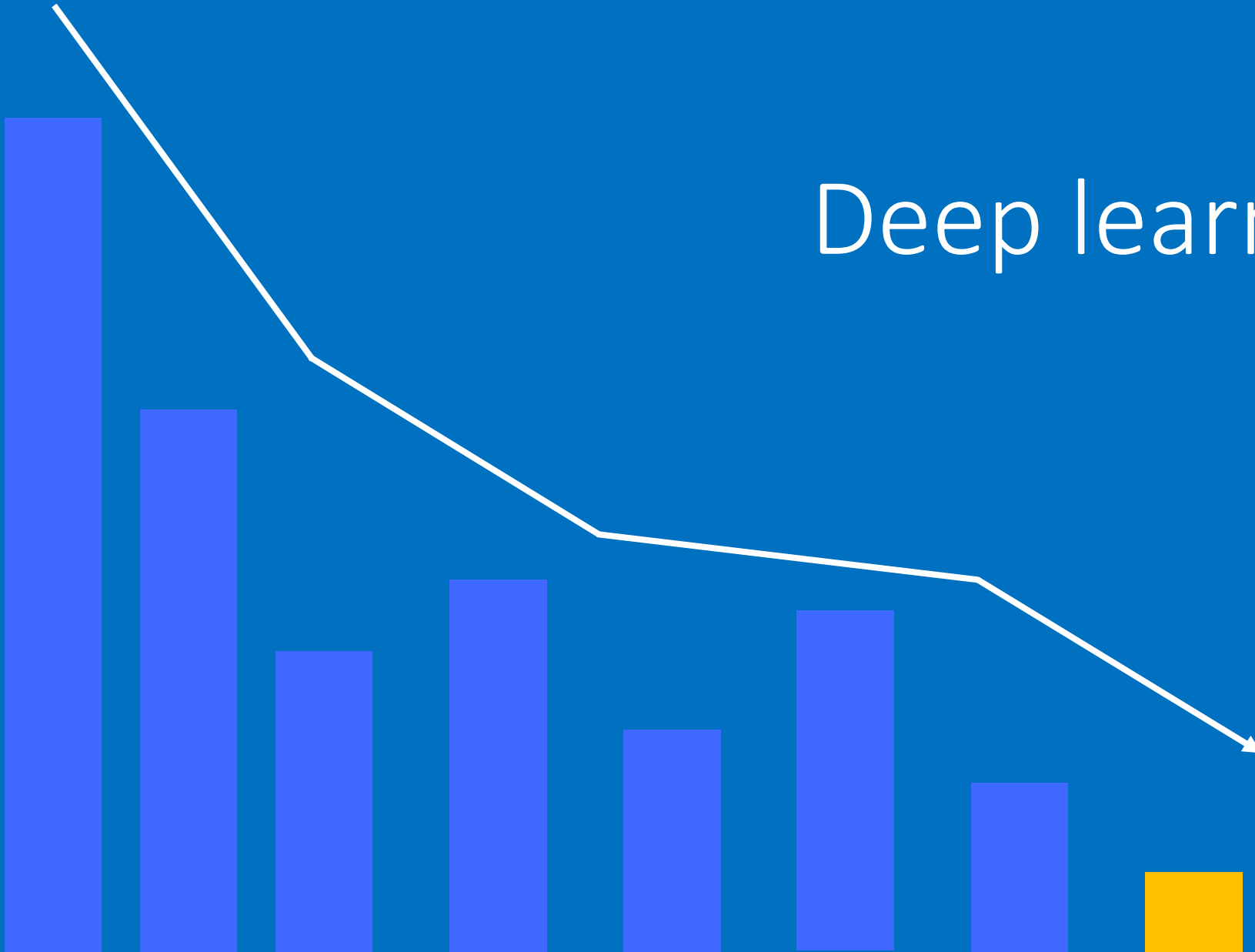


# Deep learning & Neural Nets using R

**Introducción a Data  
Science con R**



# PRESENTACIÓN

- ✓ Nombre y ocupación
- ✓ ¿Porqué tomar un curso de Data Science?
- ✓ Expectativas del módulo

# CONTENIDO TEMÁTICO

## **5. Introducción a redes neuronales artificiales**

- Diferencias entre Machine Learning y Estadística
- Antecedentes
- Componentes
- Tipos y clasificaciones
- Metodología
- Aplicaciones

## **6. Introducción a Deep learning**

- Diferencias entre Machine Learning y Deep learning
- Introducción a Tensorflow
- Aplicaciones

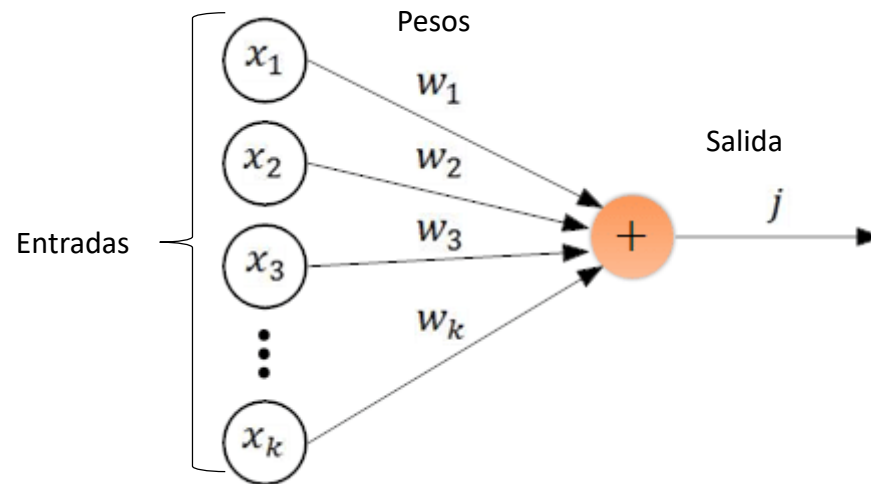
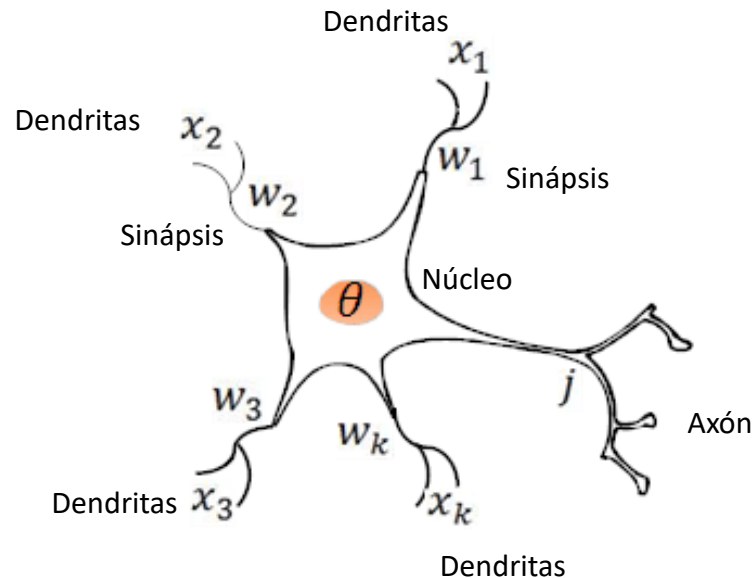
# ML vs Estadística

Machine Learning	Estadística
Se enfoca en predecir datos futuros y evaluación del modelo	Se enfoca en la explicación e inferencia del fenómeno
El modelo es un algoritmo que puede aprender de los datos sin que sean explícitamente programados	El modelo es una representación de las relaciones entre las variables
Es un subdominio de la inteligencia artificial	Es un subdominio de las matemáticas.

# 5 – INTRODUCCIÓN A REDES NEURONALES

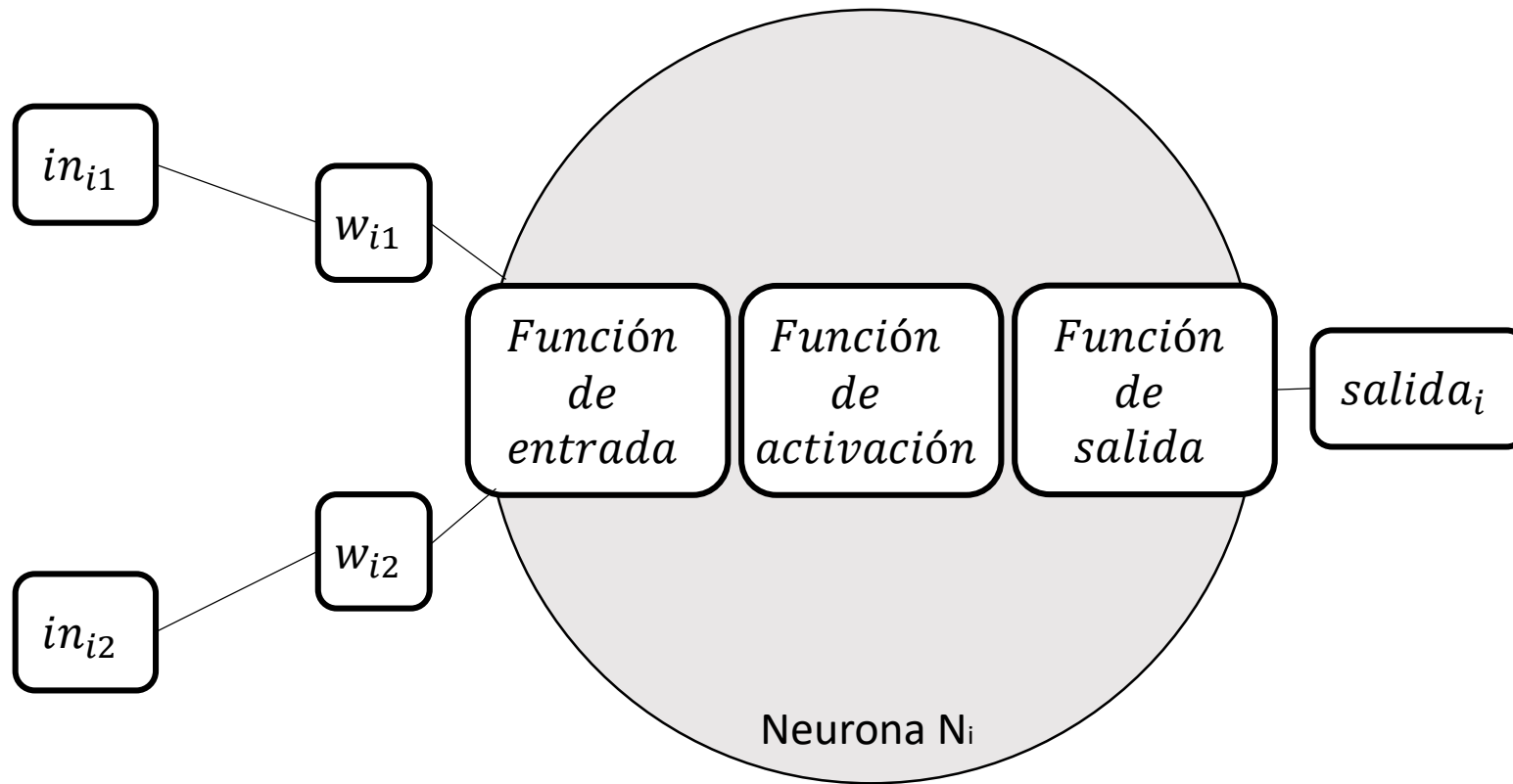
## 5.1 Antecedentes

Las Redes Neuronales Artificiales son algoritmos que emulan el comportamiento de las neuronas biológicas. Utilizan la información obtenida del entorno y, mediante procesos de aprendizaje, logran retener dicha información y utilizarla para adecuarse lo mejor posible a la estructura de los datos originales.



# 5 – INTRODUCCIÓN A REDES NEURONALES

## 5.2 Componentes



Elementos principales:

- Capas
- Pesos
- Función de entrada
- Función de activación
- Función de salida
- Estandarización de los datos

## **CAPAS**

1. Capas de entrada: aquellas que reciben la información de fuentes externas.
2. Capas ocultas: neuronas que vinculan las entradas con la salida mediante conexiones complejas.
3. Capa de salida: aquella que transfiere la información hacia el exterior.

## **PESOS**

Muestran que tan fuerte o débil es la participación (efecto) de una variable en la salida.

Bias: variable que absorbe todo el error que no se puede medir.

## **FUNCIONES**

--FUNCIÓN DE ENTRADA--

Aquella estructura que adoptarán los datos para formar el vector de entrada a la red neuronal.

--FUNCIÓN DE ACTIVACIÓN--

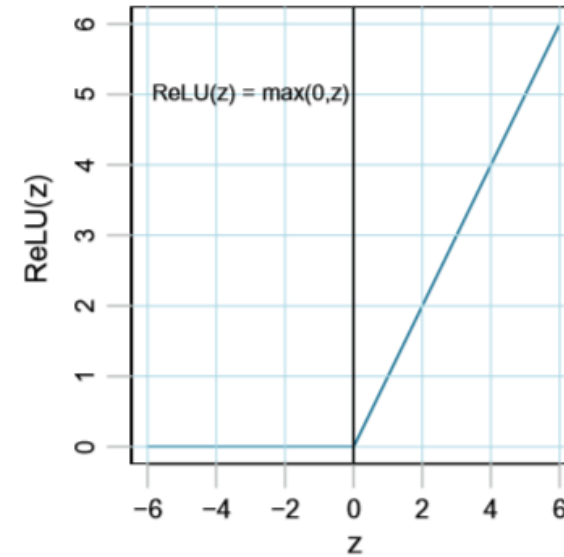
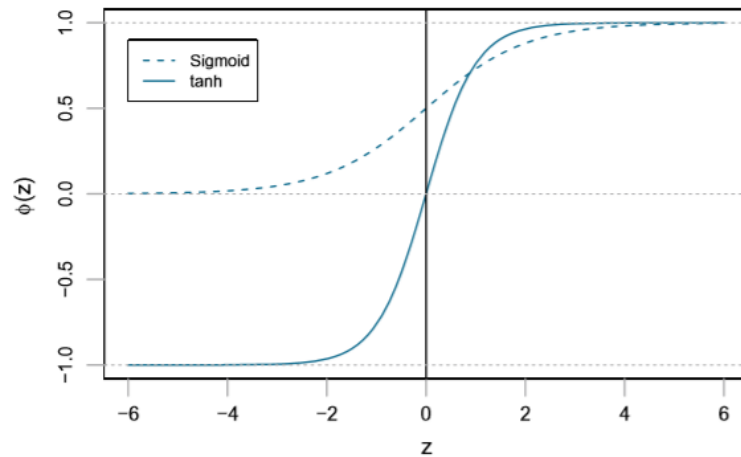
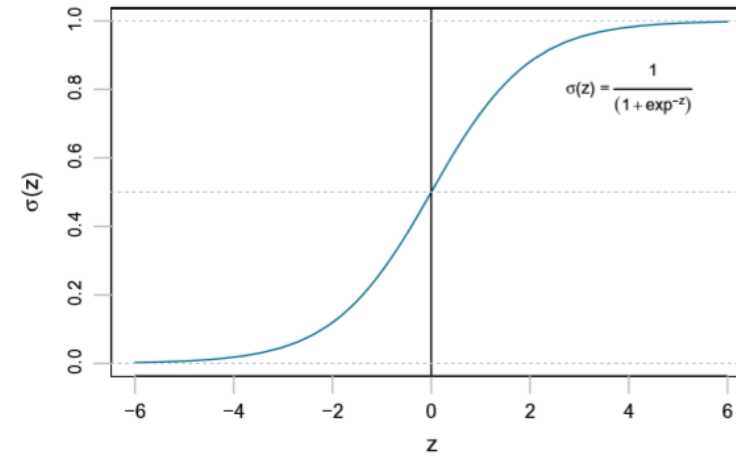
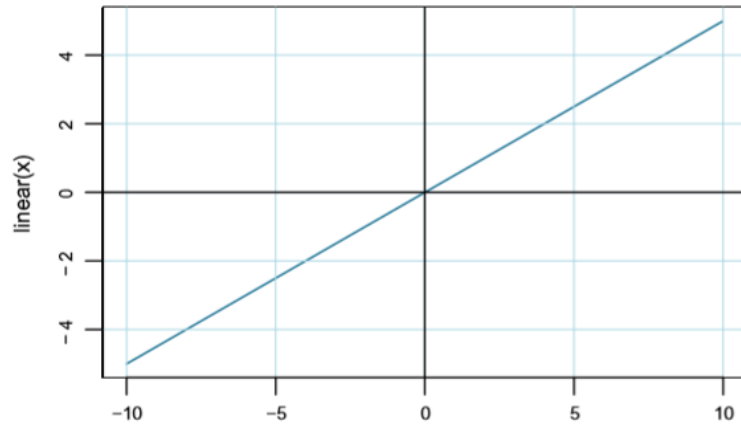
Es una función que convierte una entrada en una salida la red neuronal tiene que cumplir con una serie de características:

1. Tiene que ser diferenciable
2. Debe ser simple y rápido procesamiento

--FUNCIÓN DE SALIDA--

Determina el valor que será transferido a la siguiente neurona en cuestión

# Funciones de activación más comunes





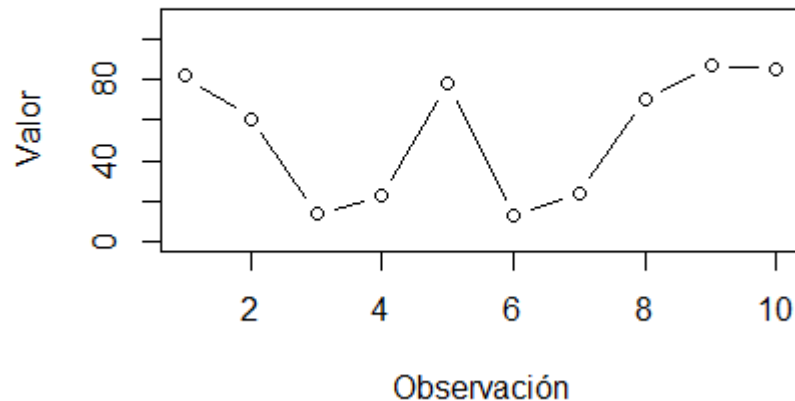
## --ESCALAMIENTO DE DATOS--

Se debe realizar una transformación de las variables para que la magnitud de las mismas no provoque un error, es decir, sesgar los resultados.

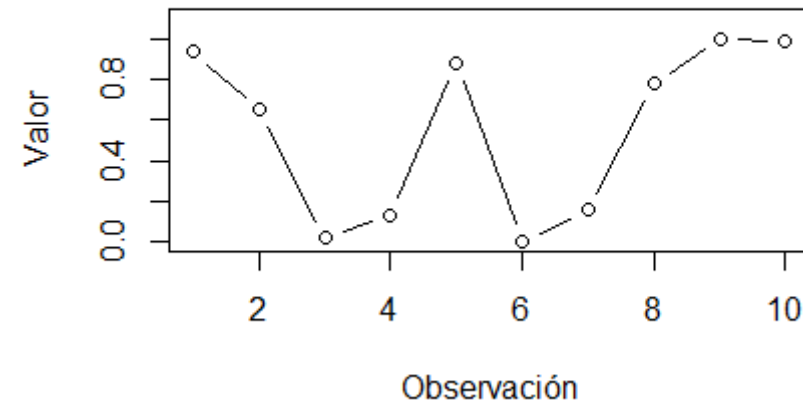
La formula para escalar los datos es irrelevante, siempre y cuando se mantenga una congruencia entre el procedimiento. Asimismo, dependerá de la distribución que tengan los datos de entrada.

$$A_i = \frac{A_{0i} - \min A_{0i}}{\max A_{0i} - \min A_{0i}}$$

**Valores originales**



**Valores escalados**

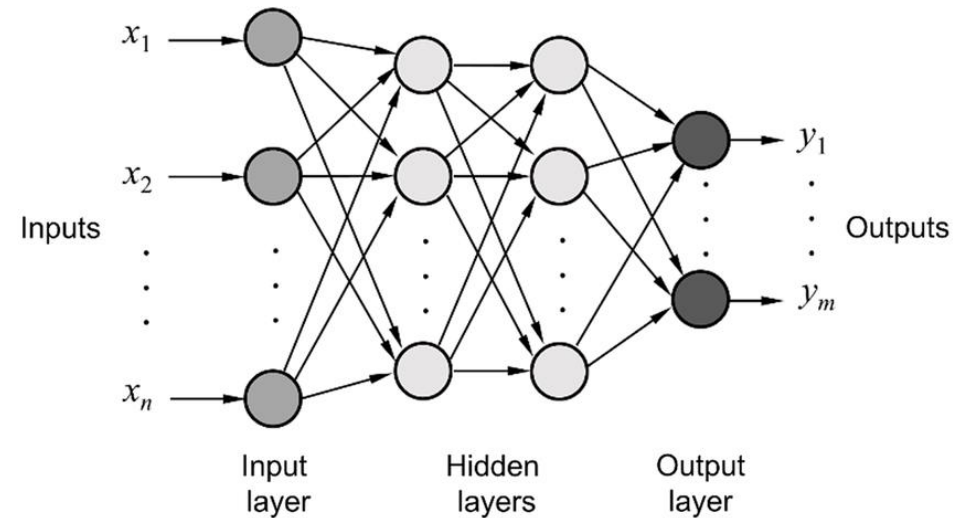
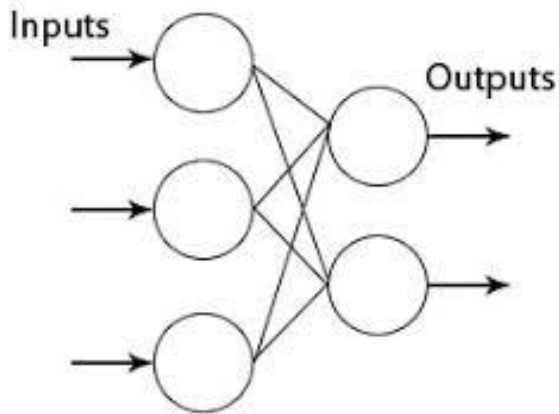


# 5 – INTRODUCCIÓN A REDES NEURONALES

## 5.3 Tipos y clasificaciones

### POR SU ARQUITECTURA

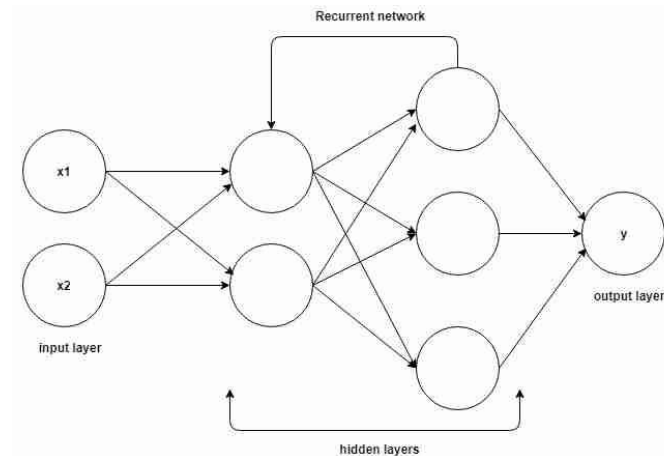
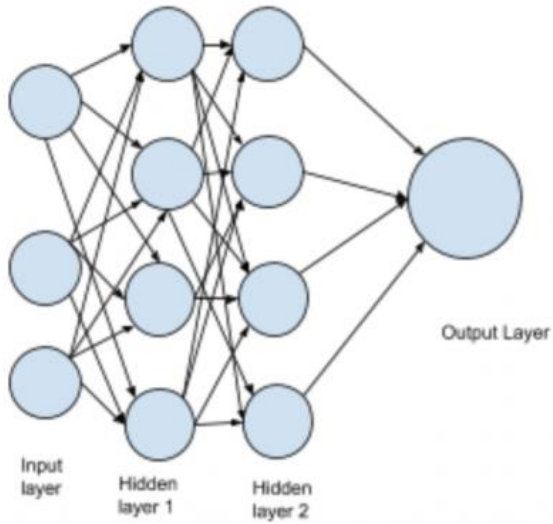
- Redes monocapa
- Redes multicapa



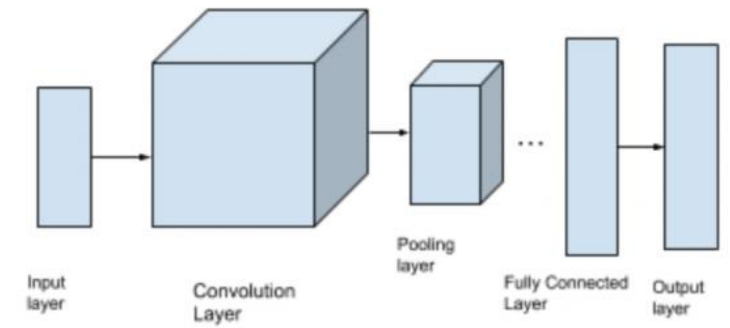
## POR FLUJO DE INFORMACIÓN

- Feedforward
- Convolutivas
- Recurrentes

Regular Neural Network



Convolutional Neural Network



## POR SU APRENDIZAJE

- Supervisadas

Tienen un punto de comparación

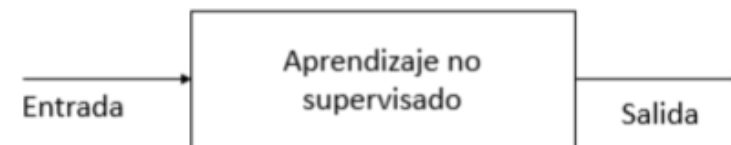
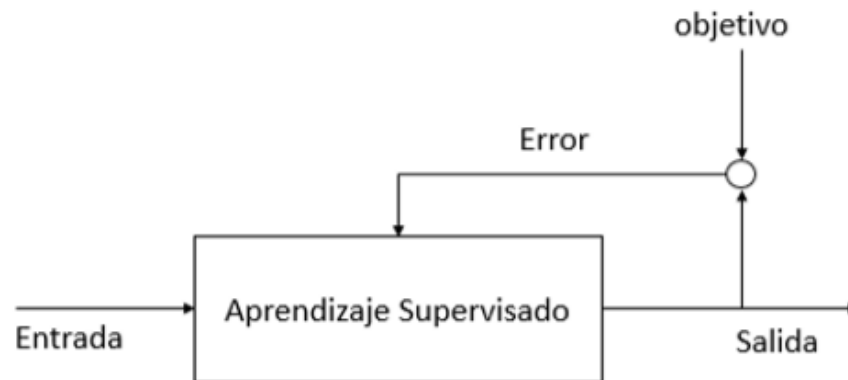
Backpropagation

- No supervisadas

No tienen punto de comparación

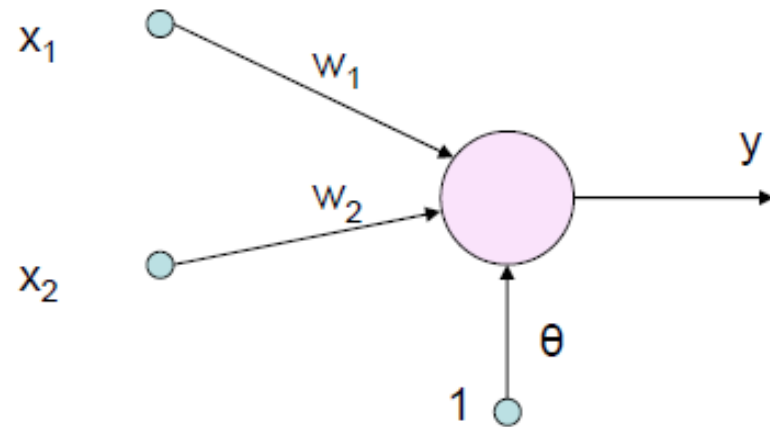
- Por refuerzo

- Indica si el valor cae dentro de un rango de aceptación



# Perceptrón: Estructura básica de red neuronal

- Frank Rosenblatt desarrolló el perceptrón en 1960. Esta estructura es conocida como la forma más primitiva de red neuronal.



Originalmente para resolver problemas de clasificación con variables que son linealmente separables.

Regla de aprendizaje del perceptron

$$W^{nuevo} = W^{viejo} + e * Entrada$$

$$b^{nuevo} = b^{viejo} + e$$

$$e = objetivo - función\ activación$$

$$y = f_{Activación}(x_1 * w_1 + x_2 * w_2 + \theta)$$

**Backpropagation:** aprender de los errores, ajustando el valor de los pesos en cada capa para minimizar la variación.

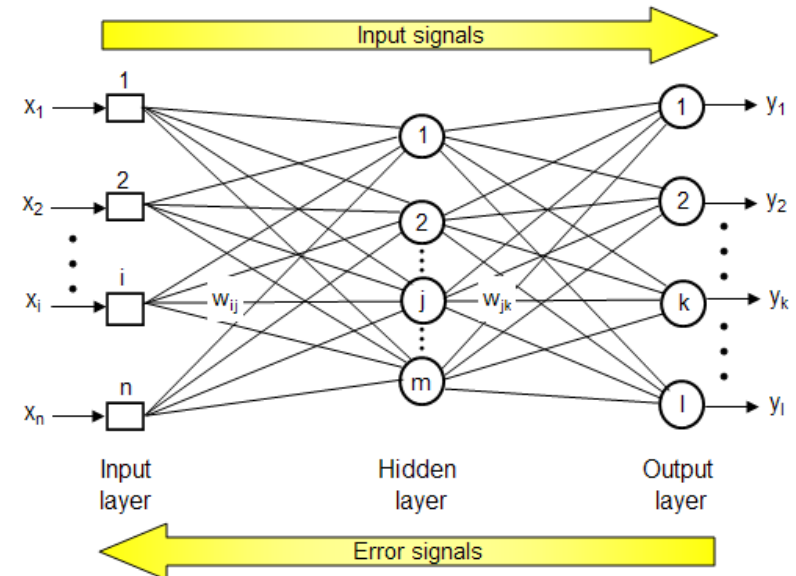
Por medio de la técnica del gradiente descendiente, la cual se convierte en una derivada parcial de la función de error con respecto al peso de la neurona.

$$\frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}}$$

Se dice que una red neuronal ha aprendido cuando la modificación de los pesos ha terminado. Es decir:

$$\frac{dw_{ij}}{dt} = 0$$

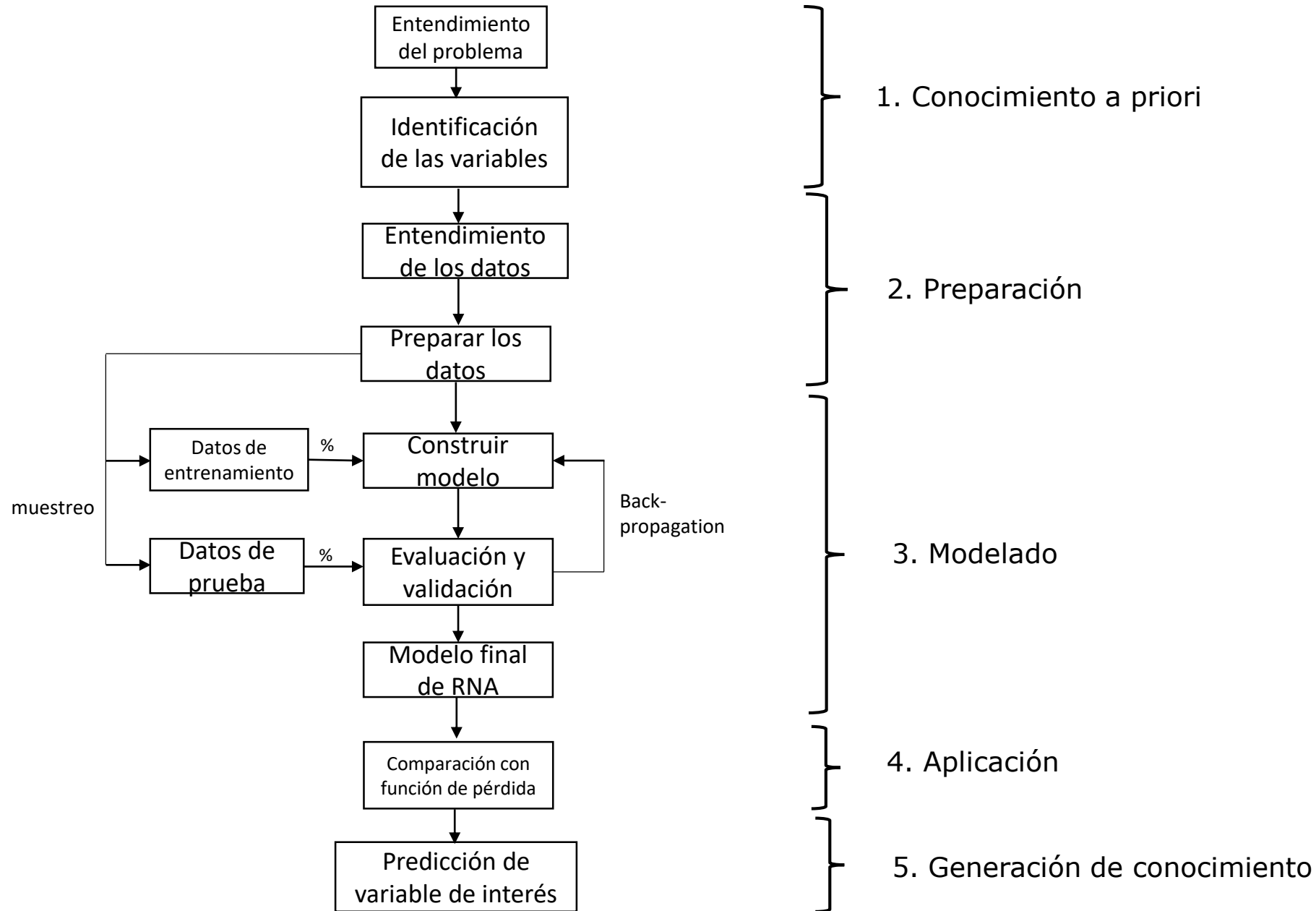


# 5 – INTRODUCCIÓN A REDES NEURONALES

## 5.4 Metodología

La metodología de diseño de una red neuronal se basa, principalmente, en los pasos de la minería de datos.

1. Conocimiento a priori
2. Preparación de los datos
3. Modelado
4. Aplicación
5. Generación de conocimiento





# Datos de entrenamiento y de prueba

- **Conjunto de entrenamiento:** en el cual se podrán ajustar los parámetros del modelo, en el caso de las redes neuronales, los pesos y bias.
- **Conjunto de validación:** En donde se verifican los parámetros
- **Conjunto de prueba:** como su nombre lo indica, se probará que los parámetros encontrados en el entrenamiento sean adecuados. Datos que el modelo no ha “visto”, para garantizar que haya aprendido y no solo haya recordado la distribución de los datos.
- Cross-validation o validación cruzada

# Validación cruzada



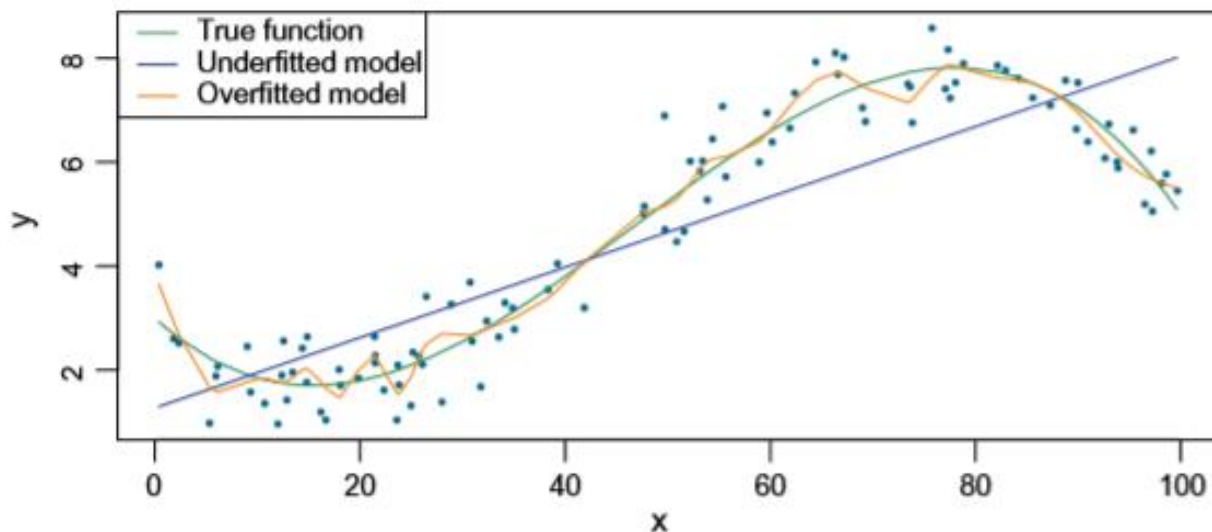
- Son técnicas para dividir los datos en los conjuntos de entrenamiento, validación y prueba.
1. The validation set approach: dividir el conjunto de manera aleatoria (80-20%)
  2. Leave one out cross validation: entrenar el modelo con  $n-1$  datos y con el dato 1 hacer la prueba. Repetir para todos los puntos.
  3. K-fold cross validation: dividir los datos en  $k$  subconjuntos. Utilizar  $k-1$  para entrenar y el restante para la prueba. Repetir para los subconjuntos restantes

# Under- & overfitting

Si no se tiene un método de validación cruzada de los datos, es probable que el modelo sufra un sobreajuste.

El **“ajuste bajo”** ocurre cuando se tiene un sesgo (bias) muy alto, y no se puede capturar el comportamiento de los datos.

El **sobre ajuste**, por otro lado, ocurre cuando el modelo diseñado es más complejo que el comportamiento de los datos. Cuando se captura el ruido presente.



- Cross validation
- Topología de red
- Cambiar hiperparámetros

**MODELADO:** “La selección del modelo (arquitectura y topología) es más bien un arte que una ciencia”.

**SELECCIÓN DE FUNCIÓN DE PÉRDIDA:** Se refiere a las medidas de error con las que se evaluará el desempeño del modelo. El objetivo es minimizar la función de pérdida.

Dependiendo de la aplicación del modelo será la función de pérdida seleccionada. Por ejemplo, se puede emplear validación categórica para problemas de clasificación y el error medio cuadrático para problemas de regresión.

$$MSE = \frac{1}{n} \sum_{t=1}^n (Y_t - \hat{Y}_t)^2$$

# 5 – INTRODUCCIÓN A REDES NEURONALES

## 5.1 Aplicaciones

Las aplicaciones de las redes neuronales artificiales son variadas:

- Abordan campos de estudio como asociaciones, clasificaciones, generación de patrones, regresión, optimización.
- Área aeroespacial, automotriz, bancos, electrónica, defensa y seguridad, entretenimiento, finanzas, seguros, manufactura, medicina, energéticos, robótica, telecomunicaciones, etc.

## **Ventajas**

1. Capaz de aproximar cualquier función continua sin necesidad de un antecedente del comportamiento de los datos
2. Capaz de adaptarse a cambios en el entrenamiento
3. Opera en tiempo real

## **Desventajas**

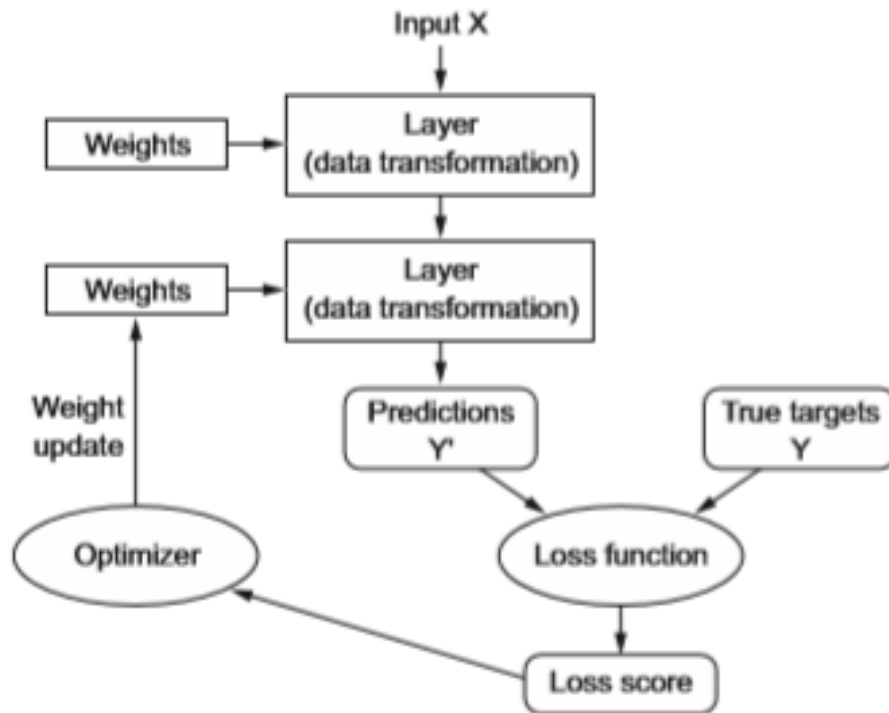
1. Mientras más profundo sea el análisis, más difíciles resulta la interpretación
2. La convergencia del algoritmo depende de los valores iniciales

# ML vs Estadística

1. Cuando no se conozca la distribución de los datos que se modelarán
2. Cuando se hagan análisis en tiempo real
3. Modelos no lineales

# 6 – DEEP LEARNING

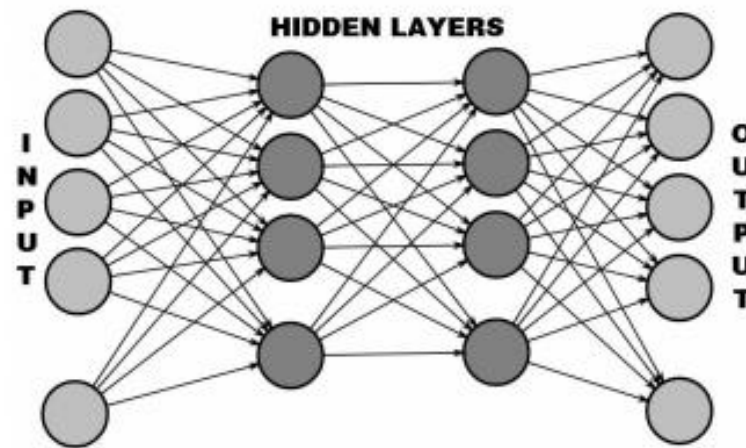
## 6.1 Deep Learning vs Machine Learning



La única diferencia que existe entre los algoritmos de deep learning y las técnicas tradicionales de machine learning, es la **cantidad de datos que se analizarán**. De acuerdo con la literatura, hasta ahora se ha comprobado que las técnicas de deep learning son superiores a las de machine learning en **tareas relacionadas con ámbito ingenieril, relacionado con problemas de reconocimiento de imágenes y voces**.



- Deep learning consiste en construir redes neuronales avanzadas. Por medio del diseño de **múltiples capas ocultas**, se puede lidiar con la complejidad de no linealidad de los datos.
- Con la llegada de la **infraestructura para datos**, tal como GPU (Graphics processing unit) es posible el desarrollo de estos algoritmos
- Al agregar capas y neuronas a una capa de una red neuronal, **se incrementa la especialización del modelo** en la base de entrenamiento pero se **disminuye el desempeño en la base de prueba**.



# Tipos de Deep Neural Networks

1. **Deep Belief Network (DBN):** una red que se alimenta hacia Adelante, en donde los datos fluyen de una capa a otra sin retroceso. Tiene, al menos, una capa oculta.
2. **Restricted Boltzmann Machine (RBM):** tiene una sola capa oculta y no hay conexiones entre los nodos de un grupo.
3. **Recurrent Neural Network (RNN):** los datos fluyen en cualquier dirección dentro y entre los nodos.
4. **Long Short Term Memory (LSTM):** al igual que las recurrentes, pero tiene la capacidad de almacenar temporalmente los registros

Una gran ventaja sobre otros modelos, es la habilidad de generalización del modelo.

# Paquetes de deep learning para R

Paquete	Tipo de redes	Observaciones
MXNet	Feed-forward, Convolutiv NN	Soporta varios lenguajes de programación. Algunos usuarios son: Amazon, Intel, Wolfram, MIT, Microsoft, etc.
darch	RBM,DBM	Disponible en MATLAB, basada en el código escrito por Hinton y Salakhutdinov
deepnet	Feed-forward, RBM,DBM	A pesar de ser una librería pequeña, tiene la capacidad de entrenar diferentes arquitecturas
h2o	Feed-forward, autoencoder	Tiene funciones de regresión lineal, k-means, principal component analysis, forest, and deep learning
nnet & neuralnet	Feed-forward	Una de las más empleadas para redes en R
keras	Variedad de DNN	Librería escrita en Python, diseñada en ser minimalista, modular y extensible
TensorFlow	Variedad de DNN	Creado por Google, para detectar y descifrar patrones y correlaciones

# 6 – DEEP LEARNING

## 6.2 Introducción a TensorFlow

- TensorFlow es una open source proveída por Google para inteligencia artificial. En conjunto con la API (Application Programming Interface) Keras se logra tener una interfaz de alto nivel para el diseño de redes neuronales.
- Si bien el lenguaje por defecto de TF es Python, se puede utilizar en R por las librerías que se han desarrollado.
- Los nodos que se generan en las redes neuronales representan las operaciones matemáticas, mientras que los ejes son los tensores o arreglos de datos multidimensionales.

# Flujo de trabajo típico de Keras

1. Definir el conjunto de datos de entrenamiento: tensor de entrada y de salida
2. Definir el modelo o capas de la red que mapearán las entradas en las salidas
3. Configurar el proceso de aprendizaje al seleccionar la función de pérdida (cálculo de error), el optimizador (método de aprendizaje) y alguna métrica para monitorear (precisión)
4. Realizar las iteraciones del modelo con los datos disponibles, utilizando la función *fit()*

Hay dos maneras de definir un modelo en Keras:

1. Utilizando la función *keras\_model\_sequential()*

Función para capas lineales. Es la arquitectura de red más empleada

2. Utilizando *API functional*

Función para capas gráficas cíclicas, permite construir arquitecturas personalizadas.

RECOMENDACIÓN: El código que se requiera para deep learning deberá tener un soporte GPU, ya que la mayoría de las aplicaciones requiere procesamiento de imágenes con redes recurrentes y convolutivas.

# 6 – DEEP LEARNING

## 6.3 Aplicaciones

Una forma de deep learning es una red neuronal con numerosas capas ocultas. Dado su poder de manejar una cantidad vasta de información, se tienen las siguientes aplicaciones:

1. Reconocimiento de imágenes
2. Reconocimiento de escritos
3. Clasificación de textos
4. Clasificación multi-clase
5. Vehículos autónomos
6. Diagnósticos médicos
7. Predicción del Mercado de valores
8. Pronóstico del estado del tiempo
9. Pronósticos basados en series de tiempo
10. Traducción de textos
11. Acciones robóticas secuenciales
12. Problemas de regresión
13. Entre otros

# Referencias

Ciaburro, Giuseppe; Venkateswaran, Balaji. (2017). *Nueral Networs with R*. Packt. Birmingham.

Chollet, Francois; Allaire, J.J. (2018) *Deep learning with R*. Capítulo 3. 35 pp.

Ghatak, Abhijit (2019). *Deep learning with R*. Springer. (eBook)  
<https://doi.org/10.1007/978-981-13-5850-0>

Shah, Tarang. (2017) *About Train, Validation and Test Sets in Machine Learning*. Disponible en: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>