Curso de

Introducción a Machine Learning por MindsDB

Natasha Seelam, PhD
Head of AI/ML Research

Conocimientos previos

- Programación con Python.

- Uso de Pandas.

- Uso de Matplotlib.

- Probabilidad, estadística y cálculo fundamentales.

Objetivos de este curso

Nuestros objetivos son:

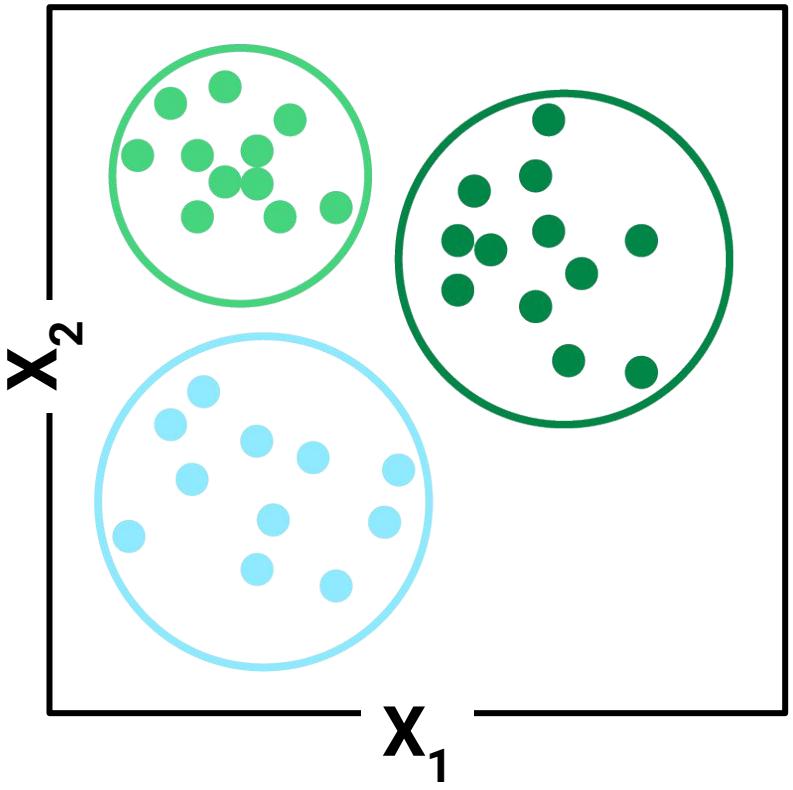
- Aprender a preparar datos y visualizarlos.
- Entender diferentes algoritmos de machine learning.
- Explorar deep learning y redes neuronales.

Qué es machine learning y su historia

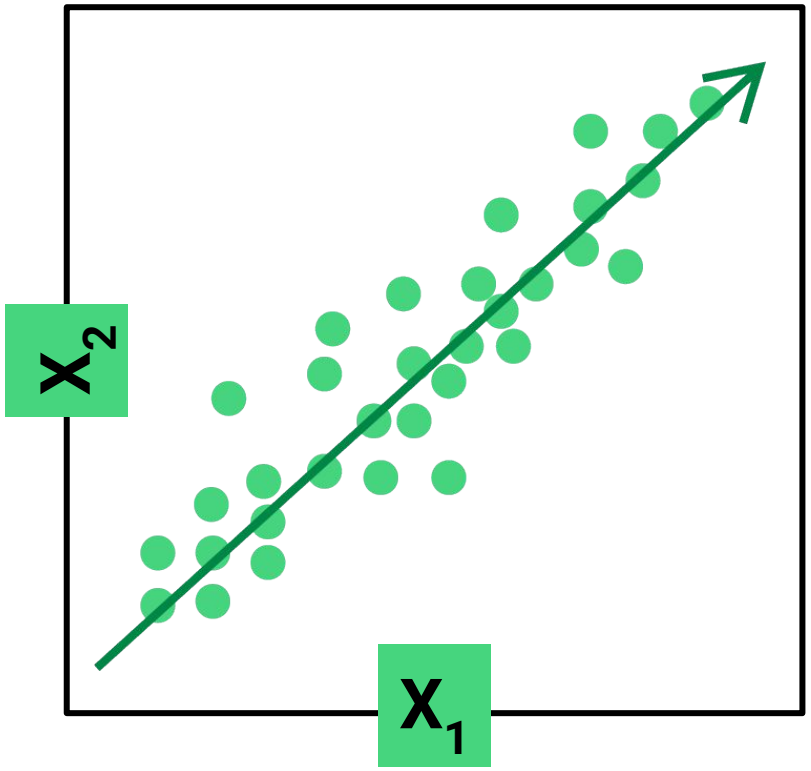
¿Qué es machine learning?

Machine learning es la ciencia de usar algoritmos para **identificar patrones en datos** con el fin de resolver un problema de interés.

Aprendizaje no supervisado



Aprendizaje supervisado



Features:
Descriptores de tus datos que pueden ayudarte en la tarea que quieres resolver.

Referencias:
Figure adapted: B. Qian et al. “Orchestrating the Development Lifecycle of Machine Learning-Based IoT Applications: A Taxonomy and Survey”, ArXiv (2019)

Machine learning es usado actualmente en la vida de todo mundo

Emails de spam cuestan a empresas cerca de $21 mil millones de dólares [1, 2] en 2021.

“

En 2015 Google dijo que puede identificar el 99.9% de emails de spam. [3]

”

Referencias:
[1] Statistic from Radicati Group, Nucleus Research
[2] Joseph Johnson. “Number of sent and received e-mails per day worldwide from 2017 to 2025”, Statista (2021)
[3] Cade Metz. “Google Says Its AI Catches 99.9 Percent of Gmail Spam”. Wired (2015)

Spam

Estimado cliente de BigBanking,

Hemos detectado una actividad sospechosa en su cuenta.

Por favor, haga clic en el siguiente enlace y proporcione sus datos bancarios.

Lo mejor,
XXX

Normal

Estimado cliente,

Su cambio de contraseña ha sido aceptado. Gracias por utilizar los servicios de BigBanking.

Lo mejor,
XXX

Historia del machine learning

1940s → 1960s → 1990s → 2020s

1940-1950
IBM crea el primer programa exitoso para jugar damas.

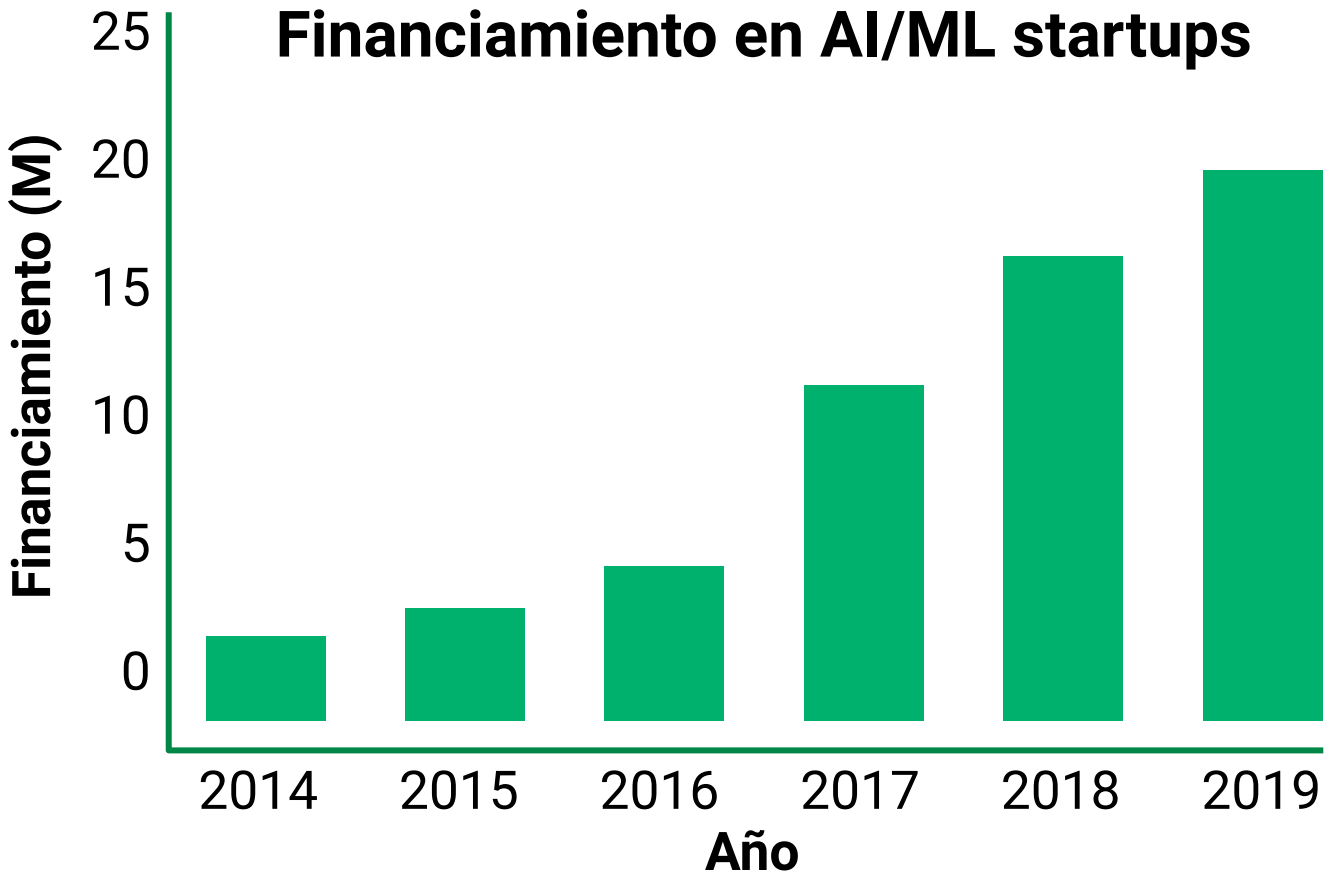
~1959
Arthur Samuel usa el término “machine learning”.

1990-2020

- Roombas son las primeras aspiradoras autónomas producidas en masa.
- Crecimiento de asistentes personales (Siri, Alexa).
- AlphaGo vence al campeón mundial en el complejo juego de Go con cerca de 2^{170} posibles posiciones jugables.

Referencias:
Figure adapted from Kelly Nguyen. “The History of Machine Learning: A Timeline”, Experian (2020)
Patricia Jean Eckhart, “Tiled DANNA: Dynamic Adaptive Neural Network Array Scaled Across Multiple Chips”, Master’s Thesis, The University of Tennessee, Knoxville (2017)
Dr. Paul Marsden. “Artificial Intelligence Timeline Infographic – From Eliza to Tay and beyond” (2017)

Machine learning sigue creciendo



Referencias:
Figure adapted from: Khari Johnson. “CB Insights: AI startup funding hit new high of $26.6 billion in 2019”, VentureBeat (2020)

El mercado para machine learning sigue creciendo

Top 10 de habilidades tech de 2020 [2]

- Python
- React (web)
- Angular
- Machine learning
- Docker
- Django
- CompTia
- Amazon AWS
- Deep learning
- React Native (mobile)

Referencias:
Figure adapted from: Louis Columbus. “AI Skills Among The Most In-Demand For 2020” Forbes (2019) from Udemy

Resumen

- Modelos de ML **encuentran patrones en datos para resolver problemas**.
- Los modelos aprenden patrones de los *features*.
- **Machine learning** (ML) es un campo en crecimiento.

Data science framework: herramientas para machine learning

Terminología para ciencia de datos

- **Data/Datos:** unidades de información o “hechos” de observaciones.
- **Features:** tipos de información acerca de tus observaciones.
- **Filas:** observaciones individuales o muestras.
- **Columnas:** features que describen tus observaciones.

Referencias:
"Statistical Language - What are Data?". Australian Bureau of Statistics. 2013-07-13.

Terminología para ciencia de datos

- **Outlier:** punto(s) de datos o data point(s) que se comporta de forma extraña.
- **Pre-processing:** preparar datos para su uso en un modelo de machine learning.
- **ETL pipeline:** framework de data science para extraer, transformar y cargar.

Referencias:
"Statistical Language - What are Data?". Australian Bureau of Statistics. 2013-07-13.

Ejemplo de un dataset

Fila: Ejemplo de 1 data point

Target: Variable de salida es la variable que te gustaría predecir. No todos los datasets tienen un target.

	Pregnancies	Glucose	Blood Pressure	Skin thickness	Insulin	BMI	Diabetes	Function	Age	Outcome
0	6	148	72	35	0	33.6	NaN	0.627	50	1
1	1	85	66	29	0	26.6	NaN	0.351	31	0
2	8	183	64	0	0	23.3	NaN	0.672	32	1
3	1	89	66	23	94	28.1	NaN	0.167	21	0
4	0	137	40	35	168	43.1	NaN	2.288	33	1

Columnas:
Representan features de los aspectos de los datos.
Estos datos tienen 10 columnas.

Referencias: Kaggle Diabetes Dataset by Mehmet A. (https://www.kaggle.com/mathchi/diabetes-data-set)

Tipos de datos

- **Numéricos:** su feature es un número de tipo entero o flotante.
- **Categórica:** sus features representan una clase o tipo; usualmente se representan como un mapeo de números o un “one-hot” vector.
- **Image:** su feature representa una imagen.
- **Texto:** su feature es en la forma de texto, sea corto (como Twitter) o largo (como en noticias).
- **NaN:** su feature es desconocido o perdido.

Convertir datos categóricos en etiquetas

Estoy tomando datos del estado del tiempo de afuera. Diario por 5 días. Etiqueto el clima si es “Soleado”, “Lluvioso”, “Nublado”, o “Nevado”.

Día 1	Día 2	Día 3	Día 4	Día 5
Soleado	Soleado	Nublado	Lluvioso	Nevado

Convertir datos categóricos en etiquetas

Etiqueto a cada tipo de clima con un número: “Soleado – 1”, “Lluvioso – 2”, “Nublado – 3”, y “Nevado – 4”. Así puedo convertir mis datos en números:

Día 1	Día 2	Día 3	Día 4	Día 5
1	1	3	2	4

Convertir etiquetas en 1-hot encodings (OHE)

“Soleado – 1”, “Lluvioso – 2”, “Nublado – 3”, and “Nevado – 4”

$$x = \langle x_1, x_2, x_3, x_4 \rangle$$

“El día 2 estuvo soleado y el día 3 estuvo nublado”

$$x^{day_2} = \langle 1, 0, 0, 0 \rangle$$

$$x^{day_3} = \langle 0, 0, 1, 0 \rangle$$

Trabajando con Pandas para cargar y entender tus datos

Trabajando con Pandas

pd.read_csv()
Leer un archivo CSV

df.head()
Muestra las primeras 5 filas

```
In [3]: # Read in your CSV
        data = pd.read_csv("diabetes.csv")

        # Get a sneak preview of the dataset
        data.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Trabajando con Pandas

```
In [4]: data.dtypes

Out[4]: Pregnancies                 int64
        Glucose                     int64
        BloodPressure               int64
        SkinThickness               int64
        Insulin                     int64
        BMI                       float64
        DiabetesPedigreeFunction  float64
        Age                         int64
        Outcome                     int64
        dtype: object
```

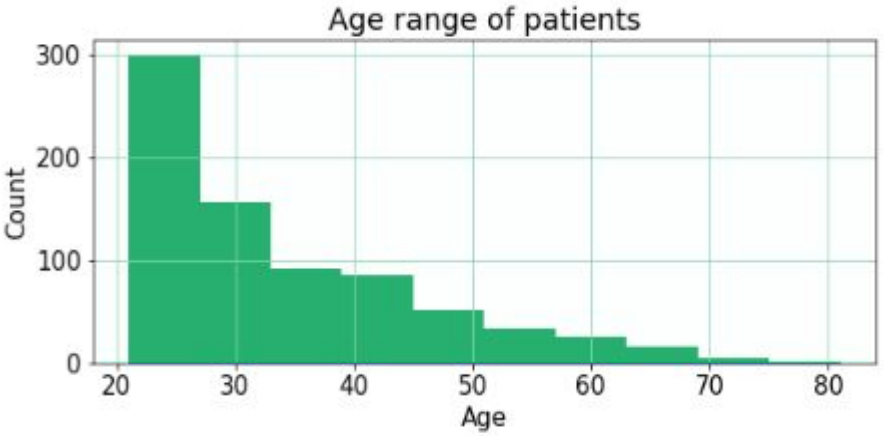
df.dtypes
Muestra el tipo de representación de los datos (float, int, object).

Visualizando tus datos

Visualización de datos – Histogramas

- Te dice qué tan “frecuentes” son ciertos valores en tus datos.
- Requiere de ti para “agrupar” los datos.

```
f = plt.figure(figsize=(8,4))
ax = f.add_subplot(1,1,1)
data["Age"].hist(ax=ax, edgecolor='black', linewidth=2)
ax.set_title("Age range of patients")
ax.set_ylim([0, 510])
ax.set_xlabel("Age")
ax.set_ylabel("Count")
f.tight_layout()
```



Ejemplo de histograma

Contando el número de monedas en los bolsillos de un grupo de personas. Obtengo los siguientes datos, x:

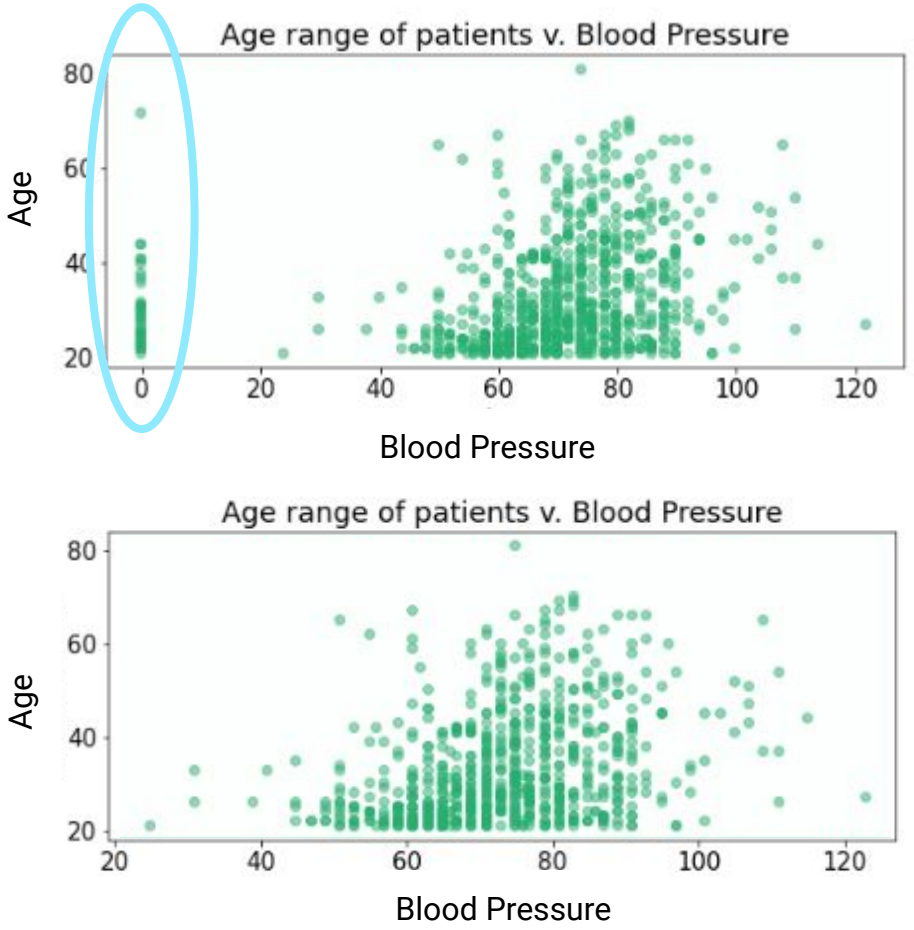
$$x = [0, 1, 1, 2, 2, 2, 3, 3, 4, 5, 6, 10, 10\ ]$$

Creamos algunos contenedores o grupos. Son:

Tiene **0 monedas** [0, 1)	Tiene **1 monedas** [1, 2)	Tiene **2 monedas** [2, 3)	Tiene **3 monedas** [3, 4)	Tiene **más de 4 monedas** [4, 11]
N = 1	**N = 2**	**N = 3**	**N = 2**	**N = 5**

Visualización de datos – Gráficas de dispersión

- Muestra la relación entre 2 features graficándolos como pares ordenados.
- En cada eje, un feature.
- Te puede ayudar a detectar anomalías.



Ejemplo de gráfica de dispersión

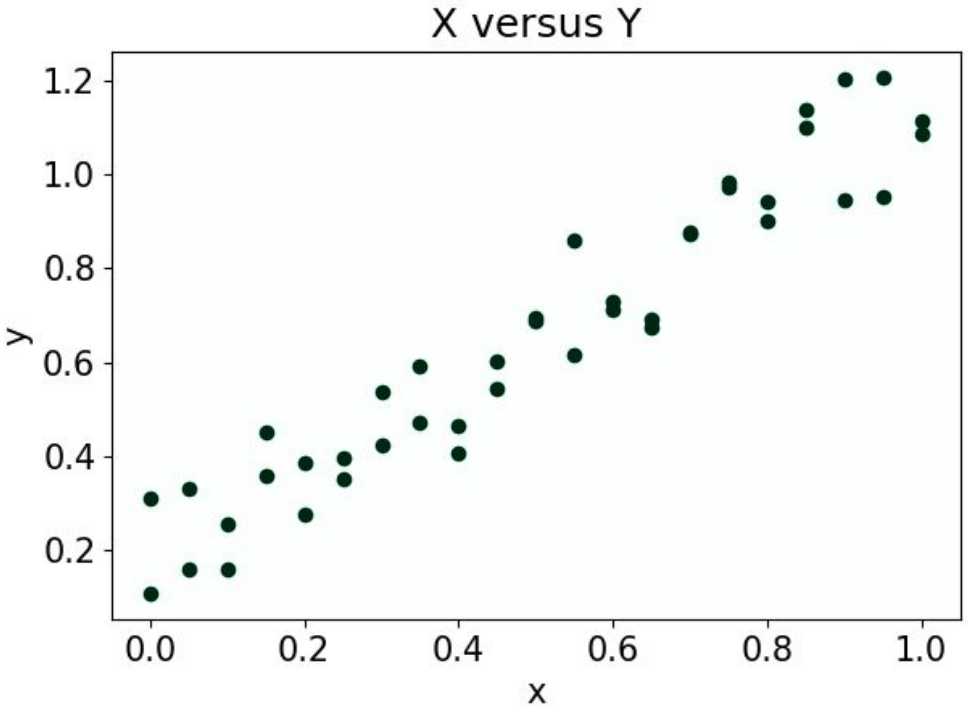
¿Cuál es la relación entre *x* y *y*?

x - Input feature

y - output target

x_i - input feature para i^{th} data point

y_i - output target para i^{th} data point



Resumen

- **Pandas** ayuda a cargar y analizar datos.
- **Histogramas** ayudan a ver la distribución de un feature.
- **Gráficas de dispersión** permiten ver la relación entre dos features.

Tipos de modelos de machine learning

Algoritmos de machine learning

Aprendizaje supervisado

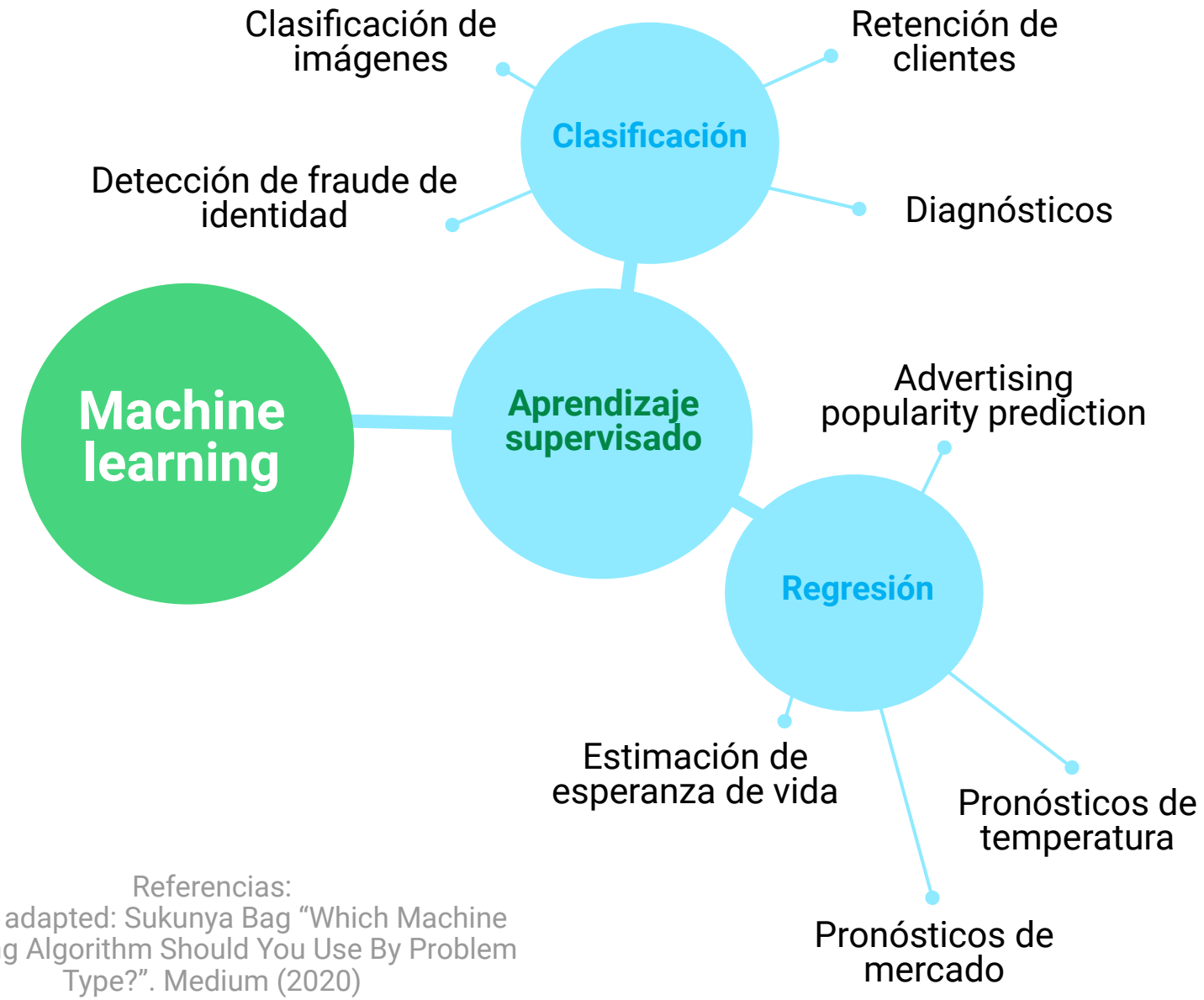
El modelo obtiene features de entrada y salida.
Hay un target/objetivo a predecir.

- **Regresión:**

Target output (objetivo de salida) es numérico.

- **Clasificación:**

Target output es una etiqueta.



Referencias:
Figure adapted: Sukunya Bag “Which Machine Learning Algorithm Should You Use By Problem Type?”. Medium (2020)

Aprendizaje no supervisado

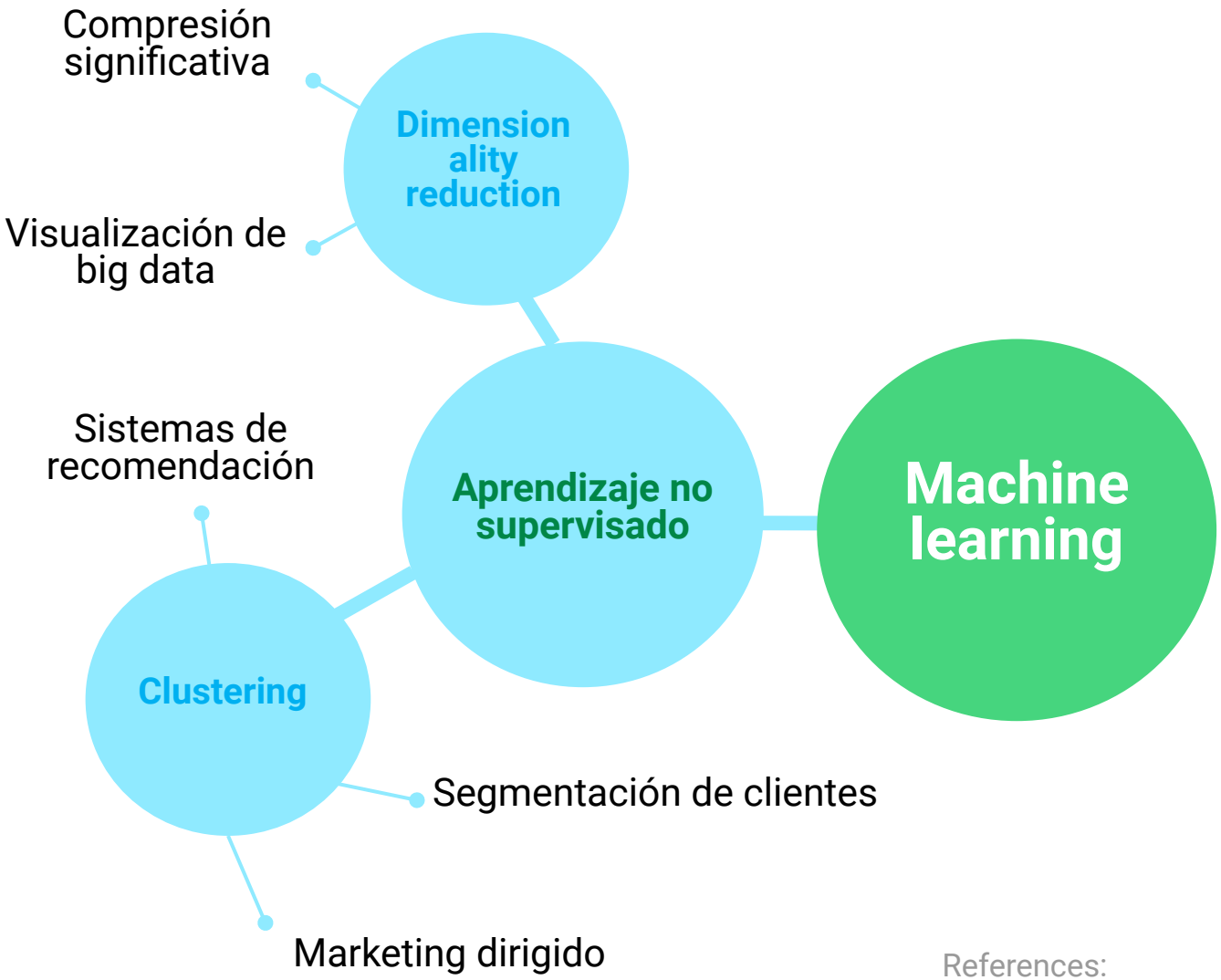
Objetivo desconocido, queremos encontrar estructura y grupos dentro de los datos.

- **Clustering:**

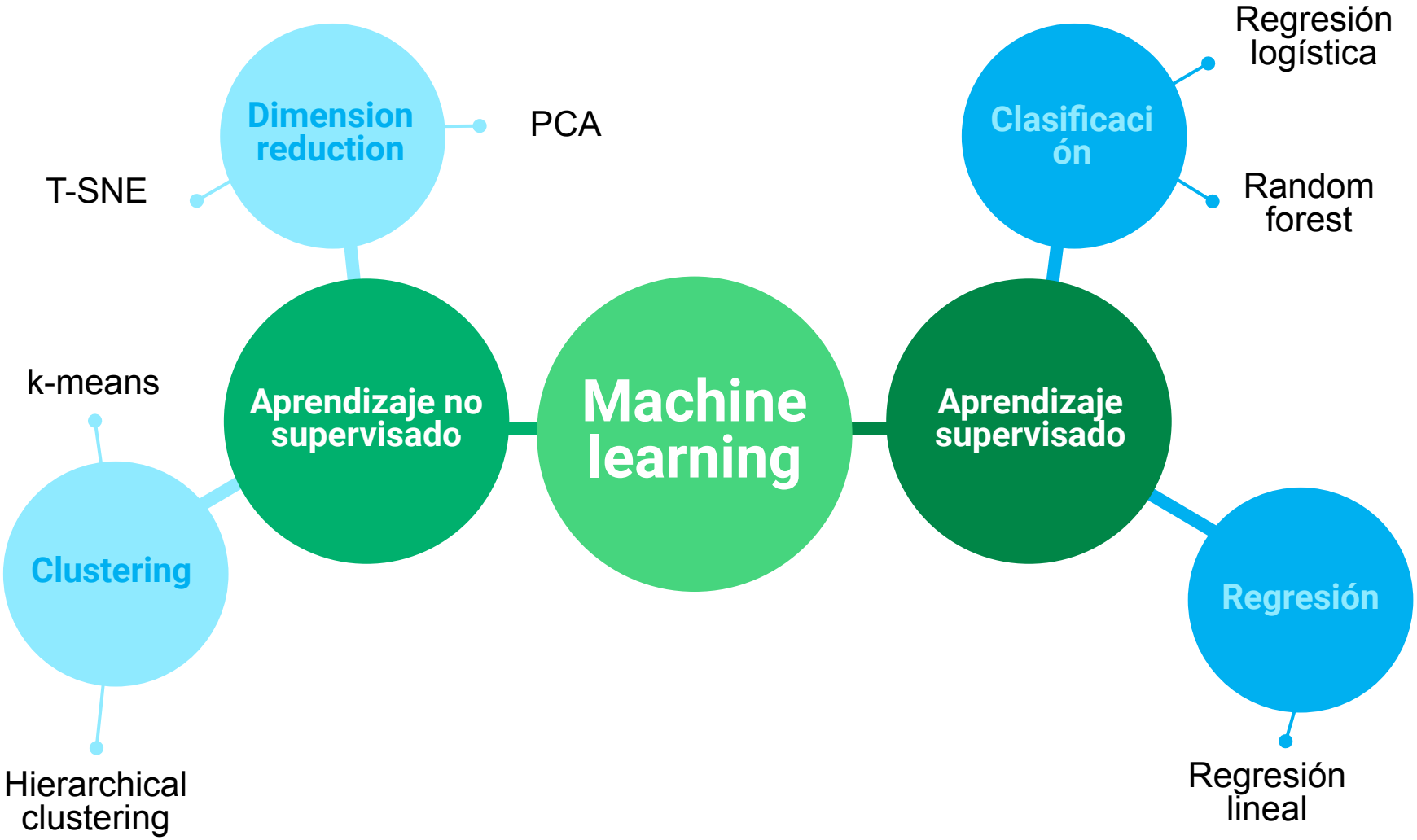
Queremos encontrar grupos en los datos.

- **Dimensionality reduction:**

Queremos encontrar qué features de entrada en los datos son de ayuda.



References:
Figure adapted: Sukunya Bag “Which Machine Learning Algorithm Should You Use By Problem Type?”. Medium (2020)


Machine learning
Aprendizaje no supervisado
Aprendizaje supervisado
Dimension reduction
PCA
T-SNE
Clustering
k-means
Hierarchical clustering
Clasificación
Regresión logística
Random forest
Regresión
Regresión lineal

Resumen

- Algoritmos de machine learning generalmente son **supervisados** o **no supervisados**.
- **Algoritmos supervisados predicen un objetivo o target**. Pueden ser de regresión o clasificación.
- **Algoritmos no supervisados predicen patrones o estructura en los datos**. Generalmente son de clustering o dimensionality reduction.

Conclusiones

Resumen de módulo

- Los modelos de machine learning **encuentran patrones en datos para resolver problemas.**
- La visualización de datos con gráficas nos ayuda a **entender qué relación existe antes de entrenar un modelo.**
- Los modelos usualmente son **supervisados** (con un objetivo) o **no supervisados** (sin un objetivo).

La “receta” para aplicar algoritmos de machine learning

Objetivos de este módulo

1. Los **“ingredientes”** de los algoritmos de machine learning.
2. Algoritmos de **aprendizaje supervisado.**
3. Algoritmos de **aprendizaje no supervisado.**

Receta de algoritmos ML

1. **Proceso de decisión:** cómo los modelos hacen una predicción, o retornan una respuesta, usando los parámetros.
2. **Función de error/coste:** cómo evaluar si los parámetros en el modelo generan buenas predicciones.
3. **Regla de actualización:** cómo mejorar los parámetros para hacer mejores predicciones (usando optimización numérica).

Referencias:
[1] Michael Tamir. “What is Machine Learning?” Hosted by UC Berkeley School of Information (2020)

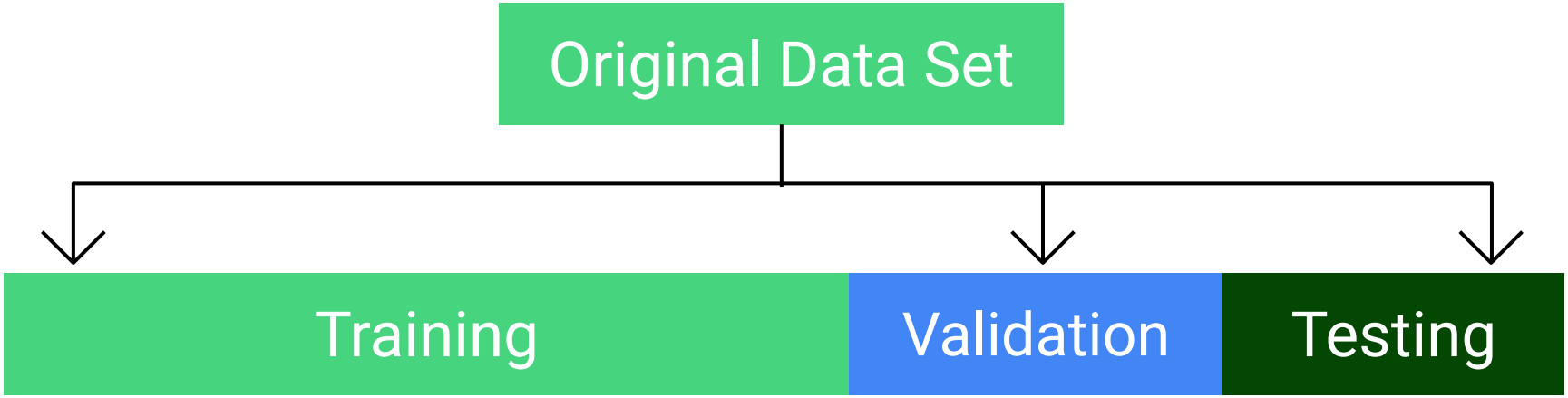
Normalizar tus datos numéricos

Para cada columna de tu dataset con valores numéricos:

1. Calcular el promedio de tus datos (μ).
2. Calcular la desviación estándar (σ) de tus datos.
3. Para cada punto de datos, realizar:

$$\tilde{x}_i = \frac{x_i - \mu}{\sigma}$$

Preparar tus datos para modelos



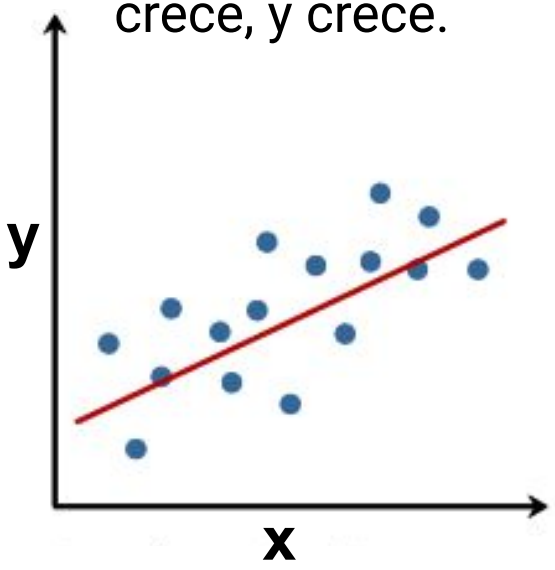
- **Training:** (60-80%) datos de los que el modelo aprende patrones.
- **Validation:** (0-20%) datos que usas para verificar que el modelo aprende.
- **Testing:** (0-20%) datos que se apartan para revisar si el modelo fue exitoso al predecir.

Algoritmos supervisados: regresión lineal

Regresión lineal: bases

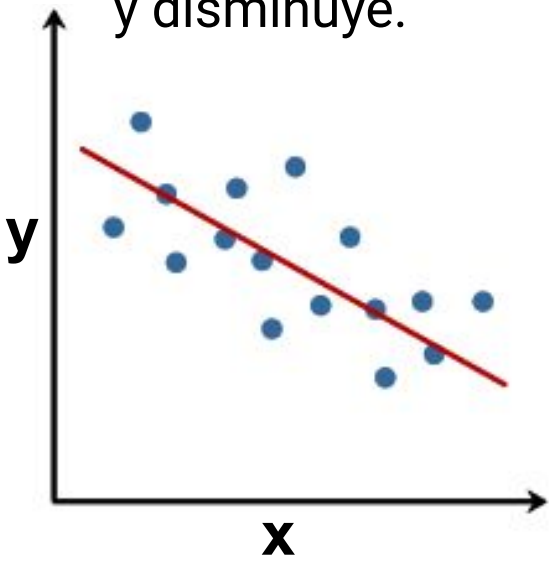
Lineal - positiva

Mientras x crece, y crece.

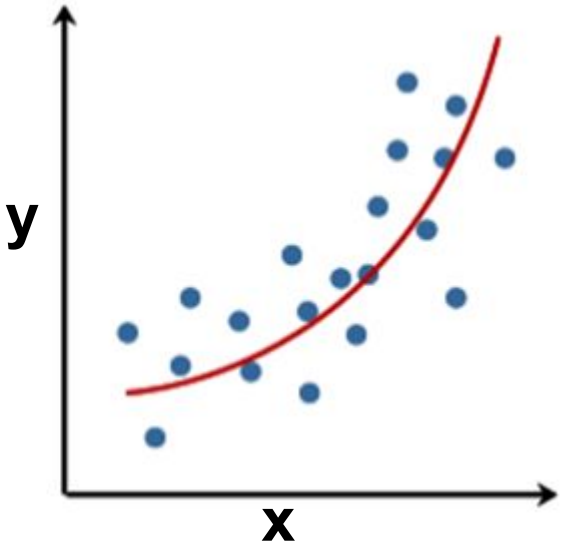


Lineal - negativa

Mientras x crece, y disminuye.



No lineal



Referencias:
[1] Laerd Statistics. “Linear Regression Analysis using SPSS Statistics” (2014)

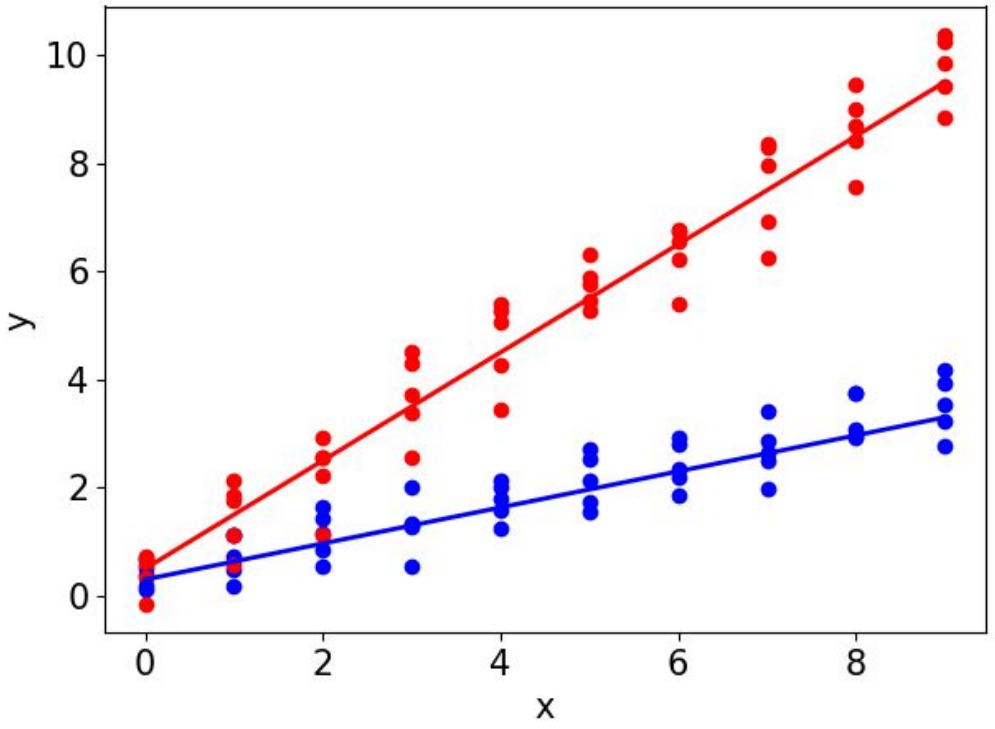
Regresión lineal: parámetros

Diferentes parámetros cambian lo que pensamos de la relación entre *input features (x)* y el *output target* (y_{pred}).

$$y_{pred} = w_1 x + w_0$$

W_1 = un cambio en "x" resulta un cambio en "y"

W_0 = El valor de "y" si "x" es 0

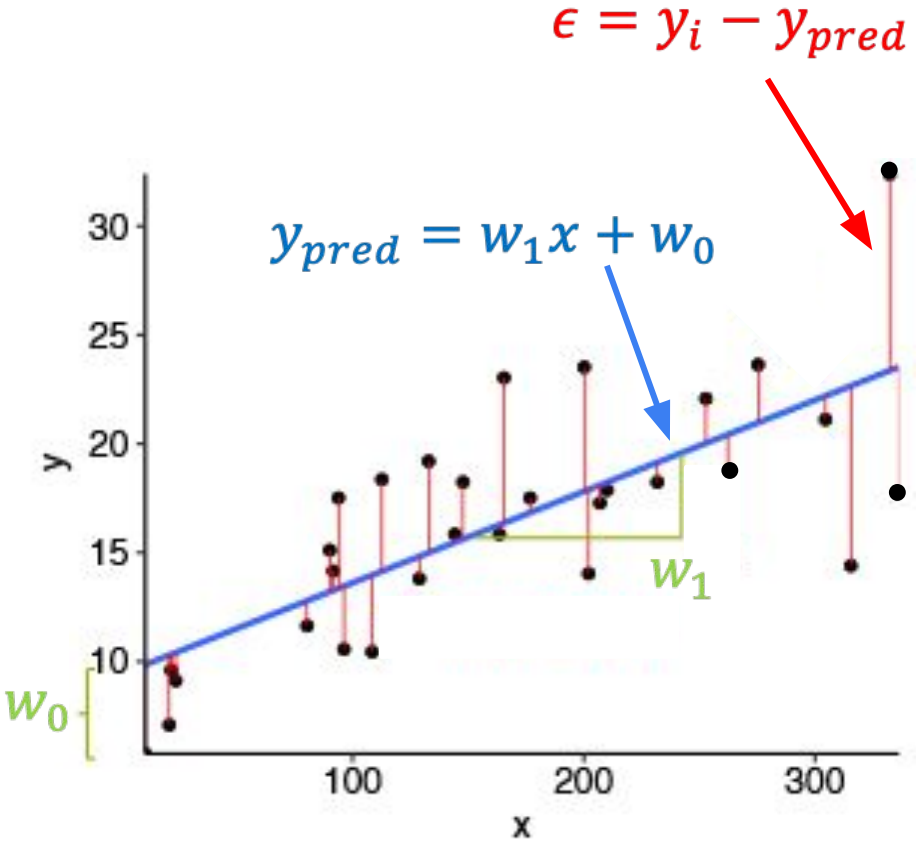


Regresión lineal: función de coste

Mean- square Error (MSE) Loss:

El costo ($\in$) es la diferencia entre el valor del target (y_i) y el valor predecido de nuestra línea (y_{pred})

$$J(w, w_0) = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{i,pred})^2$$

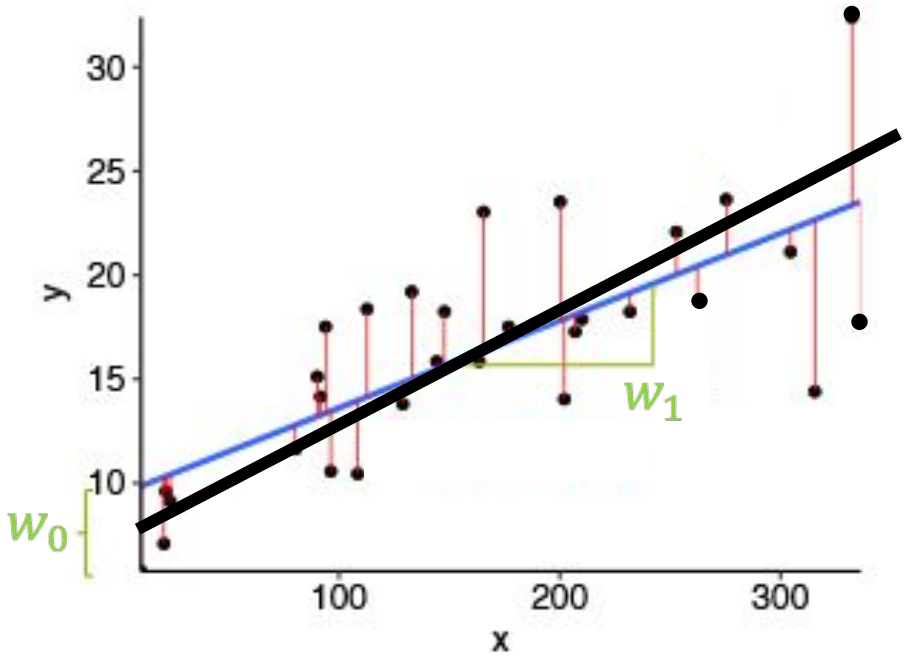


Referencias:
[1] Adapted from Mayank Tripathi. “Understanding Linear Regression with Python: Practical Guide 2”. Data Science Foundation (2020)

Regresión lineal: regla de actualización

Queremos minimizar la distancia $y_{i,\,pred}$ sobre cada punto de datos en entrenamiento.

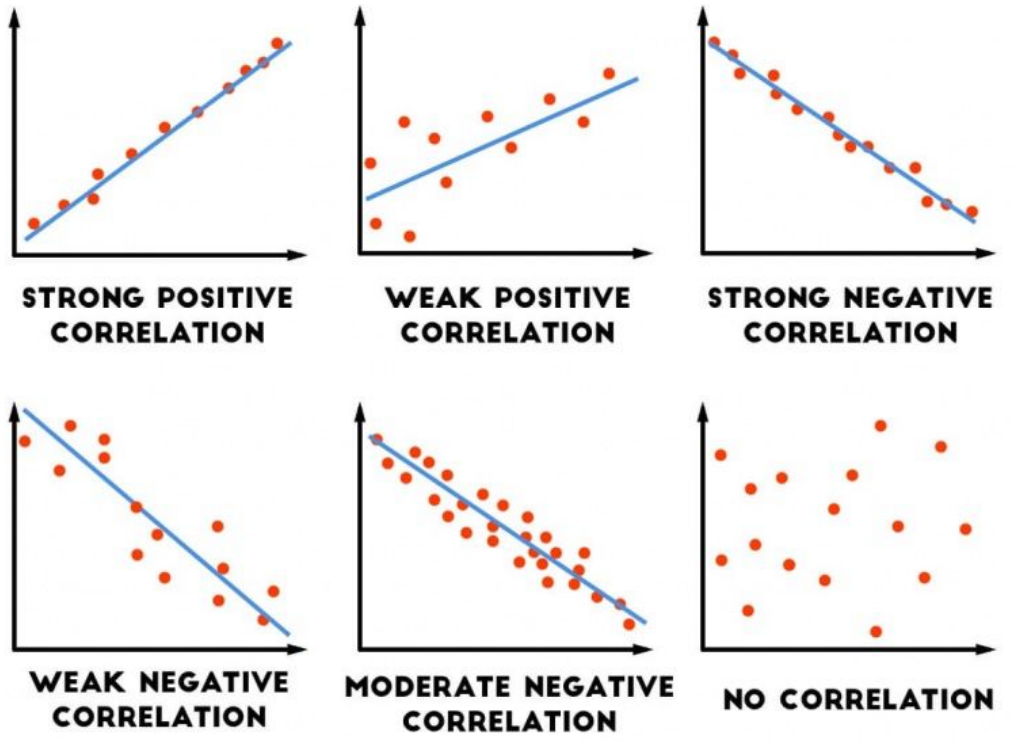
Cambiamos w_0 y w_1 hasta encontrar una suma de distancia menor.



$$\text{Update Rule} = \min J(w_0, w_1)$$

Regresión lineal: rendimiento

MSE and R^2 ayudan a identificar la fortaleza de la relación entre features de entrada y de salida.



Referencias:
[1] Figure from Dan Shiebler. “R and R^2, the relationship between correlation and the coefficient of determination.” Personal Blog (http://danshiebler.com/) (2017)

Repaso: ingredientes de la regresión lineal

Proceso de decisión:
Una función para predecir el target de salida basada en features de entrada.

$$y_{pred} = w_0 + w_1 x_1 + w_2 x_2 + \ \ldots + w_N x_N$$

Función de error/coste:
El promedio de qué tan bien se predice el target.

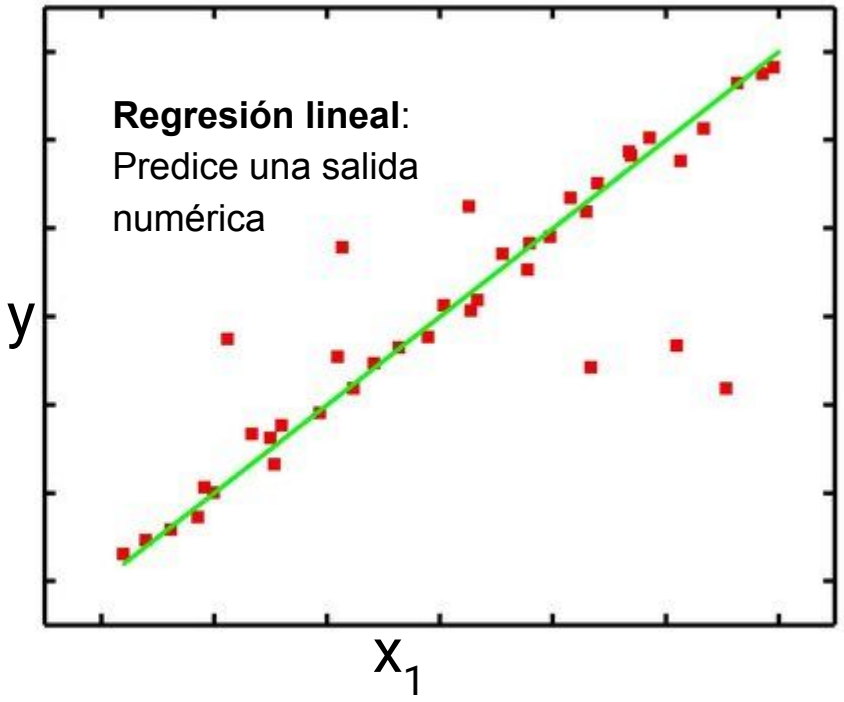
$$J(w, w_0) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - y_{i,pred}\right)^2$$

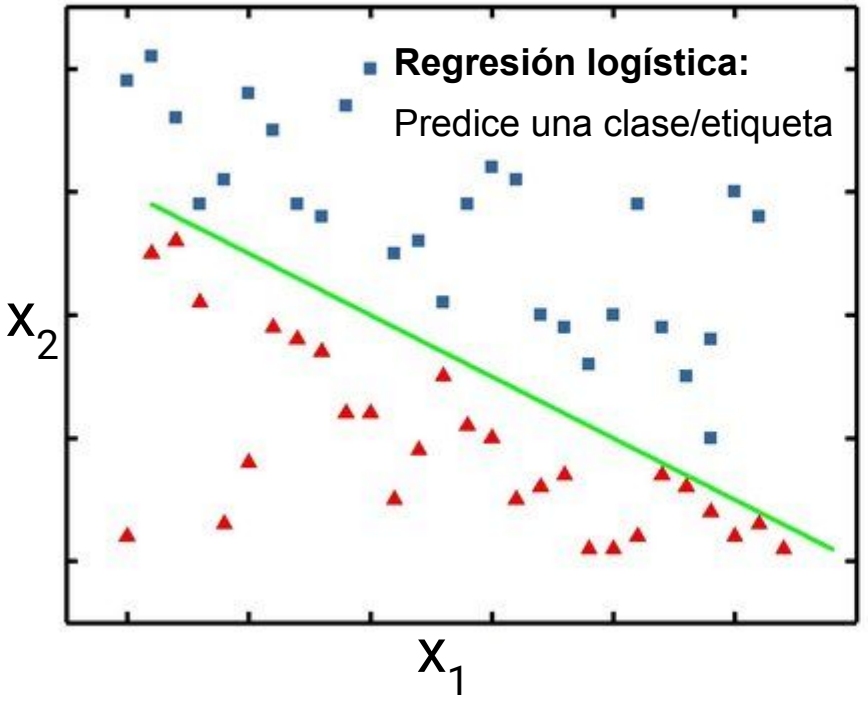
Regla de actualización:
Encuentra valores w, w_0 que mejoren las predicciones.

$$\min J(w_0, w_1, \ldots, w_N)$$

Algoritmos supervisados: regresión logística

Regresión logística: bases

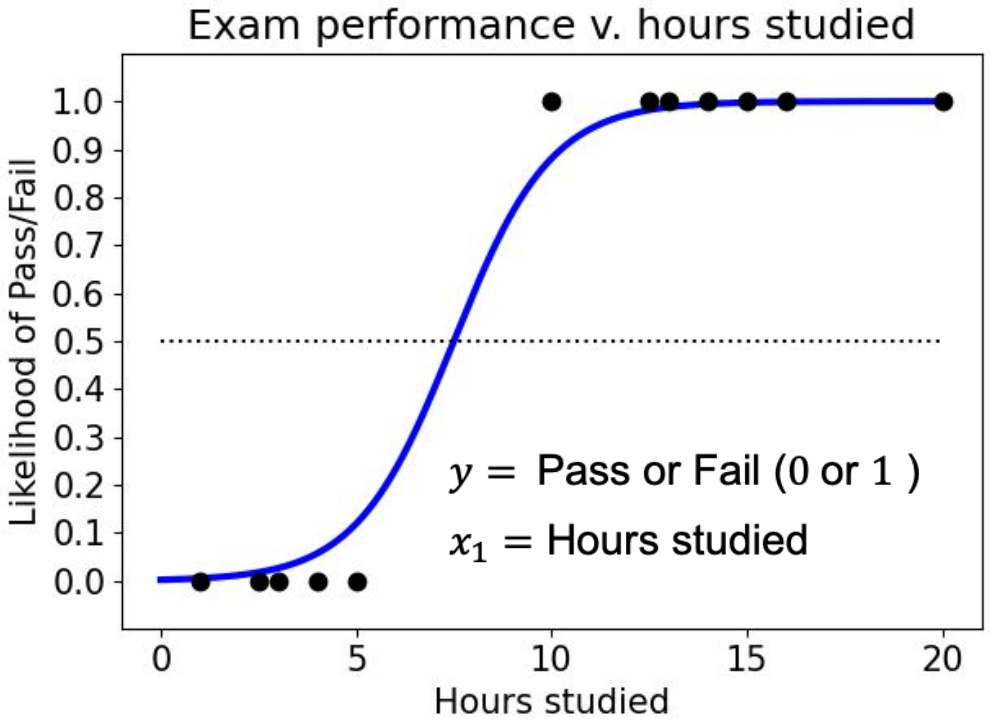




Referencias:
[1] Adapted from Xianjen Fang et al. “Regression Analysis With Differential Privacy Preserving” IEEE Access PP(99):1-1 (2019) DOI:10.1109/ACCESS.2019.2940714

Regresión logística: ejemplo

Queremos saber si un estudiante pasará un examen conociendo cuántas horas estudió:

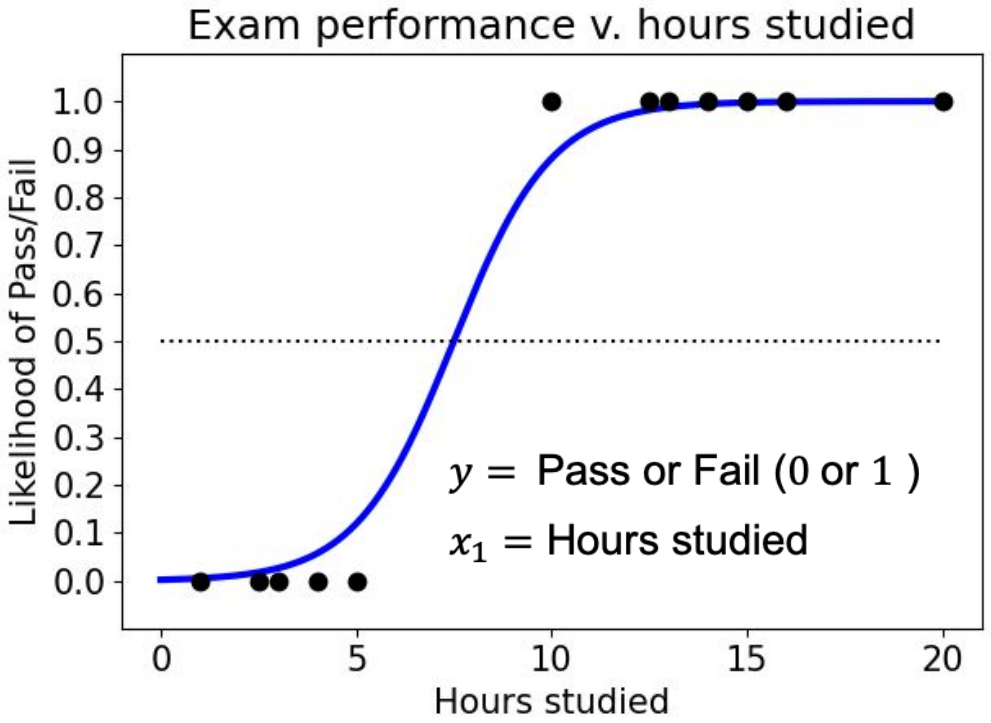


Regresión logística: parámetros

La salida es la probabilidad para cada entrada (estudiante)

Para etiquetar decimos que cualquiera “arriba” de 0.5 pasará; y por debajo, no lo hará.

$$p(Pass) = \frac{\exp(w_0 + w_1 x_1)}{1 + \exp(w_0 + w_1 x_1)}$$



Regresión logística: función de coste y regla de actualización

$$J(w, w_0) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_{\text{pass}}) + (1 - y_i)\log(1 - p_{\text{pass}})$$

Sumatoria sobre todos los puntos de datos (todos los N estudiantes).

Si el estudiante ha **pasado** el examen, este término no es 0.

Si el estudiante **no ha pasado** es "no pasando". Este término no es 0.

Regresión logística: función de coste y regla de actualización

Encuentra los valores w y w_0 que mejor predicen la probabilidad.

$$\text{Update Rule} = \min J(w_0, w_1)$$

Regresión logística: rendimiento

Confusion matrix:

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positive (TP)	False Positive (FP)
Predicted Negative (0)	False Negative (FN)	True Negative (TN)

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{\#\ Correctly\ labeled}{\#\ of\ data\ points}$$

Repaso: ingredientes de la regresión logística

Proceso de decisión: probabilidad de que un estudiante pase (y=1).

$$p(Pass) = \frac{\exp(w_0 + w_1 x_1 + \ ... + w_N x_N)}{1 + \exp(w_0 + w_1 x_1 + \cdots + w_N x_N)}$$

Función de error/coste el promedio de qué tan bien se predice que un estudiante pasa.

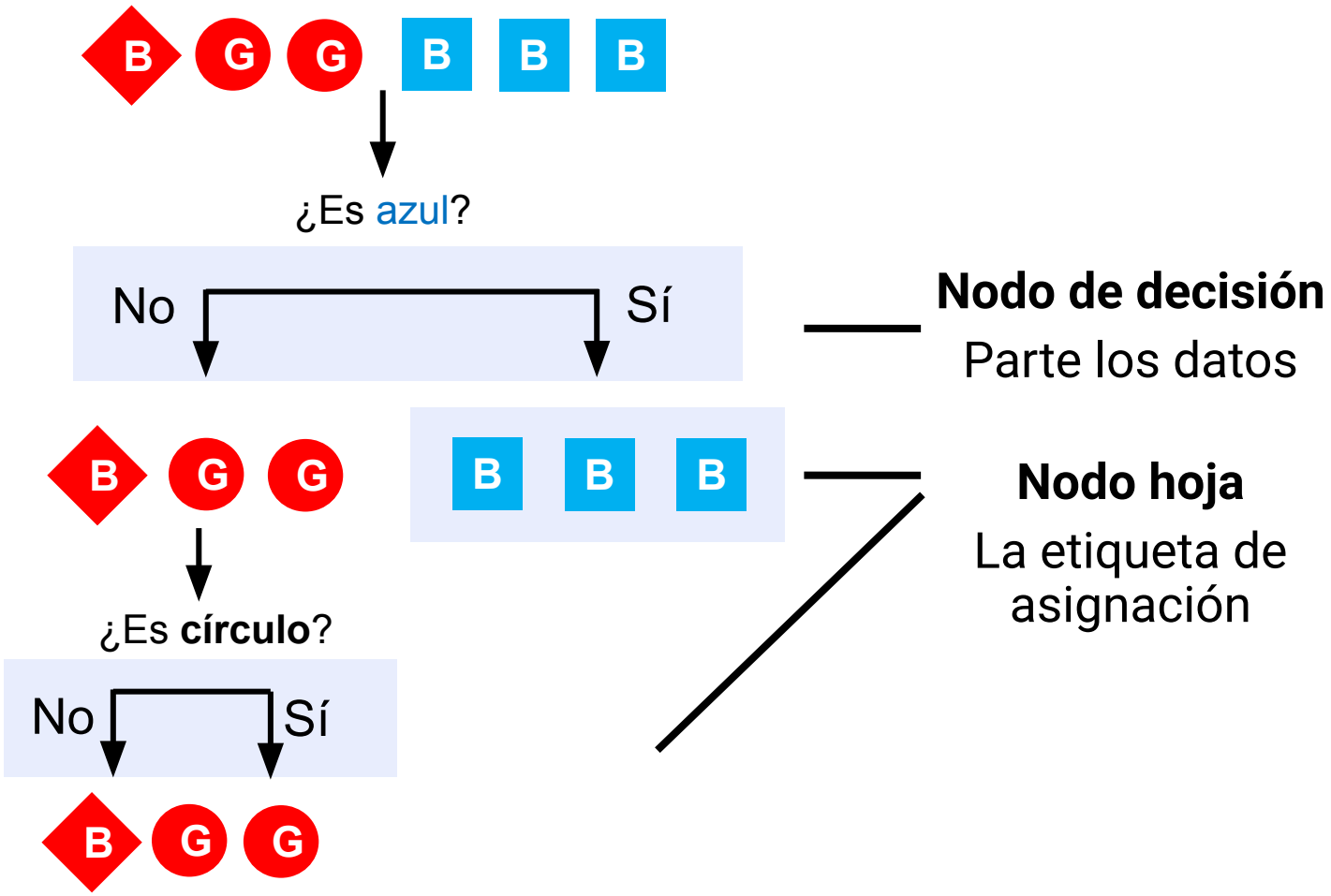
$$J(w, w_0) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_{\text{pass}}) + (1 - y_i)\log(1 - p_{\text{pass}})$$

Regla de actualización: encuentra valores w and w_0 que mejor predicen esta probabilidad.

$$\min J(w_0, w_1)$$

Algoritmos supervisados: árboles de decisión

Árboles de decisión: bases

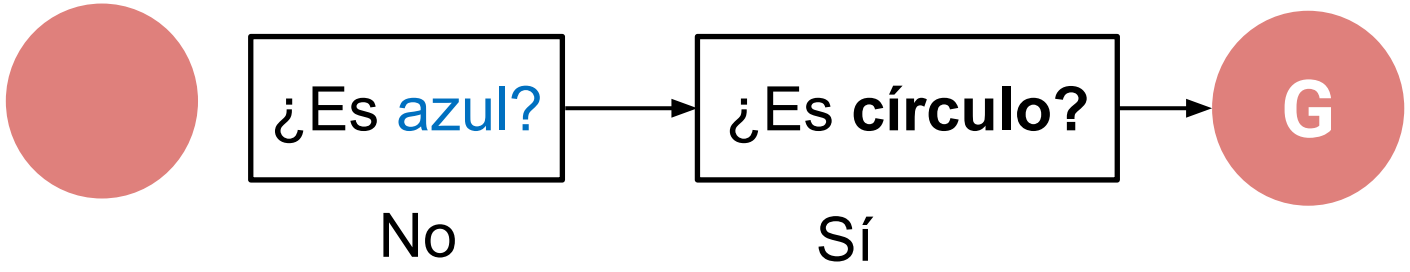


Árboles de decisión: bases

Imagina que veo estos
dos nuevos juguetes.

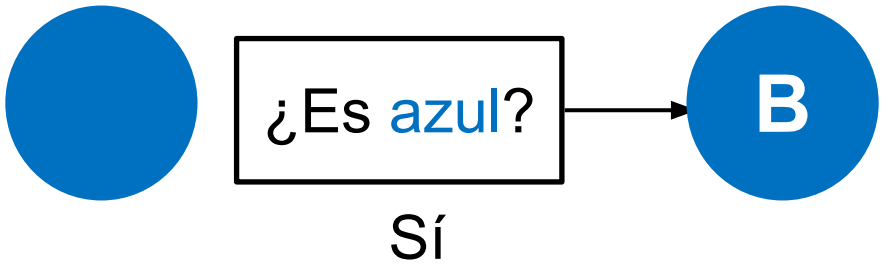
Árboles de decisión: bases

El árbol de decisión preguntará lo siguiente:



Árboles de decisión: bases

¿Cómo prevenir etiquetado incorrecto?

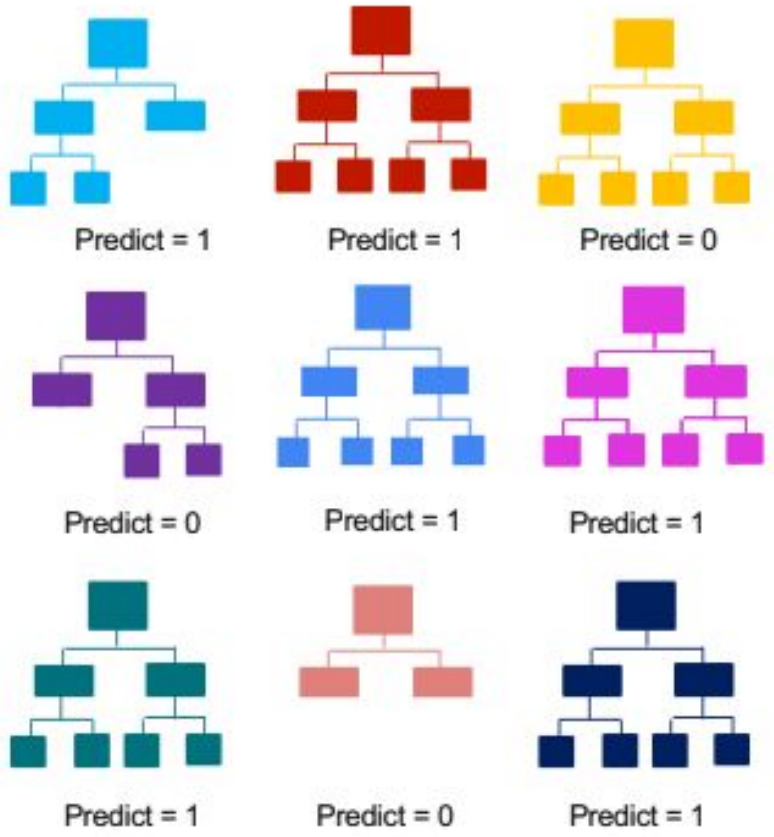


Árboles de decisión: bases

Los conjuntos nos permiten "votar" por la respuesta correcta.

6 árboles predicen 1;
3 árboles predicen 0

Respuesta final: 1

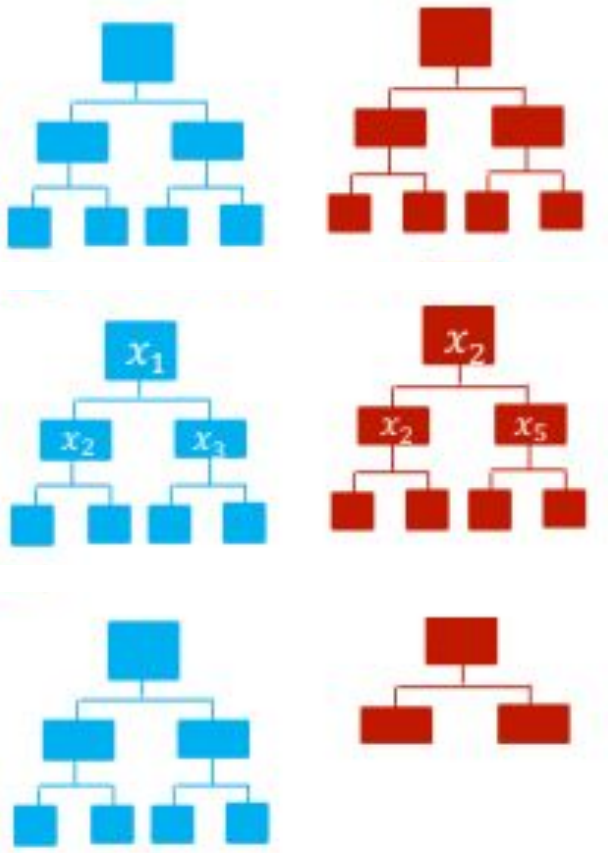


Árboles de decisión: parámetros

Número de árboles:
más árboles, menor la variación, pero más cómputo.

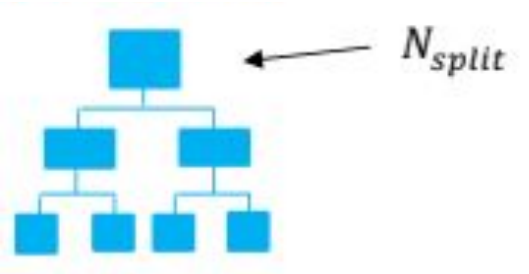
Max features:
El número de features usados para partir (split).

Max depth:
El número de niveles del árbol de decisión.

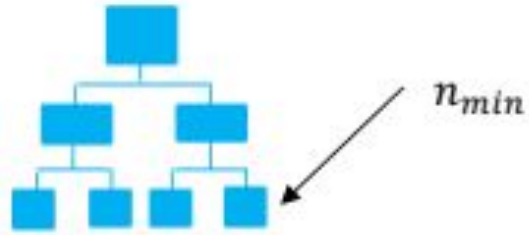


Árboles de decisión: parámetros

N_{split} = Min samples split:
Número de data points que un nodo debe tener antes de dividirse.



n_{min} = Min samples leaf:
El mínimo número de muestras requeridas para estar en una hoja (leaf).



Árboles de decisión: función de coste y regla de actualización

Función de coste: para un feature seleccionado encuentra el mejor split-point (punto de separación) y separa los datos en dos nodos más pequeños.

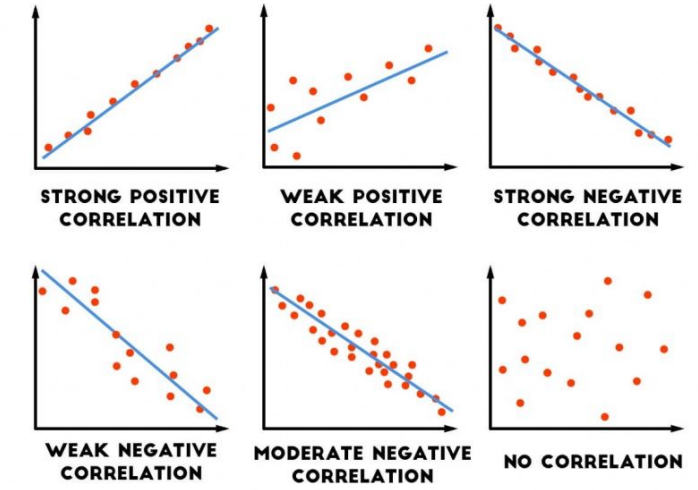
Regla de actualización: Si el nodo hoja tiene más de n_{min} puntos de datos, repite la función de coste, sino para.

Árboles de decisión: rendimiento

- Se pueden usar métricas de **clasificación** o **regresión** para evaluar qué tan bueno es el modelo.

Árboles de decisión: rendimiento

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positive (TP)	False Positive (FP)
Predicted Negative (0)	False Negative (FN)	True Negative (TN)



Clasificación:

Confusion matrix o matriz de confusión

Regresión:

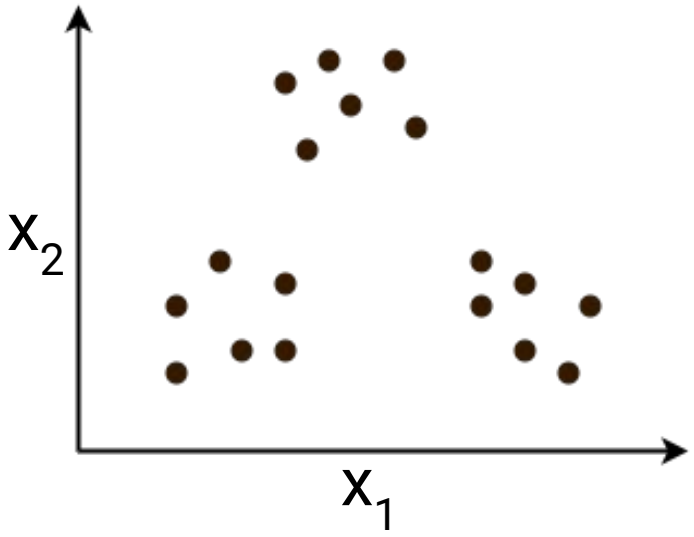
MSE / R^2

Repaso: ingredientes de árbol de decisión

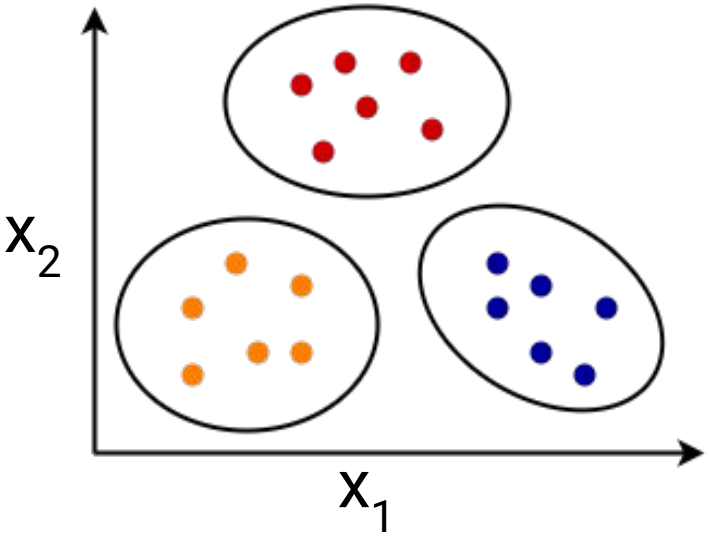
- **Proceso de decisión:**
  De un subset de features aleatorios de tus datos elige un feature. Divide tus datos.
- **Función de error/coste:**
  Usar Gini o criterio de entropía para encontrar el mejor umbral de separación de los datos.
- **Regla de actualización:**
  Si los nodos hoja no tienen n_{min} puntos de datos, repite el proceso.

Algoritmos no supervisados

K-Means clustering



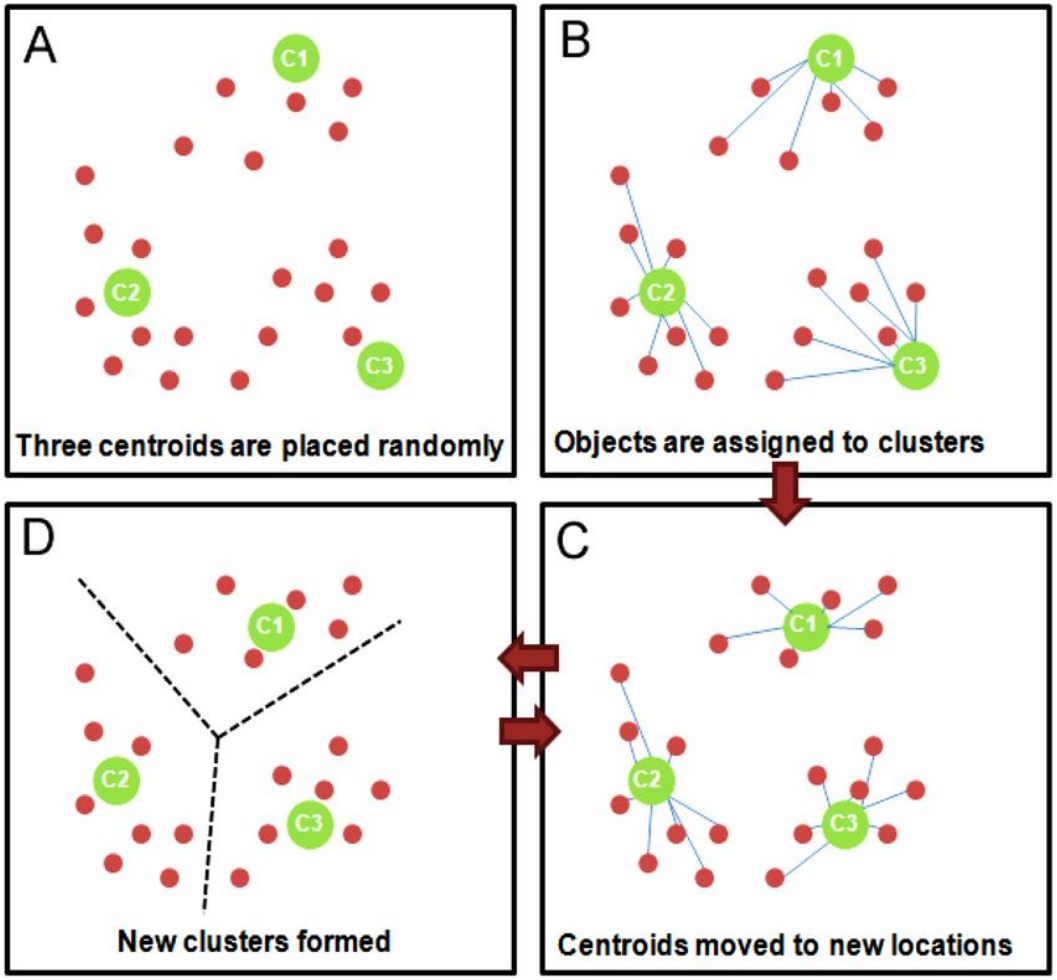
Antes k-means



Después de k-means

Referencias:
[1] Figure Adapted from https://www.gatevidyalay.com/k-means-clustering-algorithm-example/

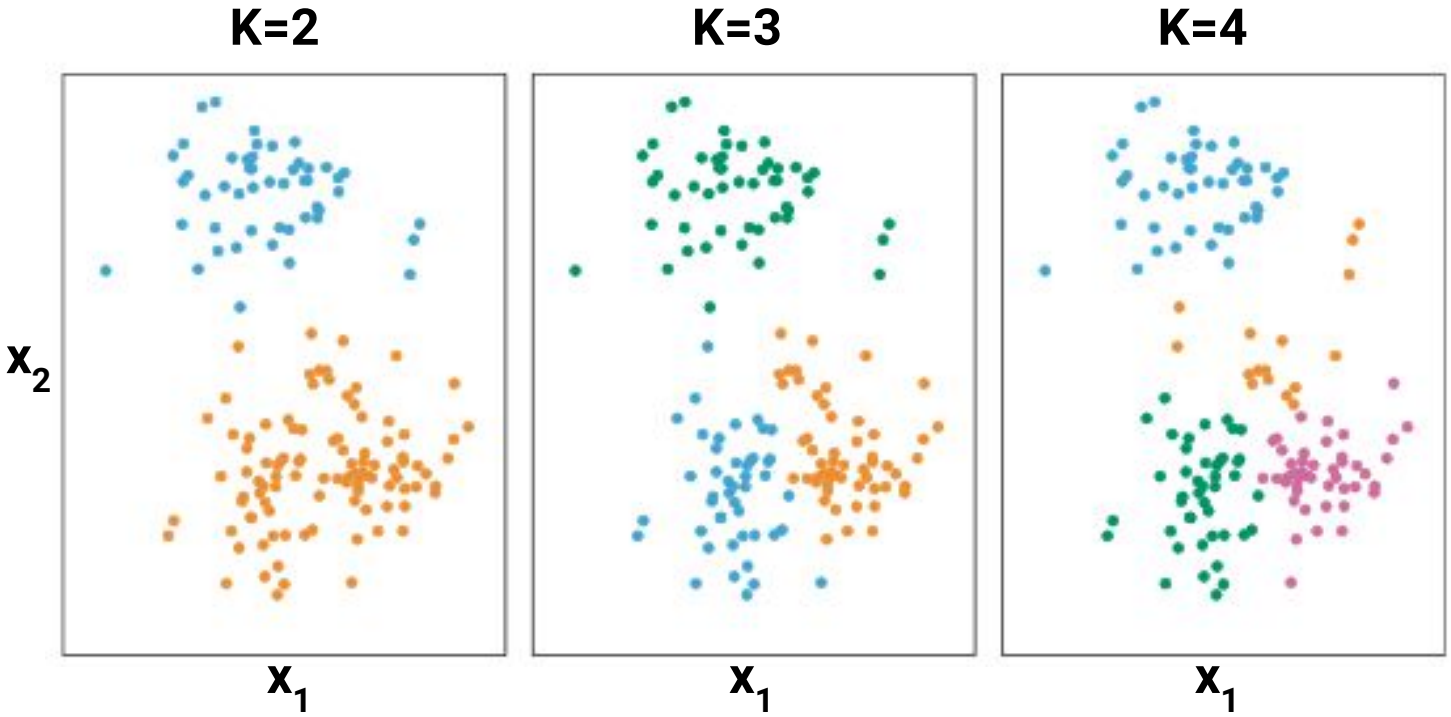
K-Means: bases



Referencias:
[1] R. K. Grewal. “Microarray data analysis: Gaining biological insights” (2013) Journal of Biomedical Science and Engineering 06(10):996-1005

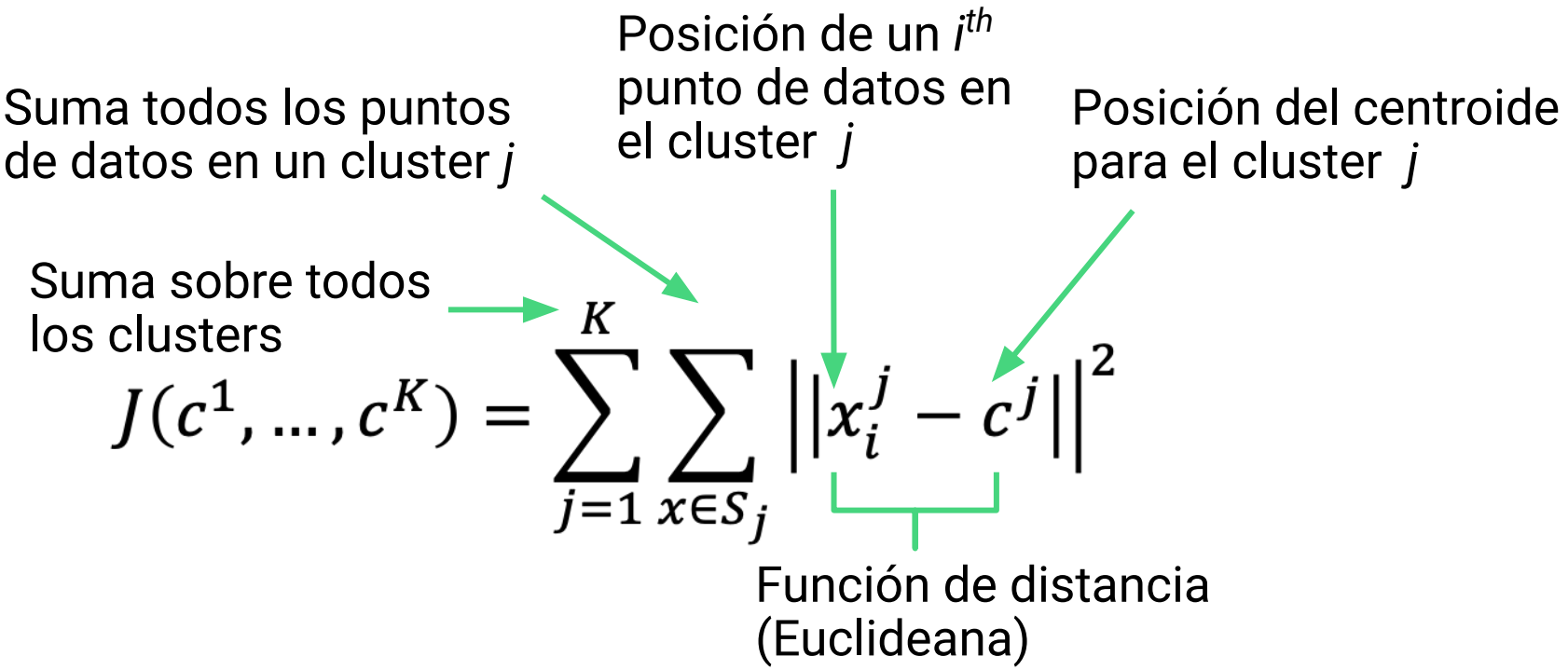
K-Means: parámetros

"K" se refiere al número de grupos que piensas existen en los datos.



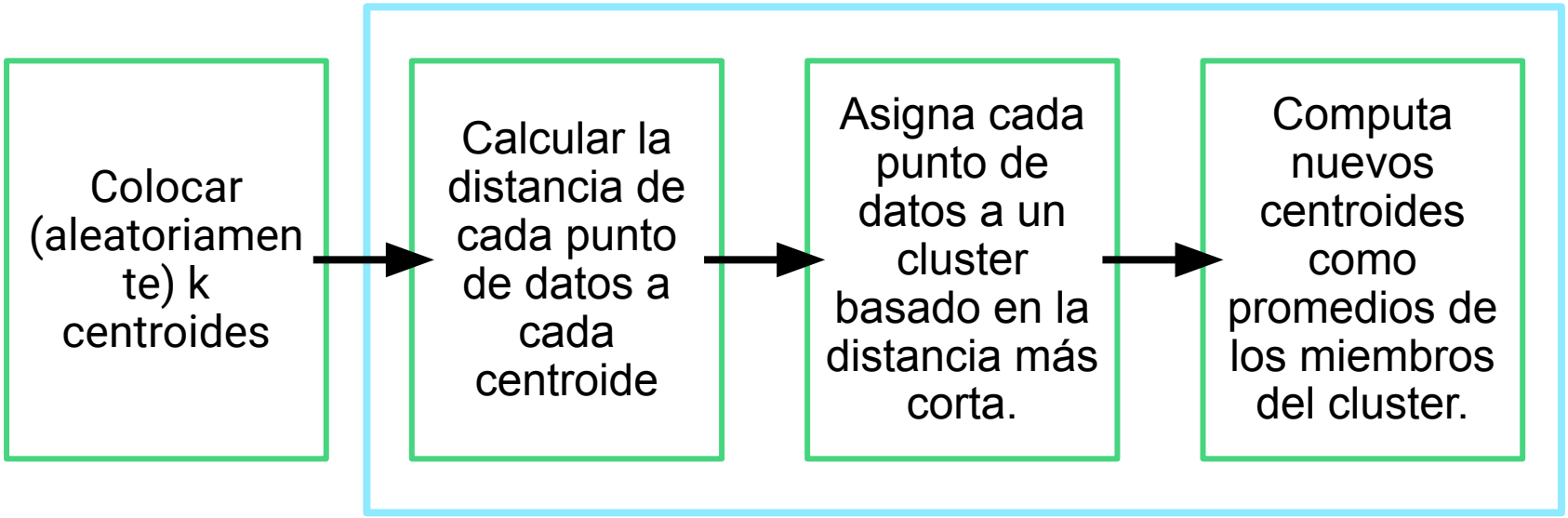
Referencias:
[1] http://andysbrainblog.blogspot.com/2015/05/k-means-analysis-with-fmri-data.html

K-Means: función de coste



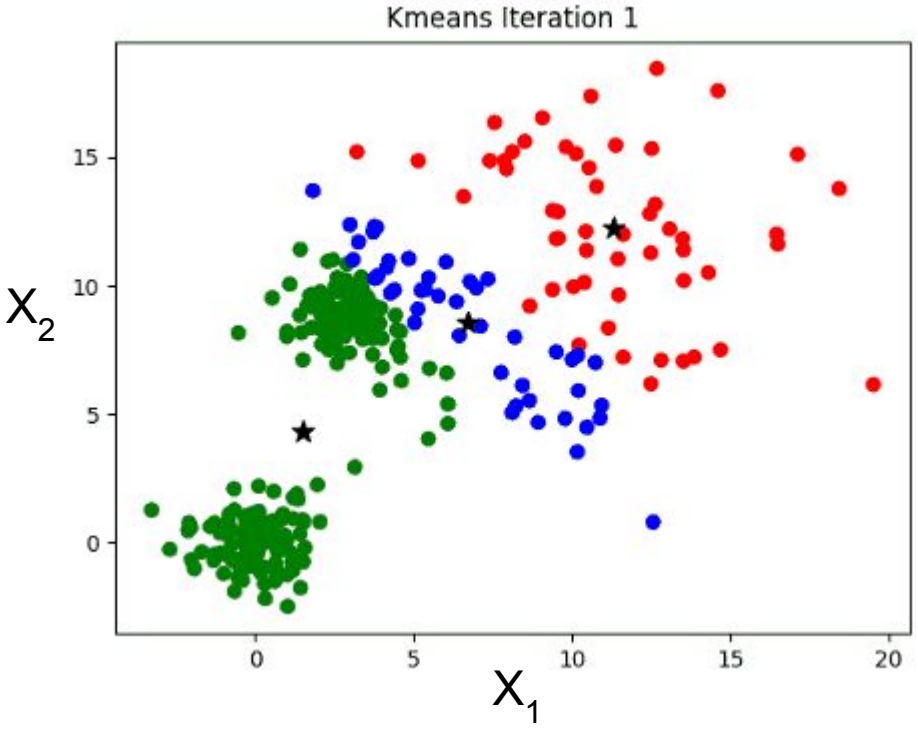
1. Asigna cada punto de datos en un cluster.
2. Calcula los centroides como el promedio de los puntos de datos.

K-Means: regla de actualización



Repetir hasta que los miembros asignados no cambien

K-Means: regla de actualización



Referencias:
[1] Azika Amelia. “K-Means Clustering: From A to Z Everything you need to know about K-means clustering” Towards Data Science (2018)
Link:
https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a

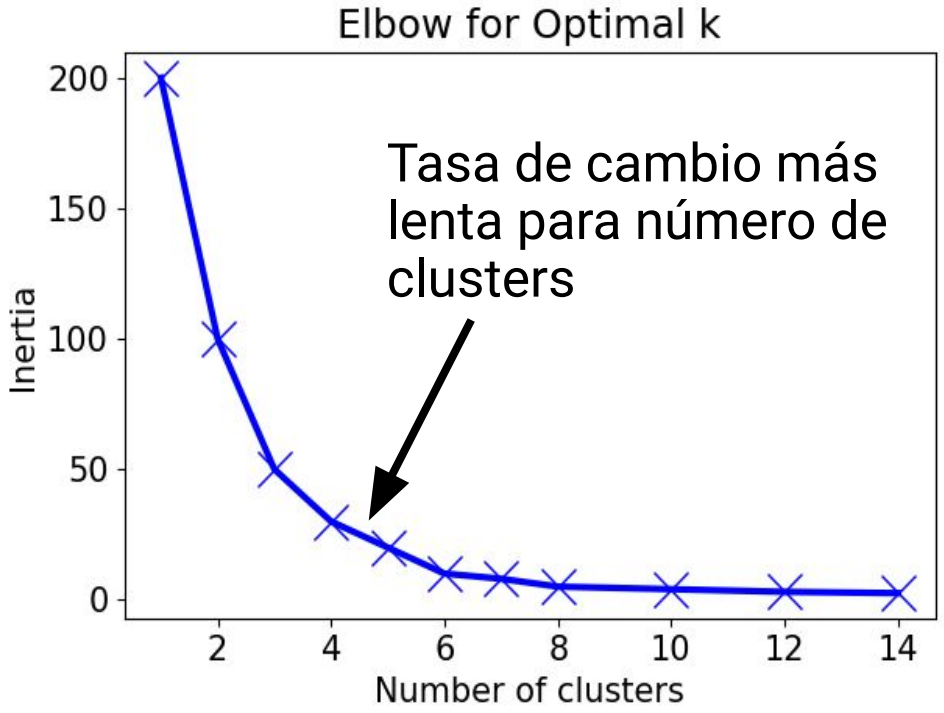
K-Means: rendimiento

Para 1 solo modelo

Inertia: qué tan cerca están los puntos de datos al centroide. Este número se necesita pequeño.

Silhouette score: qué tan lejanos son los clusters, de[-1, 1]. Este número debe ser cercano a 1.

K-Means: rendimiento



Solo para más de 1 modelo

Idealmente prueba K-Means para diferente número de k.

Repaso: ingredientes de K-Means

Proceso de decisión:

- **Calcular** la distancia de cada punto de datos a cada centroide.
- **Asignar cada** punto de datos al centroide más cercano**.**
- **Calcular** los nuevos centroides promediando los puntos de datos asignados al cluster.

Función de coste/error:
Minimizar la distancia de los puntos de datos a los clusters.

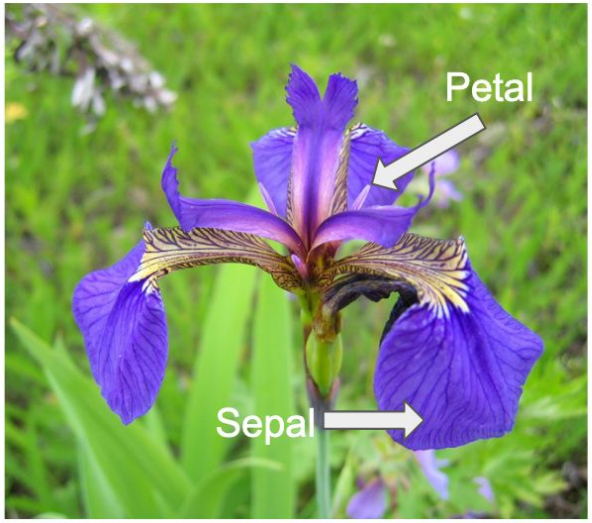
$$J(c^1, \dots, c^K) = \sum_{j=1}^{K} \sum_{x \in S_j} \left|\left| x_i^j - c^j \right|\right|^2$$

Regla de actualización:

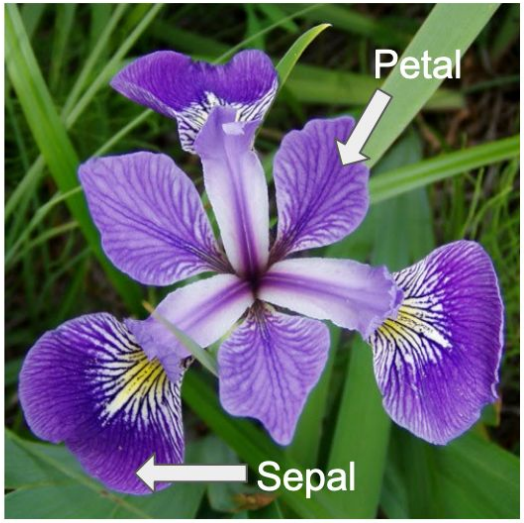
Repite el proceso de decisión hasta que los puntos de datos asignados a cada cluster sean los mismos.

Reto: aplicar machine learning al Iris dataset

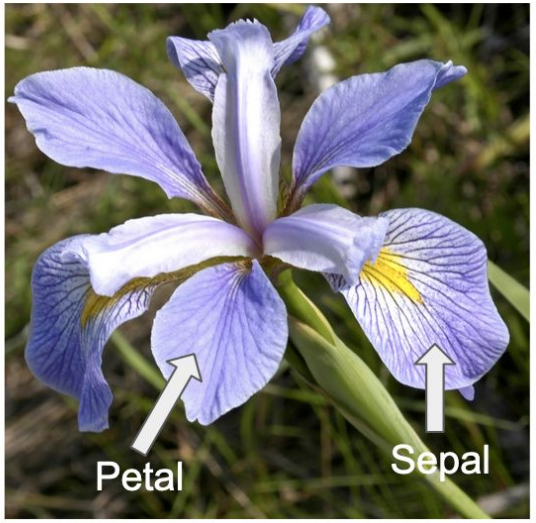
Iris setosa



Iris versicolor



Iris virginica



Queremos identificar especies de plantas con 4 features:
longitud del sépalo, ancho del sépalo, longitud del pétalo, ancho del pétalo.

References:
[1] R.A. Fisher. UCI Machine learning Repository (https://archive.ics.uci.edu/ml/datasets/iris) (1988)
[2] Figure from Yong Cui. “The Iris Dataset — A Little Bit of History and Biology” Towards Data Science, 2020

Conclusiones

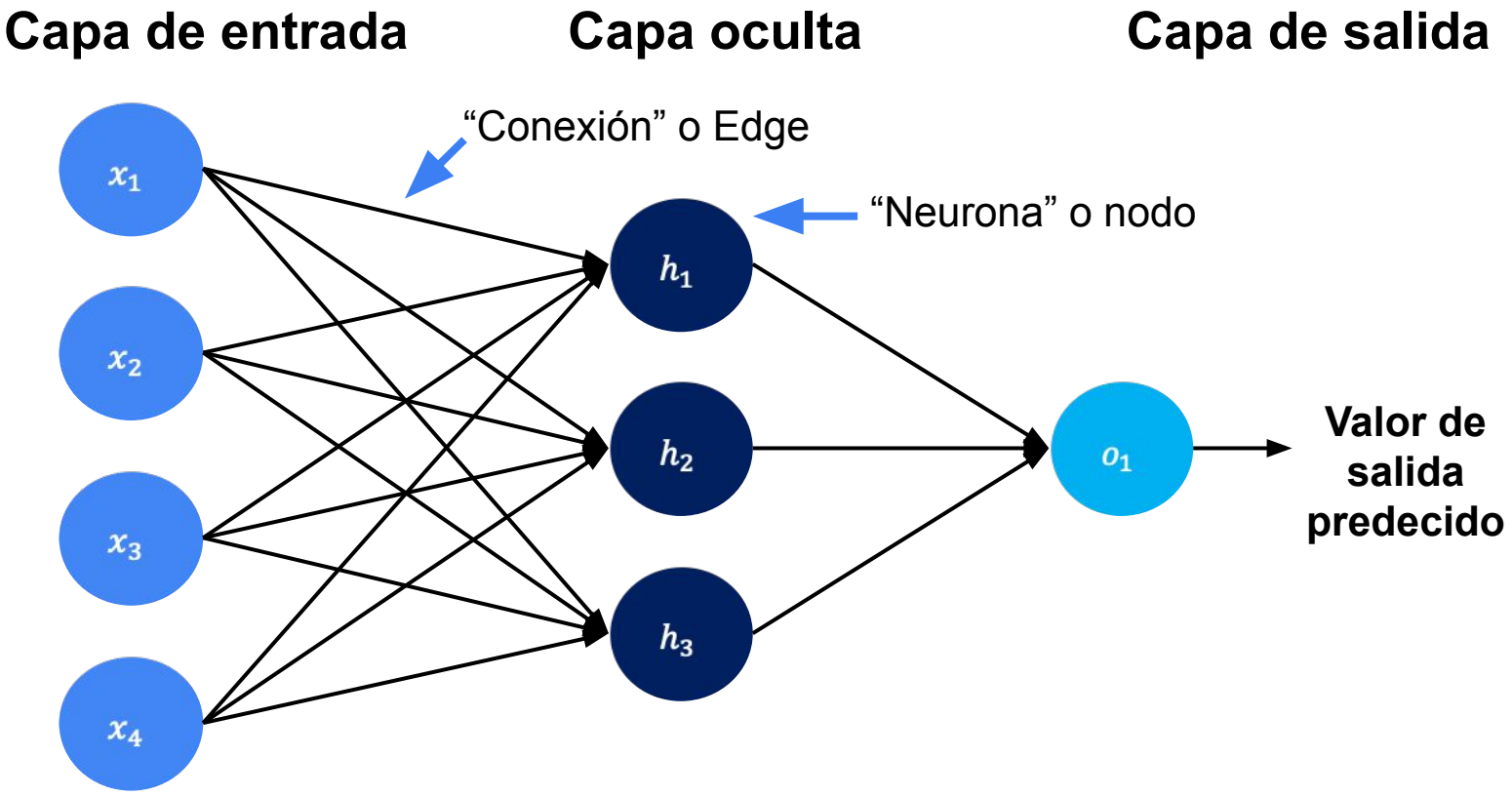
Resumen de módulo

- 3 ingredientes para problemas de ML (proceso de decisión, función de coste, regla de actualización).
- Algoritmos de aprendizaje supervisado (regresión lineal, regresión logística, árboles de decisión).
- Algoritmos de aprendizaje no supervisado (K-means clustering).

Cómo funcionan las redes neuronales

Objetivos de este módulo

1. Entender arquitectura básica de una red neuronal.
2. Entender cómo funciona el entrenamiento de redes neuronales.
3. Explorar cómo mejorar las redes neuronales.

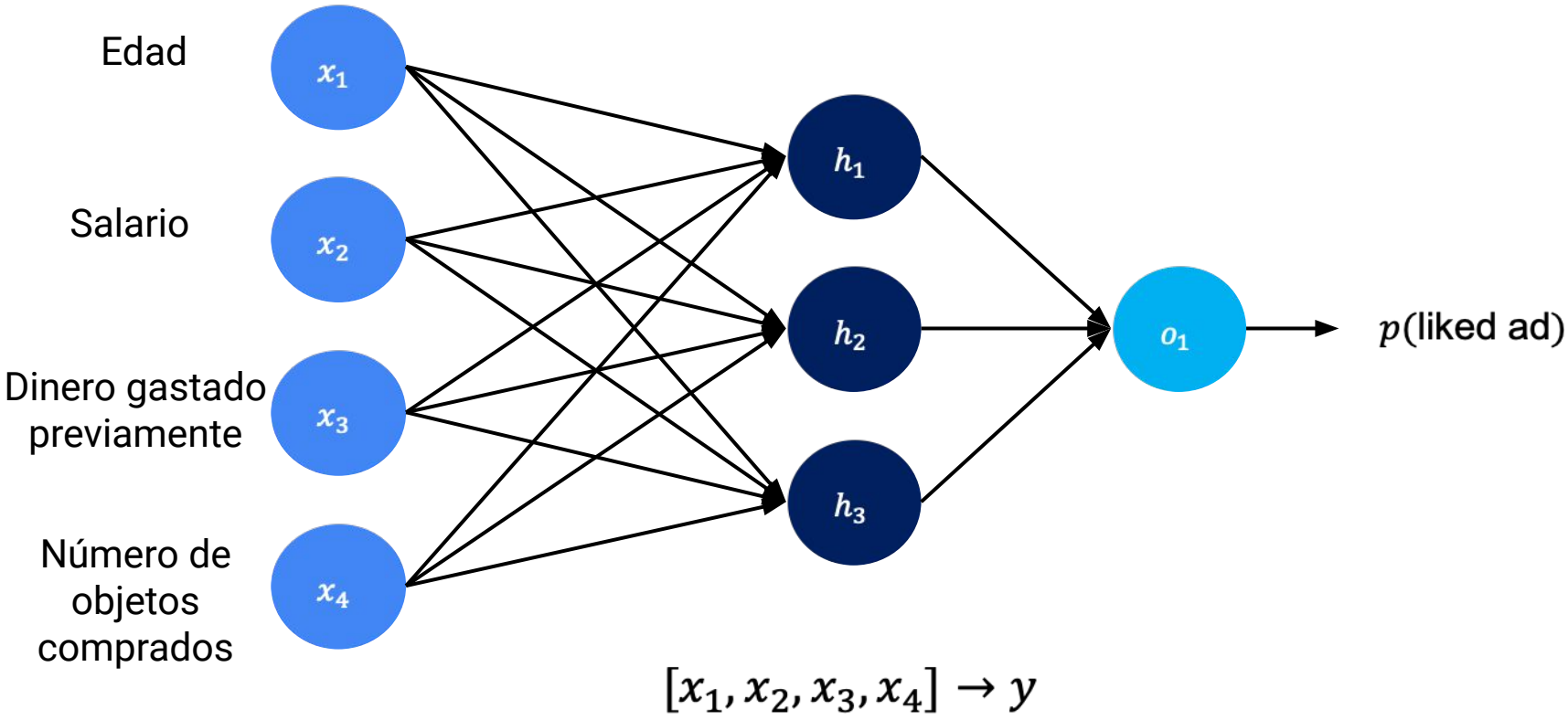


Una red neuronal es un modelo que usa neuronas y conexiones entre ellos para hacer predicciones.

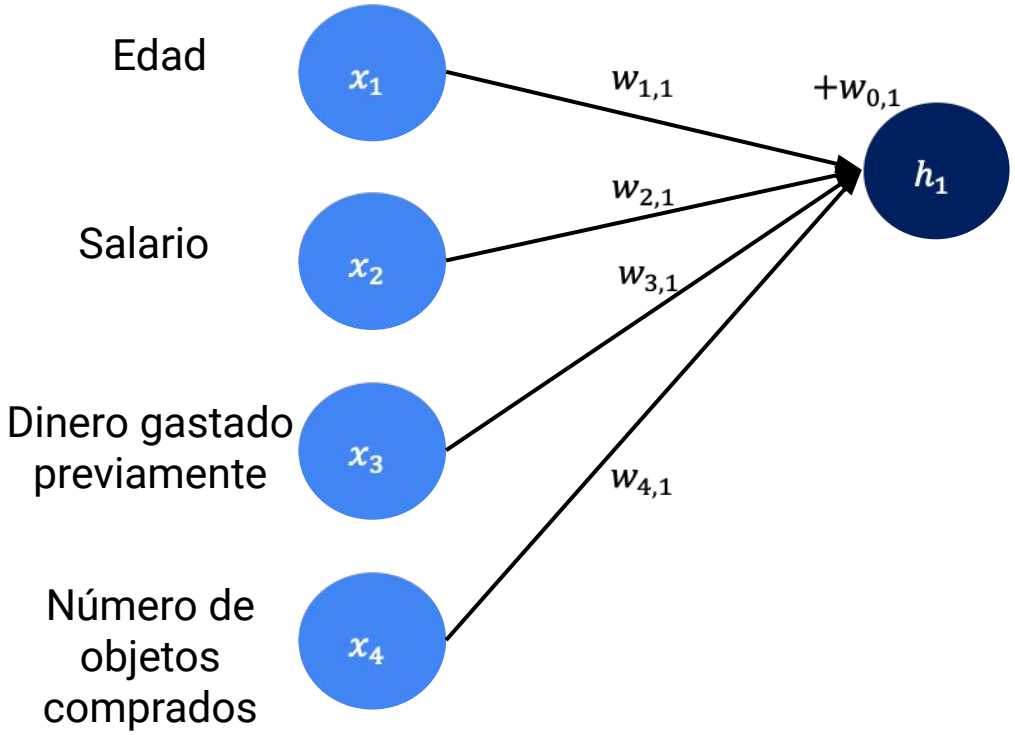
Son usadas usualmente para **aprendizaje supervisado.**

Redes neuronales: un ejemplo

Queremos predecir si a alguien le gustará un anuncio (ad) basado en su historial de compra.



Capa de entrada a una unidad oculta

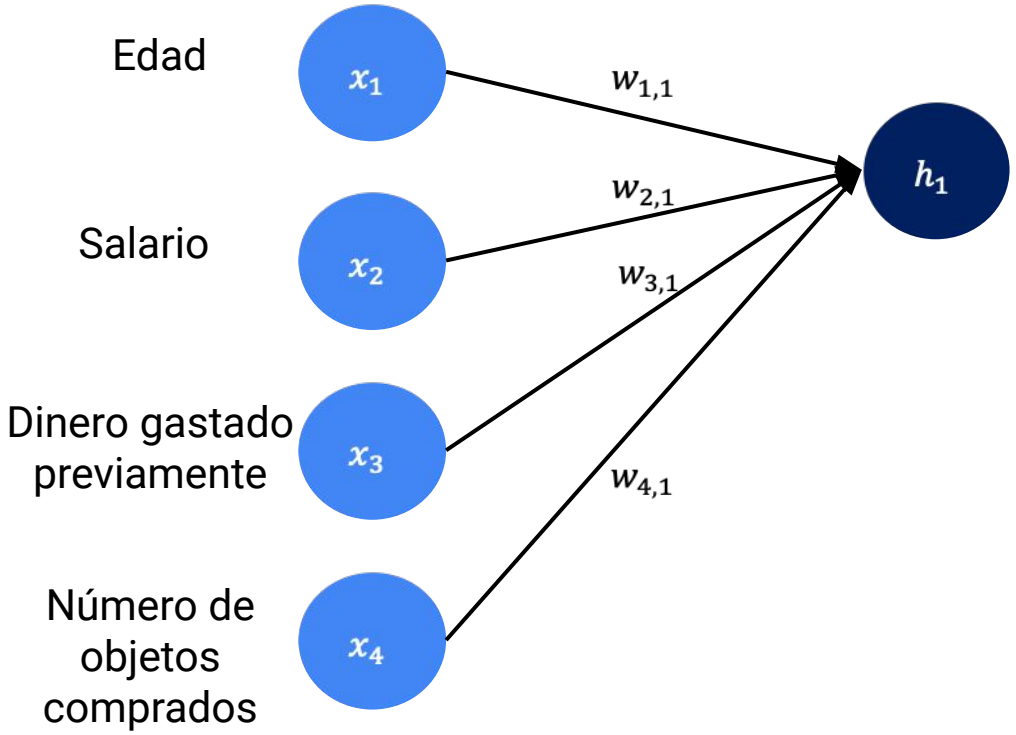


Los pesos gobiernan la **fuerza** de una conexión entre nodos. Pesos altos == conexión fuerte.

La unidad escondida (nodo) acepta una combinación lineal de nodos adjuntos previamente a él.

$$g_1(x) = w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2 + w_{3,1}x_3 + w_{4,1}x_4$$

Capa de entrada: un ejemplo



Conocemos esto de un cliente:

- Edad: 23 años
- Gana $35,000 dólares al año.
- Nuevo cliente (sin dinero gastado u objetos comprados previamente).

Capa de entrada: un ejemplo

Asumir los pesos:

$$w_{0,1} = 0.2$$

$$w_{1,1} = \frac{1}{10}$$

$$w_{2,1} = \frac{1}{10000}$$

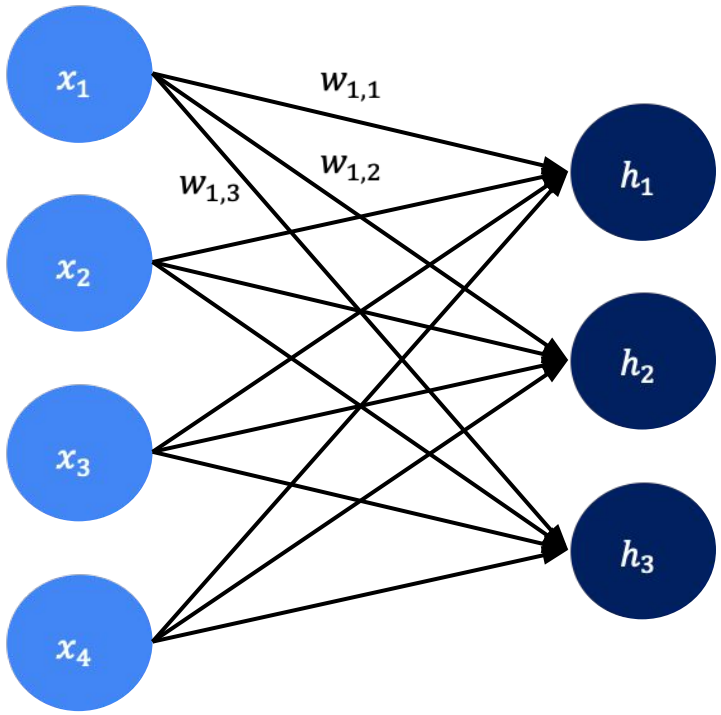
$$w_{3,1} = 1$$

$$w_{4,1} = 2$$

$$g_1(x) = \left(\frac{1}{10}\right)(23) + \left(\frac{1}{10000}\right)(35000) + (1)(0) + (2)(0) + 0.2$$

$$g_1(x) = 7$$

Capa de entrada conecta con múltiples unidades ocultas



Cada unidad oculta recibe una combinación lineal de todas las entradas.

Para la misma entrada existen diferentes pesos dependiendo de cuál unidad oculta esté adjunta. Estos pesos pueden ser distintos.

Ejemplo:
$W_{2,3}$ —— El peso de la segunda entrada a la tercera unidad.

Hidden units perform a function

La unidad oculta ejecuta una función en la combinación lineal ($g_1(x)$).
Esta función es una **función de activación.**

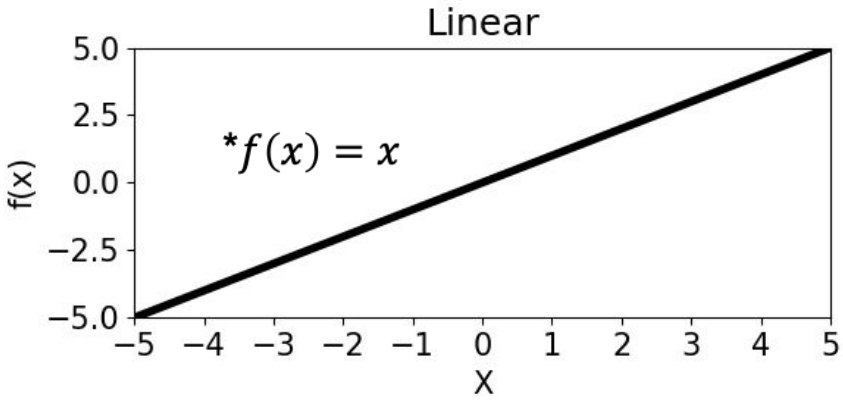
$$h_1(x) = f_1\big(g_1(x)\big)$$

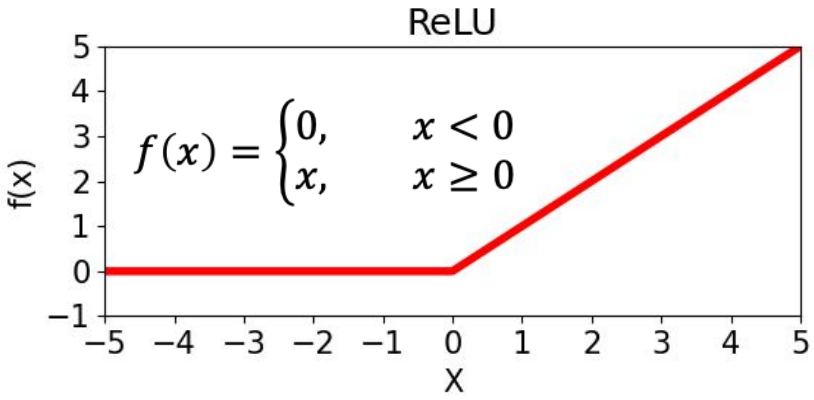
Función de activación

Combinación lineal de nodos de la capa anterior

Tipos de función de activación de capas ocultas

Función de activación “pasa” una señal.

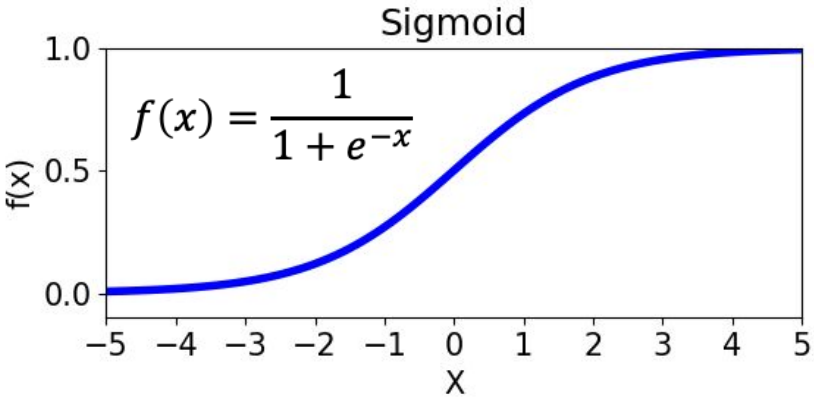


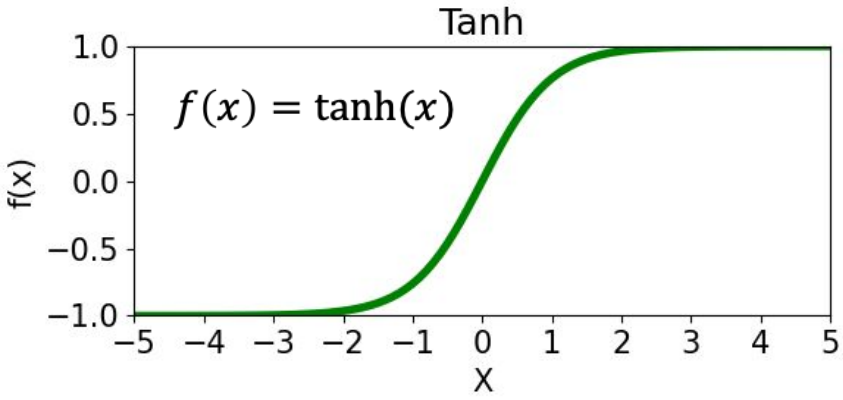


**Usar no linealidad para funciones complejas.*

Tipos de función de activación de capas ocultas

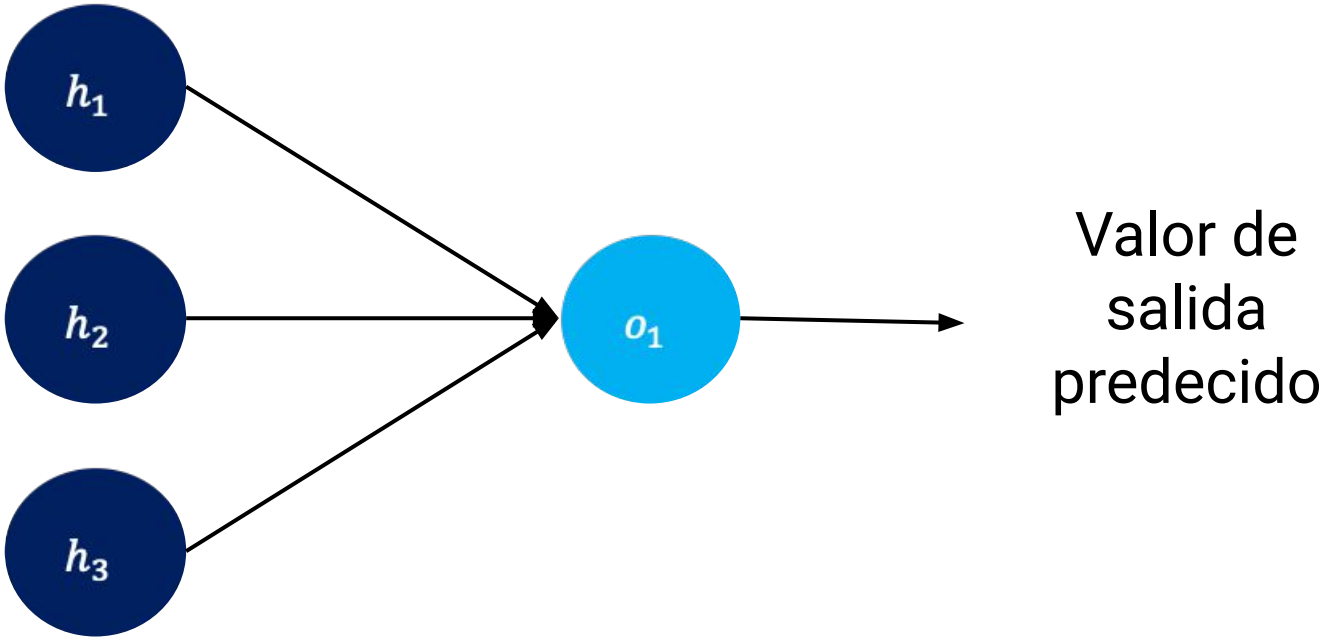
Función de activación “pasa” una señal.





**Usar no linealidad para funciones complejas.*

La predicción: la capa de salida

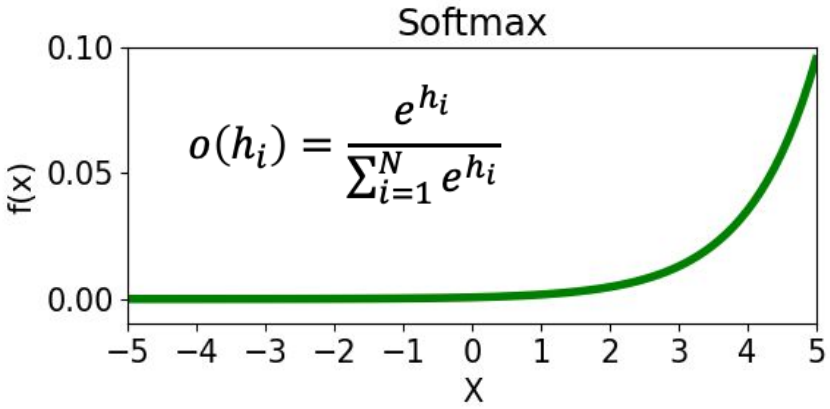


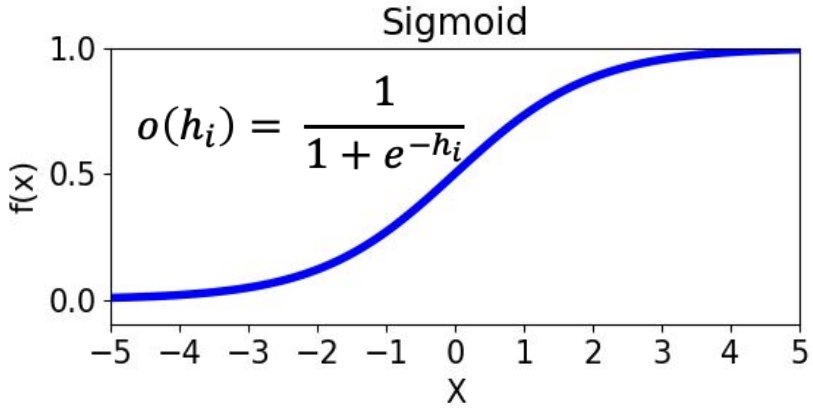
Existen **diferentes funciones de activación** para clasificación y regresión.

Tipos de función de activación para la capa de salida

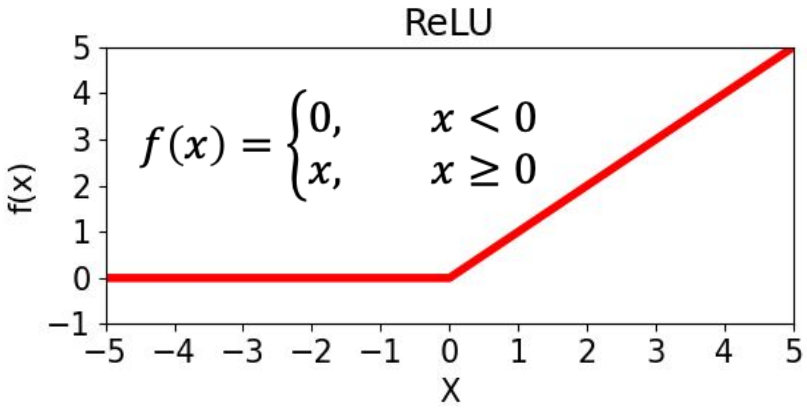
- **Algunas** funciones de activación tienen un **rango limitado** (ex: softmax/sigmoide), mientras que otras se **extienden indefinidamente** (ReLUs, lineal).

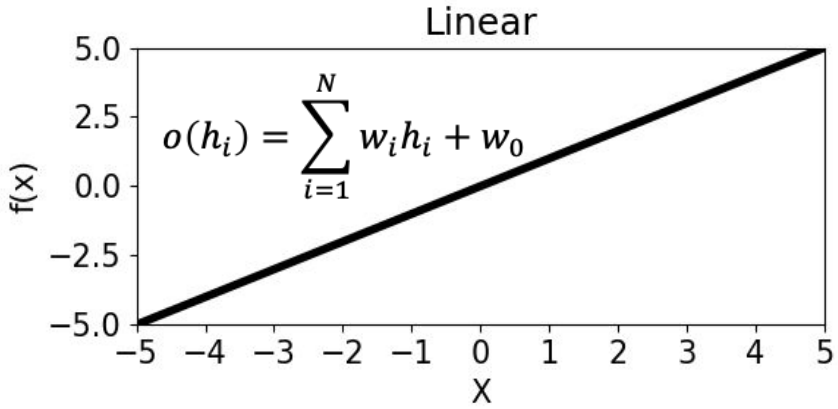
Clasificación





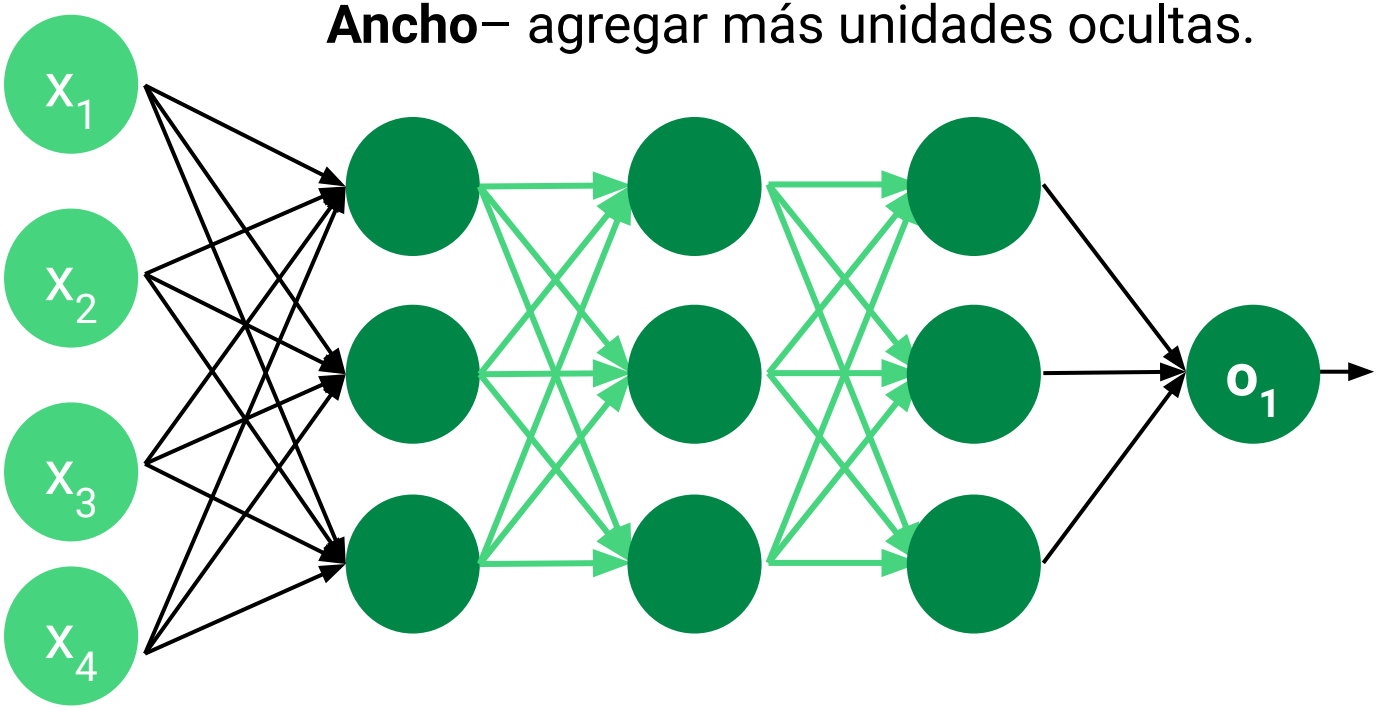
Regresión





Deep learning: añadiendo más capas

Profundidad – agregar más capas.
Ancho– agregar más unidades ocultas.



Entrenamiento de redes neuronales

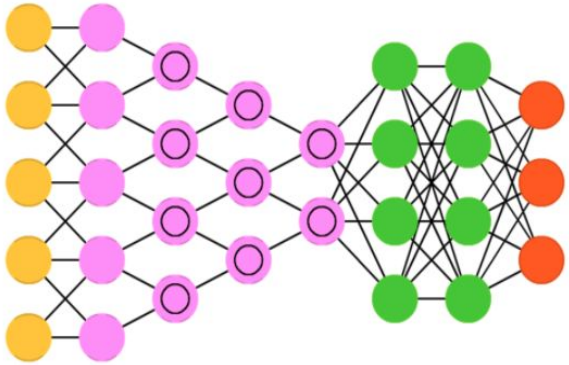
Entrenar neural networks

1. Escoge tu arquitectura.
2. La “receta” de entrenamiento.
3. Ajustar tu tasa de entrenamiento.

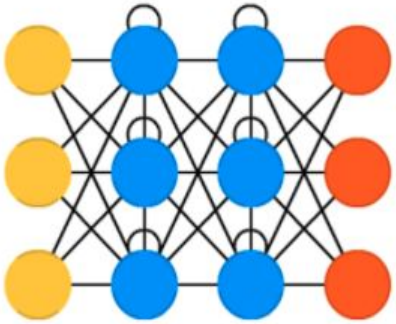
Escoge tu arquitectura

Deep Feed-Forward (DNNs)

Convolucional (CNNs)



Recurrente (RNNs)



Referencias:
[1] Fjodor van Veen and Stefan Leijnen. “The Neural Network Zoo”. The Asimov Institute (2016)
https://www.asimovinstitute.org/neural-network-zoo/

Escoge tu arquitectura

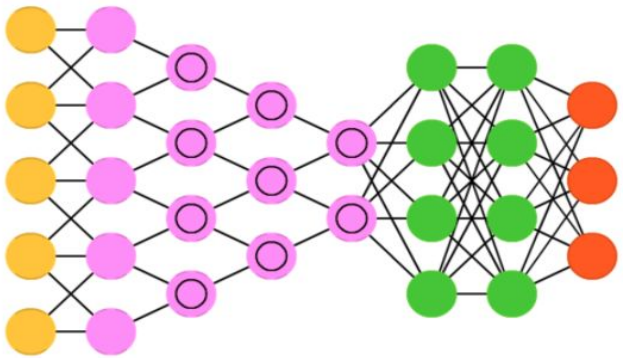
Deep Feed-Forward (DNNs)

- Funciones de activación.
- Usada en muchos problemas complejos.

References:
[1] Fjodor van Veen and Stefan Leijnen. “The Neural Network Zoo”. The Asimov Institute (2016) https://www.asimovinstitute.org/neural-network-zoo/

Escoge tu arquitectura

Convolucional (CNNs)

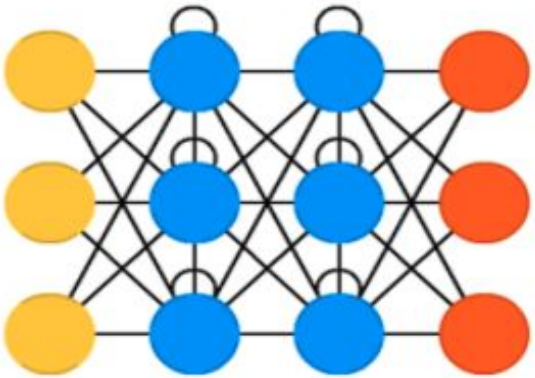


- Operador convolucional/pool y kernels.
- Usada en imágenes y genómicos.

Referencias:
[1] Fjodor van Veen and Stefan Leijnen. “The Neural Network Zoo”. The Asimov Institute (2016) https://www.asimovinstitute.org/neural-network-zoo/

Escoge tu arquitectura

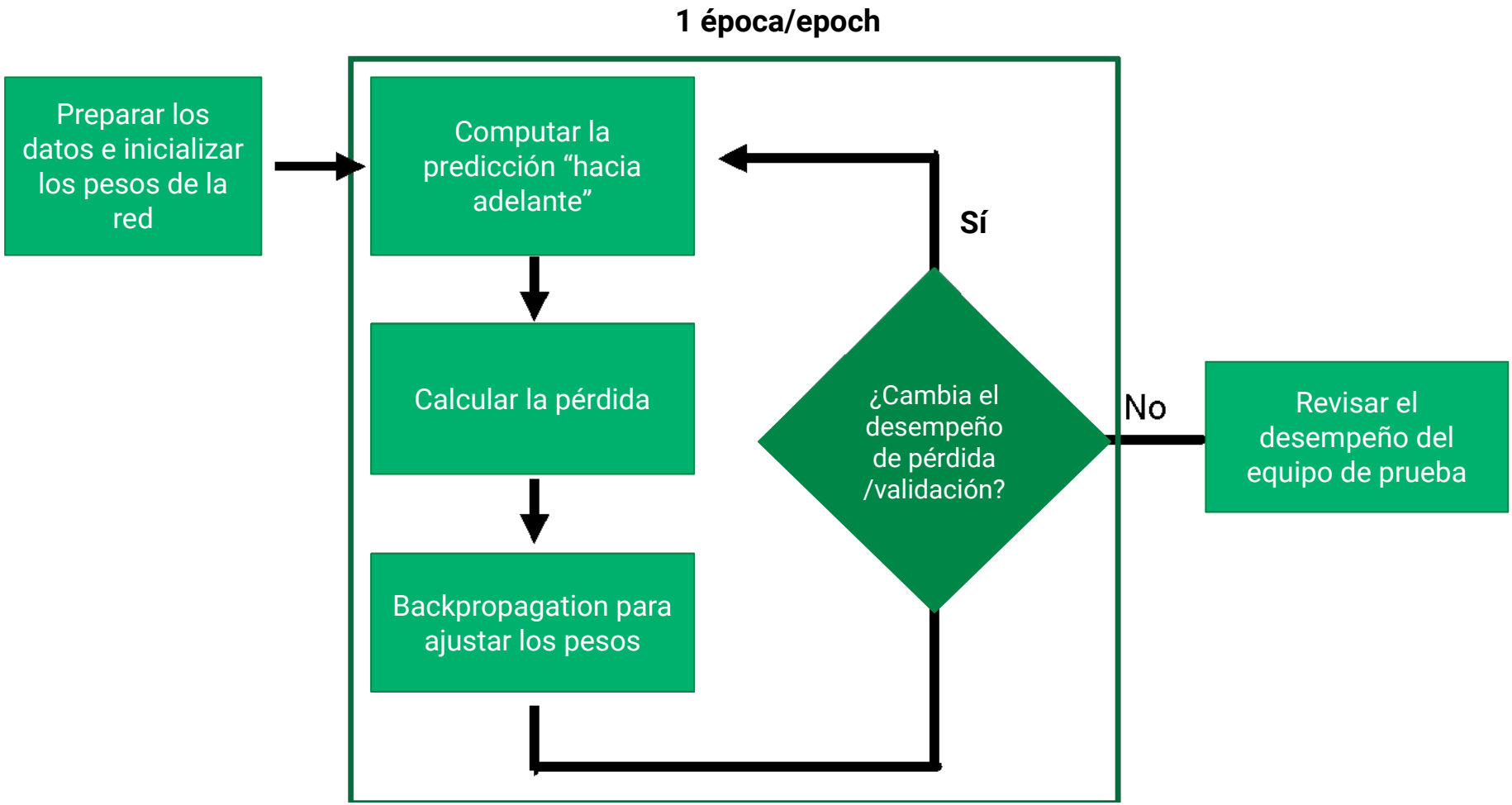
Recurrente (RNNs)



- Celdas de memoria/puerta.
- Representa secuencias.
- Usada en lenguaje.

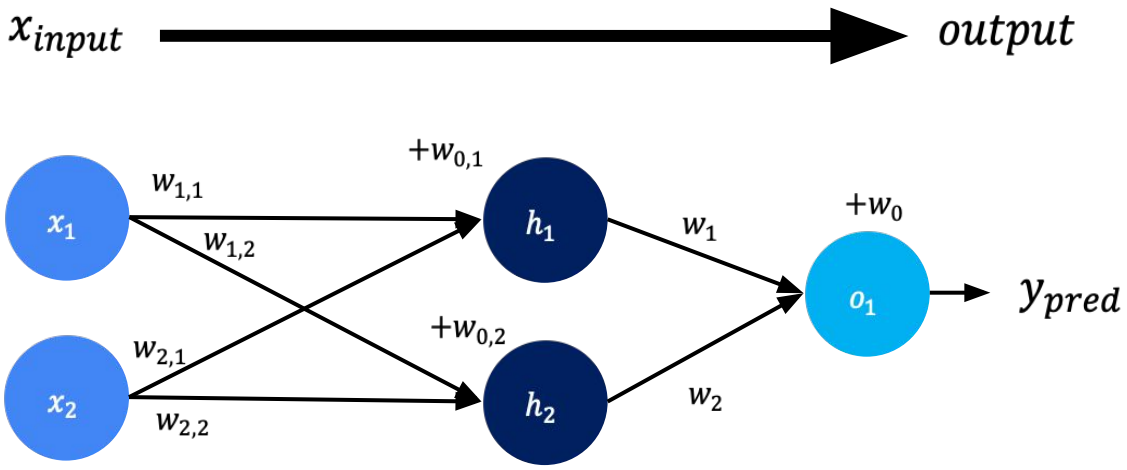
Referencias:
[1] Fjodor van Veen and Stefan Leijnen. “The Neural Network Zoo”. The Asimov Institute (2016) https://www.asimovinstitute.org/neural-network-zoo/

La receta para redes neuronales



Avance hacia adelante

Evaluar las funciones en cada nodo



$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + w_{0,1}$$
$$h_2 = w_{1,2}x_1 + w_{2,2}x_2 + w_{0,1}$$
$$y_{pred} = o_1 = w_1h_1 + w_2h_2 + w_0$$

Función de pérdida (error/coste)

Error cuadrático medio (MSE)

Para **regresión.** Diferencia en valor verdadero versus predecido.

$$J = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - y_{i,pred}\right)^2$$

Función de pérdida (error/coste)

Binary Cross-Entropy (BCE)

Para **clasificación**. Calcula qué tan confiable el modelo predice la probabilidad de una clase para 2 clases.

$$J = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\mathrm{p}(\mathrm{Class}_1)) + (1 - y_i)\log\left(1 - \mathrm{p}(\mathrm{Class}_1)\right)$$

Función de pérdida (error/coste)

Categorical Cross-Entropy (CCE)

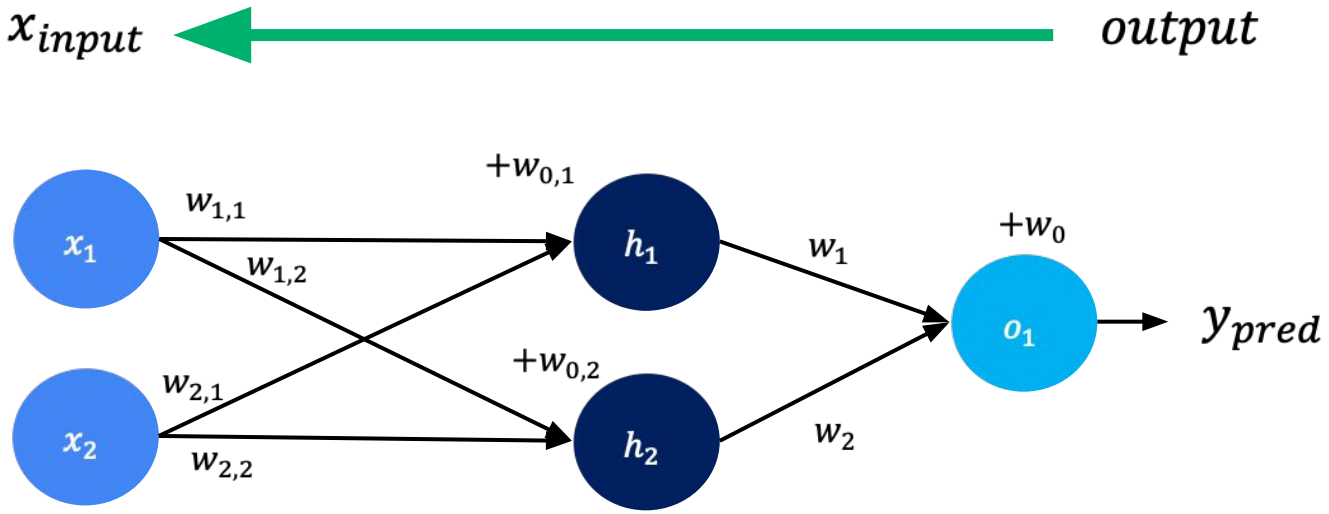
Para **clasificación.** Similar a BCE, pero para más de 2 etiquetas.

$$J = \sum_{i=1}^{C} -t_i \log(p_i)$$

C – number of classes
t_i – true label
p_i – probability of predicting label

Backpropagation

Backpropagation es la “regla de actualización” usada para ajustar los pesos en redes neuronales.



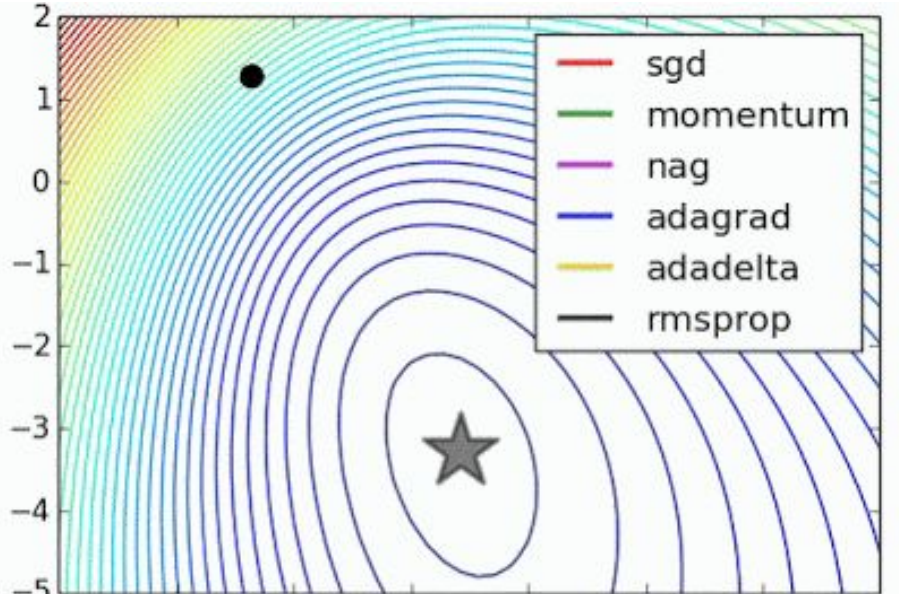
Backpropagation

Actualizamos **cada peso en la red** **tomando la derivada (parcial) de ello con respecto a cada nodo**, empezando desde la salida hacia atrás hasta las entradas.

Analogía de la tasa de aprendizaje

La optimización se usa para encontrar pesos de acuerdo a las reglas de actualización.

- Si la tasa de aprendizaje (learning rate) es **muy pequeña**, toma mucho tiempo encontrar buenos pesos.
- Si es **muy grande**, el modelo podría atorarse en una mala solución.

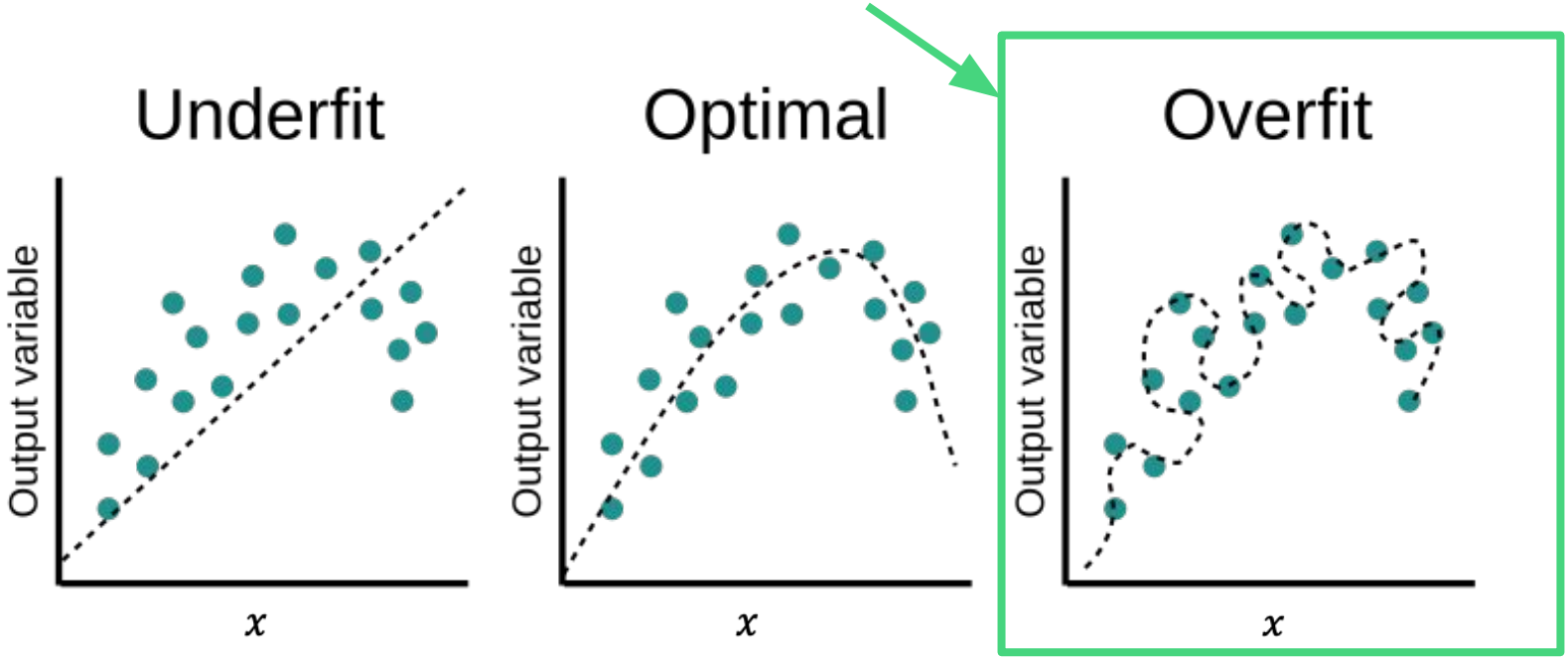


Referencias:
[1] Figure generated by Alec Radford; Hosted on Deniz Yuret's Homepage http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Cómo mejorar las predicciones de redes neuronales

Dropout protege ante el overfitting

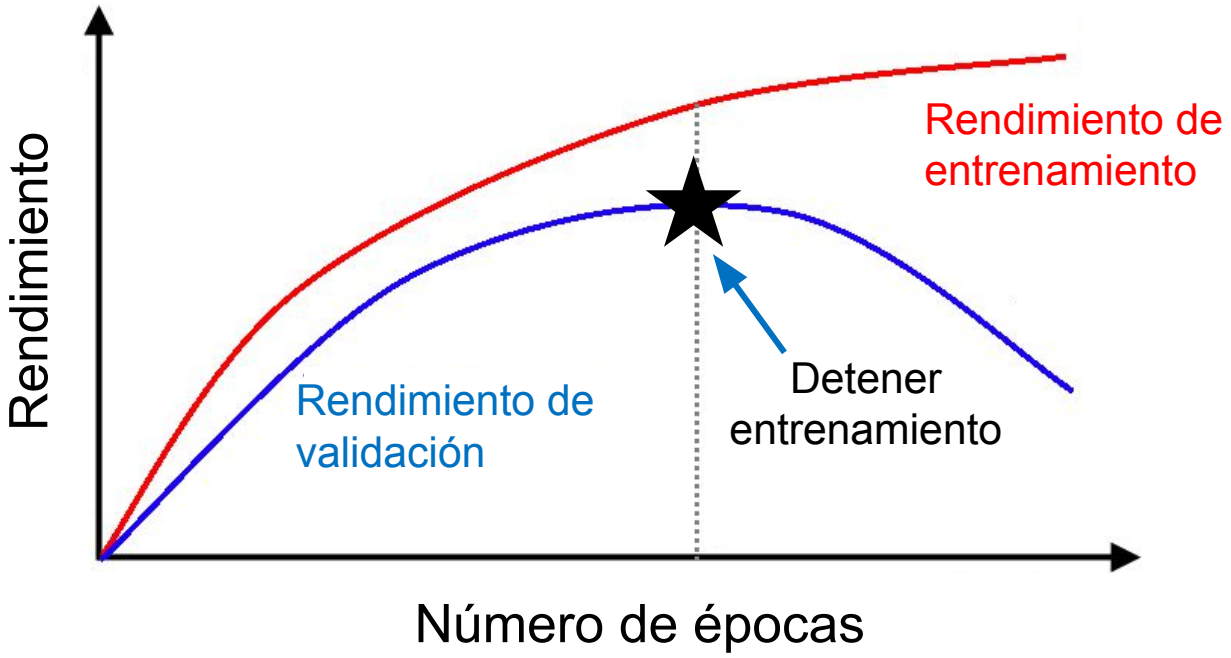
Las redes neuronales tienden al overfitting porque tienen muchos parámetros.



Dropout aleatoriamente “ignora” algunos nodos mientras entrena.

Referenclas:
[1] Figure adapted from OlexSys “Data/Parameter Ratio” https://www.olexsys.org/olex2/docs/reference/diagnostics/data-parameter-ratio/

Prevenir el overfitting



Referencias:
[1] Figure adapted from KnimeAnalytics “How to generate Accuracy Curve by Epoch to detect Overfitting with Training & Validation Data” https://forum.knime.com/t/how-to-generate-accuracy-curve-by-epoch-to-detect-overfitting-with-training-validation-data/11916

Conclusiones

Resumen del módulo

- Las redes neuronales tienen **tres capas generales:** entrada, ocultas, salida.
- **Funciones de activación** para capas ocultas y de salida.
- Entrenar involucra **avanzar en la red**, **calcular la pérdida** y **backpropagation.**
- Tasa de aprendizaje y dropout son importantes para el entrenamiento.
- Revisar la pérdida y el rendimiento en el set de validación.

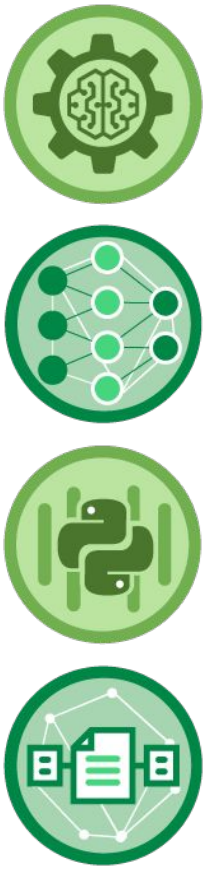
¿Qué más aprender de inteligencia artificial?

Qué aprendimos

- Bases de ciencia de datos y visualizaciones.
- Algoritmos de aprendizaje supervisado.
- Algoritmos de aprendizaje no supervisado.
- Redes neuronales y deep learning.

Qué más aprender

- **Uso práctico y avanzado** de algoritmos de machine learning y redes neuronales.
- Uso avanzado de **librerías ML** como scikit-learn y TensorFlow.
- **Despliegue** de aplicaciones de ML.
- **Procesamiento de lenguaje natural** y más.



¡Felicidades!

- ¡Completa todos los retos!
- Aprueba el examen.
- Deja un review de 5 estrellas.
- ¡Nos vemos la próxima!