Repaso de redes neuronales

Visión por Computador, curso 2024-2025

Silvia Martín Suazo, silvia.martin@u-tad.com

24 de octubre de 2024

U-tad | Centro Universitario de Tecnología y Arte Digital

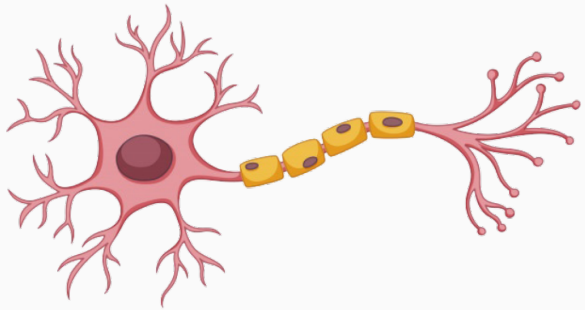
La neurona artificial

Redes neuronales: concepto general

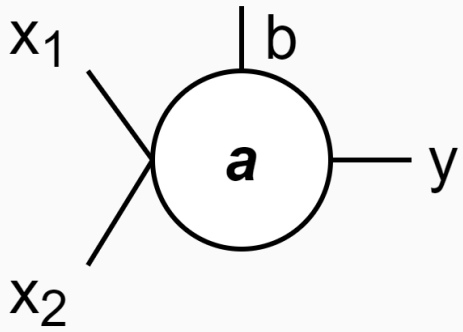
Una red neuronal es un sistema matemático capaz de aprender ha realizar predicciones a partir de unos datos de entrada, fue propuesta por McCulloch y Pitts en 1943[1], se basaba en imitar el comportamiento de una neurona biológica.

- Toma ciertos estímulos de entrada, los procesa y genera una nueva salida.
- En el caso biológico los estímulos provienen de impulsos nerviosos, mientras que en el caso de las neuronas artificiales esto es replicado a través de cálculos matemáticos.

Neurona biológica



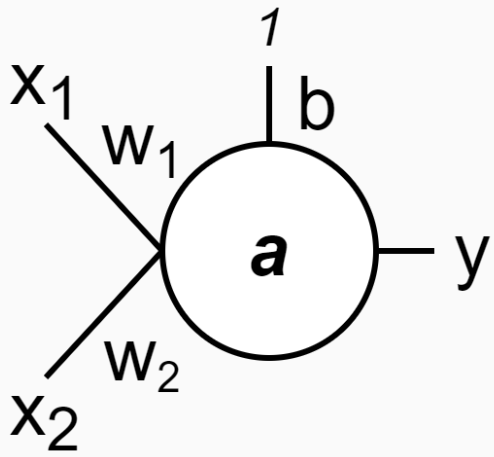
Neurona artificial



Redes neuronales: concepto general

Una neurona artificial realiza cálculos matemáticos para transformar ciertos valores numéricos. Para dicha labor, existen diversos elementos dentro de una neurona artificial:

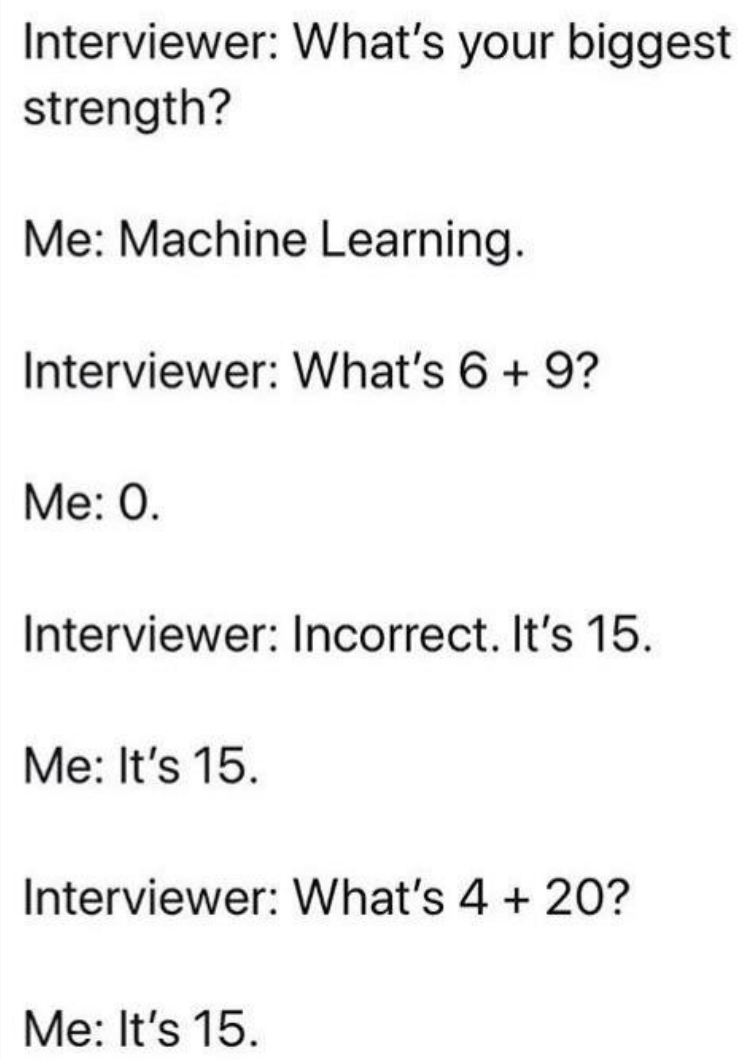
- Entradas (x_i): Introducen valores numéricos en la neurona
- Salida (y): Suministra la salida de la neurona
- Pesos (w_i): Son parámetros capaces de cambiar, realizan el aprendizaje de la neurona
- Bias (b): Realiza la misma función que cualquier otro peso, pero el valor de su entrada (x_0) es siempre 1
- Función de activación (a): Es una función que participa en el cálculo de la salida de la neurona



Redes neuronales: entrenamiento

El entrenamiento de las redes neuronales se puede dividir en tres fases:

- Predicción: calculamos para las entradas la predicción de valor de salida de la red.
- Calculo de error: con la predicción y el resultado que queremos obtener calculamos el error obtenido.
- Ajuste de pesos: con el error se reajustan los parámetros de la red.



Algoritmo de propagación

La fase de predicción de una red neuronal se realiza a través del algoritmo de propagación.

Este se encarga de procesar la entrada y generar la salida correspondiente.

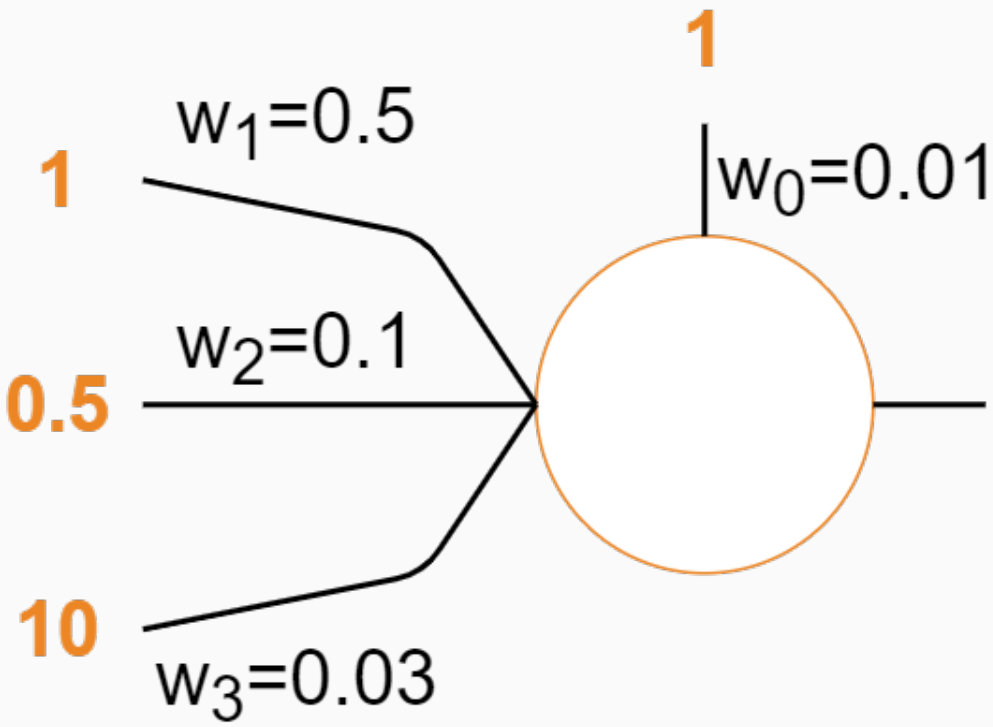
Para ello, la computación de cada neurona es la siguiente:

- Cada entrada x_i es multiplicada por el valor de su peso correspondiente w_i.
- La entrada de *bias* siempre es "1", y se multiplica por su peso correspondiente (a veces indicado como w_0).
- Todas las entradas de la neurona se combinan haciendo una suma de todas ellas, de tal manera que se realiza una combinación lineal.
- El resultado de la combinación lineal se pasa por una función no lineal para generar la salida de la neurona.

Algoritmo de propagación

La ecuación que define este proceso es la siguiente:

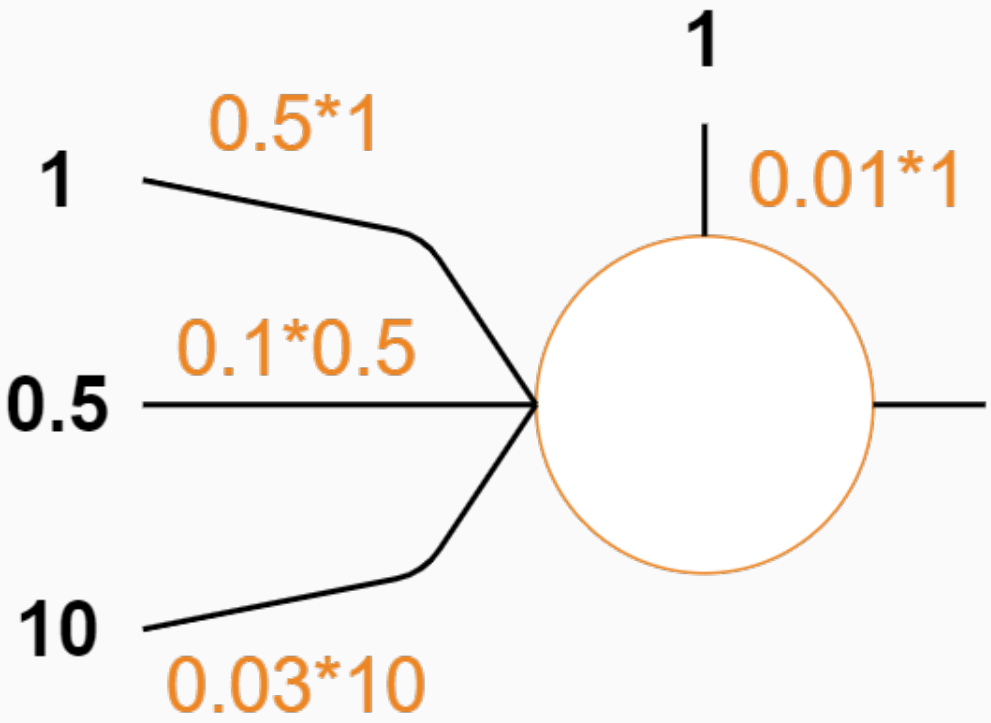
$$f\left(\sum_{i=0}^{n} W_i X_i\right) \tag{1}$$



Algoritmo de propagación

La ecuación que define este proceso es la siguiente:

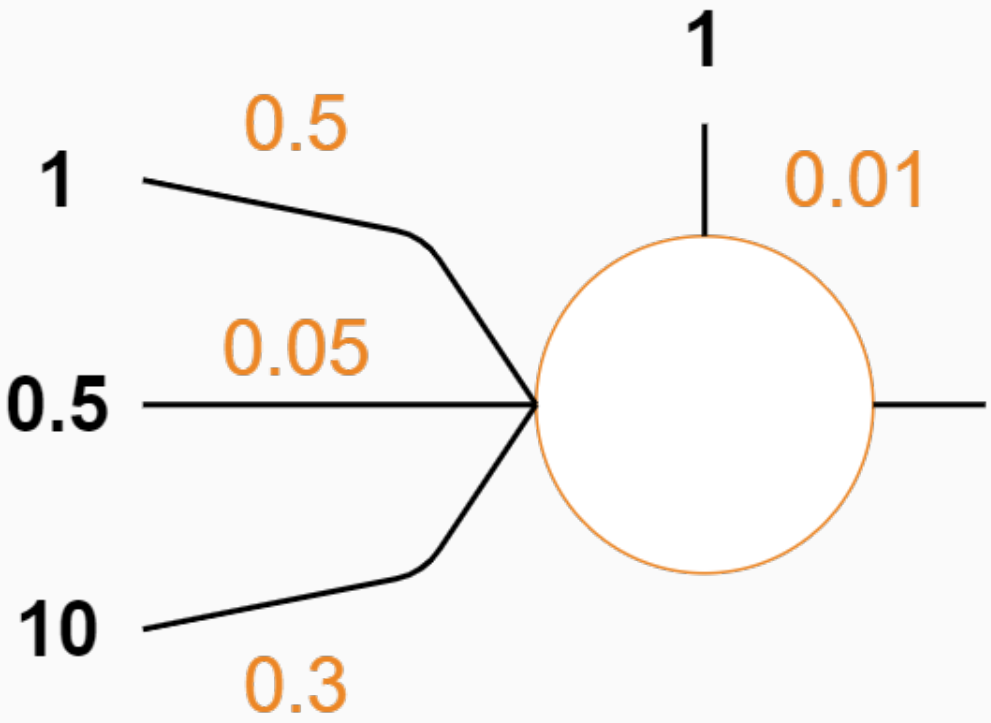
$$f\left(\sum_{i=0}^{n} W_i X_i\right) \tag{1}$$



Algoritmo de propagación

La ecuación que define este proceso es la siguiente:

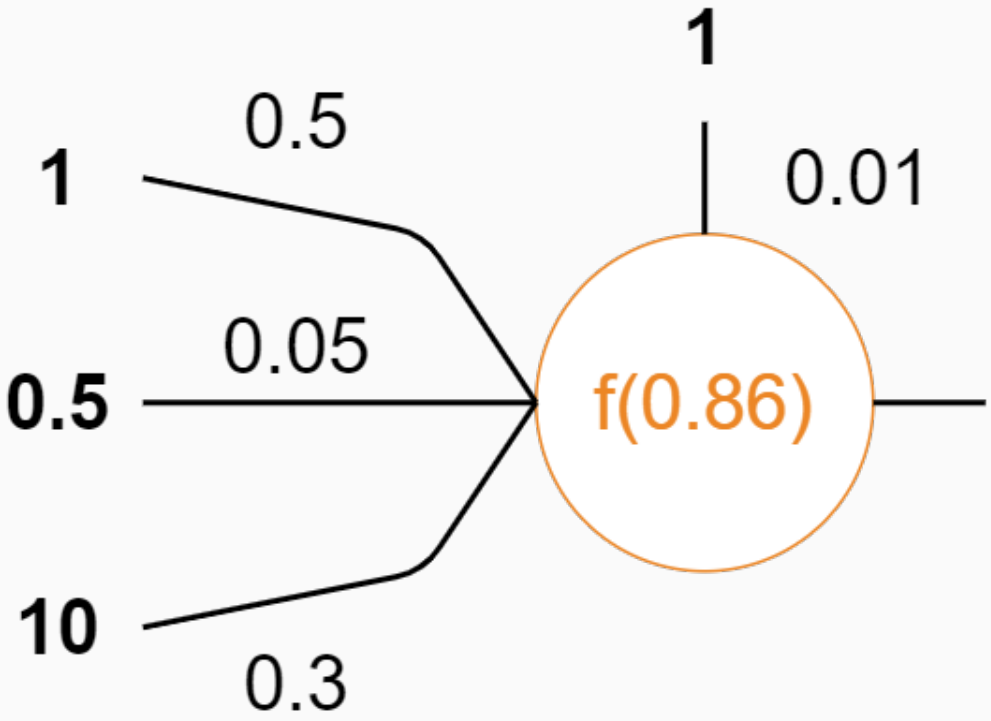
$$f\left(\sum_{i=0}^{n} W_i X_i\right) \tag{1}$$



Algoritmo de propagación

La ecuación que define este proceso es la siguiente:

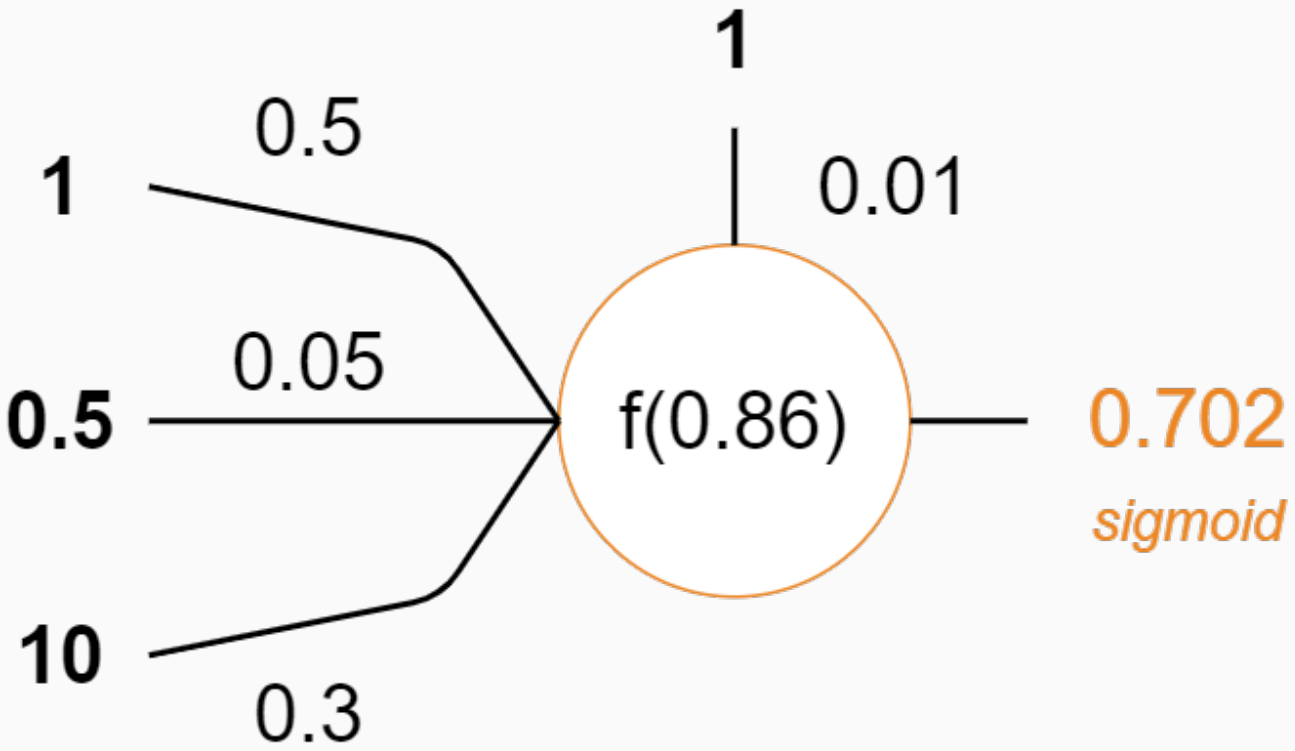
$$f\left(\sum_{i=0}^{n} W_i X_i\right) \tag{1}$$



Algoritmo de propagación

La ecuación que define este proceso es la siguiente:

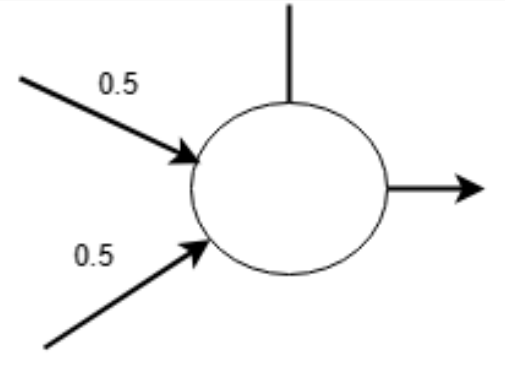
$$f\left(\sum_{i=0}^{n} W_i X_i\right) \tag{1}$$



Ejercicio algoritmo de propagación

Podemos entrenar neuronas para que actúen como puertas lógicas. Por ejemplo si queremos implementar una OR con la siguiente neurona, función de activación escalón y $b = 1$:

$$S = \begin{cases} 1 & \text{si } \sum_i x_i \times w_i + b \geq 0 \\ 0 & \text{en otro caso} \end{cases}$$



Ejercicio algoritmo de propagación

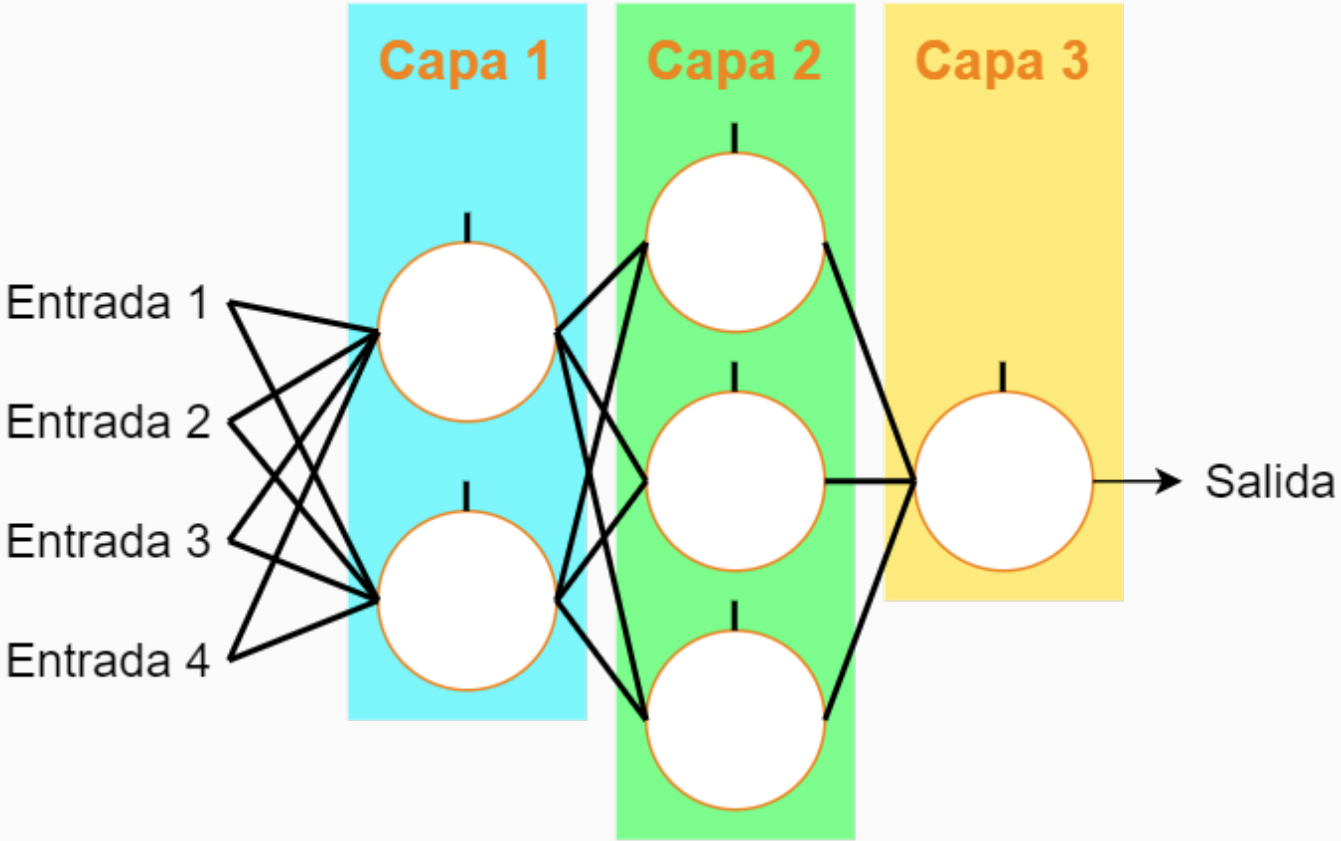
It	(x_1,x_2)	$\sum$	S_{neu}	S_{real}	Nuevo w_i	Nuevo b
1	(0, 0)	1	1	0	-	-1
2	(0, 0)	-1	0	0	-	-
3	(0, 1)	-0.5	0	1	(0.5, 1)	-
4	(0, 1)	0	1	1	-	-
5	(1, 0)	-0.5	0	1	(1,1)	-
6	(1, 0)	0	1	1	-	-
7	(1, 1)	1	1	1	-	-

Redes de neuronas artificiales

Estructura de capas

Una red de neuronas se organiza en capas, las cuales se componen por un número determinado de neuronas.

Cada capa de neuronas se conecta con la siguiente y recibe datos de la anterior. De esta manera se produce el flujo de datos a lo largo de la red.



Estructura de capas

Se distinguen tres tipos de capas:

- Capa de entrada: recibe los datos de entrada. Únicamente hay una en la red.
- Capas ocultas: son las capas intermedias y principales encargadas de realizar el aprendizaje. Se denominan ocultas porque la información en estas no es accesible “a priori”. Pueden tener de 1 a n, y pueden ser de distintos tipos: densa, de convolución, de normalización, ...
- Capa de salida: produce el resultado final de la red, una clase en un problema de clasificación o valores en un problema de regresión. Solo hay una.

Backpropagation

El algoritmo de retropropagación o backpropagation es fundamental para optimizar la red de neuronas para su cometido específico.

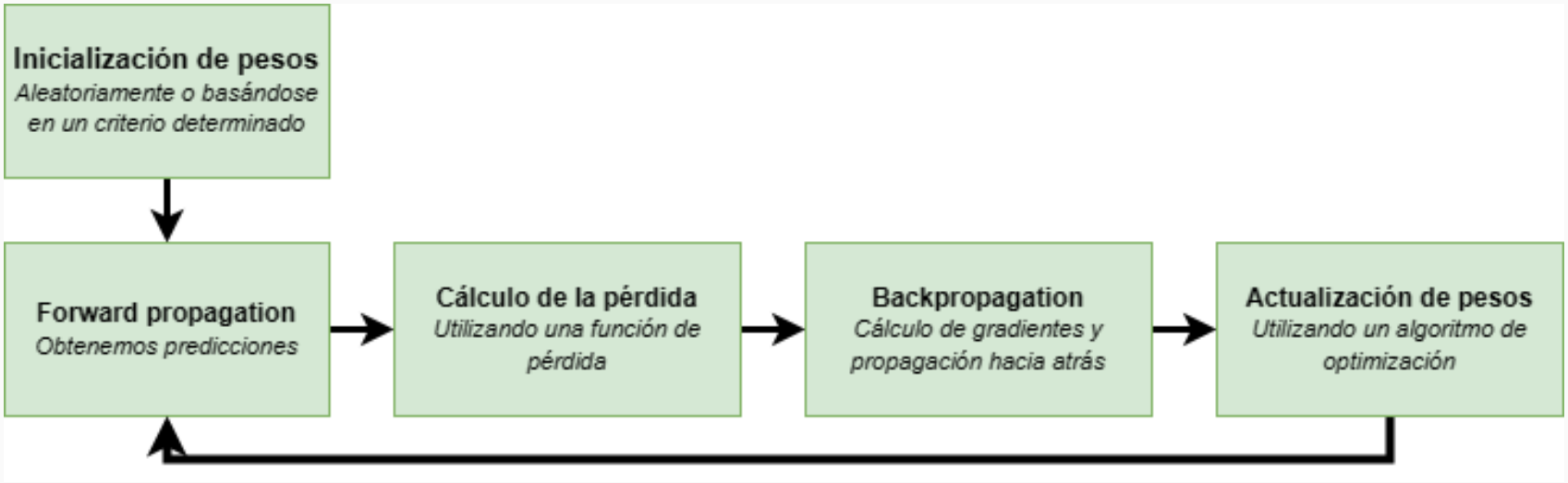
Se basa en actualizar los pesos de la red en función del error cometido al predecir salidas concretas. Para ello se calculan los gradientes (derivadas) de la función de pérdida con respecto a cada peso en cada capa oculta.

La actualización de los pesos (w_i) se lleva a cabo según la siguiente fórmula, donde α representa la tasa de aprendizaje que determina la magnitud con la que la red ajusta sus pesos:

$$\triangle w_i = w_i - \alpha \cdot \nabla L \tag{2}$$

Aquí, ∇L representa el gradiente de la función de pérdida. Este proceso se repite iterativamente hasta finalizar el entrenamiento.

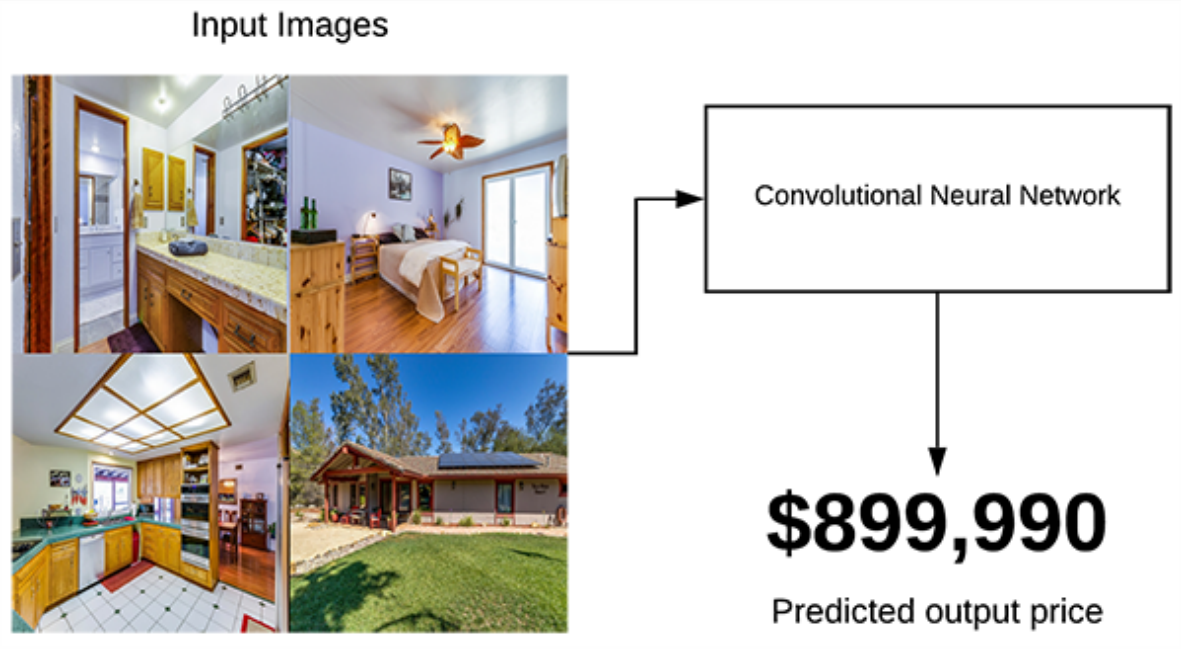
Esquema general de funcionamiento



Salida de una red neuronal

Gracias a la utilización de neuronas organizadas en redes, se puede aprender relaciones más complejas entre los datos y llevar a cabo tareas más complicadas. Estas tareas se pueden dividir en dos grupos:

- Regresión: en esta categoría se obtienen uno o varios valores de salida, generalmente representando predicciones.

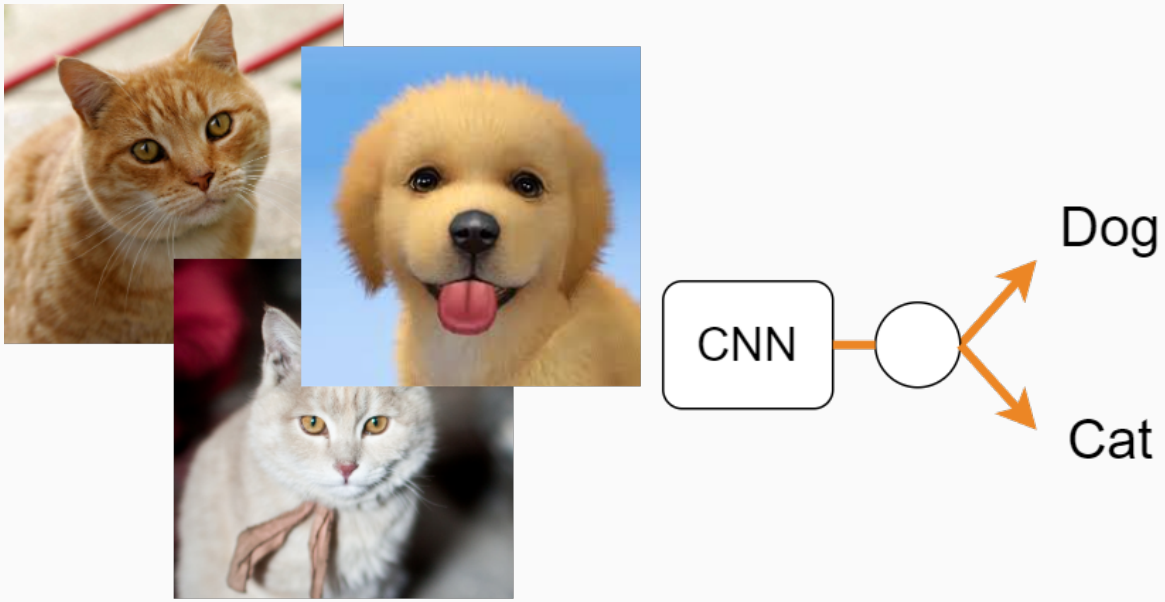


[2]

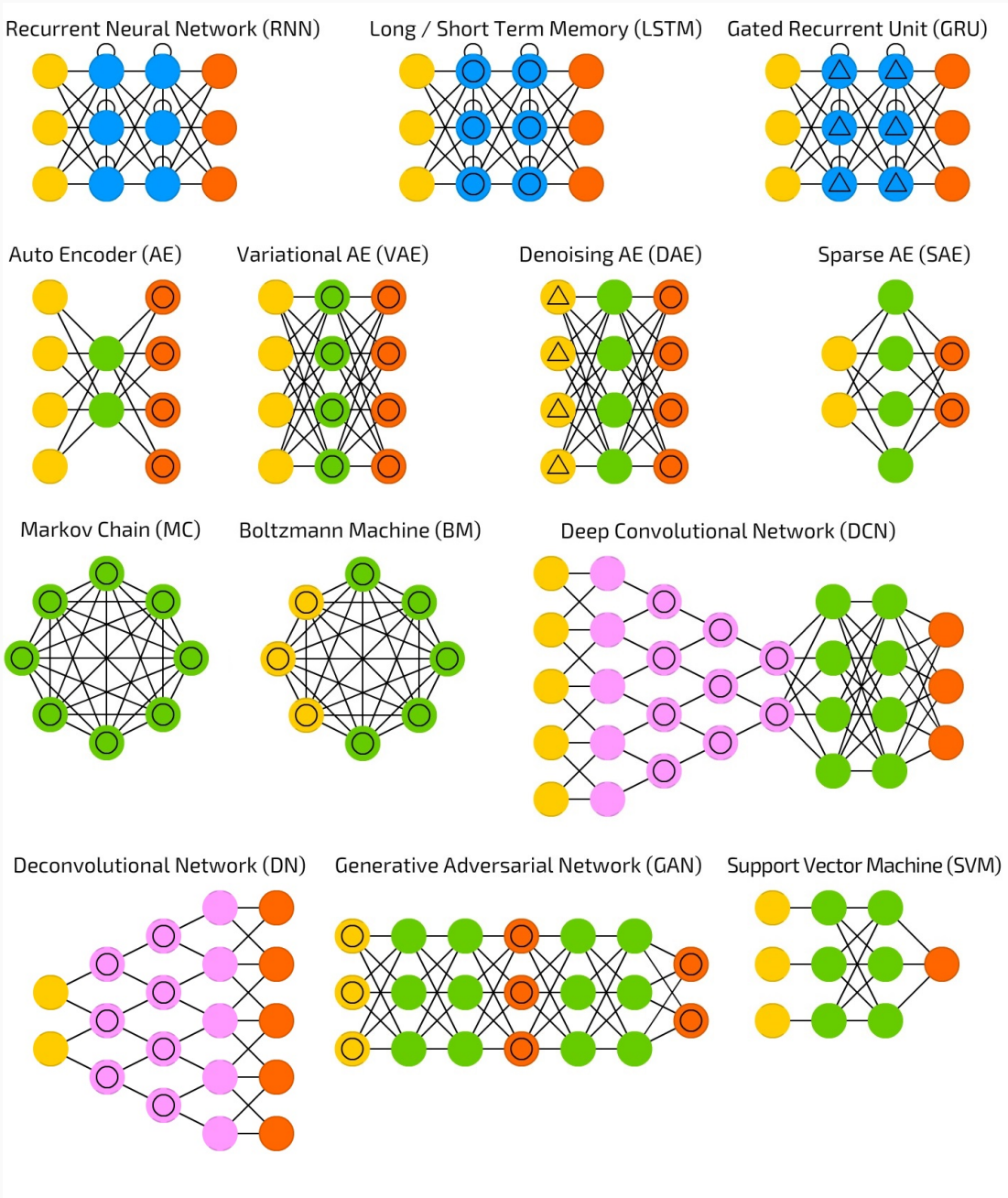
Salida de una red neuronal

Gracias a la utilización de neuronas organizadas en redes, se puede aprender relaciones más complejas entre los datos y llevar a cabo tareas más complicadas. Estas tareas se pueden dividir en dos grupos:

- Clasificación: en esta categoría se obtienen las probabilidades de que los datos de entrada pertenezcan a cada una de las clases indicadas previo entrenamiento.



Tipos de redes neuronales



Consideraciones de las redes neuronales

Algoritmos de optimización

Los algoritmos de optimización se utilizan para ajustar los pesos de la red neuronal en cada iteración. Generalmente se basan en la utilización de gradientes. Algunos de los más utilizados son:

- Descenso de gradiente (Gradient Descent): únicamente se actualizan los pesos tras procesar todo el conjunto de entrenamiento.
- Descenso de Gradiente Estocástico (SGD): se ajustan los pesos tras cada lote de entrenamiento.
- Adam (Adaptive Moment Estimation): extensión de SGD que ajustar las tasas de aprendizaje de manera adaptativa para cada parámetro del modelo.
- RMSprop (Root Mean Square Propagation): adapta la tasa de aprendizaje disminuyéndola exponencialmente cuando el cuadrado del gradiente está bajo cierto umbral, y usa medias móviles exponenciales de los gradientes al cuadrado para ajustar los parámetros del modelo.

Funciones de pérdida

Existen múltiples métodos para calcular la distancia de la predicción $\hat{y}$ con respecto de la salida deseada y. Es decir, múltiples funciones de pérdida que nos permiten calcular el error.

Regresión:

$$\text{MAE} = \frac{1}{n}\sum |y - \hat{y}| \tag{3}$$

$$\text{MSE} = \frac{1}{n}\sum (y - \hat{y})^2 \tag{4}$$

$$\text{MAPE} = \frac{1}{n}\sum \left(\frac{|y - \hat{y}|}{|y|}\right) \times 100 \tag{5}$$

*También pueden utilizarse como métricas

Funciones de pérdida

Existen múltiples métodos para calcular la distancia de la predicción $\hat{y}$ con respecto de la salida deseada y. Es decir, múltiples funciones de pérdida que nos permiten calcular el error.

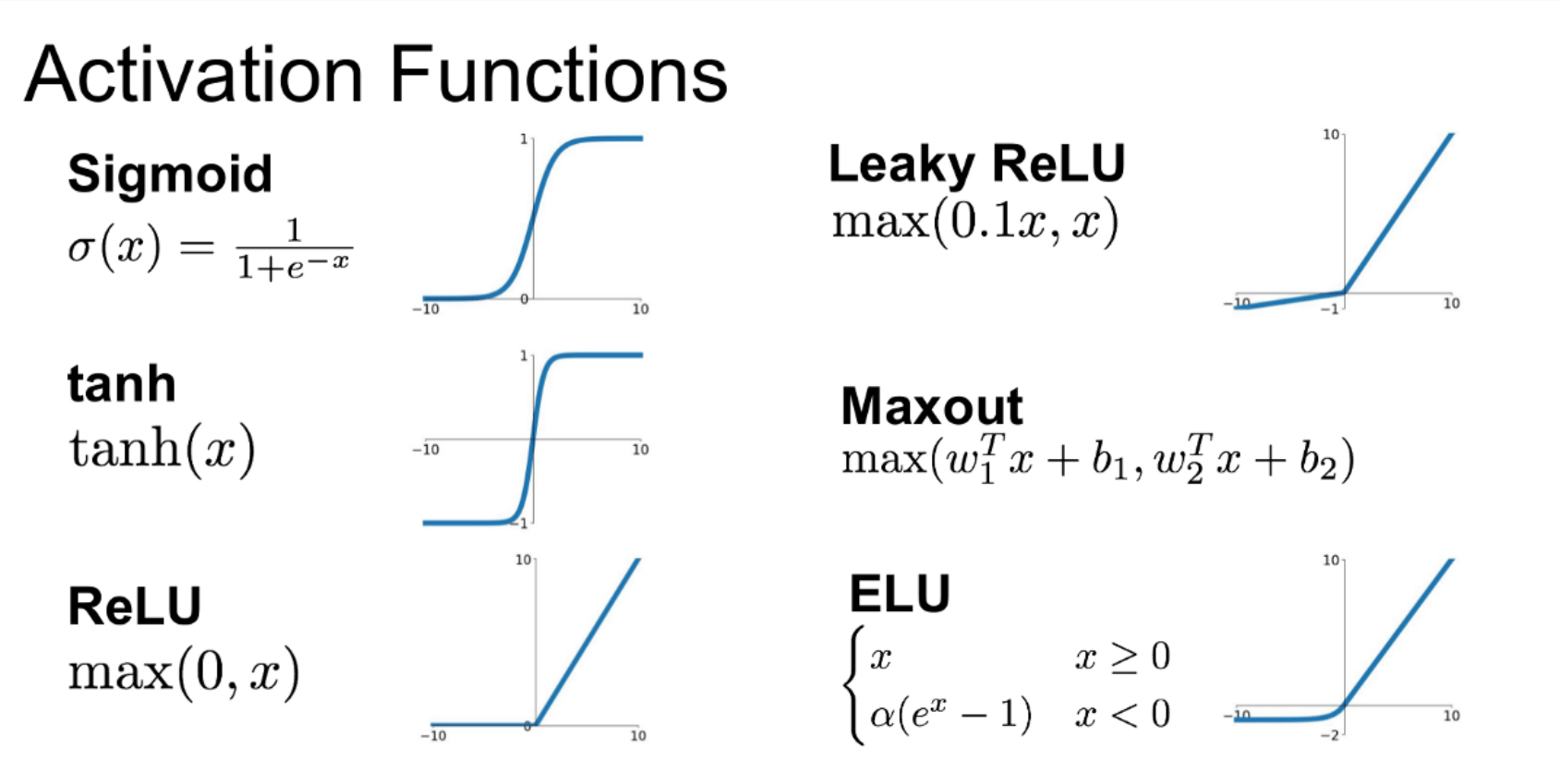
Clasificación:

$$\text{Cross-Entropy} = -\sum y \cdot \log \hat{y} \tag{6}$$

$$\text{Hinge loss} = \max(0, 1 - y \cdot \hat{y}) \tag{7}$$

Funciones de activación

Las funciones de activación de cada neurona pueden variar, entre las más populares se encuentran:



[3]

Problema de la neurona muerta

El problema de la neurona muerta o dying ReLU sucede cuando una red utiliza la función de activación ReLU y llega a un punto donde únicamente genera ceros.

Esto sucede cuando una actualización de los pesos de la red hace que la salida de una neurona sea siempre negativa. Entonces esa salida será siempre cero y la neurona se considera muerta porque no es capaz de aprender.

Se propone la Leaky ReLU para solucionar este problema, aunque su rendimiento sea menor. Además, puede caer en el problema de "gradient explosion".

Problemas del gradiente

Los problemas derivados del gradiente son comunes a todas las redes neuronales. Estos están directamente influenciados por el número de capas de la red.

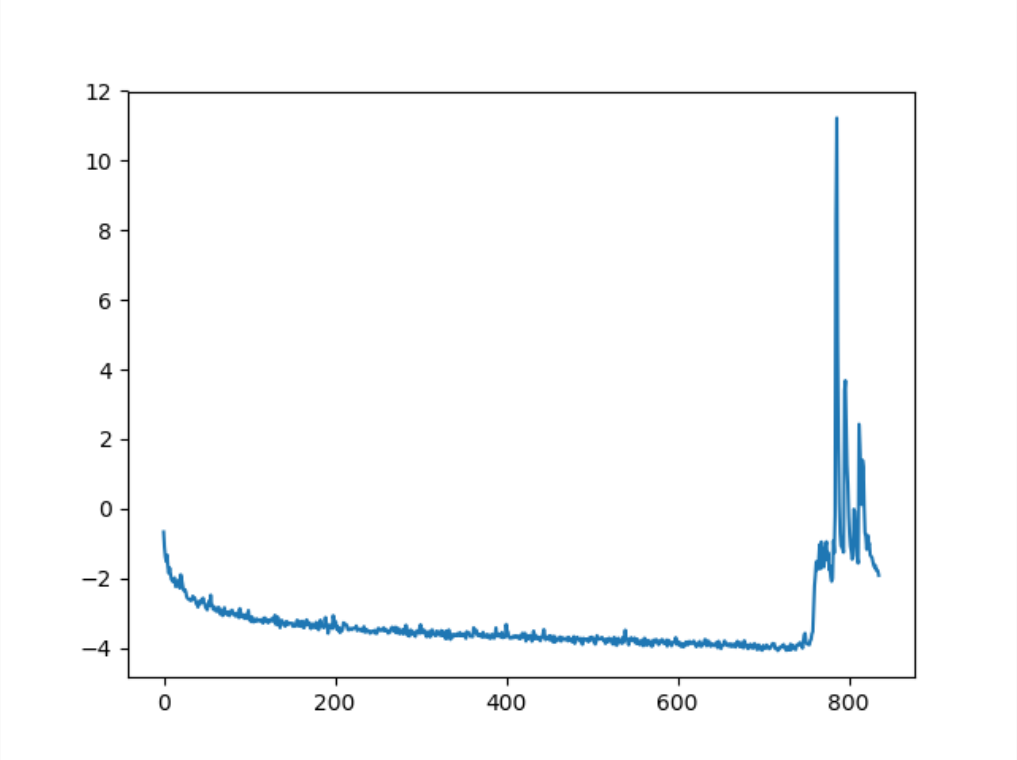
Se diferencian dos tipos:

- Gradient explosion.
- Gradient vanishing.

Al realizarse la retropropagación los valores de pérdida pasan de unas capas a otras. En este algoritmo las derivadas de cada neurona pueden llegar a descontrolarse.

$$W'_x = W_x - \alpha \left(\frac{\partial \text{Loss}}{\partial W_x} \right) \tag{8}$$

Problemas del gradiente: Gradient explosion

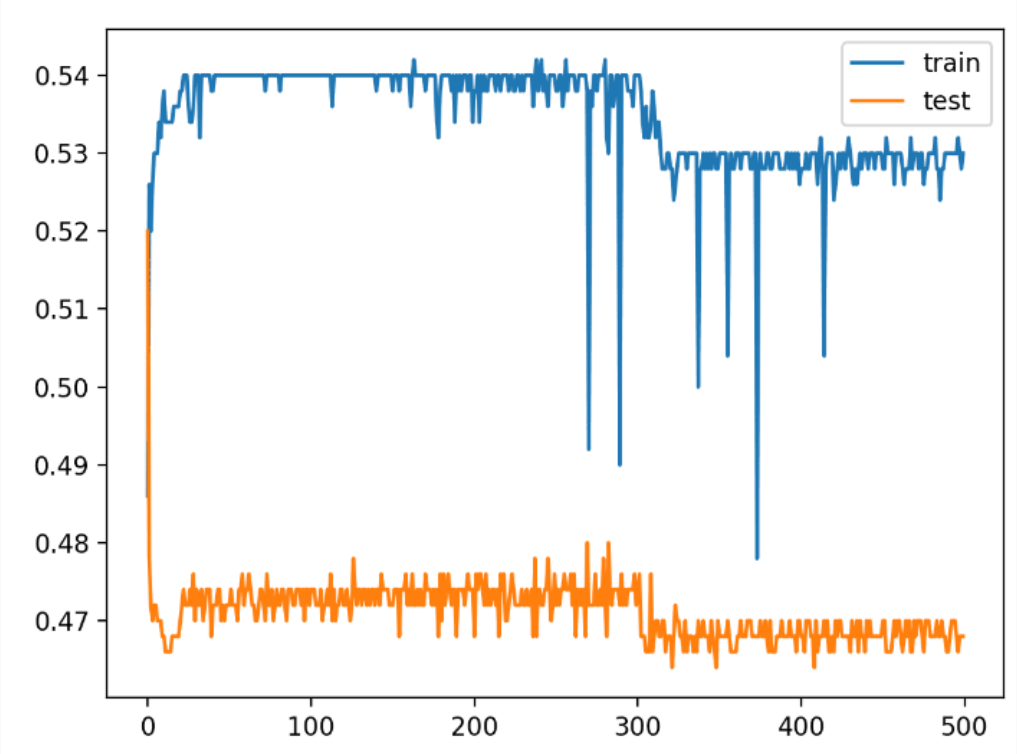


[4]

El gradient explosion, también conocido como exploding gradients sucede cuando la actualización de pesos toma valores muy elevados.

Se identifica con valores de pérdidas de NaN o muy exageradas.

Problemas del gradiente: Gradient Vanishing



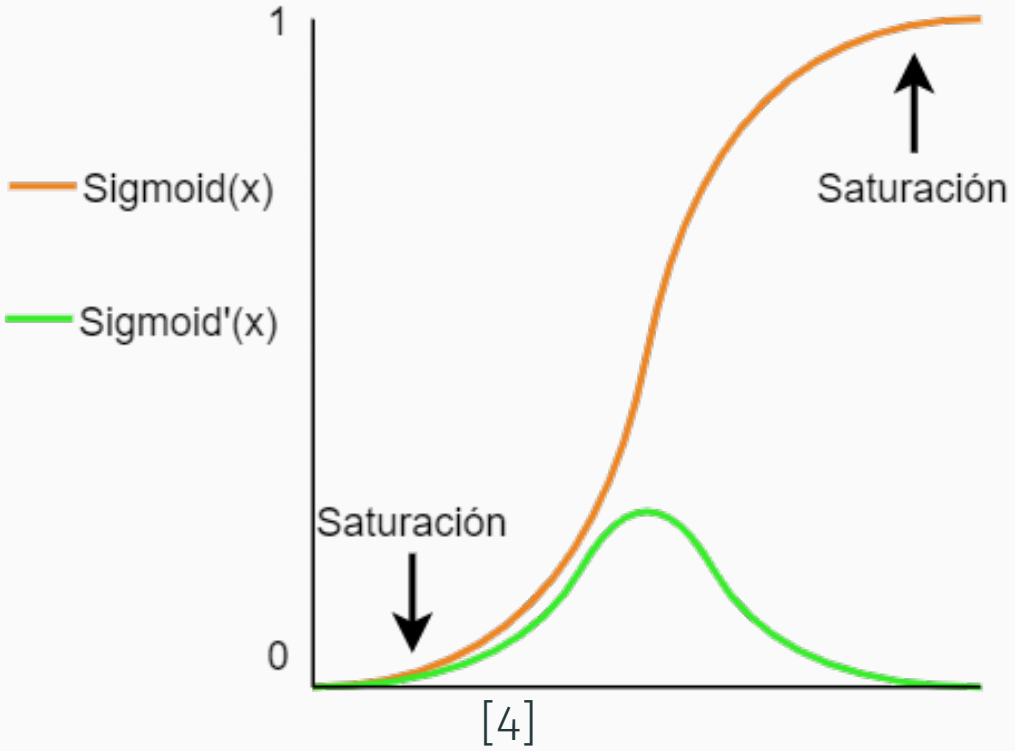
[5]

Cuando sucede gradient vanishing, también llamado vanishing gradients, la actualización de pesos se hace nula por tener valores muy pequeños.

Se identifica cuando la pérdida es constante en el tiempo.

Origen de los problemas derivados del gradiente

La principal causa de estos problemas es usar funciones de activación cuya derivada satura a 0.



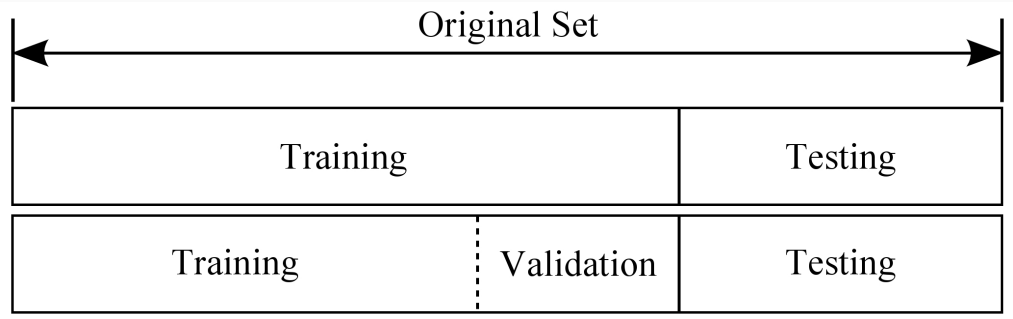
[4]

Sucede principalmente con las funciones tanh y sigmoid, por lo tanto se recomienda el uso de ReLU para capas ocultas en una red.

División de datos

Al realizar un entrenamiento con modelos de aprendizaje se realiza una división del conjunto de datos con el que se entrena. Este proceso ayuda a comprobar la fiabilidad de la red. Generalmente se utilizan 3 conjuntos de datos: el de entrenamiento, el de validación (utilizado para ajustar hiperparámetros) y el de testeo (evaluación del rendimiento final).

Generalmente se recomienda destinar un 70-15-15.

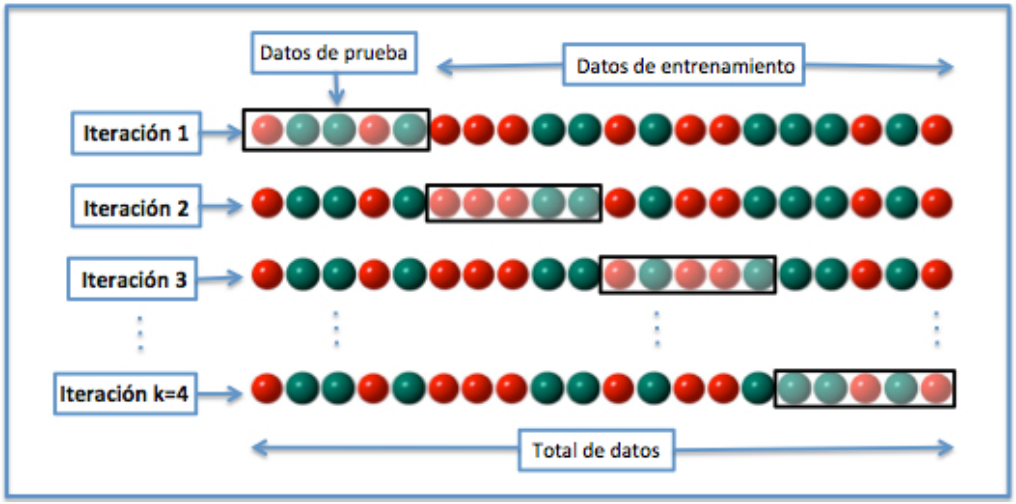


En muchos casos el conjunto de datos de entrenamiento se alimenta a la red en lotes o batches por eficiencia computacional. Además, tras cada lote se actualizan los pesos en vez de tras todo el conjunto.

K-fold

El método k-fold cross-validation o validación cruzada k-fold es una técnica de división de los datos y evaluación del modelo.

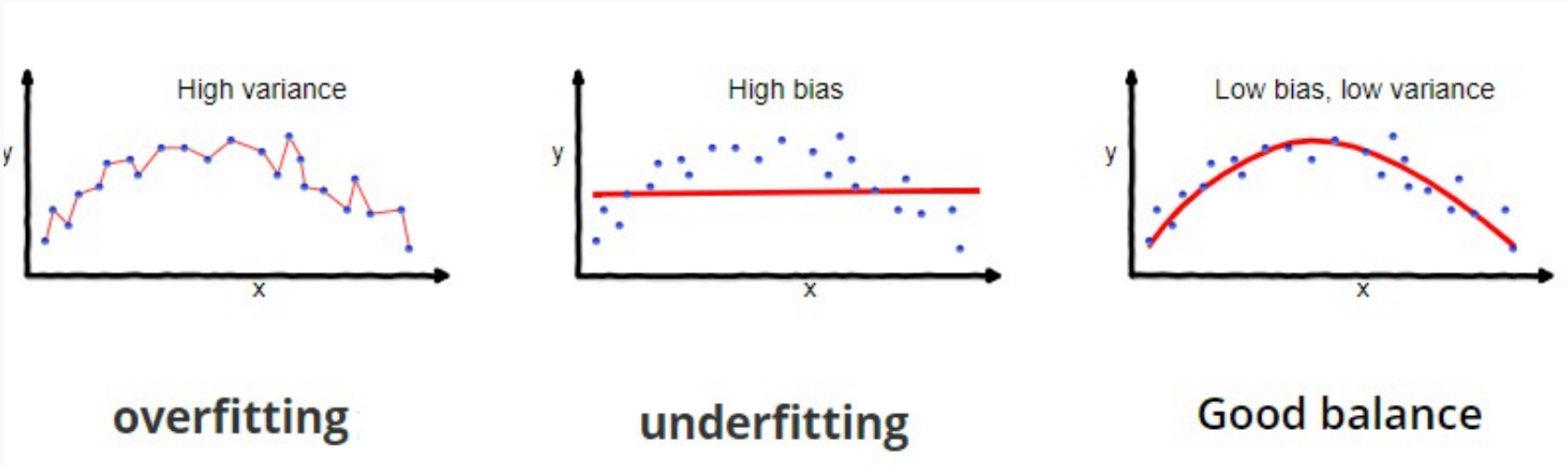
Consiste en la división de los datos en *k* pliegues y la realización de *k* iteraciones. En cada iteración se utiliza un pliegue de prueba y el resto para entrenamiento. Finalmente se promedian los resultados.



Bias-variance tradeoff

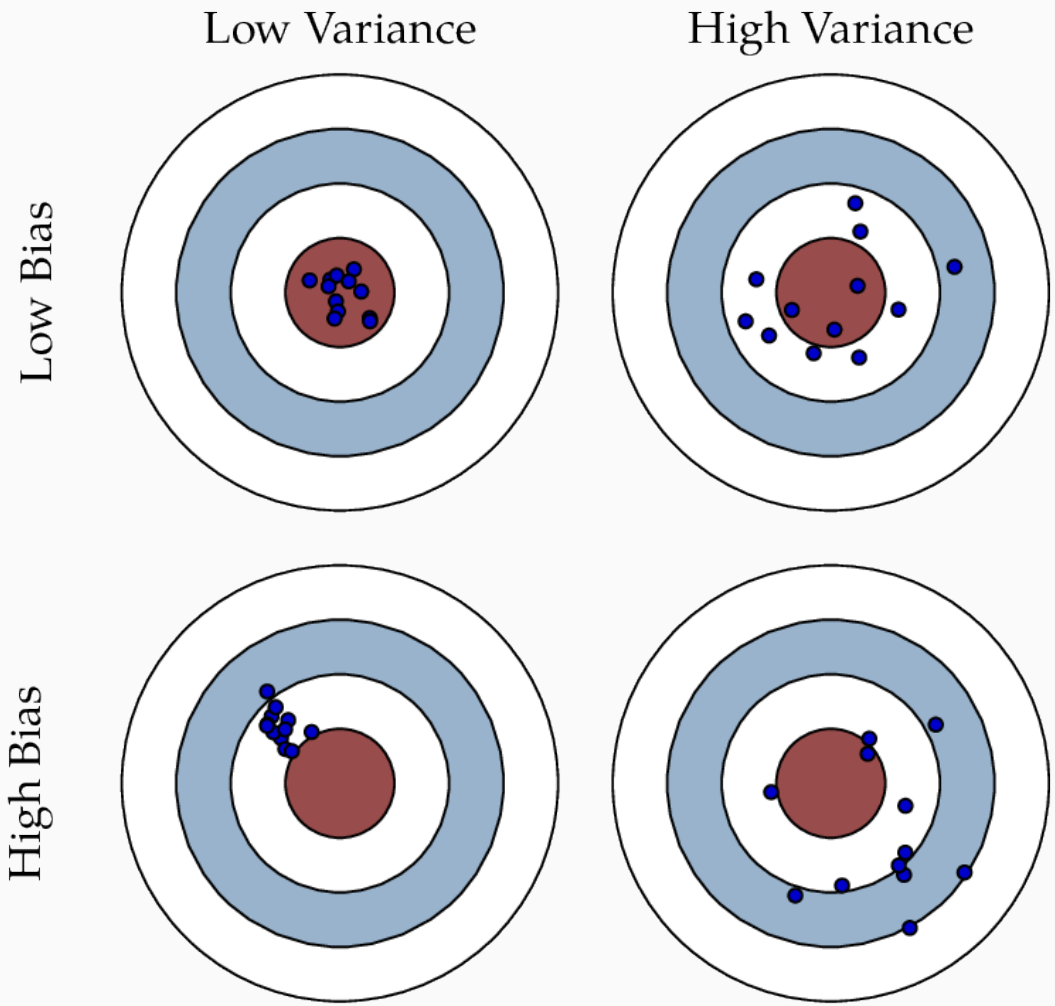
Existen dos métricas de alto nivel que evalúan el rendimiento de una red neuronal:

- Bias: Es el error del modelo ante el conjunto de datos de entrenamiento.
- Variance: Es el error del modelo ante el conjunto de datos de testeo respecto los de entrenamiento.



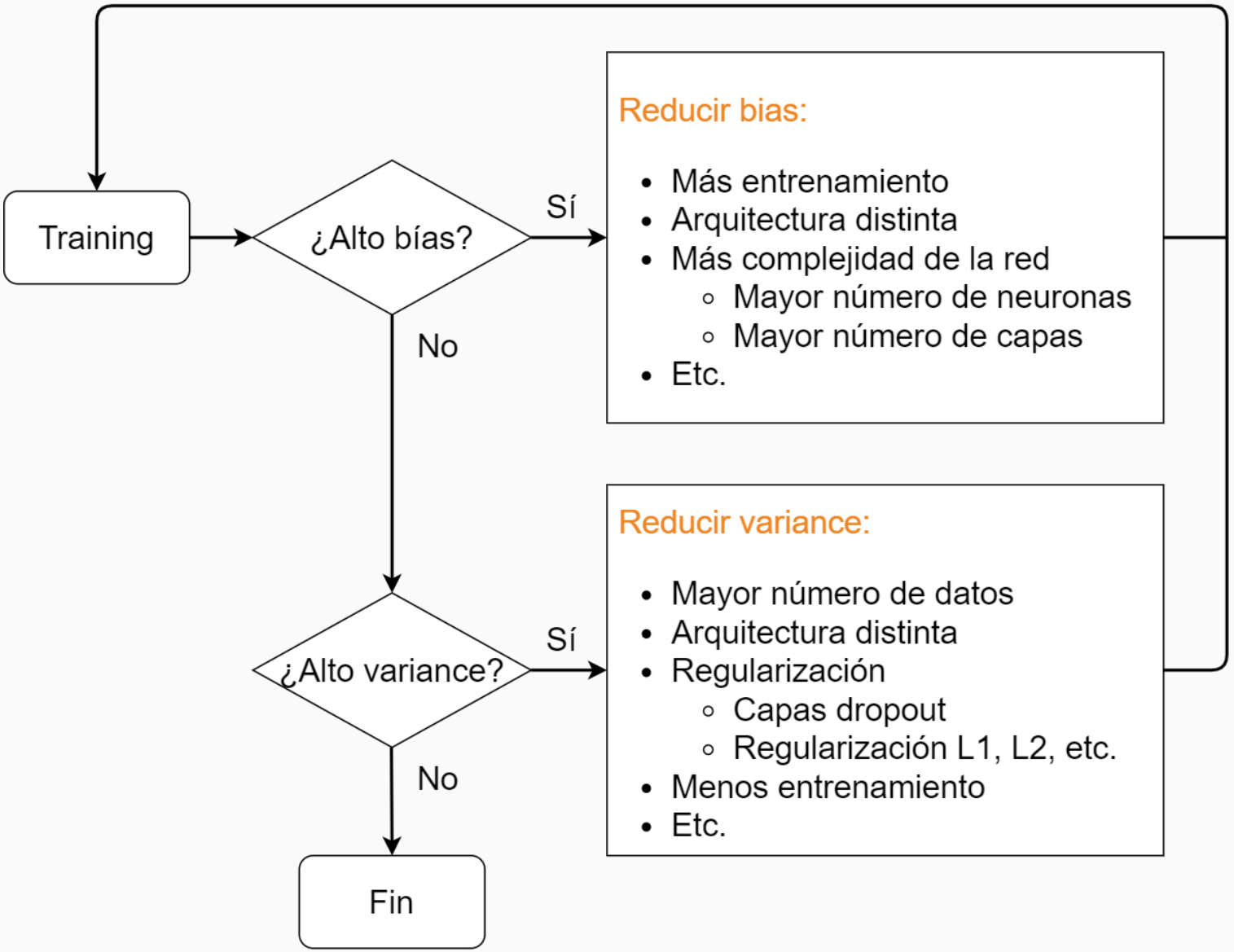
[6]

Bias-variance tradeoff

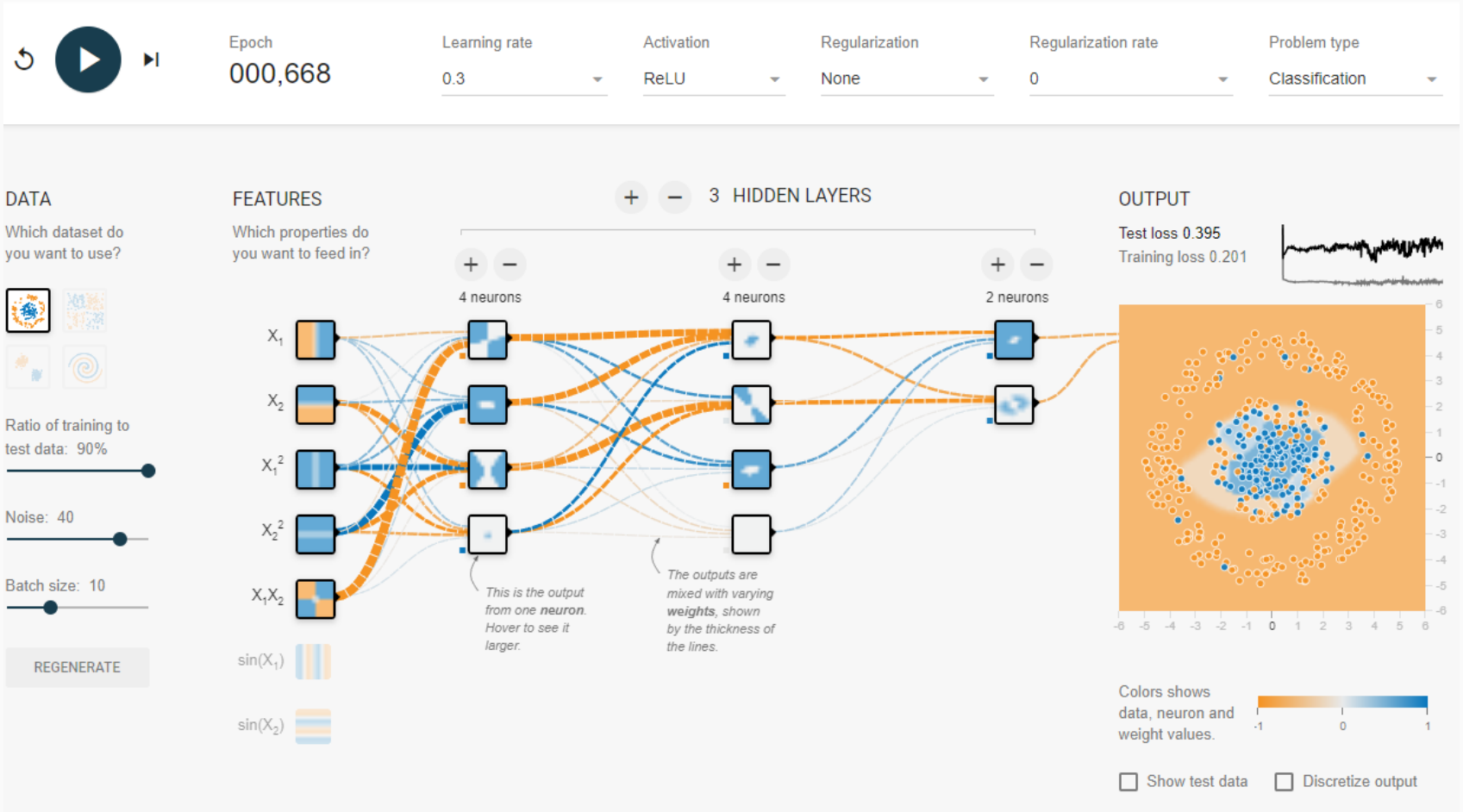


[7]

Esquema general de entrenamiento de redes neuronales



Playground

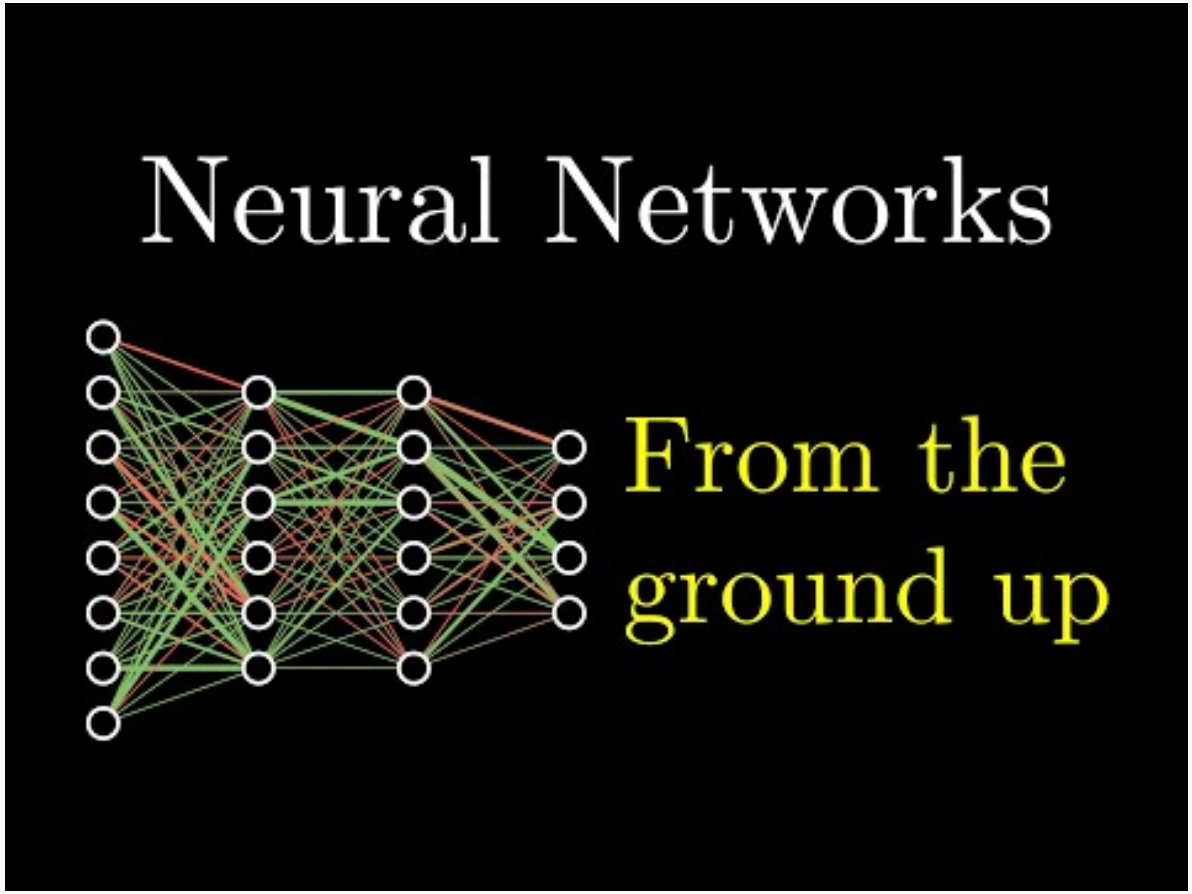


Tensorflow playground

Perceptrón multicapa para procesar imágenes

¿Cómo procesar imágenes?

La idea más básica para procesar imágenes con redes neuronales es transformar la matriz numérica de datos a un vector unidimensional.



Vídeo youtube

Capa de reshape

La capa de keras llamada “reshape” se encarga de realizar esa transformación de matriz a vector.

Sin embargo el principal inconveniente al tratar las imágenes de esta manera es la pérdida total de información espacial de la imagen.

Notebook de ejemplo, perceptrón clasificador

El siguiente notebook contiene un ejemplo de clasificador usando un perceptrón multicapa como red neuronal.

- 03.01-PerceptronMNIST.ipynb

Referencias i

[1] Warren S McCulloch and Walter Pitts.
A logical calculus of the ideas immanent in nervous activity.
The bulletin of mathematical biophysics, 5(4):115–133, 1943.

[2] Pyimagesearch.
Regression with keras and cnns.
[Online; accessed October, 2023].

[3] Data Science Interview Preparation.
Activation functions image.
[Online; accessed August, 2022].

[4] acrosson (GitHub).
Gradient explosion image.
[Online; accessed August, 2022].

Referencias ii

[5] Jason Brownlee (Machine Learning Mastery).
Gradient vanishing image.
[Online; accessed August, 2022].

[6] The Machine Learners.
Bias variance tradeoff image.
[Online; accessed August, 2022].

[7] endtoend.ai.
Bias variance tradeoff image.
[Online; accessed August, 2022].