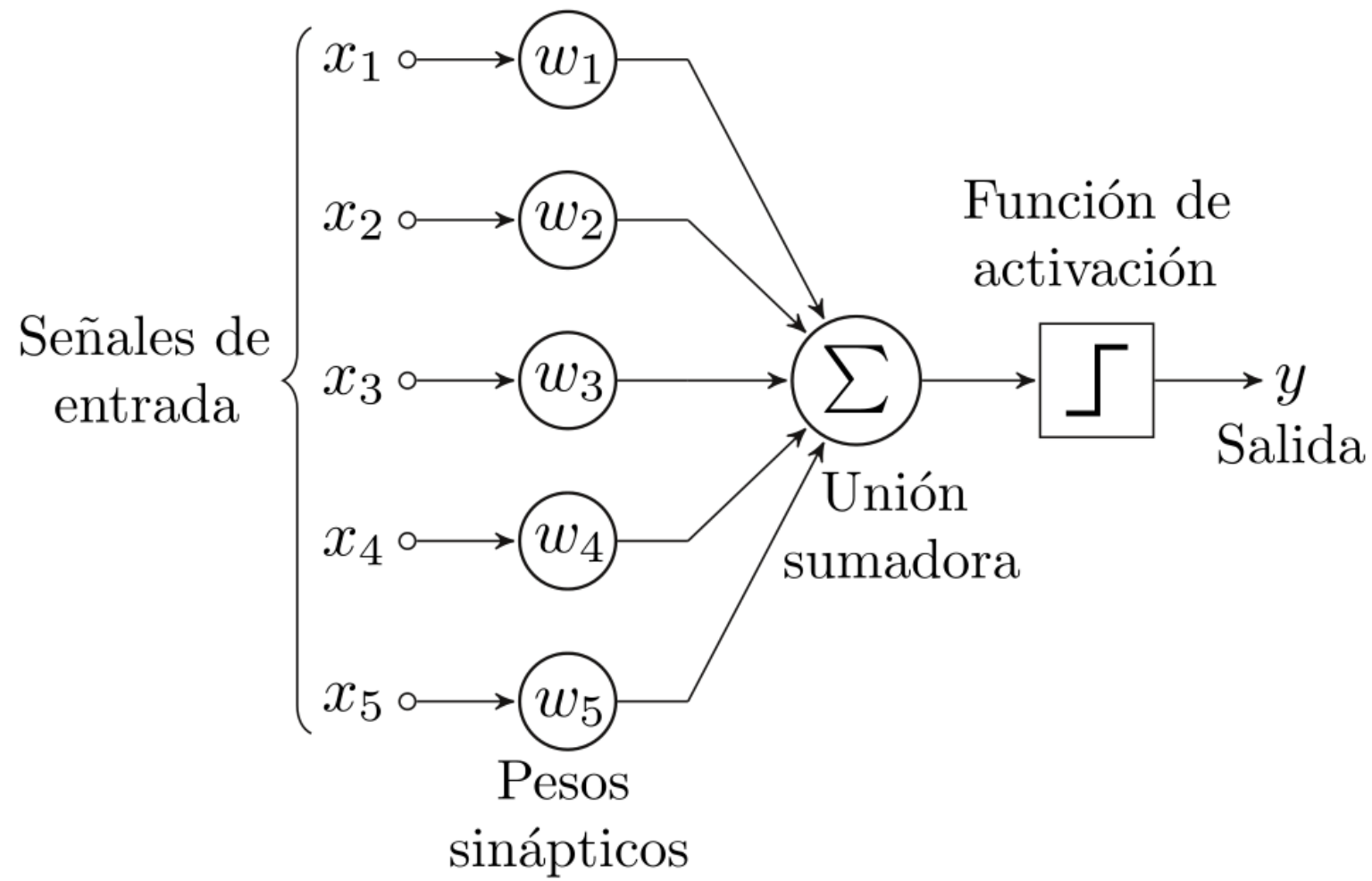


# Redes Neuronales Artificiales

Semestre Otoño 2022

Modulo 2



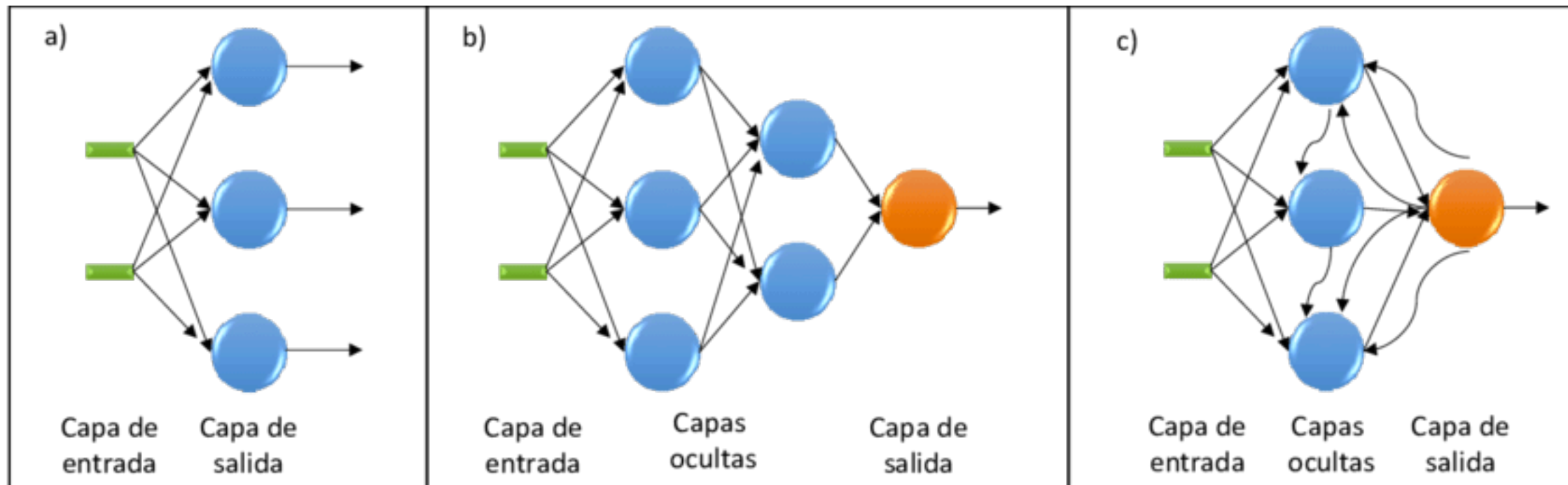
- Las funciones discriminantes lineales puras tienen limitaciones: las fronteras de decisión son hiperplanos.
- Con funciones no lineales  $\phi$  se pueden obtener fronteras de decisión complejas: mayor poder de expresión.
- ¿Cómo determinar o aprender las no linealidades  $\phi$  para lograr bajos errores de clasificación?

# Redes Multicapa


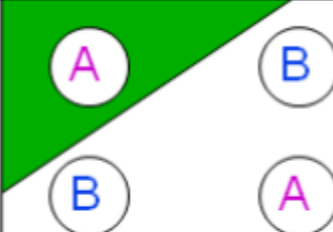
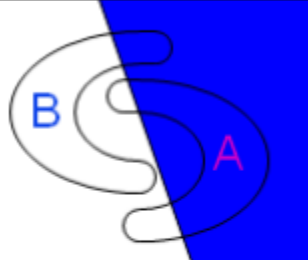

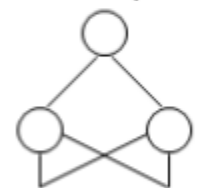
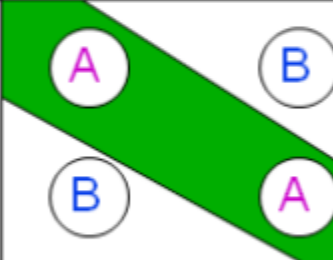
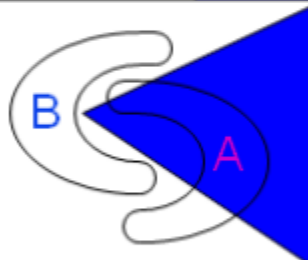
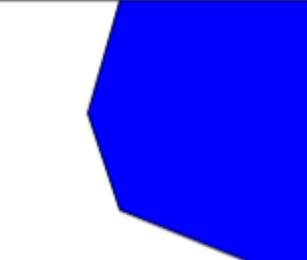

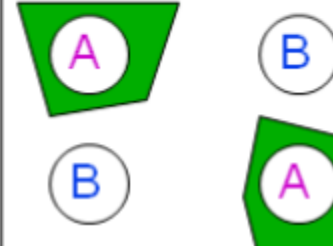

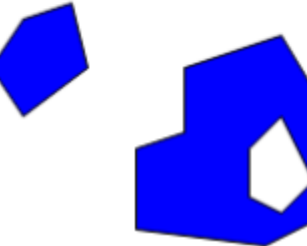
Son un tipo de arquitectura de redes neuronales artificiales compuestas por múltiples capas de neuronas interconectadas.

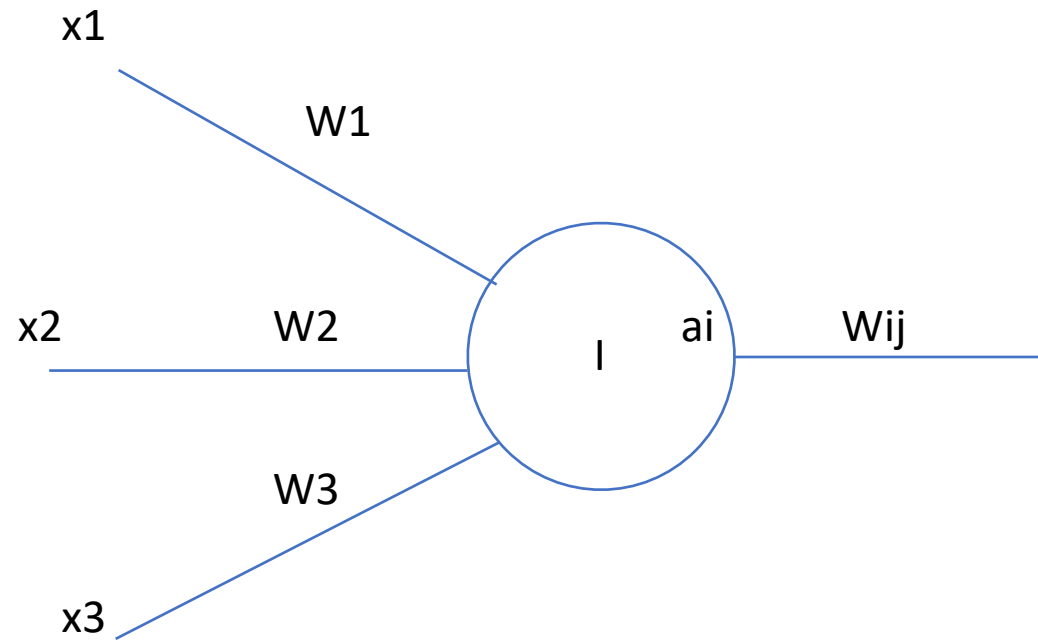
Estas capas se organizan en una estructura secuencial, donde cada capa se comunica con la siguiente, formando una secuencia desde la capa de entrada hasta la capa de salida.

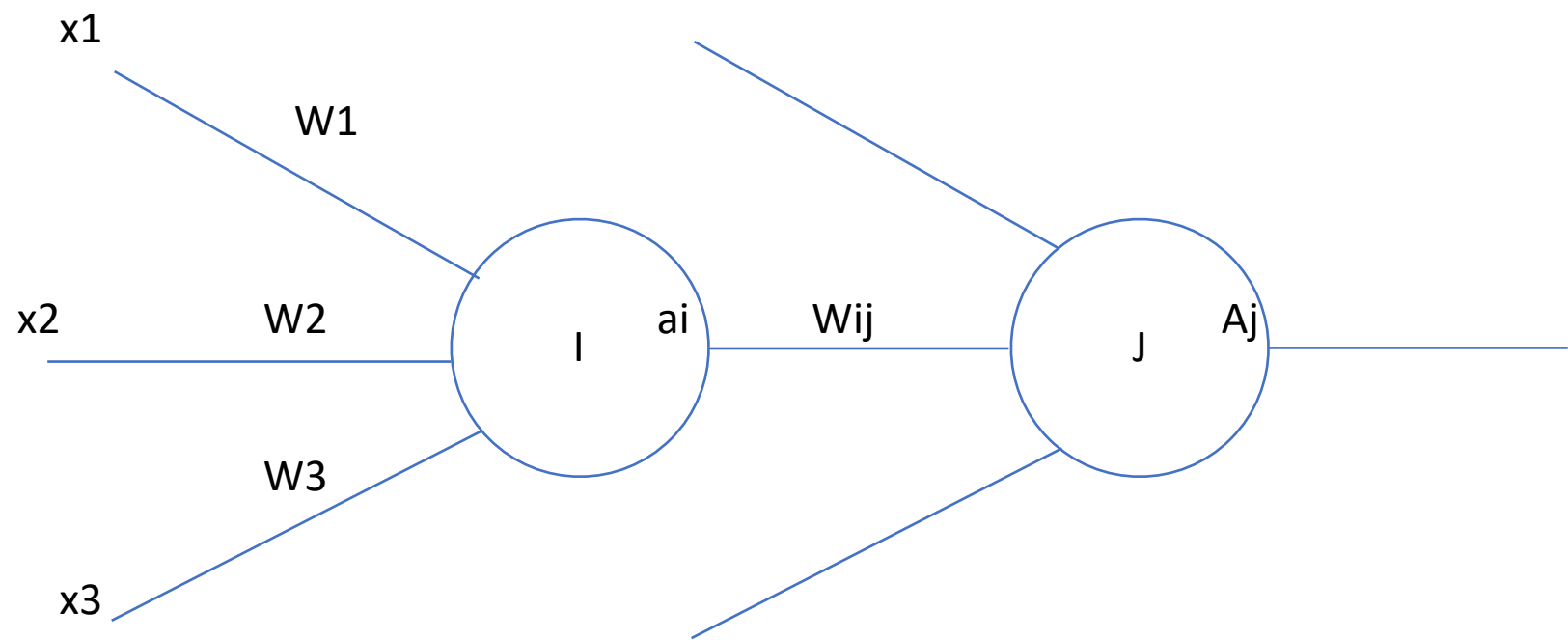
# Redes Multicapa



# Perceptrón Multicapa

<i>Estructura</i>	<i>Tipos de regiones de decisión</i>	<i>Problema XOR</i>	<i>Separación en clases</i>	<i>Formas regiones más generales</i>
<i>Una capa</i> 	<i>hemiplano limitado por hiperplano</i>			
<i>Dos capas</i> 	<i>Regiones convexas abiertas o cerradas</i>			
<i>Tres capas</i> 	<i>Arbitrarias (Complejidad limitada por N°. de Nodos)</i>			



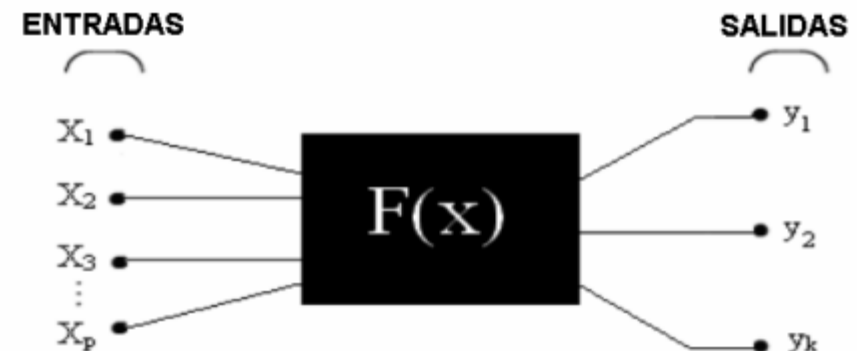




# Diagrama de caja negra

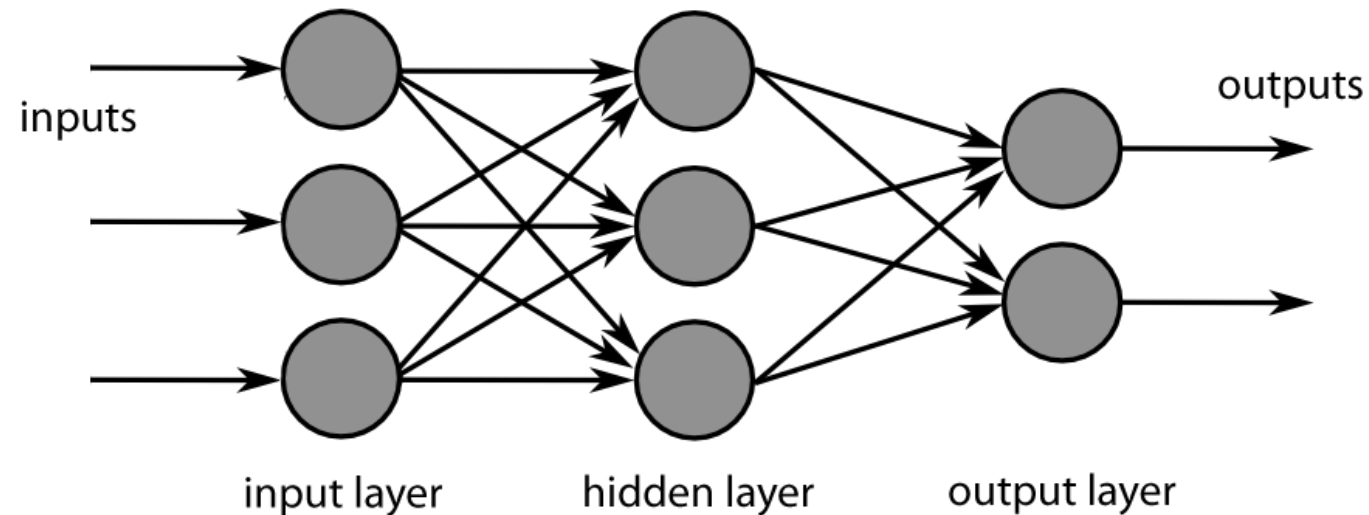
La arquitectura del Perceptrón Multicapa se puede observar que las múltiples entradas conectadas en la primera capa son mapeadas en las salidas en función de las distintas capas de neuronas intermedias y de los parámetros libres de la red.

Se puede ver como una caja negra que realiza una operación sobre las entradas, produciendo un rango de salidas en función de los parámetros libres.



# Redes Feed-forward

Las redes feed-forward, también conocidas como redes neuronales feed-forward o redes neuronales de propagación hacia adelante, son un tipo de arquitectura de redes neuronales artificiales donde la información fluye en una dirección, desde la capa de entrada hasta la capa de salida, sin ciclos ni retroalimentación.



# Funciones feedforward

- La forma general de las salidas de una red de tres capas es:

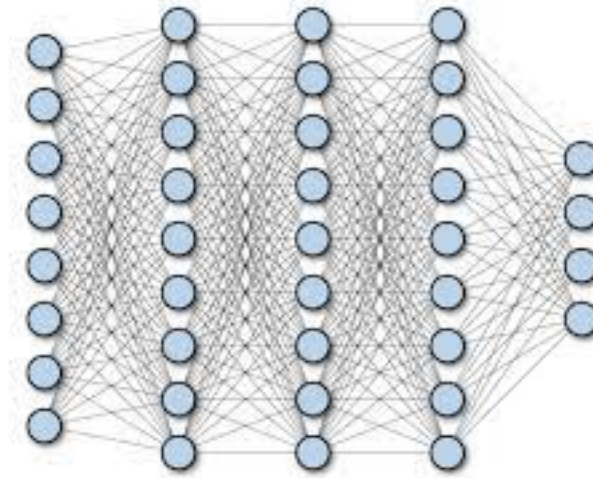
$$z_k(\mathbf{x}, \mathbf{w}) = f \left( \sum_{j=1}^{n_H} w_{kj}^{(2)} f \left( \sum_{i=1}^d w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

La evaluación de esta función se puede ver como una propagación de información en una red, hacia adelante (feedforward). Interesa saber la capacidad expresiva de este modelo: ¿Se puede aproximar cualquier función?

# Tipos Redes Multicapa

## Conexiones completamente conectadas (Fully Connected)

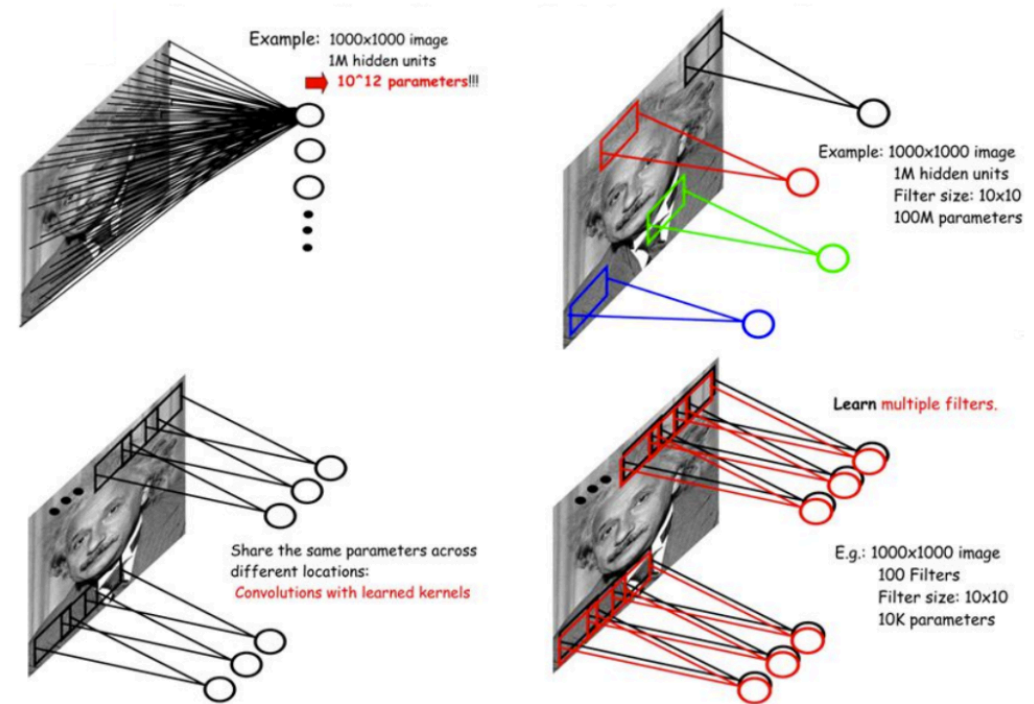
- En este tipo de conexión, cada neurona de una capa está conectada con todas las neuronas de la capa siguiente.
- También se conoce como conexión densa.
- En una red multicapa completamente conectada, todas las salidas de las neuronas de una capa se convierten en entradas de las neuronas de la capa siguiente.



# Tipos Redes Multicapa

## Conexiones locales (Local Connections)

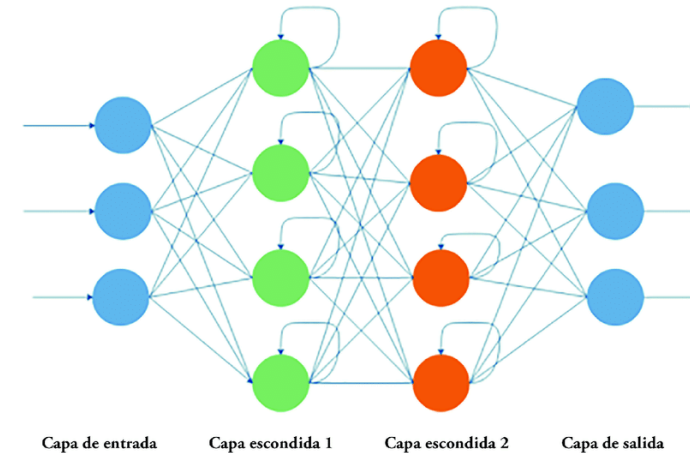
- Estas conexiones son típicas en arquitecturas como las redes neuronales convolucionales (CNN).
- Las neuronas de una capa están conectadas solo con un subconjunto de las neuronas de la capa siguiente, en lugar de estar conectadas con todas las neuronas.
- Esto se logra utilizando filtros o ventanas deslizantes que se aplican a la entrada para extraer características locales.



# Tipos Redes Multicapa

## Conexiones recurrentes (Recurrent Connections)

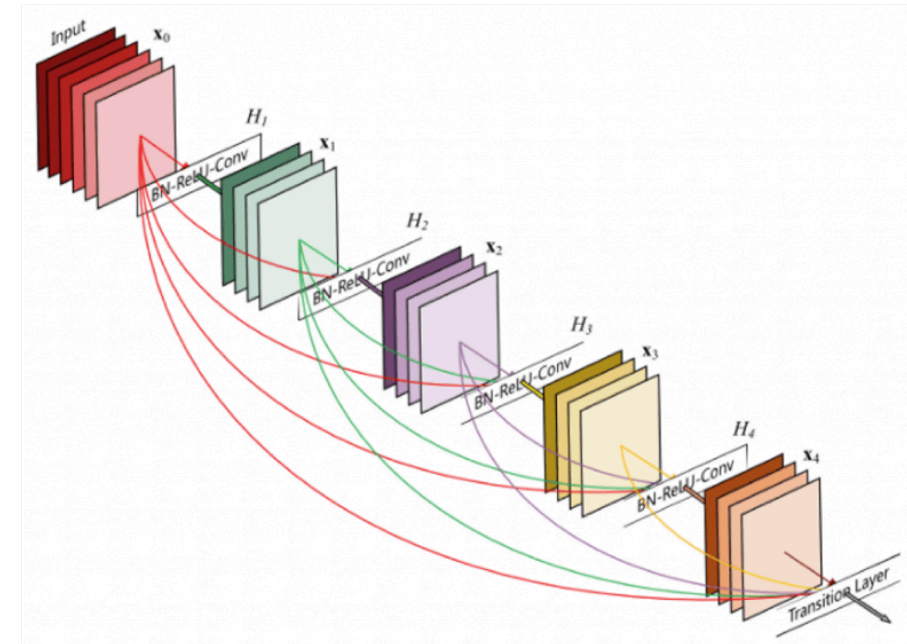
- En las redes recurrentes, las conexiones se establecen entre las neuronas de una capa y las neuronas de la misma capa o de capas anteriores.
- Esto permite que la información fluya en bucles, lo que permite a la red tener memoria o mantener estados internos.
- Las conexiones recurrentes son útiles en tareas donde la secuencia o el orden de los datos es importante, como en el procesamiento de secuencias de texto o en las redes neuronales de memoria a corto plazo (LSTM).



# Tipos Redes Multicapa

## Conexiones saltadas (Skip Connections):

- Las conexiones saltadas, también conocidas como conexiones residuales, se utilizan en arquitecturas como las redes neuronales residuales (ResNet).
- En este tipo de conexión, la salida de una capa se agrega directamente a la entrada de una capa posterior en lugar de pasar por un procesamiento adicional.
- Esto permite que la información fluya directamente a través de la red sin obstrucciones, facilitando el entrenamiento de redes neuronales profundas.



\* BRAIN-  
STATE-IN-A-  
BOX

\* Memorias  
Asociativas  
Lineales

\* Additive  
Grossberg

\* Shunting  
Grossberg

\* RED de  
HOPFIELD

\* Maquina de  
Cauchy

\* Maquina de  
Boltzmann

Conexiones  
Laterales

Adeline  
Madelaine

Perceptron  
Simple

Drive-  
Reinforcement

Conexiones  
Laterales

Learning  
Vector  
Quantizer  
(LVQ)

Topology  
Preserving  
Map (TPM)

Conexiones  
Laterales

Bidirectional  
Associative  
Memory (BA)

Adaptive B

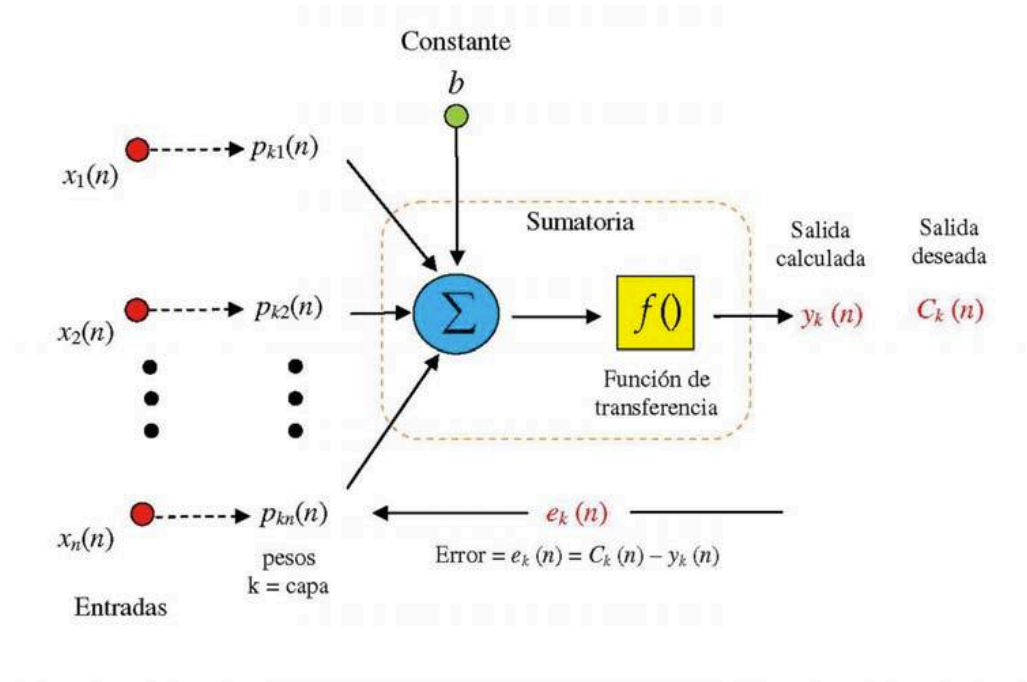
Temporal  
Associative  
Memory (TA)

Fuzzy  
Associative  
Memory (FA)



# Algoritmos retropropagación

El algoritmo de retropropagación (backpropagation) es el principal algoritmo utilizado para entrenar redes neuronales multicapa, permitiendo ajustar los pesos y sesgos de las neuronas en función de la función de pérdida o error.



# Algoritmos retropropagación

## 1. Inicialización de pesos

Se asignan valores iniciales aleatorios a los pesos y sesgos de todas las conexiones de la red neuronal. Estos valores se pueden inicializar de diferentes maneras, como con una distribución normal o uniforme.

## 2. Propagación hacia adelante (Forward Propagation):

- Se pasa un conjunto de datos de entrenamiento a través de la red neuronal.
- Los datos se propagan desde la capa de entrada hasta la capa de salida, pasando por las capas ocultas, utilizando los pesos y sesgos actuales de la red.
- Cada neurona realiza una combinación lineal de las entradas ponderadas por los pesos, y luego aplica una función de activación para producir la salida.

## 2. Cálculo de la función de pérdida:

- Se compara la salida predicha por la red neuronal con la salida deseada para el conjunto de entrenamiento.
- Se calcula la función de pérdida, que mide la discrepancia entre las salidas predichas y las salidas deseadas.
- El objetivo es minimizar esta función de pérdida durante el entrenamiento.

# Algoritmos retropropagación

## 4. Retropropagación del error:

- El error calculado en el paso anterior se propaga hacia atrás a través de la red neuronal.
- Comienza en la capa de salida y se calcula el gradiente del error con respecto a los pesos y sesgos en cada capa.
- Esto se realiza utilizando el algoritmo de diferenciación automática, donde el gradiente se calcula mediante la regla de la cadena.

## 5. Ajuste de pesos y sesgos:

- Utilizando los gradientes calculados en el paso anterior, se actualizan los pesos y sesgos de todas las conexiones de la red neuronal.
- Se realiza un ajuste utilizando un algoritmo de optimización, como el descenso de gradiente, que toma en cuenta el gradiente y la tasa de aprendizaje para determinar el tamaño del paso de ajuste.

## 6. Repetición del proceso:

- Los pasos 2-5 se repiten para cada conjunto de entrenamiento en el conjunto de datos de entrenamiento.
- Este proceso de propagación hacia adelante, cálculo del error, retropropagación del error y ajuste de pesos se repite durante múltiples épocas hasta que se alcance una condición de finalización, como un número máximo de épocas o cuando la función de pérdida alcanza un umbral aceptable.

# Algoritmos retropropagación

El algoritmo de retropropagación permite que el error se propague hacia atrás a través de la red neuronal, actualizando los pesos y sesgos de manera iterativa para reducir el error en la salida y mejorar el rendimiento de la red.

A medida que se repite el proceso, la red neuronal ajusta los pesos para encontrar configuraciones que minimicen la función de pérdida y produzcan salidas más precisas.

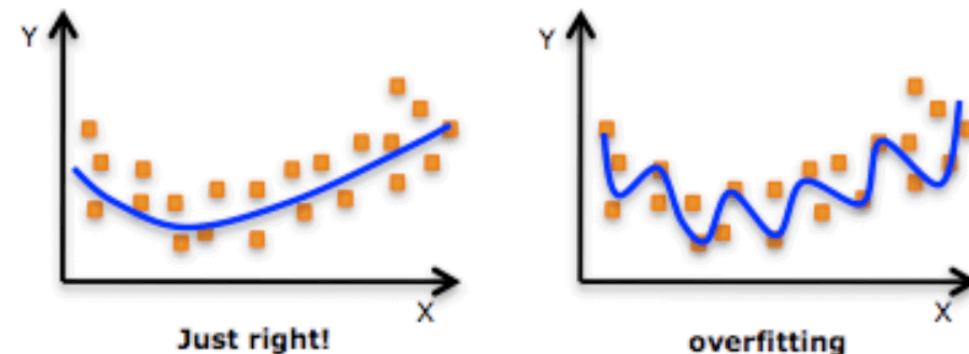
# Regularizacion



# Overfitting

Fenómeno en el aprendizaje automático donde un modelo de machine learning se ajusta demasiado a los datos de entrenamiento, lo que resulta en un rendimiento deficiente al generalizar a nuevos datos.

El sobreajuste ocurre cuando un modelo se vuelve demasiado complejo y capta tanto los patrones reales de los datos de entrenamiento como el ruido o las fluctuaciones aleatorias presentes en esos datos. Como resultado, el modelo se adapta demasiado específicamente a los datos de entrenamiento y no puede generalizar correctamente a nuevos datos que no ha visto antes.



# Overfitting

signos comunes de sobreajuste

- Un rendimiento excelente en los datos de entrenamiento, pero un rendimiento deficiente en los datos de prueba o validación.
- Diferencias significativas entre el rendimiento en los datos de entrenamiento y en los datos de prueba o validación.
- El modelo muestra una alta varianza y es muy sensible a pequeños cambios en los datos de entrenamiento.

# Overfitting

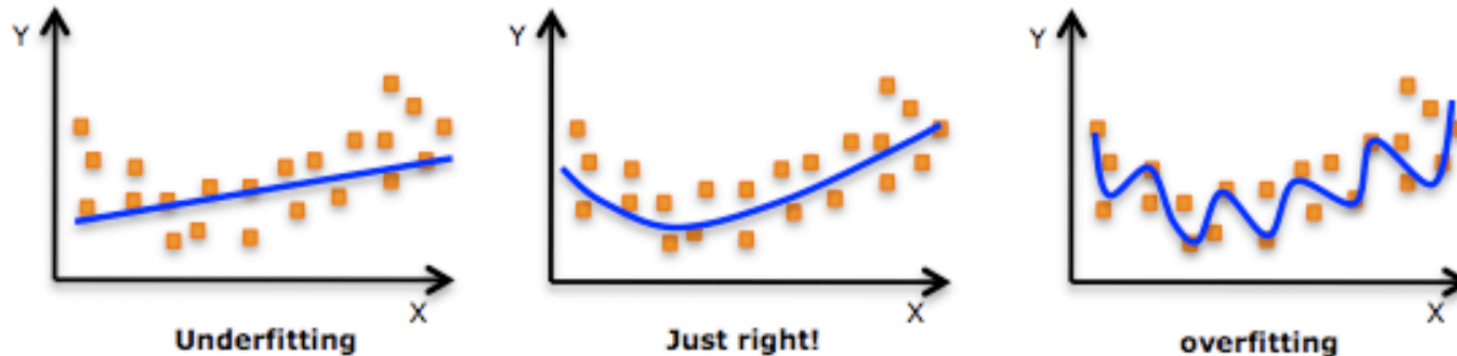
## Causas

- La complejidad excesiva del modelo, como tener demasiados parámetros o capas.
- Una cantidad insuficiente de datos de entrenamiento.
- Datos de entrenamiento ruidosos o con errores.
- Sesgo en los datos de entrenamiento, donde los datos no representan adecuadamente la distribución del mundo real.



# Underfitting

- Ocurre cuando un modelo es demasiado simple o tiene una capacidad insuficiente para representar la complejidad de los datos subyacentes.
- En lugar de aprender patrones útiles, el modelo subajustado generaliza demasiado y produce resultados inexactos o insatisfactorios.



# Underfitting

signos comunes de subaste

- Rendimiento deficiente tanto en los datos de entrenamiento como en los datos de prueba o validación.
- Diferencias mínimas o inexistentes entre el rendimiento en los datos de entrenamiento y en los datos de prueba o validación.
- El modelo muestra una alta sesgo y no es capaz de capturar la estructura subyacente de los datos.

# Underfitting

## Causas

- Utilizar un modelo demasiado simple con pocos parámetros o capas.
- No proporcionar suficientes datos de entrenamiento para que el modelo pueda aprender correctamente.
- Datos de entrenamiento ruidosos o con errores que dificultan la identificación de los patrones relevantes.

# Que es la regularización?

Técnica utilizada en el aprendizaje automático para evitar el sobreajuste (overfitting) de un modelo a los datos de entrenamiento.

La regularización se basa en la idea de agregar una penalización adicional a la función de pérdida del modelo durante el entrenamiento.

# Regularización

Existen dos técnicas comunes de regularización utilizadas en modelos de aprendizaje automático:

1. **Regularización L1 (Lasso):** Agrega una penalización proporcional al valor absoluto de los coeficientes de los pesos. Esto favorece la reducción de coeficientes a cero y, por lo tanto, permite la selección automática de características, eliminando algunas características irrelevantes.
3. **Regularización L2 (Ridge):** Agrega una penalización proporcional al cuadrado de los coeficientes de los pesos. Esta penalización mantiene los coeficientes pequeños sin eliminar completamente ninguno de ellos, lo que puede ser útil cuando todas las características son potencialmente relevantes.

### L1 Regularization

$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

### L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^M W_j^2}_{\text{Regularization Term}}$$

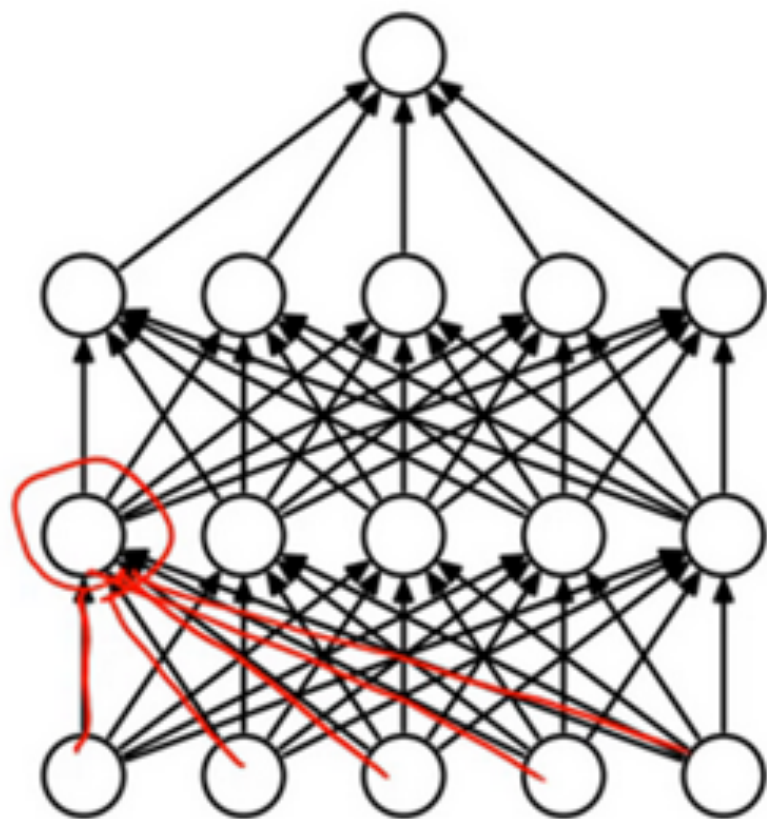


Aun hay mas !

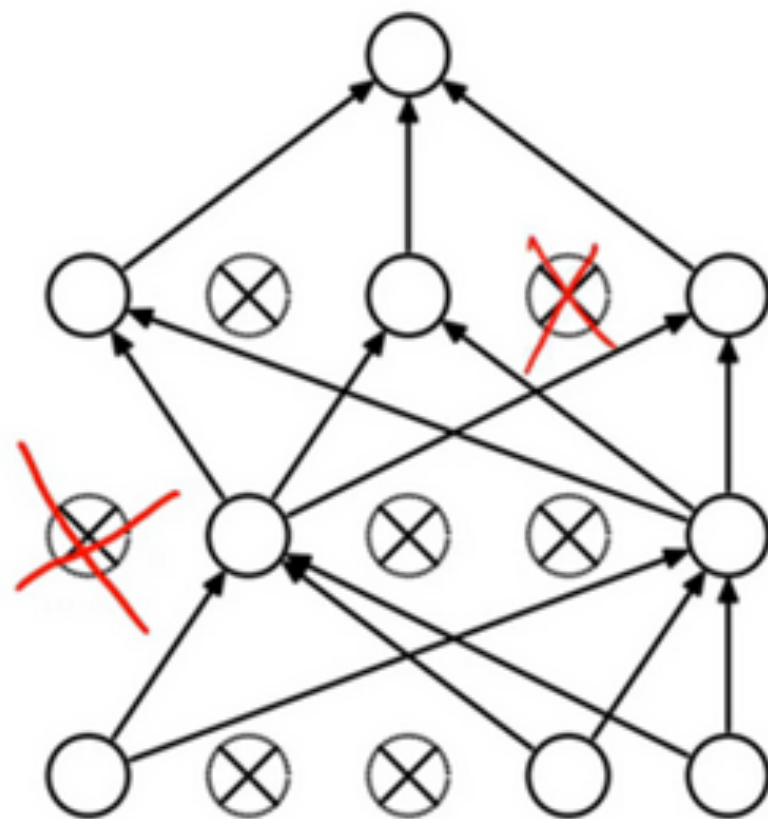
# Dropout

- El Dropout es una técnica de regularización utilizada en redes neuronales para evitar el sobreajuste y mejorar la generalización del modelo.
- Consiste en desactivar aleatoriamente un porcentaje de las neuronas (unidad de procesamiento) durante el entrenamiento de la red.





(a) Standard Neural Net



(b) After applying dropout.

# ElasticNet

- Es una combinación de la regularización L1 y L2. Aplica una penalización tanto a los valores absolutos de los coeficientes (L1) como a los valores cuadrados de los coeficientes (L2). Esto permite beneficiarse de las propiedades de ambas regularizaciones.

$$\frac{\sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2}{2n} + \lambda \left( \alpha \sum_{j=1}^p |\beta_j| + \frac{1 - \alpha}{2} \sum_{j=1}^p \beta_j^2 \right)$$

# Early Stopping

- Técnica que detiene el entrenamiento del modelo cuando el rendimiento en un conjunto de validación deja de mejorar.
- Se utiliza un conjunto de validación separado para evaluar periódicamente el rendimiento del modelo durante el entrenamiento.
- Esto ayuda a evitar el sobreajuste y a encontrar un equilibrio entre el rendimiento en el conjunto de entrenamiento y la capacidad de generalización del modelo.

# Optimizadores

Algoritmos utilizados durante el entrenamiento de la red neuronal para ajustar los pesos y sesgos de manera iterativa, con el objetivo de minimizar la función de pérdida y mejorar el rendimiento de la red.

Los optimizadores determinan cómo se actualizan los pesos y sesgos en cada iteración del proceso de entrenamiento.

# Optimizadores

- **Descenso de gradiente estocástico (SGD, Stochastic Gradient Descent):** En cada iteración, se calcula el gradiente de la función de pérdida con respecto a los pesos y sesgos y se actualizan mediante un paso en la dirección opuesta al gradiente, multiplicado por una tasa de aprendizaje.
- **Momento (Momentum):** El optimizador de momento mejora el descenso de gradiente estocástico al agregar un término de momento que acumula una fracción de la actualización anterior de los pesos. Esto ayuda a acelerar el proceso de aprendizaje y superar los mínimos locales en la superficie de la función de pérdida.
- **RMSprop (Root Mean Square Propagation):** adapta la tasa de aprendizaje para cada parámetro individual de la red neuronal. Utiliza un promedio en ejecución de los cuadrados de los gradientes anteriores para ajustar la tasa de aprendizaje de manera adaptativa. Esto permite un aprendizaje más rápido en direcciones más empinadas y un aprendizaje más lento en direcciones más planas de la función de pérdida.
- **Adam (Adaptive Moment Estimation):** Ajusta adaptativamente las tasas de aprendizaje para cada parámetro individual y utiliza una versión corregida del momento. Adam es conocido por su eficiencia computacional y buen rendimiento en una amplia gama de problemas.

# Arquitectura en redes neuronales

La arquitectura de una red neuronal puede incluir varios aspectos, como:

- **Tipo de capas:** Se eligen diferentes tipos de capas, como capas densas (fully connected), capas convolucionales, capas recurrentes, etc., según el tipo de datos y la tarea a realizar.
- **Número de capas:** Se define cuántas capas se incluirán en la red. Puede haber una o más capas ocultas, además de la capa de entrada y la capa de salida.
- **Tamaño de las capas:** Se especifica el número de neuronas en cada capa. Un mayor número de neuronas puede permitir una mayor capacidad de representación, pero también puede aumentar la complejidad y el costo computacional.
- **Conexiones entre las capas:** Se establecen las conexiones entre las capas. Por ejemplo, en una red neuronal completamente conectada, todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente. En una red convolucional, las conexiones están basadas en regiones locales y comparten pesos.
- **Funciones de activación:** Se seleccionan las funciones de activación que se utilizarán en cada capa para introducir no linealidad en el modelo.