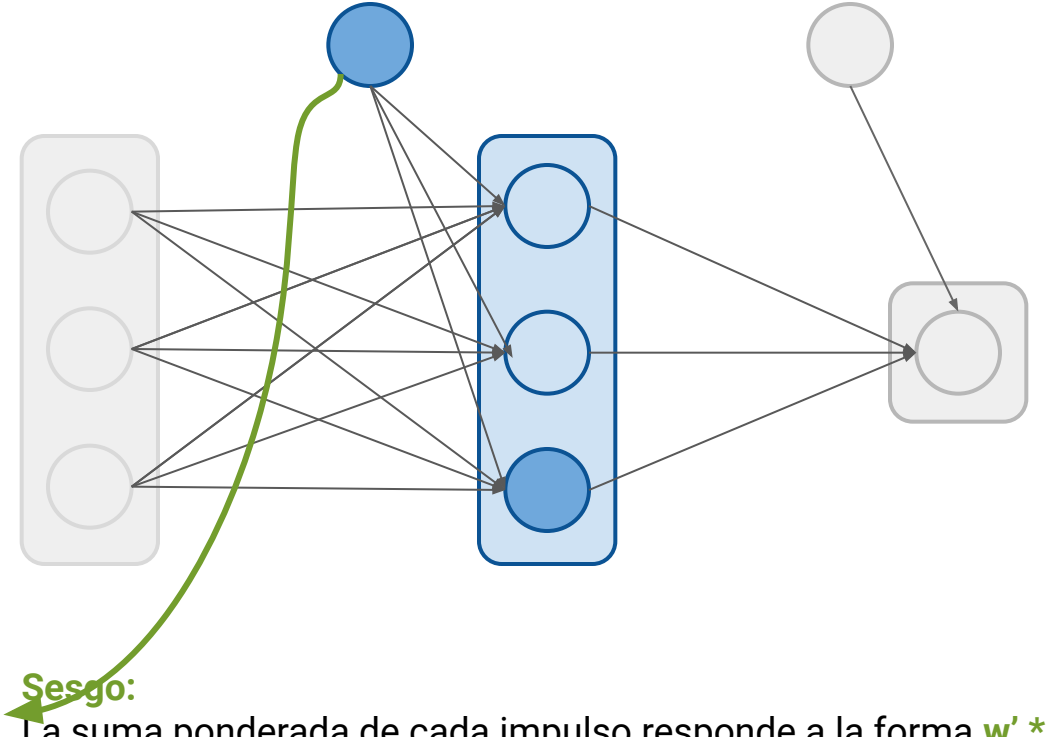
Capa Oculta (Hidden Layer):
La cantidad de neuronas alojadas en una capa permite representar un fenómeno:
En la medida que aumentamos las neuronas, incorporamos más información sobre la cantidad de conexiones entre los impulsos.
En la medida que disminuímos las neuronas, reducimos la información capturada.

Elementos conformantes



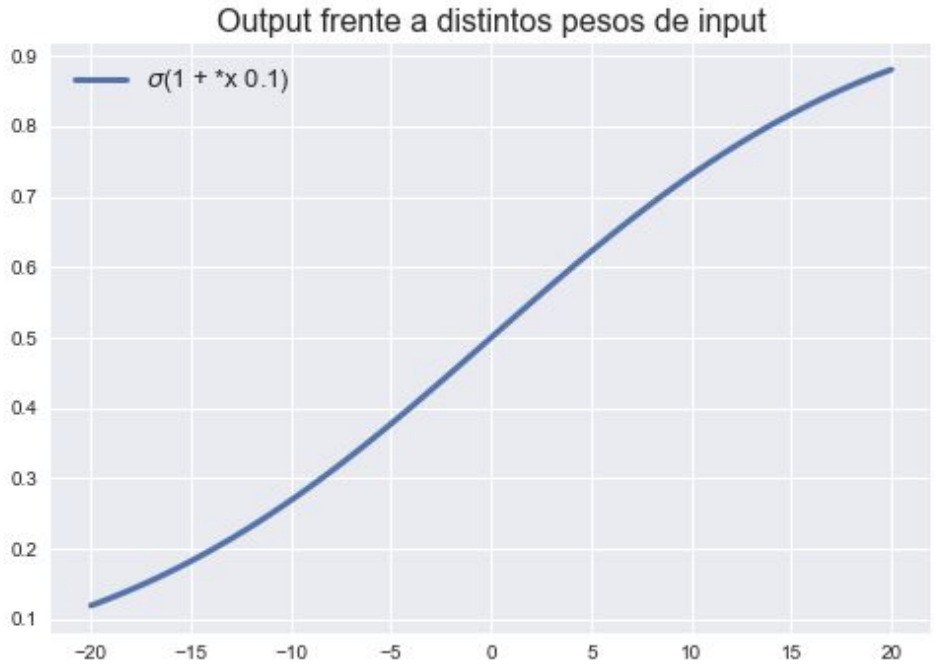
Neuronas:
Elemento basal. A grandes rasgos opera como un perceptrón.
Elementos conformantes:
Suma ponderada.
Función de activación

Elementos conformantes

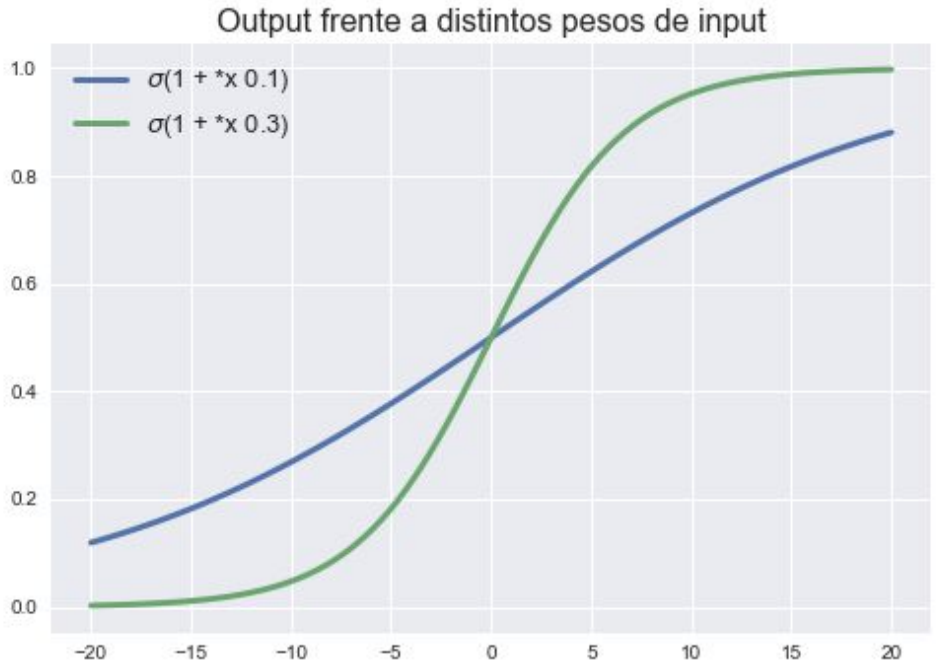


Sesgo:
La suma ponderada de cada impulso responde a la forma **w' * x + b**. El sesgo es una neurona declarada a nivel de capa que afecta el procesamiento del impulso por igual a cada neurona.

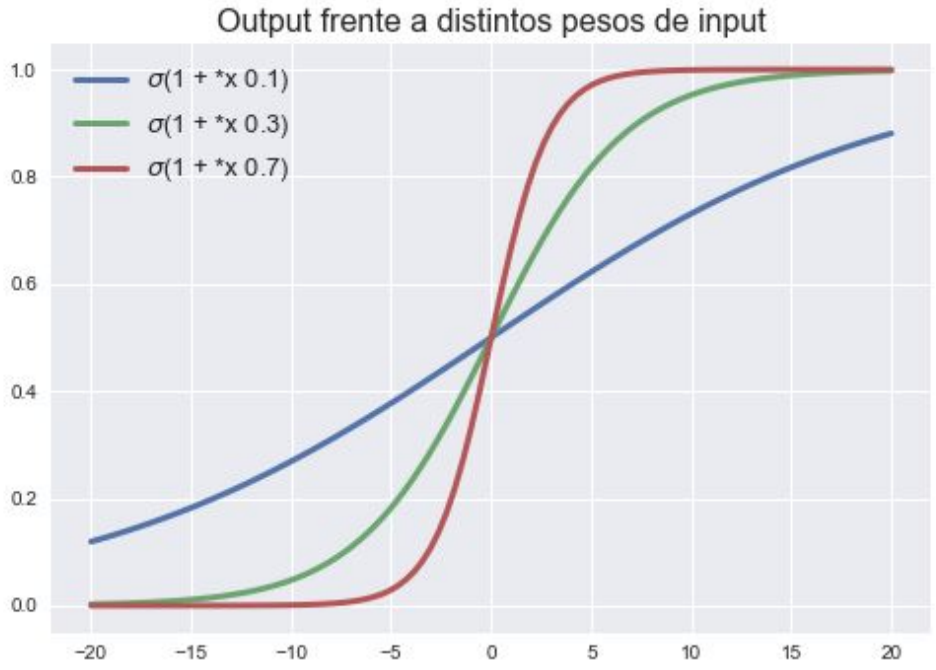
Pesos y Sesgos



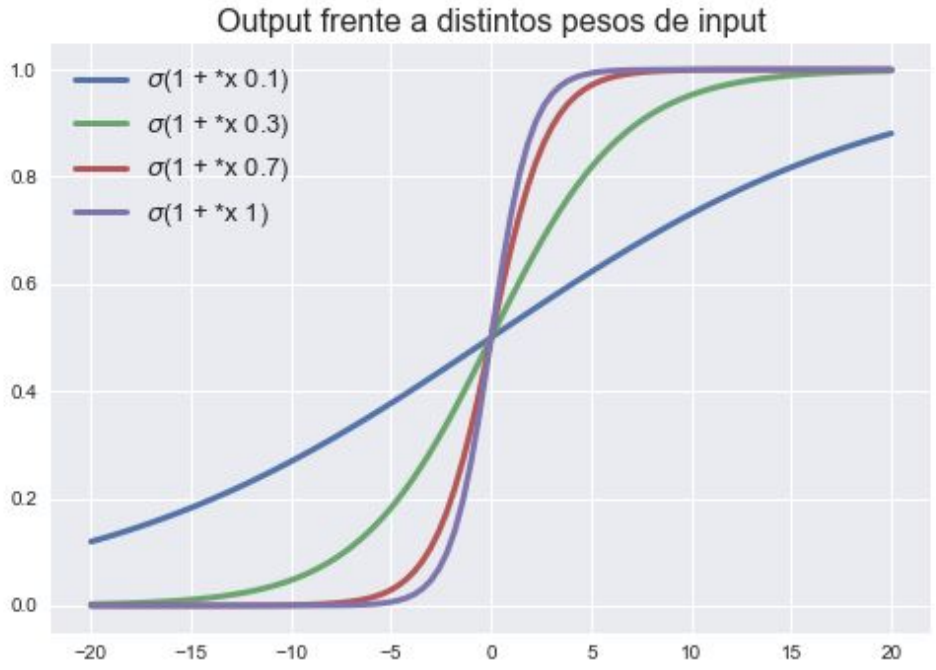
Pesos y Sesgos



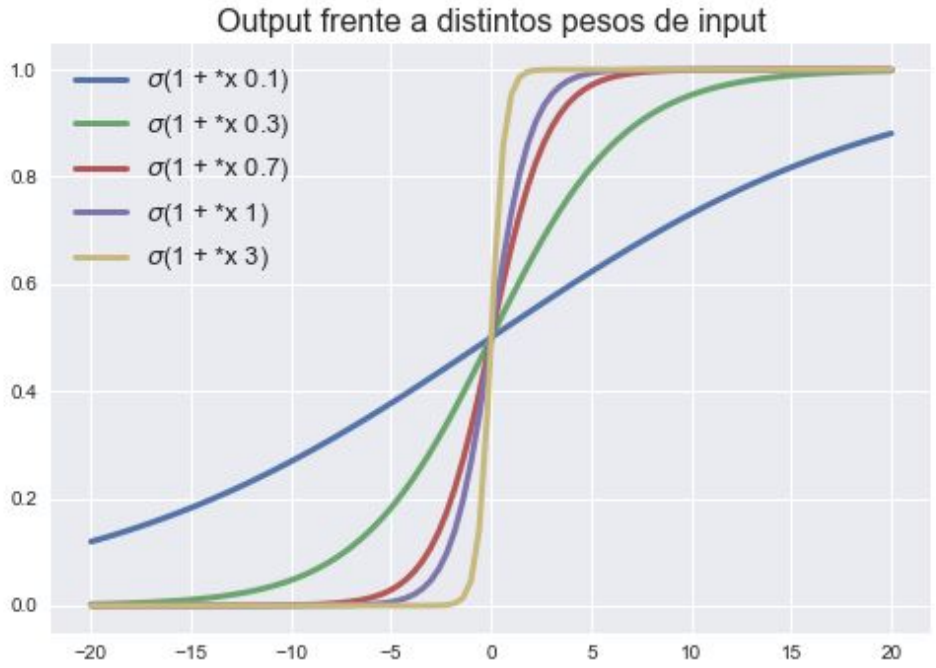
Pesos y Sesgos



Pesos y Sesgos

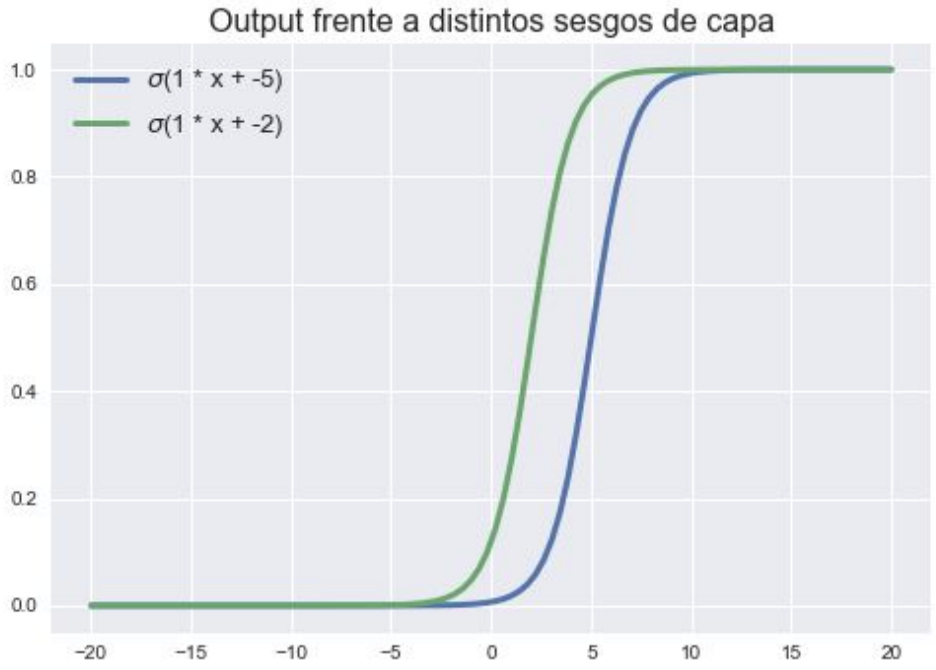


Pesos y Sesgos

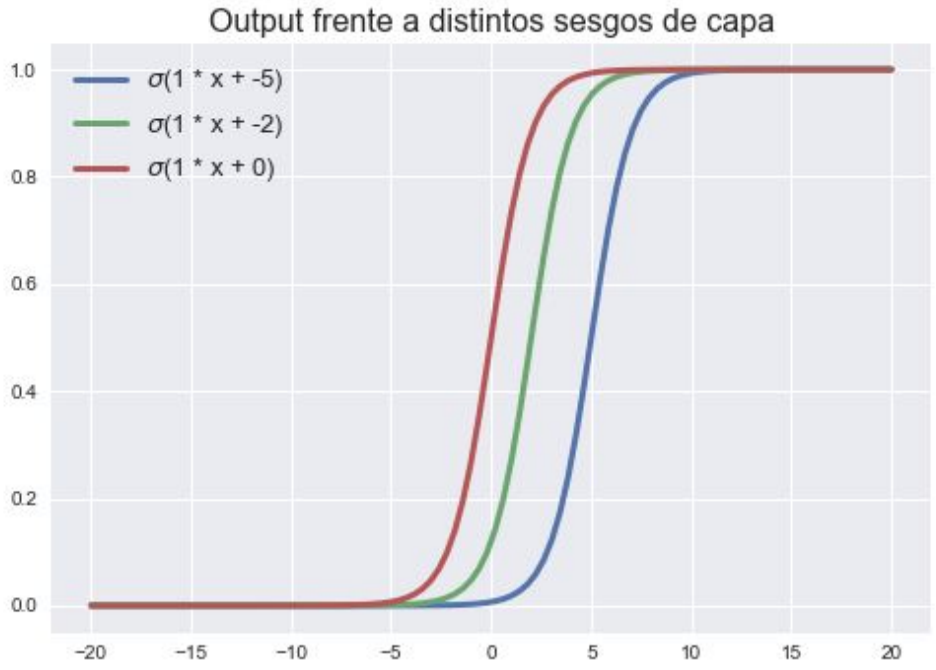


Pesos y Sesgos

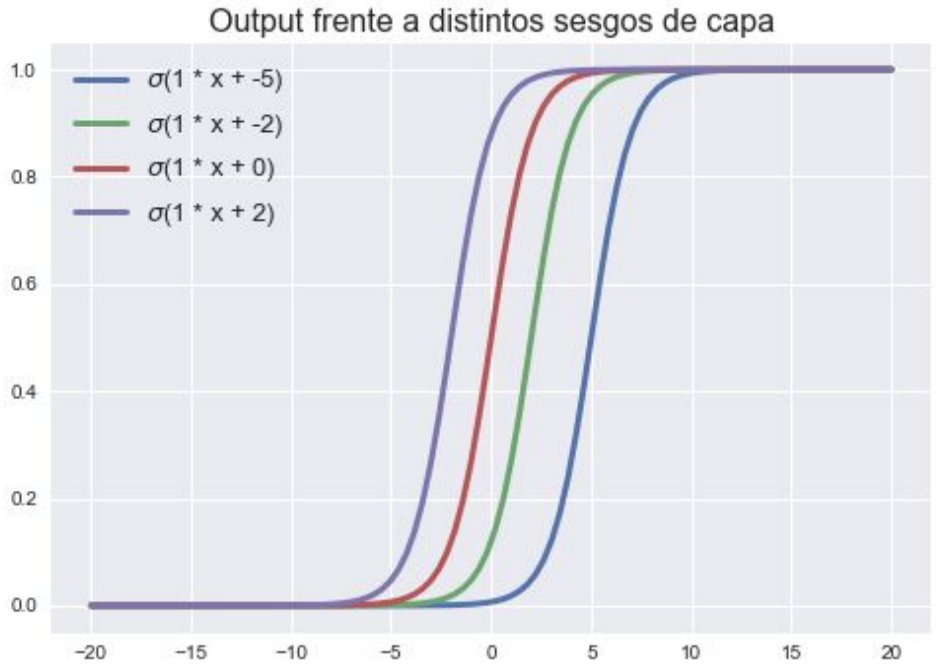
Pesos y Sesgos



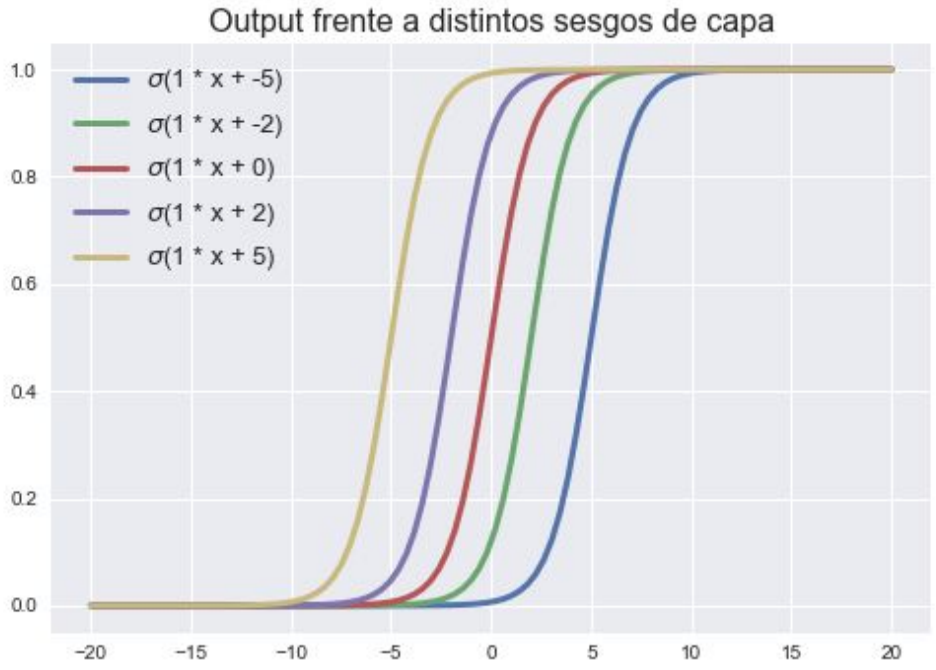
Pesos y Sesgos



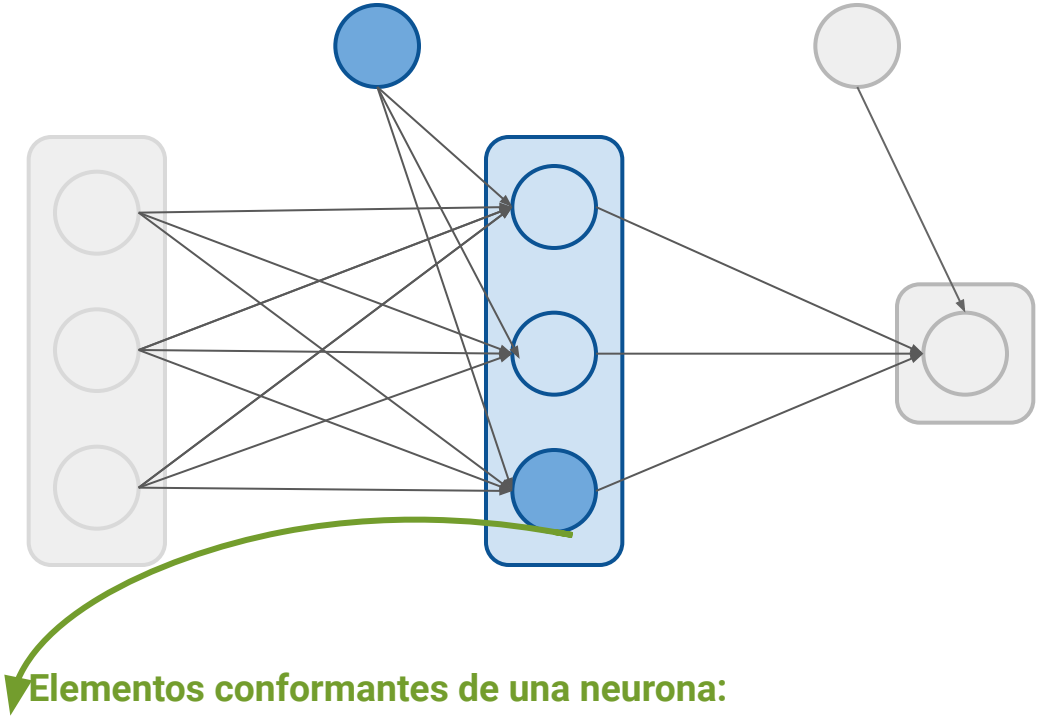
Pesos y Sesgos



Pesos y Sesgos

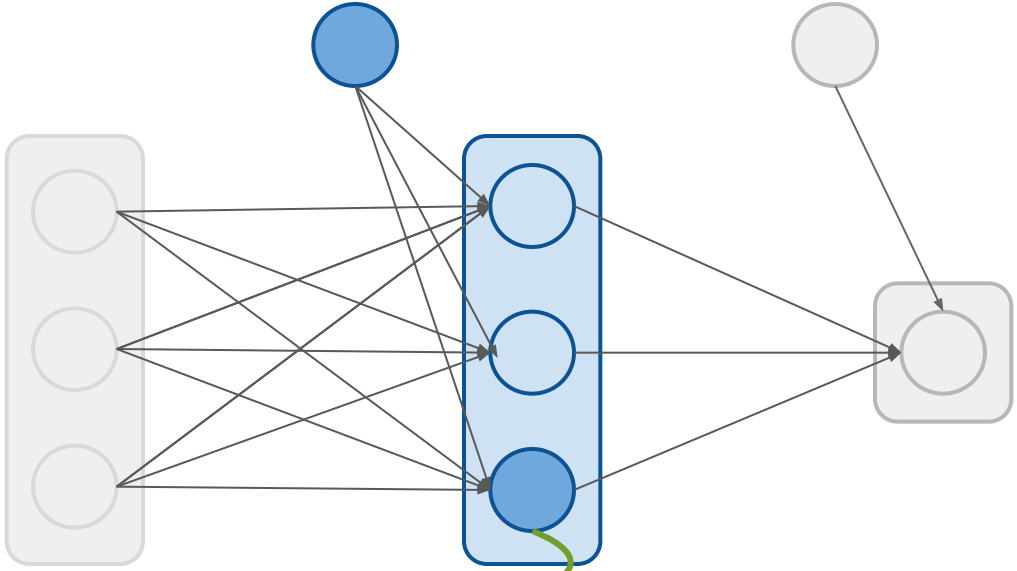


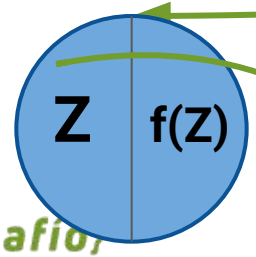
Elementos conformantes



Elementos conformantes de una neurona:
Inputs: Cantidad de impulsos en la capa previa.
Pesos: Coeficiente asociado a la neurona y su representación específica.
Función de activación: Reexpresión no-lineal de la suma ponderada

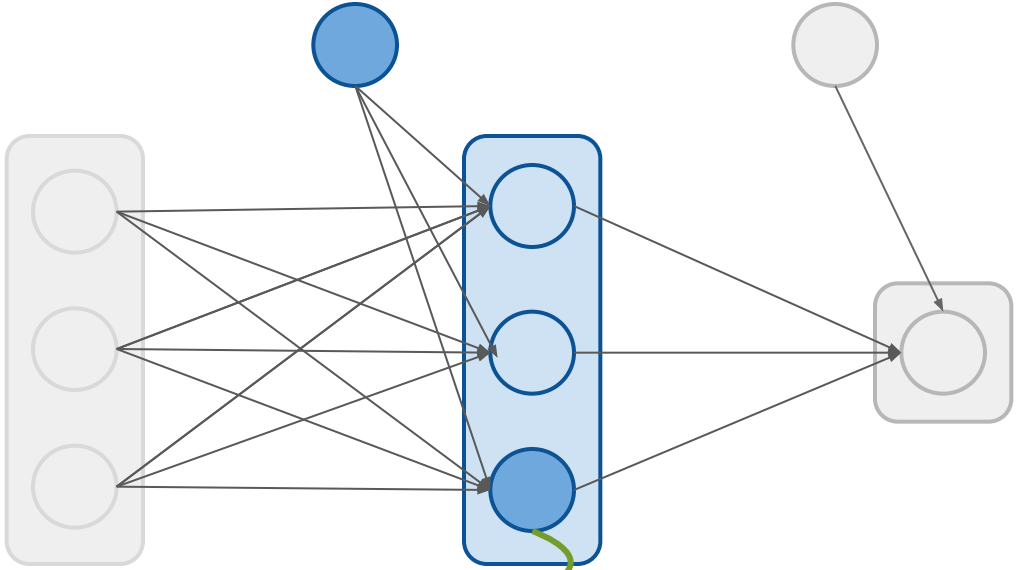
Elementos conformantes

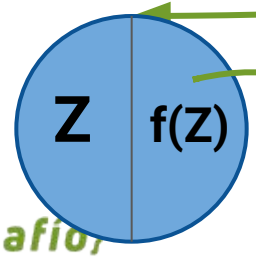




Suma ponderada de los inputs:
Ajusta los atributos del ejemplo de entrada por un peso específico definido mediante Backpropagation.

Elementos conformantes



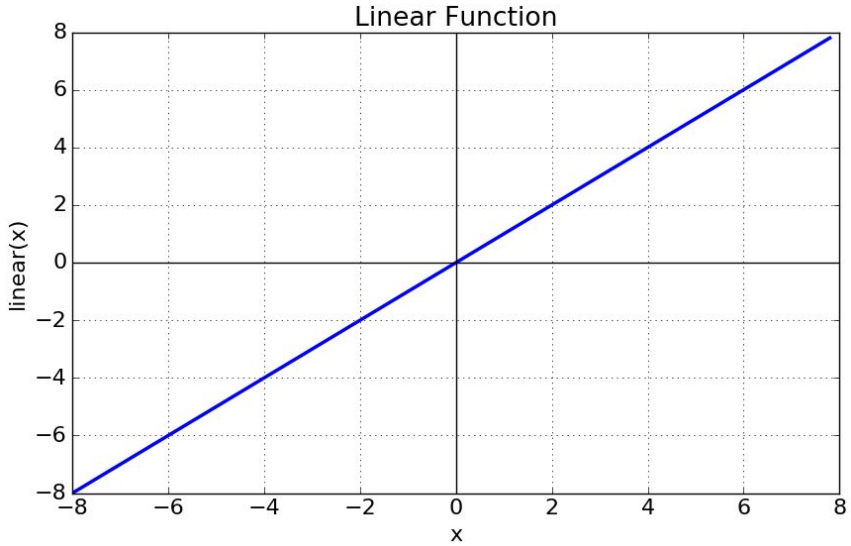


Función de Activación:
Reexpresa la suma ponderada en un espacio nolineal

Funciones de Activación

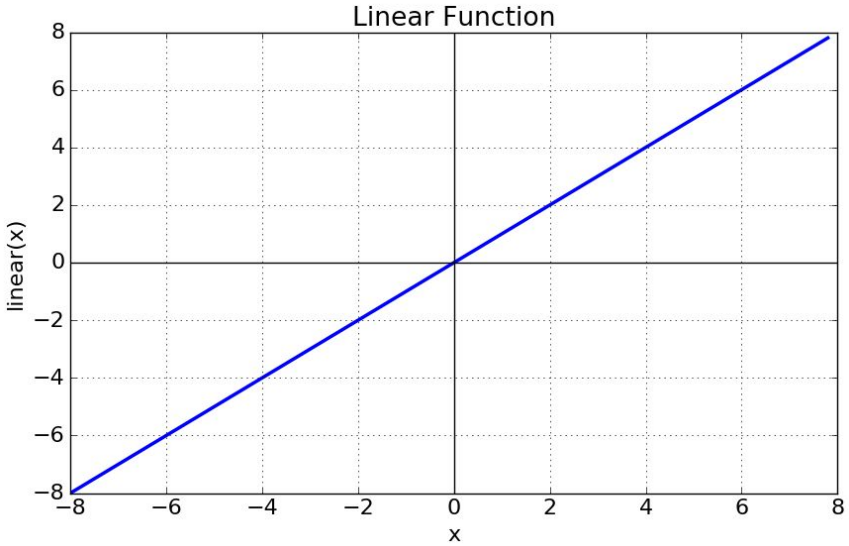
$$\text{Lineal} = \mathbf{w}^T \cdot \mathbf{x}$$

- Podemos implementar una variante de la función signum del perceptrón.



Funciones de Activación

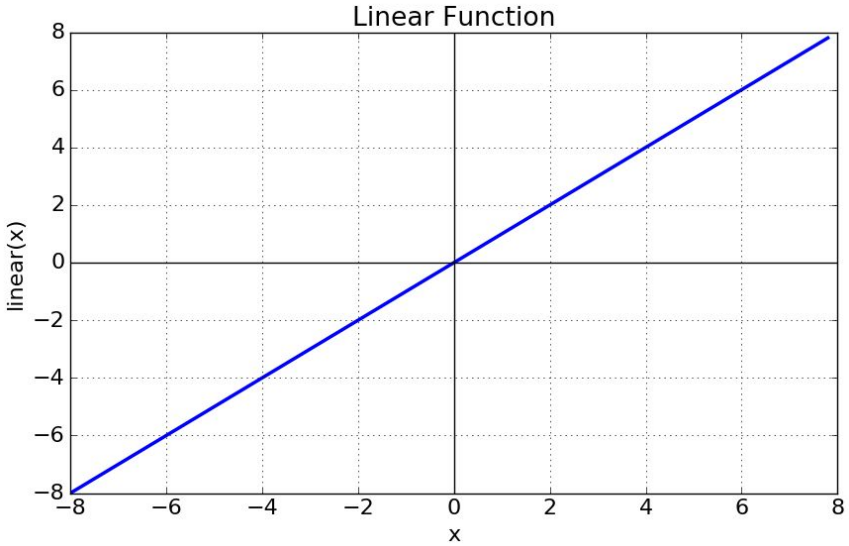
$$\text{Lineal} = \mathbf{w}^T \cdot \mathbf{x}$$



- Podemos implementar una variante de la función signum del perceptrón.
- En este contexto, la activación no será discreta al valor o signo del input.

Funciones de Activación

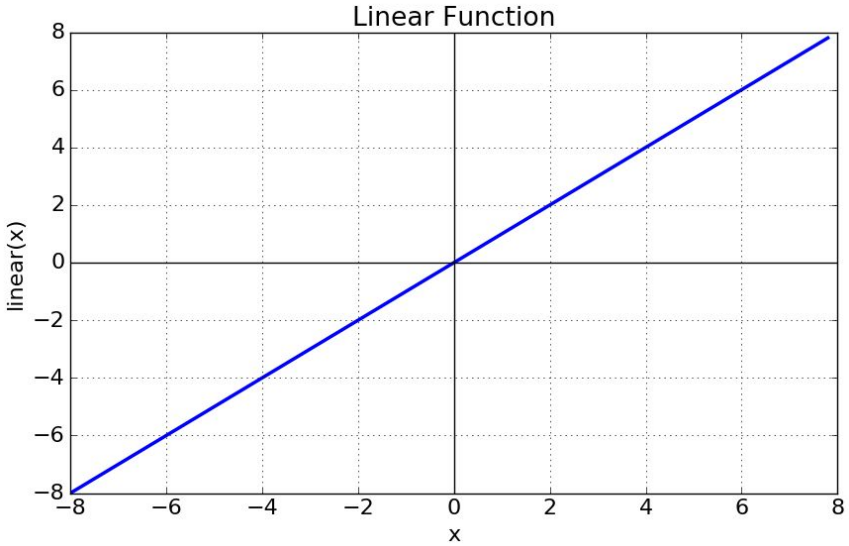
$$\text{Lineal} = \mathbf{w}^T \cdot \mathbf{x}$$



- Podemos implementar una variante de la función signum del perceptrón.
- En este contexto, la activación no será discreta al valor o signo del input.
- En la medida que el input aumente, la activación asociada incrementará proporcionalmente.

Funciones de Activación

$$\text{Lineal} = \mathbf{w}^T \cdot \mathbf{x}$$

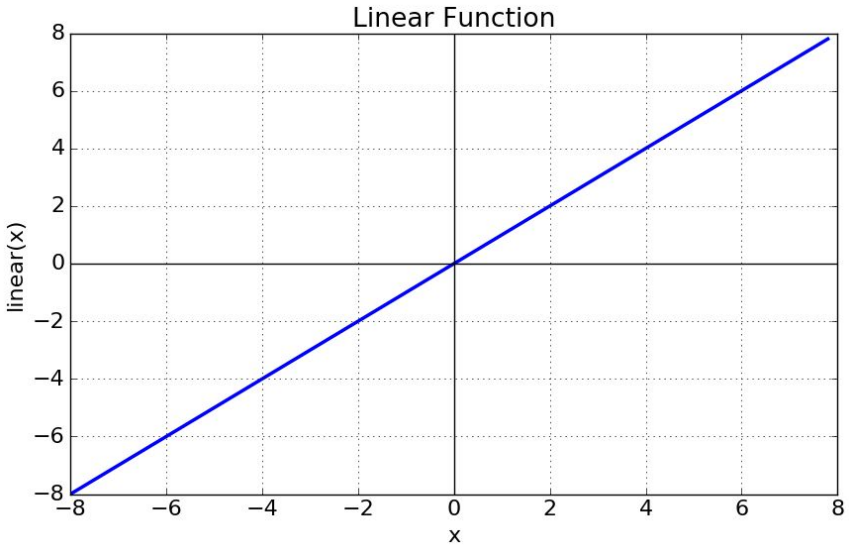


- Podemos implementar una variante de la función signum del perceptrón.
- En este contexto, la activación no será discreta al valor o signo del input.
- En la medida que el input aumente, la activación asociada incrementará proporcionalmente.

Problemas:

Funciones de Activación

$$\text{Lineal} = \mathbf{w}^{\mathsf{T}} \cdot \mathbf{x}$$



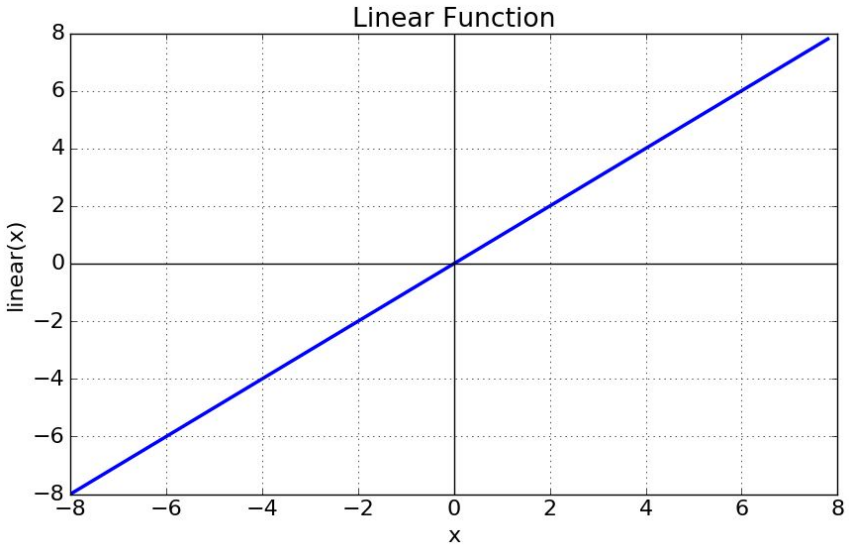
- Podemos implementar una variante de la función signum del perceptrón.
- En este contexto, la activación no será discreta al valor o signo del input.
- En la medida que el input aumente, la activación asociada incrementará proporcionalmente.

Problemas:

- Al final de cuentas no entrega una nueva reexpresión de los impulsos intermedios.

Funciones de Activación

$$\text{Lineal} = \mathbf{w}^T \cdot \mathbf{x}$$



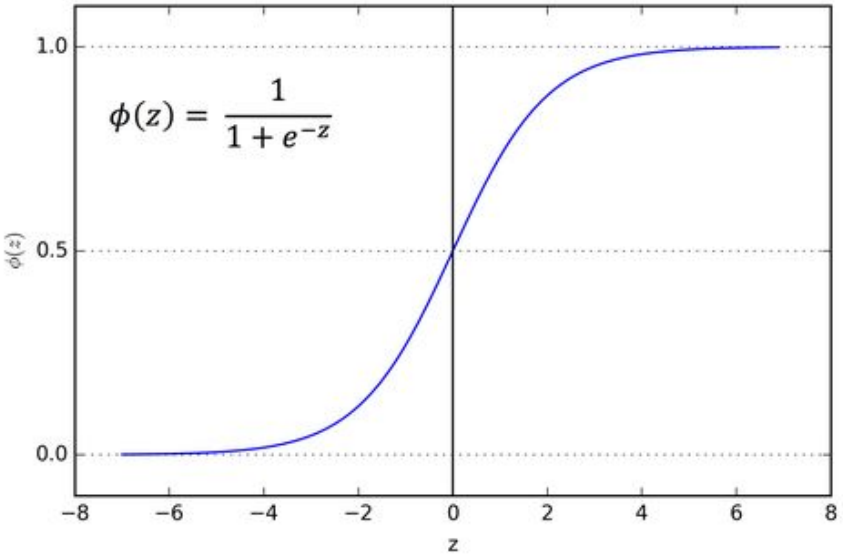
- Podemos implementar una variante de la función signum del perceptrón.
- En este contexto, la activación no será discreta al valor o signo del input.
- En la medida que el input aumente, la activación asociada incrementará proporcionalmente.

Problemas:

- Al final de cuentas no entrega una nueva reexpresión de los impulsos intermedios.
- No presenta límites definidos.

Funciones de Activación

$$\text{Sigmoide} = \frac{1}{1+\exp(-x)}$$



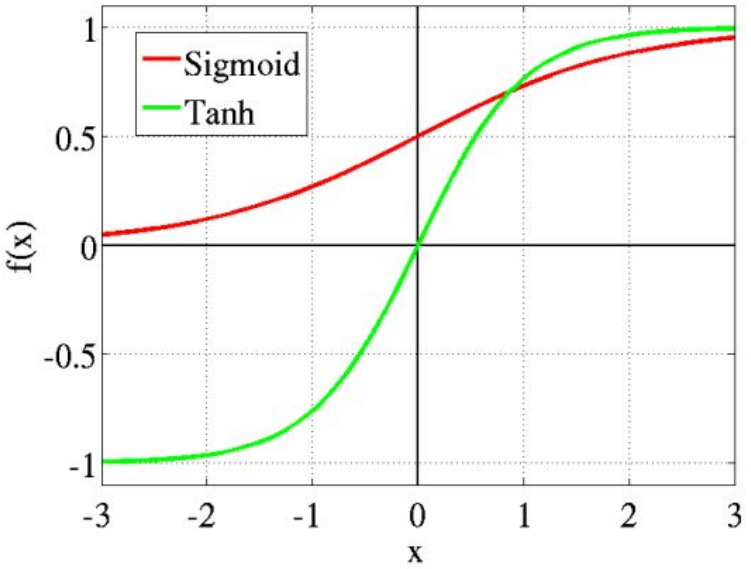
- Asociada a la función logística inversa: nos permite reexpresar un valor X entre 0 y 1.
- Es un híbrido entre una función de activación binaria, suavizada por la forma lineal.
- Reexpresa los impulsos entre 0 y 1, regularizando la influencia de los pesos.

Problemas:

- En los extremos de la función, los valores de mi eje Y tienden a ser inflexibles. Este es el problema de gradientes desvanecientes.

Funciones de Activación

$$\text{tahn} = \frac{-2}{1+\exp(-2x)} - 1$$

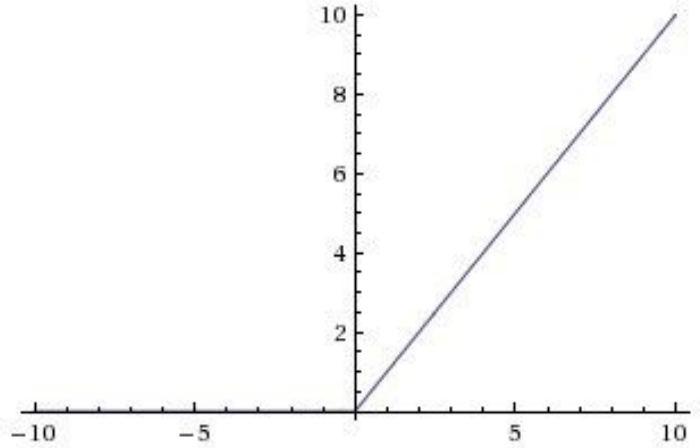


Función de Arco Tangente.

- Permite reexpresar un impulso entre -1 y 1.
- Una de las principales ventajas es que tiene una mayor resistencia al problema de los gradientes desvanecientes.

Funciones de Activación

$$\text{RELU} = \max(0, x)$$

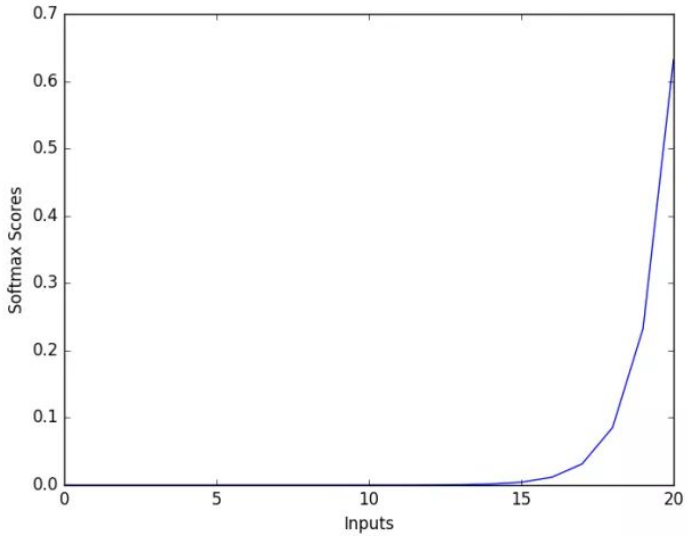


Función Lineal Unitaria Rectificada

- Permite reexpresar un input de manera lineal si es que satisface alguna condición de corte.
- Por lo general esta condición de corte se representa mediante el operador signum.
- Permiten capturar nolinealidades en los impulsos de entrada.
- Sirve como una forma bruta de regularización, forzando a la neurona a capturar señales sólo si son positivas.
- Esto es útil cuando estemos ante problemas de capas dispersas.

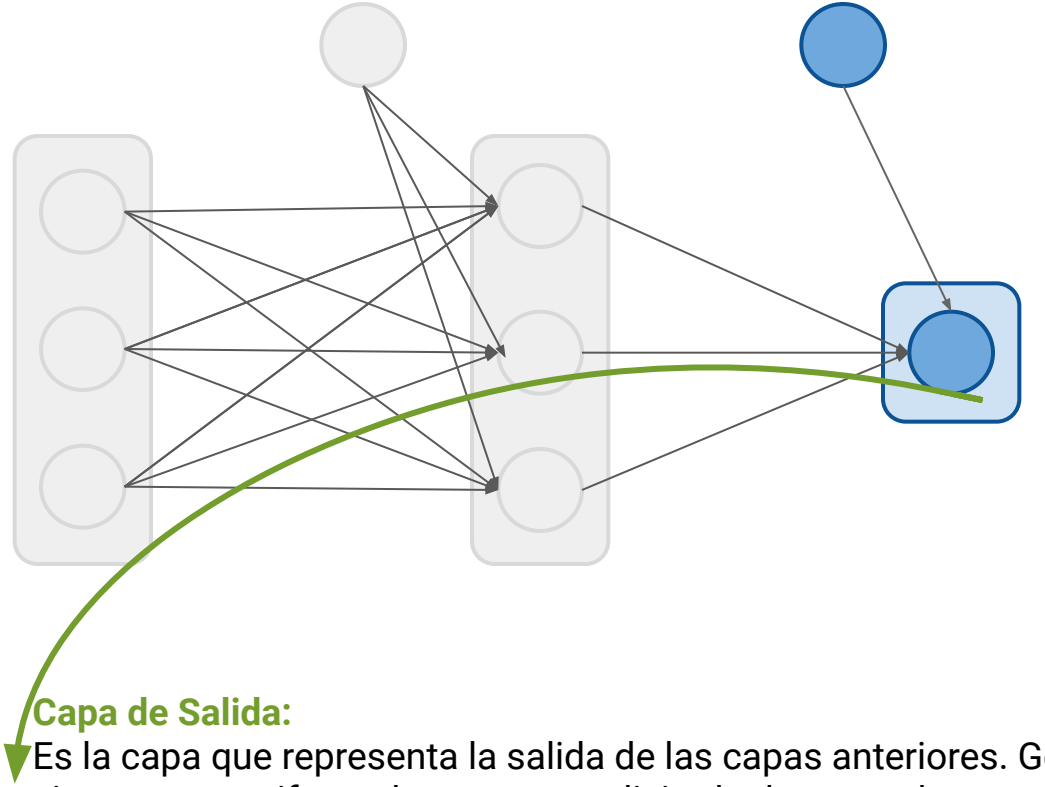
Funciones de Activación

$$\text{Softmax} = \frac{\exp(x)}{\sum \exp(x)}$$



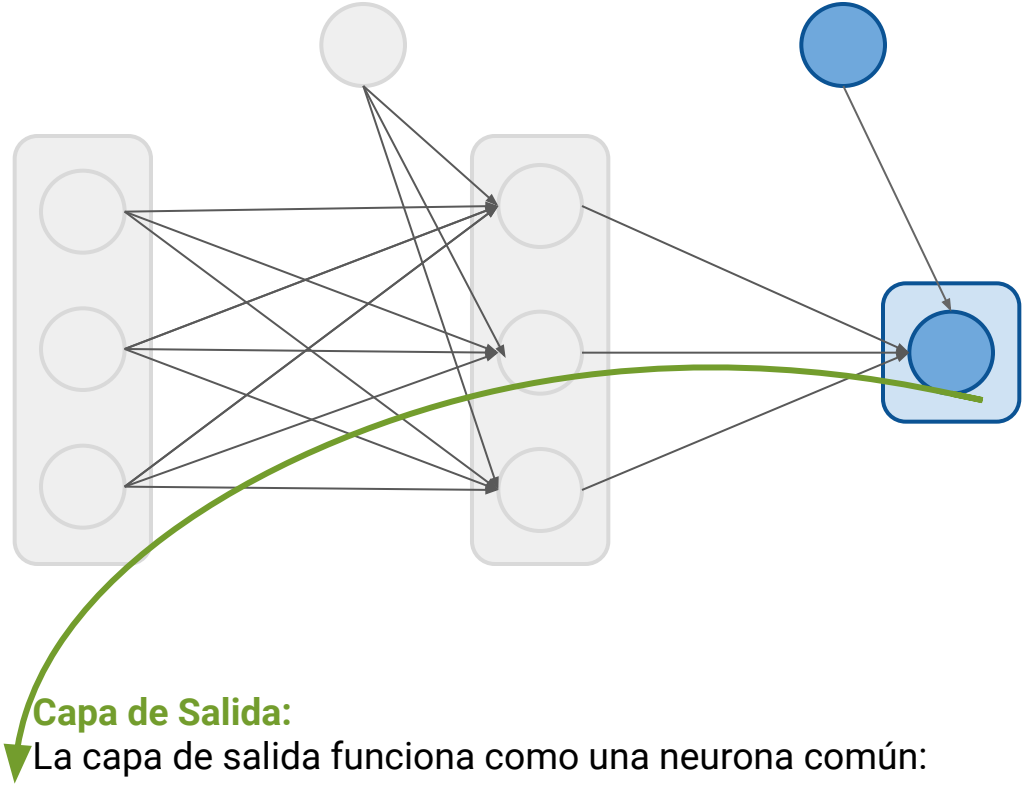
- También conocida como **activación multinomial.**
- Resuelve el problema de más de 2 clases que la activación Sigmoide no logra.
- Asegura que las probabilidades de cada clase sumen 1.

Elementos conformantes



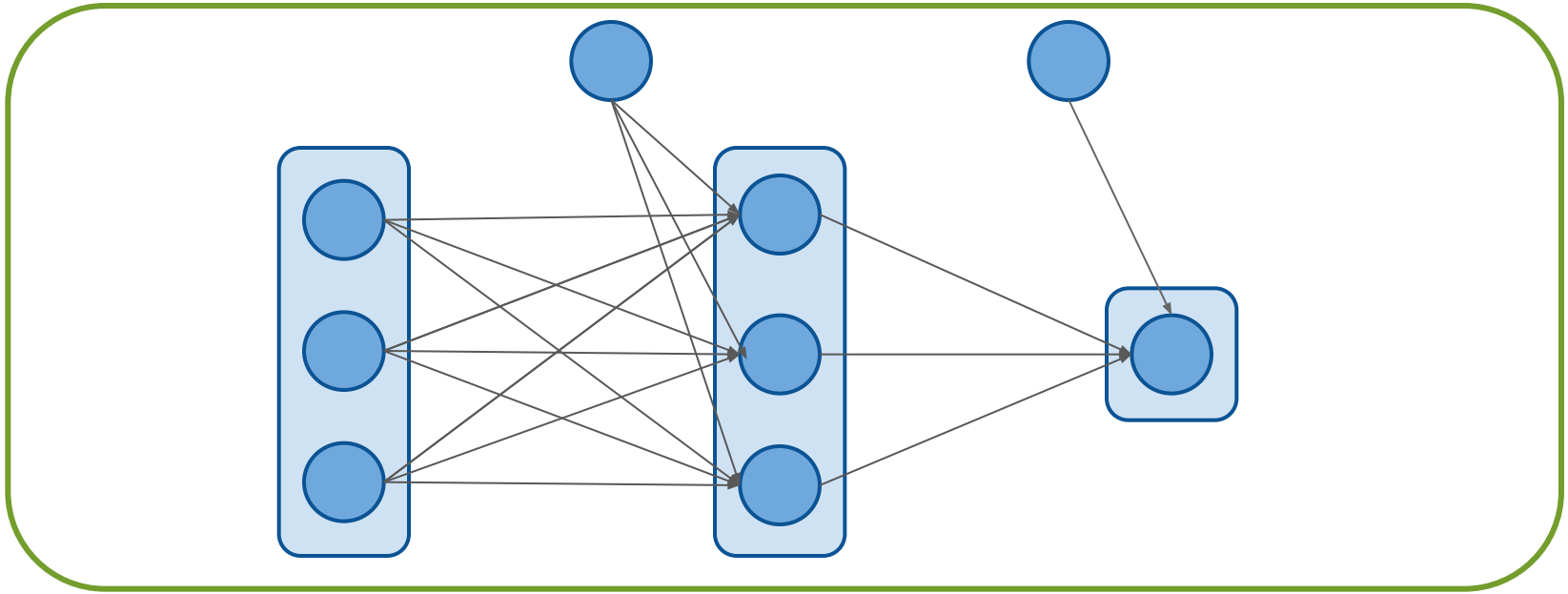
Capa de Salida:
Es la capa que representa la salida de las capas anteriores. Generalmente sirve para manifestar la respuesta elicitada de una red neuronal.

Elementos conformantes



Capa de Salida:
La capa de salida funciona como una neurona común:
Debe contener una neurona que recolecte los impulsos.
Y debe poseer una función de activación que permita reexpresar los resultados.

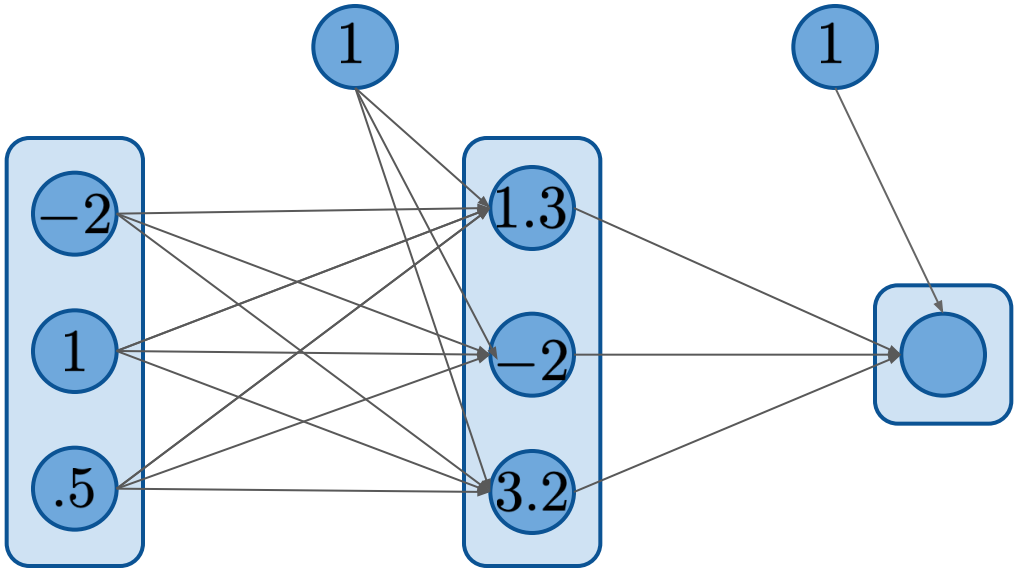
Elementos conformantes



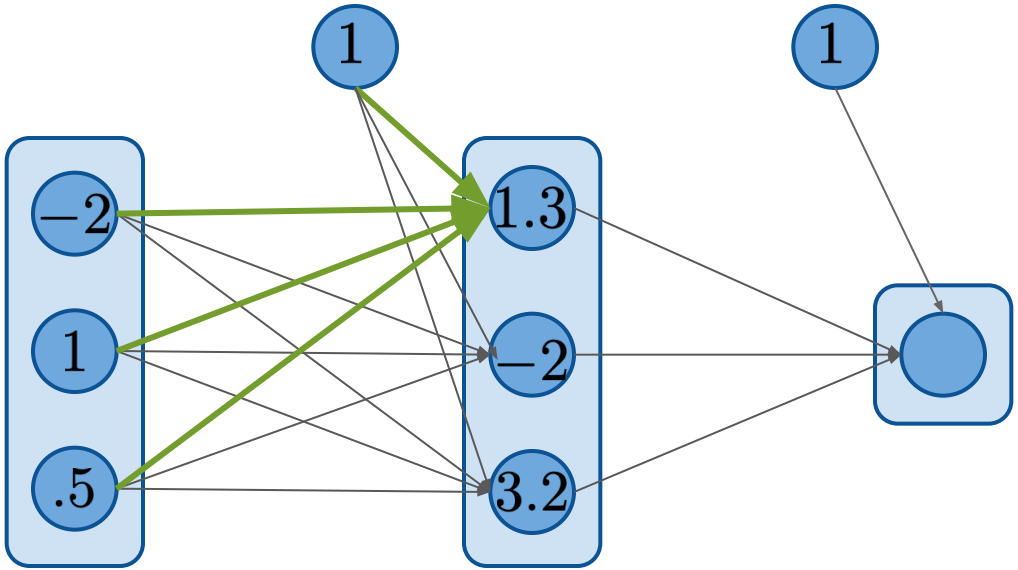
Agenda de entrenamiento y optimización:
Tipo de optimizador: Algún método basado en Gradiente Estocástico.
Épocas: Debemos definir la cantidad de ciclos (o iteraciones) con las cuales vamos a entrenar nuestro modelo y calibrar los pesos.

El proceso de decisión en una Red Neuronal

El flujo de un ejemplo en una RNA: Setup

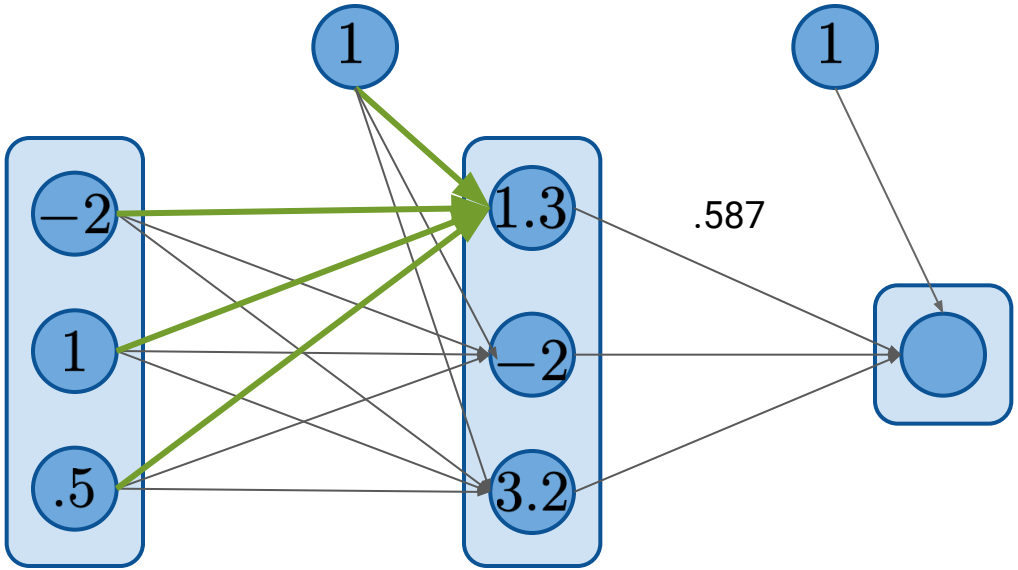


El flujo de un ejemplo en una RNA: Primer peso



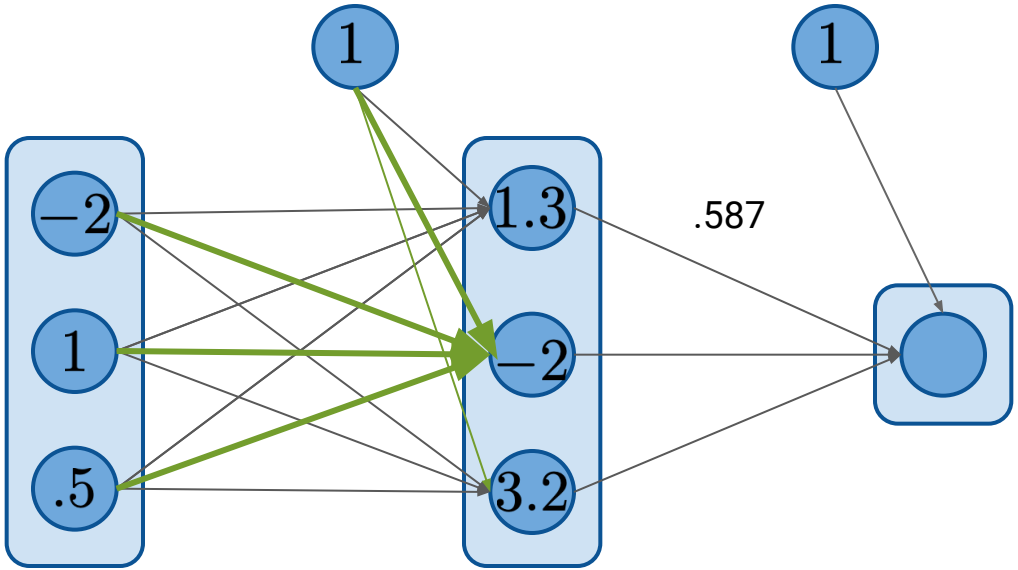
$$z_1 = \sigma\Big((-2 \cdot 1.3) + (1 \cdot 1.3) + (0.5 \cdot 1.3) + 1\Big) = 0.35$$

El flujo de un ejemplo en una RNA: Activación del primer peso



$$z_1 = \sigma\Big((-2 \cdot 1.3) + (1 \cdot 1.3) + (0.5 \cdot 1.3) + 1\Big) = \sigma(0.35) = 0.587$$

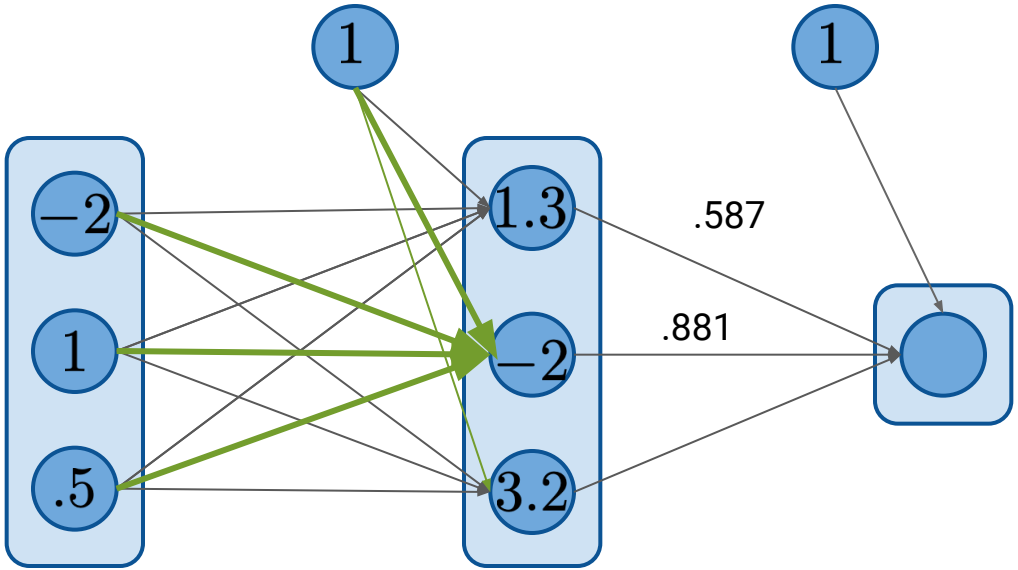
El flujo de un ejemplo en una RNA: Segundo peso



{desafío} latam_

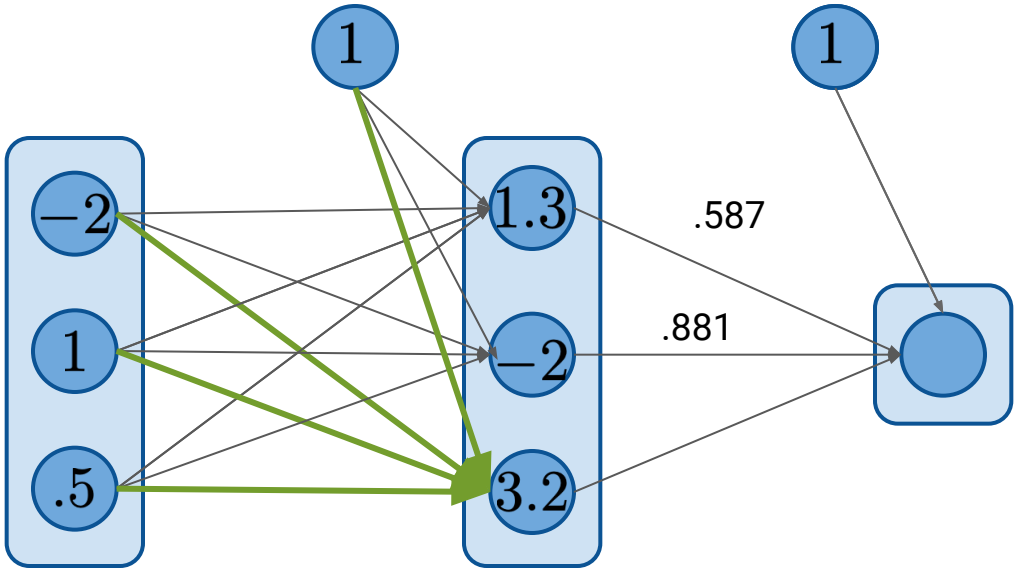
$$z_2 = \sigma\Big((-2 \cdot -2) + (1 \cdot -2) + (0.5 \cdot -2) + 1\Big) = 2.0$$

El flujo de un ejemplo en una RNA: Activación del segundo peso



$$z_2 = \sigma\Big((-2 \cdot -2) + (1 \cdot -2) + (0.5 \cdot -2) + 1\Big) = \sigma(2.0) = 0.881$$

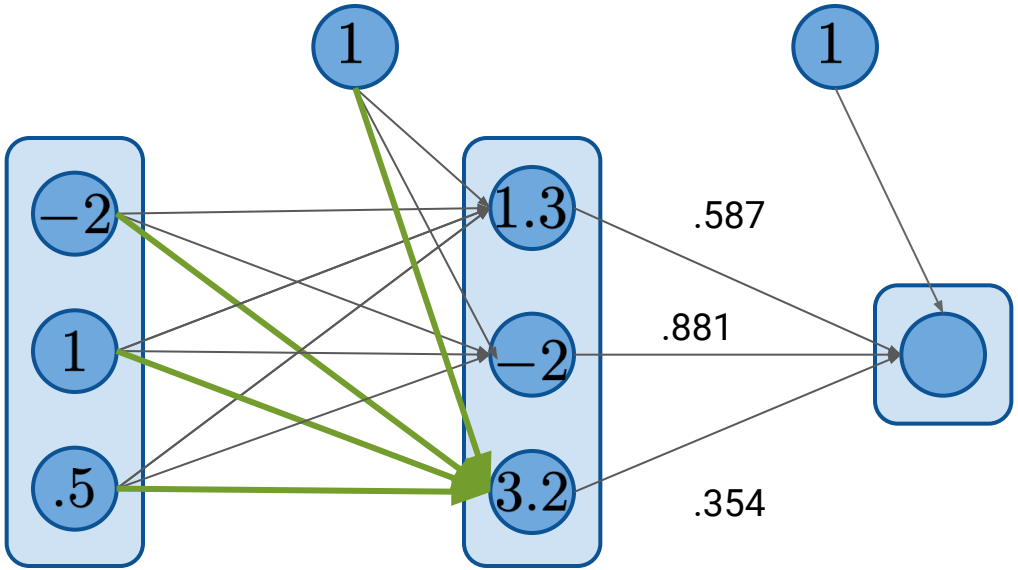
El flujo de un ejemplo en una RNA: Tercer peso



{desafío} latam_

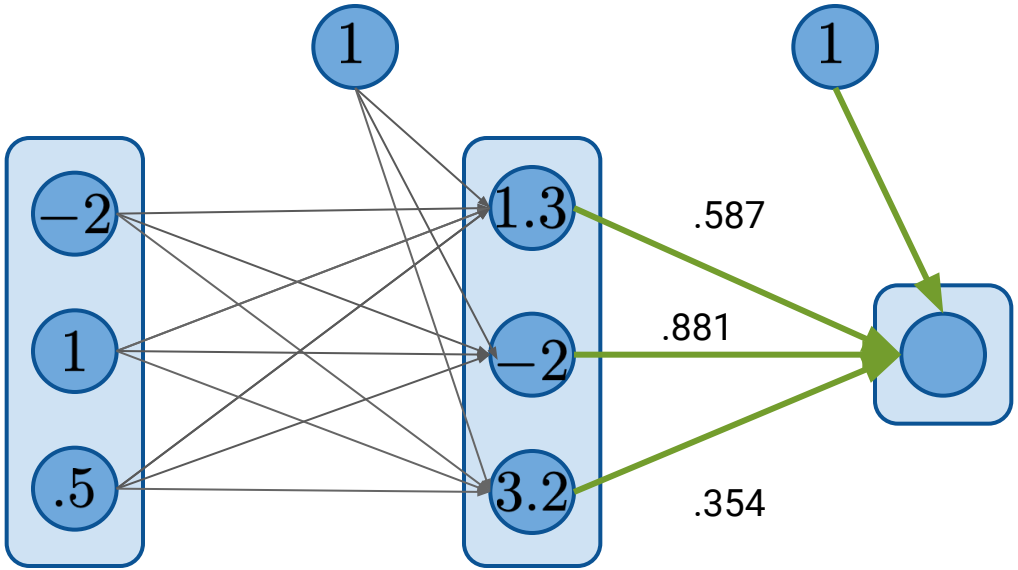
$$z_3 = \sigma\Big((-2 \cdot 3.2) + (1 \cdot 3.2) + (0.5 \cdot 3.2) + 1\Big) = -0.6$$

El flujo de un ejemplo en una RNA: Activación del tercer peso



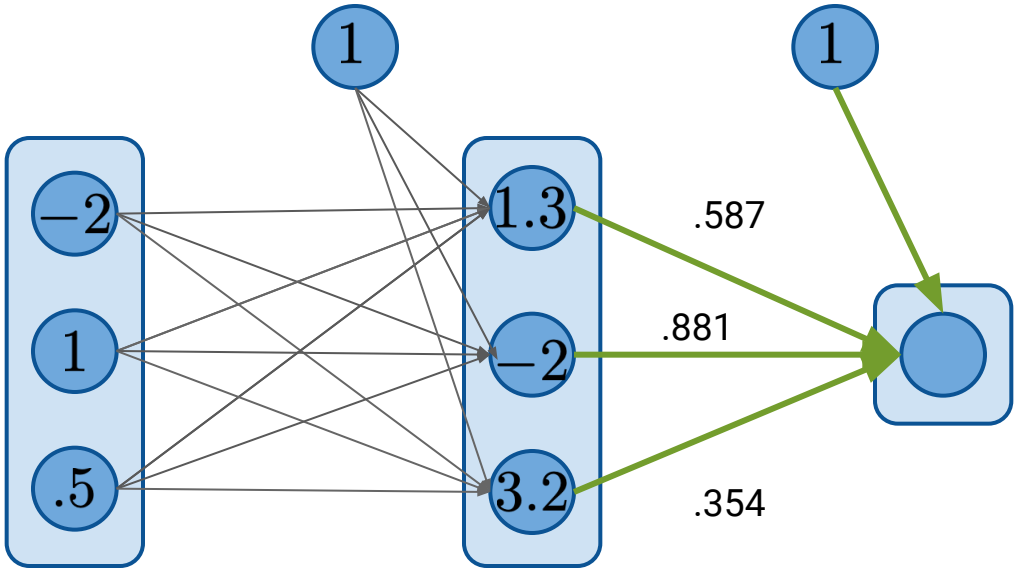
$$z_3 = \sigma\Big((-2 \cdot 3.2) + (1 \cdot 3.2) + (0.5 \cdot 3.2) + 1\Big) = \sigma(-0.6) = 0.354$$

El flujo de un ejemplo en una RNA: Capa de salida



$$z_4 = \sigma\left(0.587 + 0.881 + 0.354 + 1\right) = 2.822$$

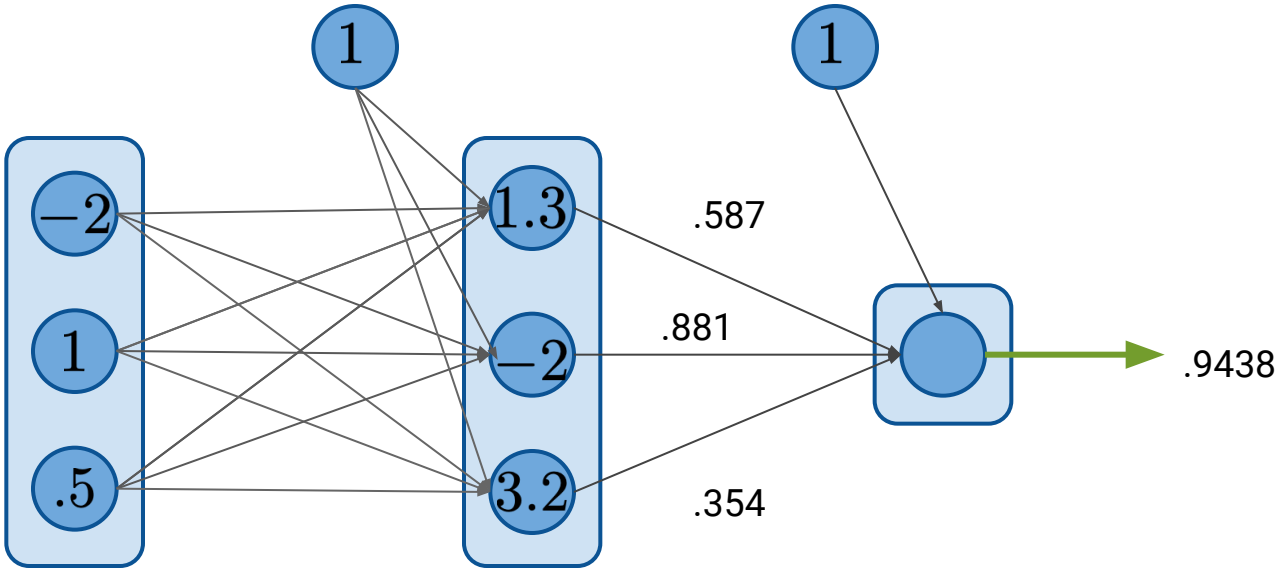
El flujo de un ejemplo en una RNA: Activación en la capa de salida



$$a_4 = \sigma\left(0.587 + 0.881 + 0.354 + 1\right) = \sigma(2.822) = 0.9438$$

{desafío} latam_

El flujo de un ejemplo en una RNA: Resultado



$$z_4 = \sigma\Big(0.587 + 0.881 + 0.354 + 1\Big) = \sigma(2.822) = 0.9438$$

El proceso, resumido

$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

- Nuestro primer paso es implementar una suma ponderada a nivel de capa.

El proceso, resumido

$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

- Nuestro primer paso es implementar una suma ponderada a nivel de capa.
- De esta forma, debemos considerar el sesgo a nivel de capa para cada uno de las neuronas integrantes de la capa.

El proceso, resumido

$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

- Comenzamos por multiplicar cada peso de la neurona por cada atributo y posteriormente sumar los resultados de este procedimiento.
- Una vez finalizado, procedemos con la adición del sesgo a nivel de capa.
- Este va a ser el resultado específico a nivel de neurona dentro de una capa.

El proceso, resumido

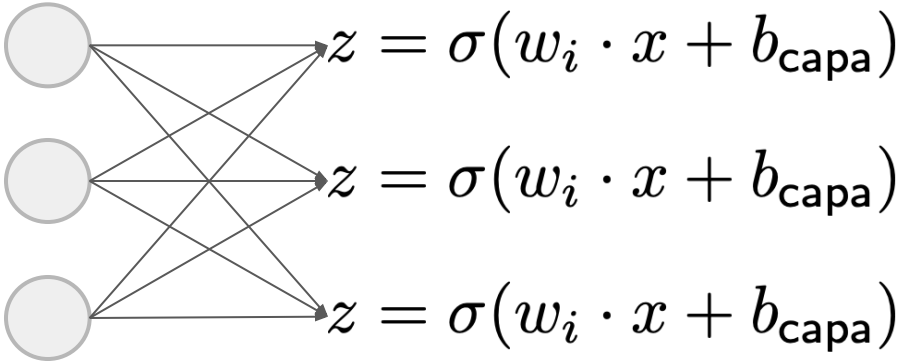
$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

La suma ponderada a nivel de neurona es evaluada mediante una función de activación que permitirá reexpresar el peso actualizado en una forma no-lineal.

El proceso, resumido

El proceso, resumido

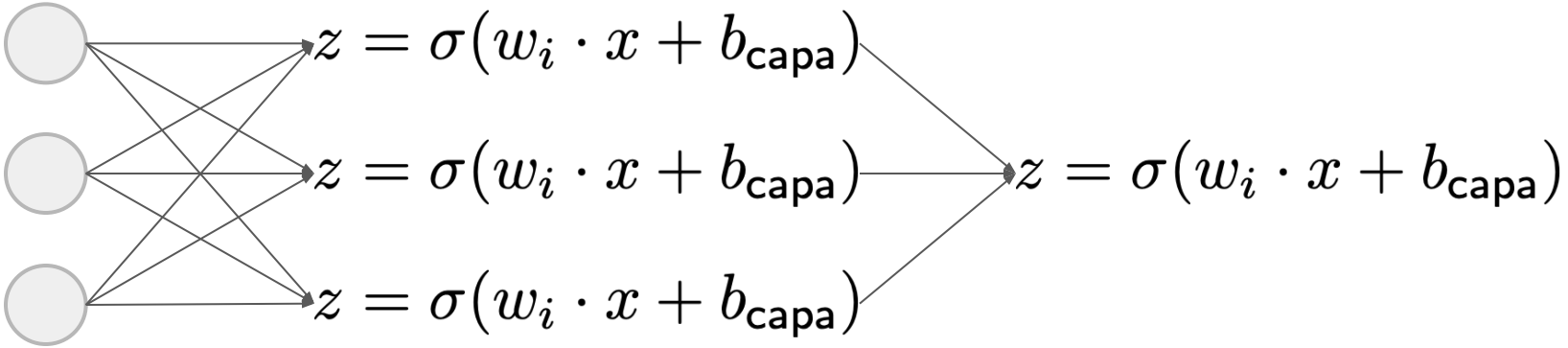


$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

$$z = \sigma(w_i \cdot x + b_{\text{capa}})$$

El proceso, resumido



Efecto de la cantidad de neuronas en una capa

Nuestra primera gran decisión

Resulta que el problema de asignar neuronas a las capas de entrada y salida no es complejo:

- La capa de entrada reflejará la cantidad de atributos en un ejemplo de entrenamiento.
- La capa de salida reflejará nuestro vector objetivo o la cantidad de clases a predecir.

"One hidden layer is sufficient for the large majority of problems" (Sarle, W. 2000; Neural Networks FAQ).
"The optimal size of the hidden layer is usually between the size of the input and the size of the output layers" (Heaton, J. Introduction to Neural Networks in Java).

- Podemos generar el promedio de ambas capas.
- Otras reglas sugieren incorporar ⅔ de neuronas en una capa.
- Intervalo superior del número de neuronas que no conduce a overfitting:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

{desafío} latam_

Academia de talentos digitales

www.desafiolatam.com