1ª edición

Machine Learning:

50 conceptos clave para entenderlo

eBooks
Paradigma

Paradigma

Créditos:

Equipo Paradigma Digital

Coordinación: Manuel Zaforas.

Edición: Ana Domínguez.

Maquetación: David Mota.

Texto: Marco Russo.

Beatriz Blanco.

Alberto Grande.

Francisco Rodes.

Juan Iglesias.

Carlos Navarro.

José Manuel Carral.

Manuel Zaforas.

En esta edición:

Introducción. 8

Glosario de términos. 10

— 01. Accuracy. 12

— 02. Algorithm. 14

— 03. AutoML. 16

— 04. Backpropagation. 18

— 05. Bagging. 20

— 06. Batch. 22

— 07. Bias. 24

— 08. Boosting. 26

— 09. Classification. 28

— 10. Clustering. 30

— 11. Confusion matrix. 34

— 12. Convolutional Networks. 36

— 13. Decision tree. 38

— 14. Deep Learning. 40

— 15. Dimensionality Reduction. 42

— 16. Epoch. 44

— 17. Explainability. 46

— 18. Exploratory Data Analysis. 48

— 19. Feature. 50

— 20. Feature Engineering. 52

— 21. Generative Adversarial Networks (GAN). 56

—22.	GPT-2.	58
—23.	Gradient Descent.	60
—24.	Hyperparameters.	62
—25.	Learning rate.	64
—26.	Long Short-Term Memory (LSTM).	66
—27.	Loss.	68
—28.	MLOps.	70
—29.	MNIST.	72
—30.	Neural Networks.	74
—31.	NLP.	78
—32.	Overfitting.	80
—33.	Perceptron.	82
—34.	Principal Component Analysis (PCA).	84
—35.	Random Forest.	86
—36.	Regression.	88
—37.	Regression metrics.	90
—38.	Reinforcement Learning.	92
—39.	Rectified Linear Unit (ReLU).	94
—40.	Sigmoid function.	96
—41.	Singularity.	100
—42.	Softmax.	102
—43.	Supervised/unsupervised learning.	104
—44.	SVM (Support Vector Machine).	106

—45. TensorFlow. 108

—46. Time series analysis. 110

—47. Training Set. 112

—48. Transfer Learning. 114

—49. Underfitting. 116

—50. Variance. 118

Autores. 122

—01. Alberto Grande. 124

—02. Beatriz Blanco. 125

—03. Carlos Navarro. 126

—04. Francisco Rodes. 127

—05. José Manuel Carral. 128

—06. Juan Iglesias. 129

—07. Manuel Zaforas. 130

—08. Marco Russo. 131

Conclusión. 132

Introducción:

En los últimos años las técnicas de Machine Learning han ganado relevancia. Han demostrado tener aplicaciones muy útiles y un gran impacto en los procesos de negocio de las empresas.

Esto se debe al enorme aumento de la capacidad de cómputo, el gran volumen de datos que las empresas manejan y las técnicas o algoritmos que se han desarrollado en los últimos años.

“Nuestras máquinas son tontas, y solo estamos tratando de hacerlas menos tontas.”

Yoshua Bengio, Investigador en IA en la Universidad de Montreal y uno de los padres del Deep Learning.

Sin embargo, se trata de un dominio complejo, con multitud de conceptos y términos particulares, lo que hace difícil adentrarse en este universo y poder entender en profundidad las ventajas, aplicaciones y el funcionamiento de estas técnicas.

En este ebook hemos recopilado y explicado, de una forma sencilla, los 50 principales términos o conceptos en torno al mundo del Machine Learning que cualquier persona que quiera comprender este mundo debe conocer.

— 01

Glosario de términos.

— 01.01

Accuracy.

(Exactitud)

La exactitud es una métrica que nos indica cómo de preciso es un modelo de Machine Learning a la hora de hacer predicciones.

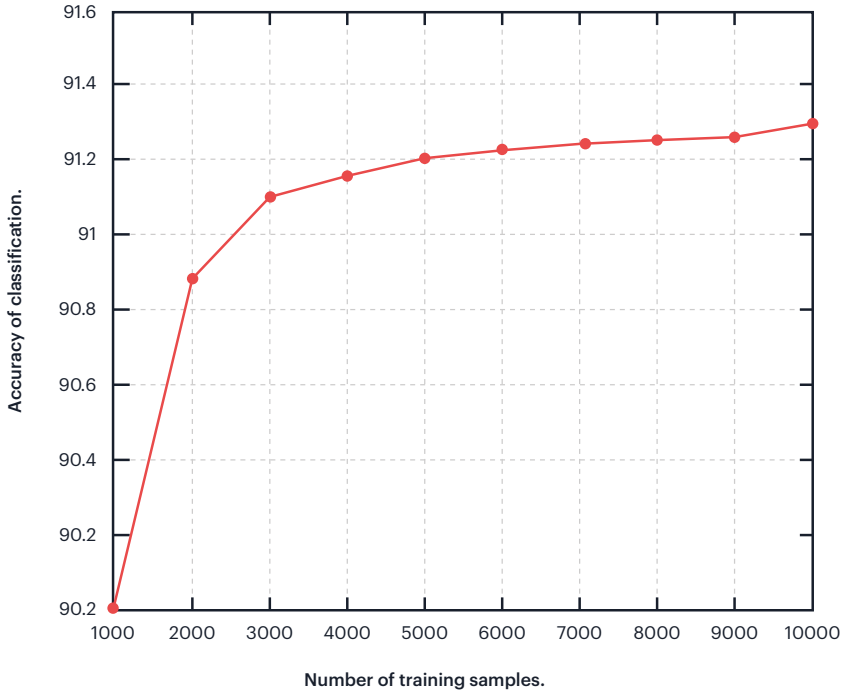
Formalmente, se define como el ratio entre las predicciones correctas respecto al número total de ejemplos. Cuanto más alta sea la exactitud, es decir más cercano a 1, más acertará nuestro modelo y, por lo tanto, será más preciso.

Fijarnos solo en la exactitud no es suficiente y puede ser engañoso en ocasiones. Existen otras métricas que podemos observar a la hora de evaluar cómo de bueno es un modelo, por ejemplo el Recall, la F1 score o la curva de ROC.

Una manera de aumentar la precisión de un modelo puede ser aumentar el número de datos de entrenamiento.

— 01.01 Accuracy.

$$\text{Exactitud} = \frac{\text{Predicciones correctas}}{\text{Número total de ejemplos}}$$



— 01.02

Algorithm.

(Algoritmo)

Un algoritmo es una secuencia lógica de instrucciones que describen paso a paso la forma de resolver un problema.

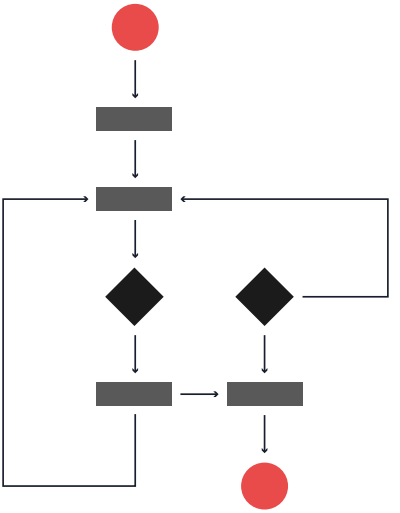
Los algoritmos son un elemento básico en el mundo del Machine Learning y la Inteligencia Artificial.

Muy a menudo, el algoritmo funciona como una secuencia de instrucciones simples if ⟶ then. Otros son más complejos e incluyen ecuaciones o fórmulas matemáticas.

El objetivo de un algoritmo de Machine Learning es definir los pasos necesarios para aprender de los datos y resolver un problema de forma autónoma.

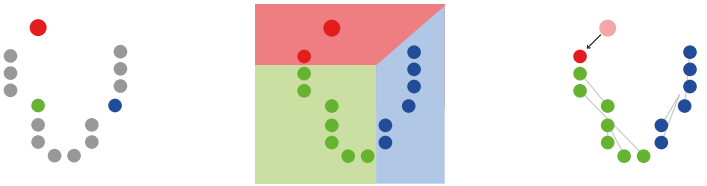
Algunas familias de algoritmos de Machine Learning muy populares son los algoritmos de clustering, regresión o recomendación.

— 01.02 Algorithm.



1. Selecciona los K centroides iniciales.
2. Asigna los elementos xi del conjunto de datos X a su centroide más cercano.
3. Recalcula los nuevos centroides y regresa al paso 2 hasta que converge.

Ejemplo del pseudocódigo del algoritmo K-means



— 01.03

AutoML.

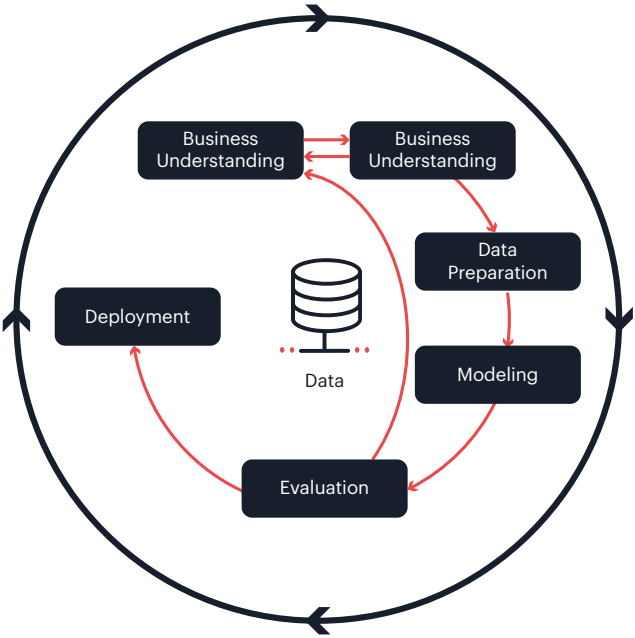
(Auto Aprendizaje Automático)

El proceso de creación de un modelo de Machine Learning suele ser complejo y requiere de personas especializadas con distintas cualidades.

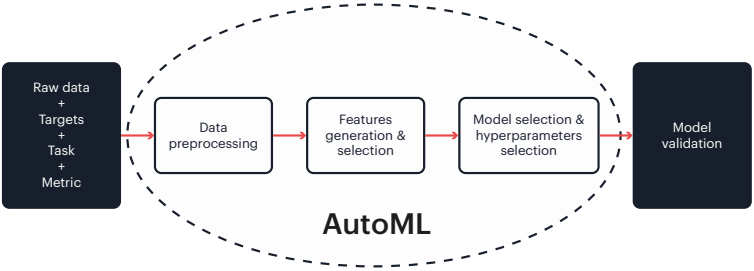
Dentro de estas fases, el propio modelado requiere: especialistas capaces de realizar las tareas de feature engineering, conocimiento de los algoritmos para la optimización de hiperparámetros, experiencia en el desarrollo de software para implementar en código el resultado. AutoML nos ayudará en esta tarea, principalmente en las fases de Modeling y Evaluation de un proyecto típico. AutoML recibirá un conjunto de datos preparados y una tarea a realizar, para ello:

- Buscará una estrategia para preprocesar los datos (cómo tratar los datos no balanceados, cómo codificar las categorías...).
- Generará nuevas características y seleccionará las más significativas.
- Elegirá un algoritmo (algoritmos lineales, redes neuronales...).
- Realizará el tuning de los hiperparámetros del modelo elegido.
- Creará un conjunto estable de modelos para intentar mejorar el scoring obtenido si fuera posible

— 01.03 AutoML.



Ciclo básico de un proyecto.



— 01.04

Backpropagation.

(Retropropagación)

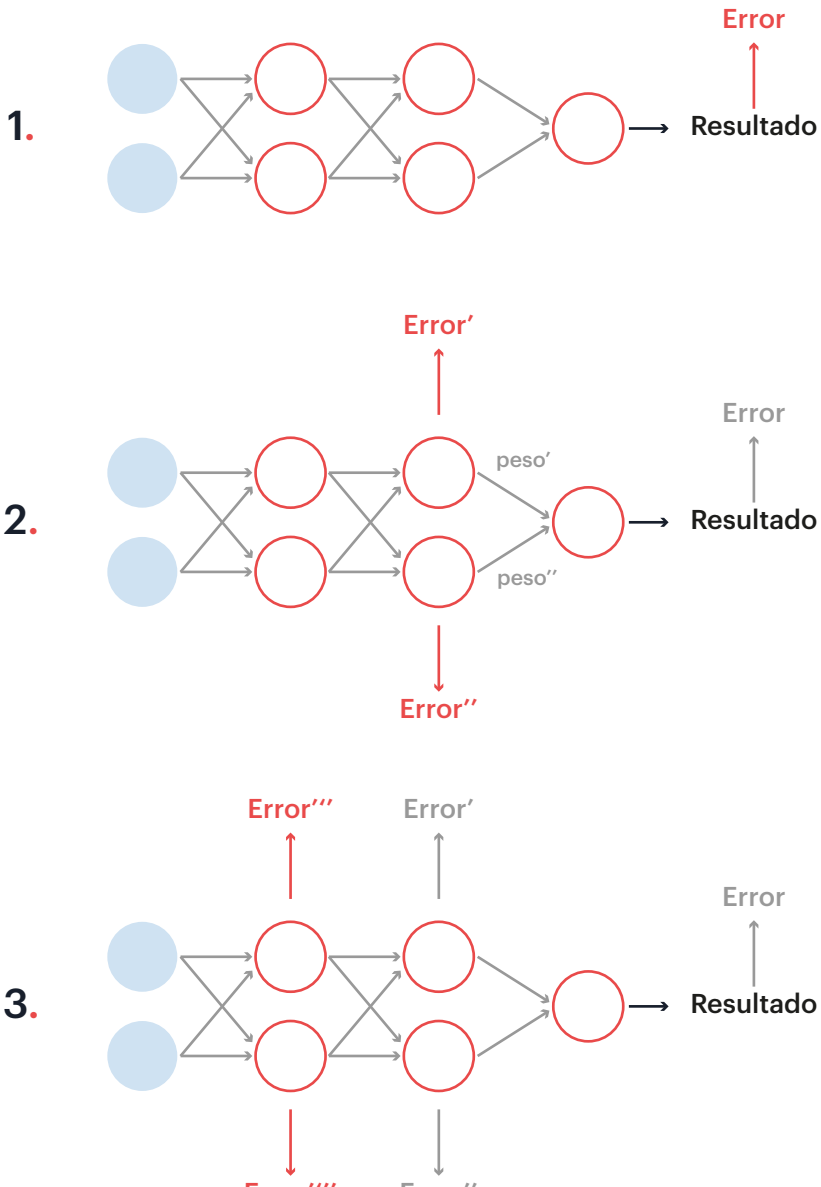
Es uno de los principales algoritmos que se aplica para el entrenamiento de redes neuronales.

Nos permite optimizar los pesos dentro de la red de neuronas en base al ratio de error obtenidos en la anterior iteración.

Calcularemos el error en nuestra neurona de salida y realizaremos un análisis de responsabilidad de cada una de las neuronas dado el error obtenido (en base al peso de cada una de las conexiones), es decir, retro-propagamos los errores.

1. Calculamos el error del resultado obtenido.
2. Propagamos hacia la capa anterior el error, calculando el error asociado a cada una de las neuronas en función del peso.
3. Aplicamos recursivamente propagando el error hacia atrás capa por capa.

— 01.04 Backpropagation.



— 01.05

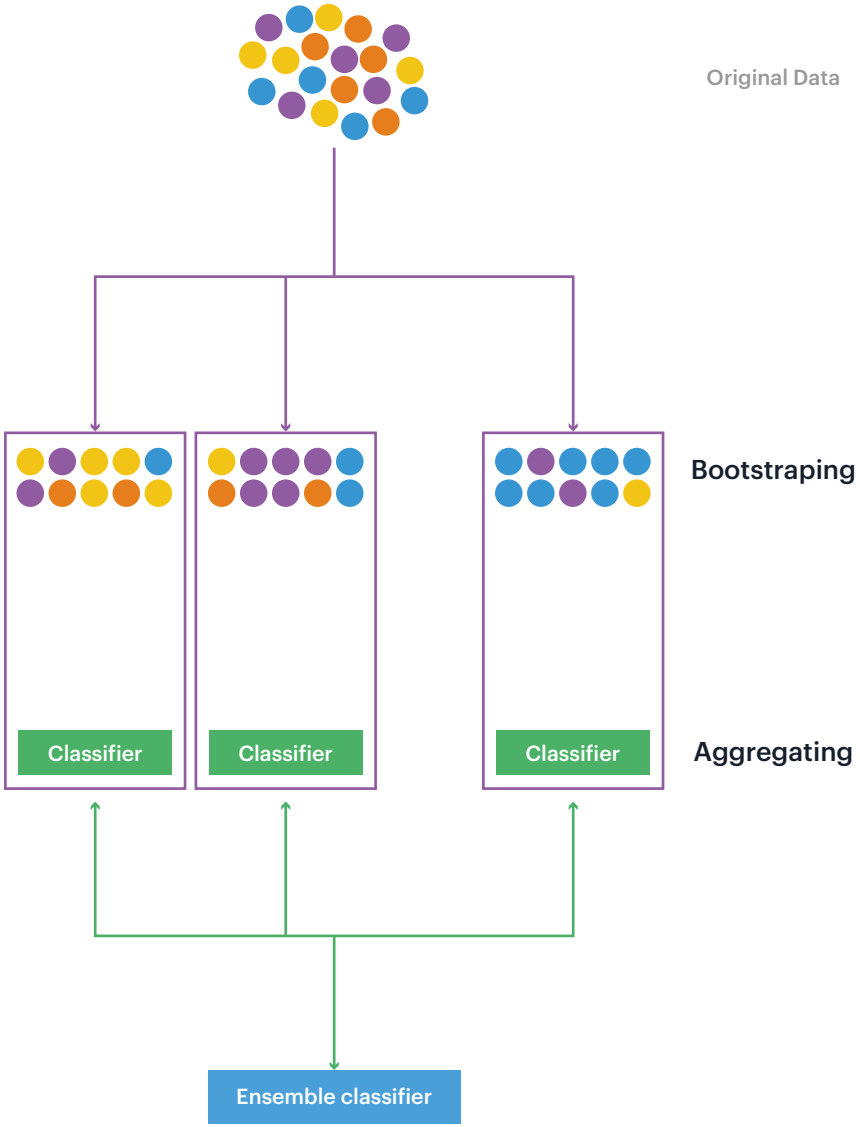
Bagging.

El Bagging (Bootstrapping and aggregating) es una de las técnicas que intentan obtener clasificadores más estables y actualmente este es uno de los campos de investigación abiertos en el ámbito de sistemas de clasificación.

Son efectivos cuando nos encontramos con una distribución de clase equilibradas, sin embargo, se han propuesto muchas modificaciones a los algoritmos que adaptan su comportamiento y los hacen más adecuados para un desequilibrio de clases.

Implica, primero, seleccionar muestras aleatorias de un conjunto de datos de entrenamiento con reemplazo, lo que significa que una muestra dada puede contener cero, uno o más de una copia de ejemplos en el conjunto de datos de entrenamiento (muestra de arranque). Luego se ajusta un modelo de aprendizaje débil en cada muestra de datos. Finalmente, las predicciones de todas las clases débiles en forma se combinan para hacer una predicción única (por ejemplo, agregada).

— 01.05 Bagging.



— 01.06

Batch.

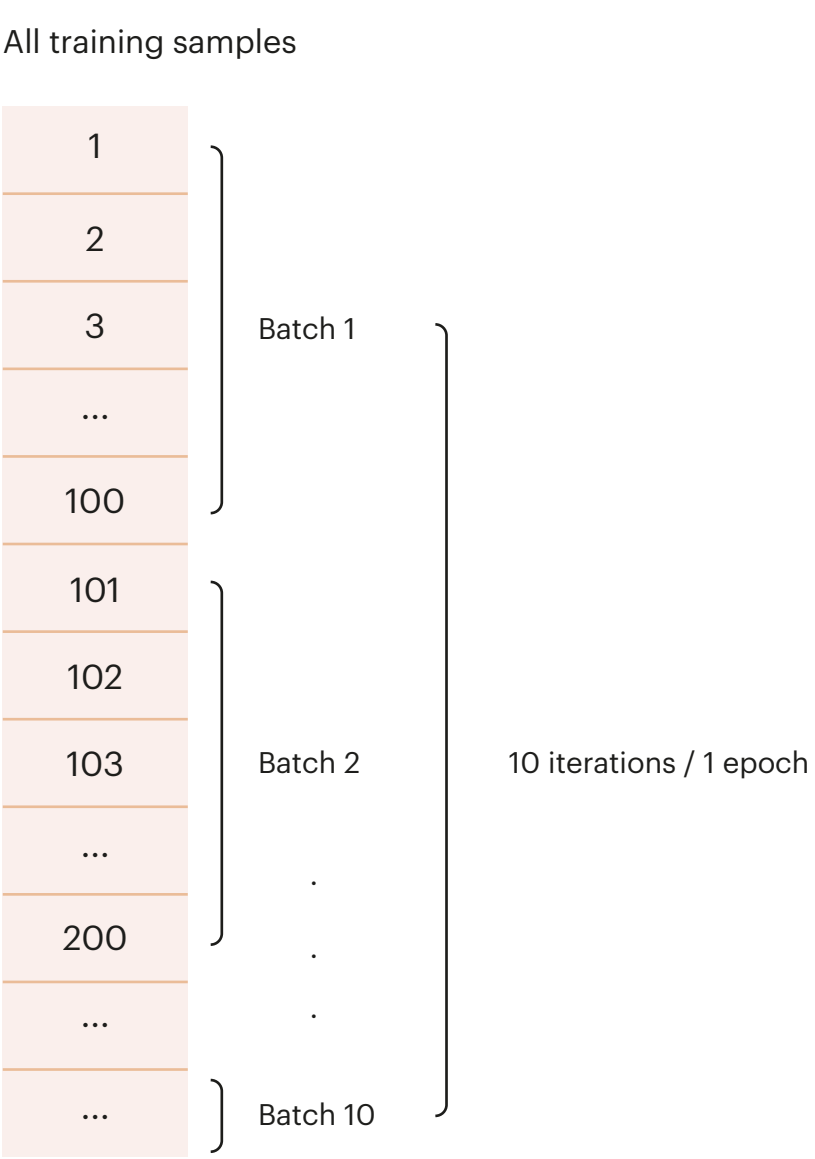
(Lote)

Por Batch se entiende el grupo de muestras de entrenamiento (training samples).

Supongamos que tenemos un problema de clasificación binaria (binary classification) que buscamos resolver utilizando un perceptrón multicapa, del que disponemos de 1000 ejemplos de cada una de las clases.

Una vez que se inicia el entrenamiento, para poder ir actualizando el cálculo de pesos sin tener que esperar a que el modelo haya completado el entrenamiento, se toman, por ejemplo, 100 ejemplos aleatorios de cada clase. A esto lo llamamos lote (batch), se entrena el modelo con este lote, se va actualizando los pesos y pasando al siguiente lote (batch), hasta que se haya procesado todo el conjunto de datos de entrenamiento.

— 01.06 Batch.


All training samples
1
2
3
...
100
101
102
103
...
200
...
...
Batch 1
Batch 2
Batch 10
10 iterations / 1 epoch

— 01.07

Bias.

(Sesgo)

El sesgo en el aprendizaje automático hace referencia al fenómeno de asunciones incorrectas que se dan durante el proceso de entrenamiento y que se verán reflejadas sistemáticamente en los resultados producidos. Los motivos que conducen a un mayor o menor sesgo del modelo tienen origen en el algoritmo y en los datos empleados.

En el primer caso, si durante el proceso de aprendizaje un algoritmo es capaz de predecir la variable objetivo con un acierto del 80%, el 20% de error restante será considerado como el sesgo asociado a ese modelo, dado que las futuras predicciones partirán de ese 20% de error base.

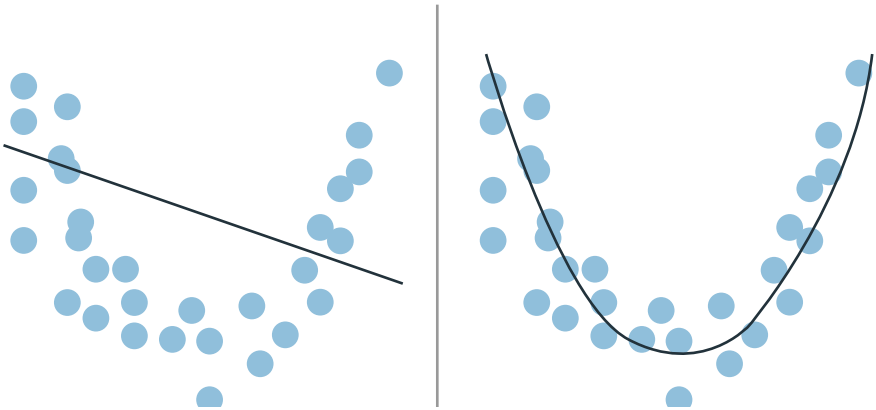
En segundo lugar, los datos están sesgados cuando no recogen toda la información relativa al problema en cuestión o no reflejan la situación real. En este caso, los patrones aprendidos por el modelo se verán directamente afectados y los resultados estarán sesgados.

Por último, es importante mencionar que el término sesgo también es utilizado para referirse a la ordenada en el origen de la función que se optimiza durante el entrenamiento en redes neuronales:

$$y' = b + w^1 x^1 + w^2 x^2 + \dots w^n x^n$$

— 01.07 Bias.

Ejemplo de la función aprendida por dos modelos después del entrenamiento (a), (b).



(a) Presenta mayor sesgo que (b) ya que el error durante el proceso de entrenamiento es claramente menor en (b).

*https://medium.com/thoughts-and-reflections/racial-bias-and-gender-bias-examples-in-ai-systems-7211e4c166a1

— 01.08

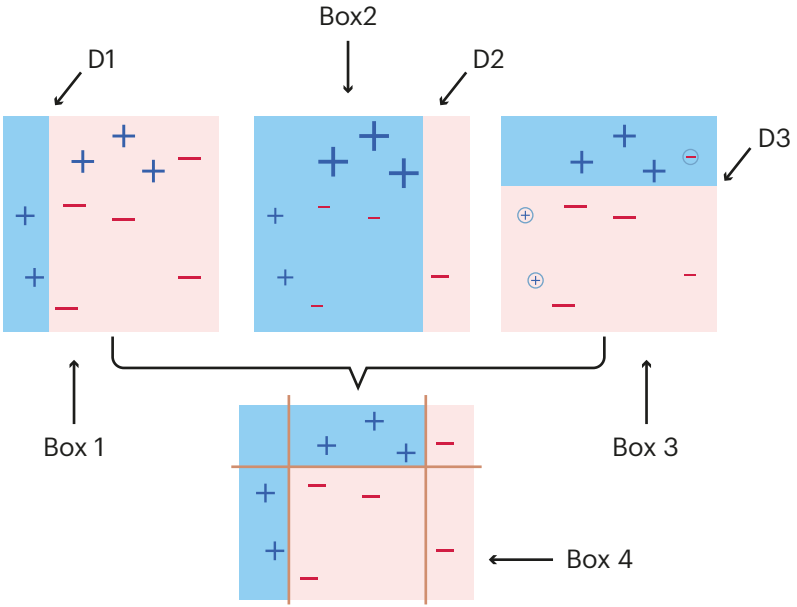
Boosting.

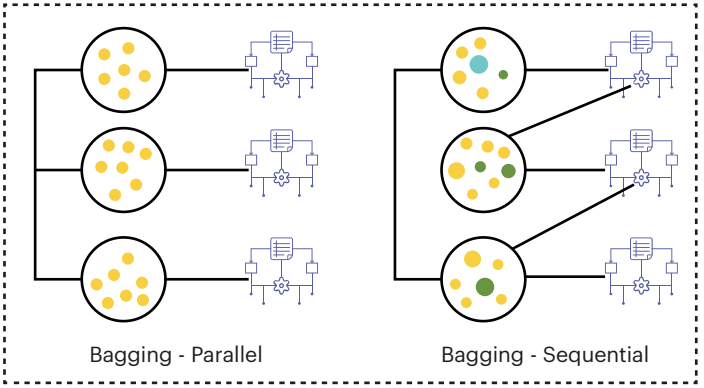
Boosting es un método que aumenta la precisión de un clasificador a partir de varios clasificadores débiles. Utiliza el método de clasificación como una subrutina para producir un clasificador que consigue una alta precisión en el conjunto de entrenamiento.

Boosting aplica el sistema de clasificación varias veces sobre el conjunto de entrenamiento, pero cada vez dirige la atención del aprendizaje a diferentes ejemplos del mismo. Una vez que el proceso ha terminado, los clasificadores básicos obtenidos se combinan en un único clasificador final que será muy preciso en el conjunto de entrenamiento.

El clasificador final, normalmente, logra también una precisión elevada en el conjunto de test, según han demostrado diversos autores tanto teórica como empíricamente. Aunque existen diversas versiones de algoritmos boosting, la más extendida es la que se conoce como AdaBoost y Gradient boosting.

— 01.08 Boosting.





— 01.09

Classification.

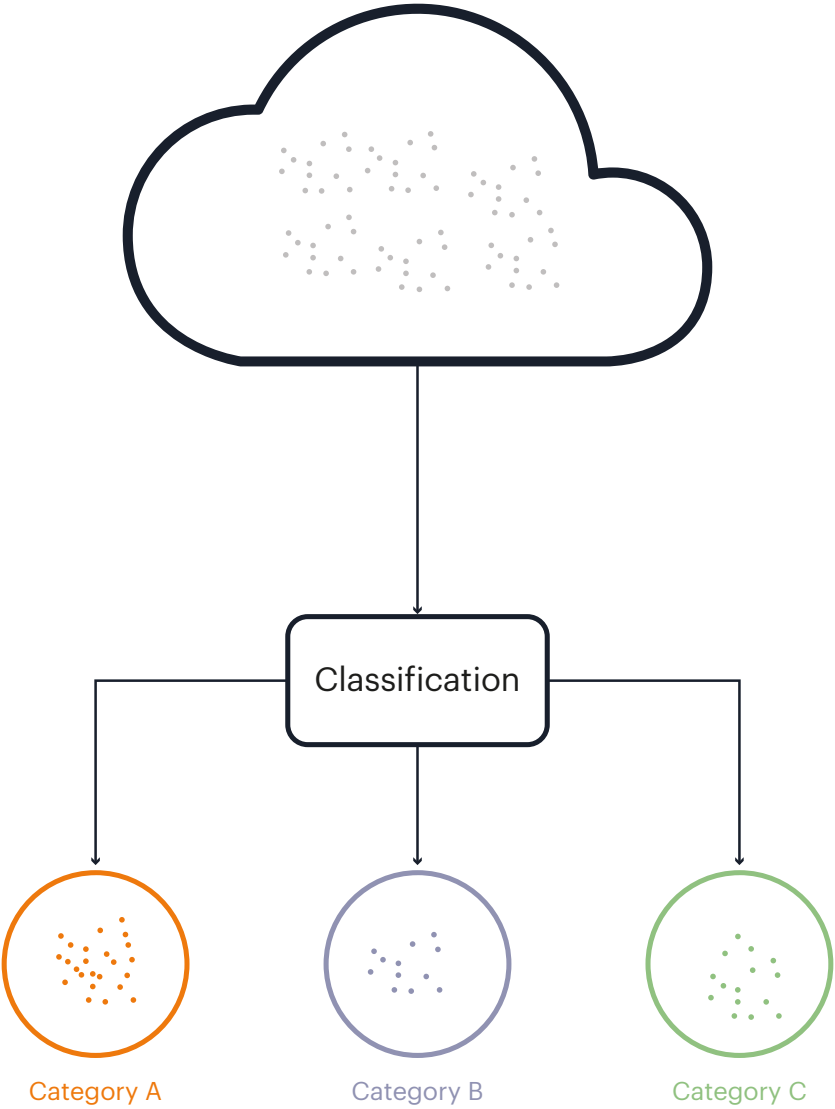
(Clasificación)

La clasificación es una técnica de aprendizaje supervisado que consiste en agrupar elementos en una serie de categorías predefinidas. Por ejemplo, un sistema para etiquetar artículos periodísticos en las categorías nacional, economía, ocio y deportes o un sistema que clasifique opiniones entre positivas, negativas y neutrales.

La diferencia con las técnicas de clustering, que también se encargan de agrupar elementos, es que las técnicas de clustering son no supervisadas, es decir, que no requieren que se introduzca información adicional. En cambio, las técnicas de clasificación requieren que, como mínimo, se definan las categorías en las que un elemento se puede clasificar.

Uno de los retos recurrentes de las técnicas de clasificación es que normalmente requiere de datasets correctamente etiquetados y suficientemente grandes, los cuales pueden ser difíciles de obtener.

— 01.09 Classification.



— 01.10

Clustering.

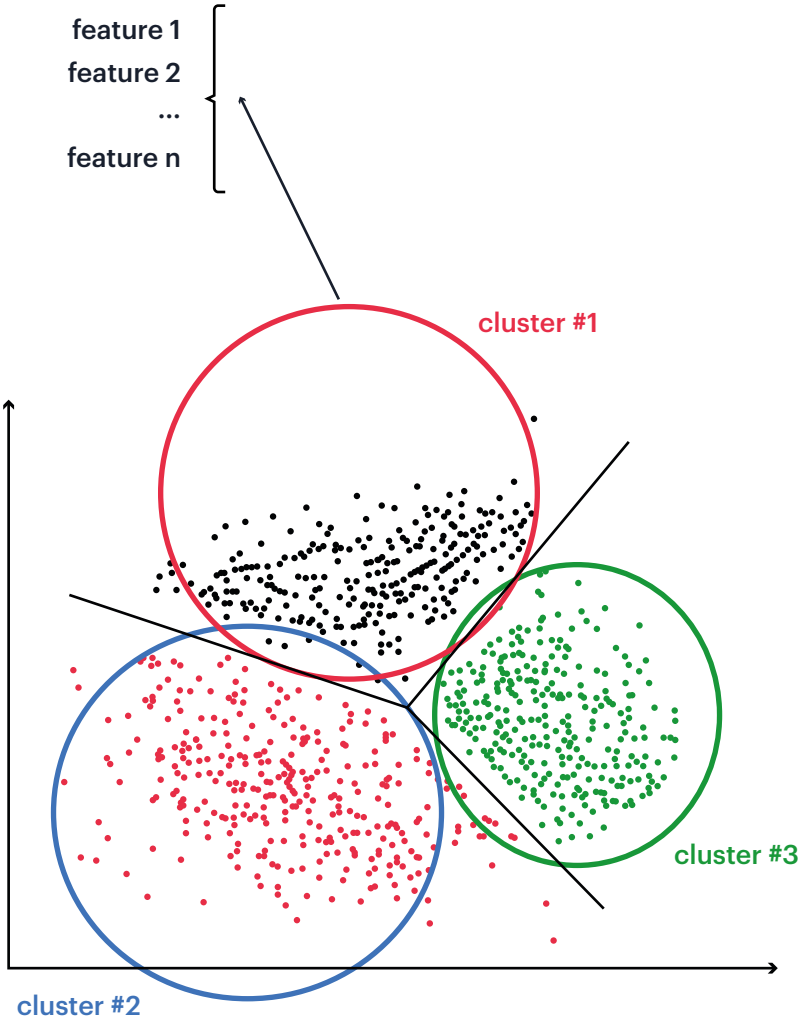
(Agregación)

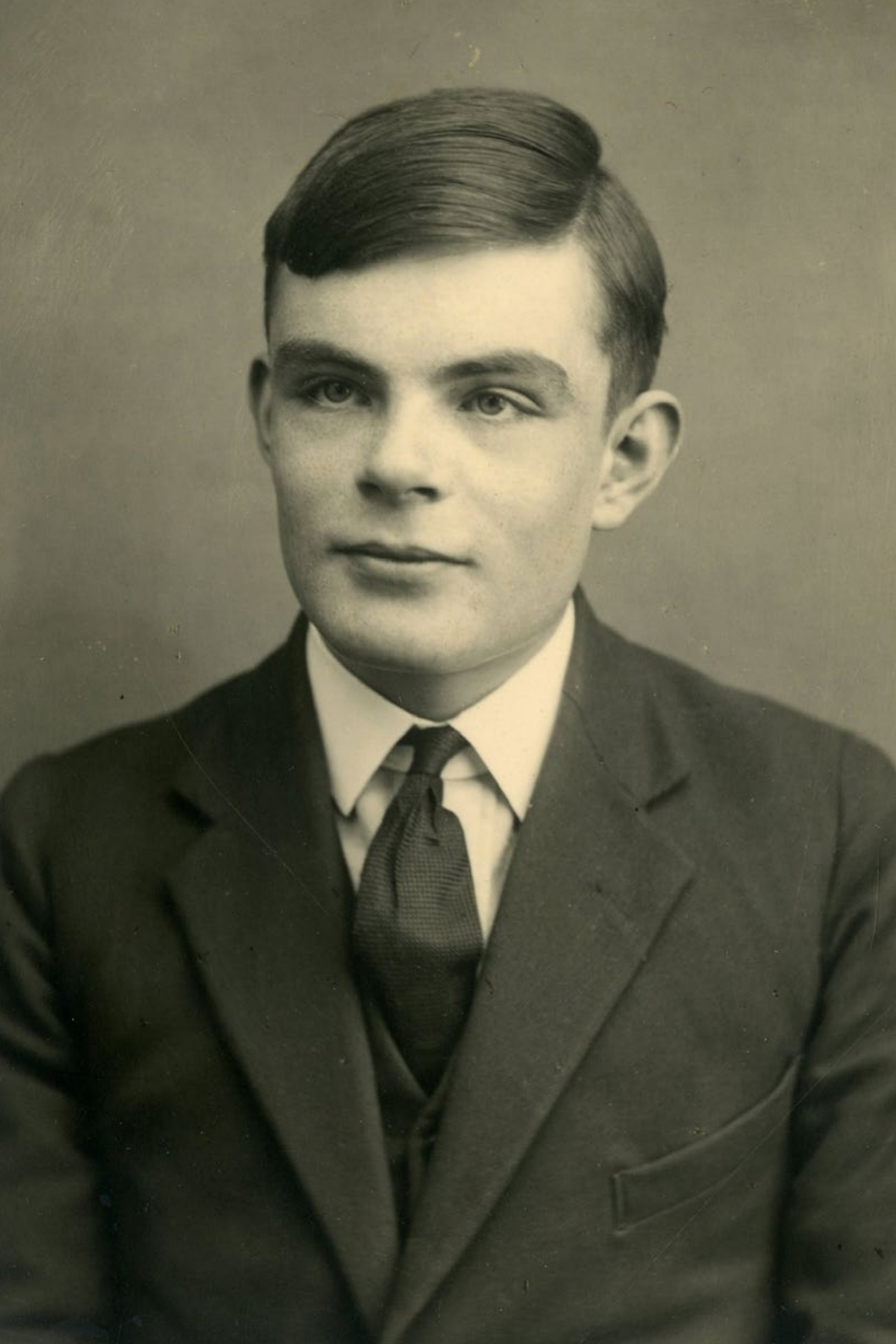
Los métodos de agregación, que son parte de los métodos no supervisados en Machine Learning, se utilizan en una situación de relativo desconocimiento del dominio. Se trata de obtener una descripción inicial que separe grupos de objetos con características parecidas. Esta primera separación debe permitirnos analizar las características comunes de los objetos que pertenecen a cada grupo, qué los hace parecidos y por qué, y qué los diferencia de los otros grupos y por qué.

Por lo tanto, el objetivo de la agregación consiste en determinar cómo podemos separar un conjunto de objetos en varios grupos a partir de las combinaciones presentes de atributos y valores, de manera que los objetos más parecidos estén en el mismo grupo y los objetos diferentes, en grupos distintos.

Es importante darse cuenta de que en este espacio podemos definir una distancia entre objetos en función de los valores de los atributos correspondientes, que nos permiten asimilar en este espacio los conceptos de objetos parecidos y objetos próximos.

— 01.10 Clustering.





“

Creo que a finales de siglo se podrá hablar de máquinas pensando sin esperar que nadie te contradiga.

Alan Turing.
Padre de las ciencias de la computación y precursor de la IA.

— 01.11

Confusion matrix.

(Matriz de confusión)

Una matriz de confusión es una tabla que a menudo se usa para describir el rendimiento de un modelo de clasificación (o "clasificador") en un conjunto de datos de prueba para los que se conocen los valores verdaderos.

Veamos un ejemplo aplicado a un caso de uso, si un paciente tiene o no una enfermedad, aplicaremos un algoritmo con función logística, donde el resultado será de tipo binario (0 = NO tiene y 1 = SÍ tiene).

Ahora definamos los términos más básicos y los ratios calculados:

- **Positivos Verdaderos (TP):** estos son casos en los que predijimos que sí (tienen la enfermedad) y tienen la enfermedad.
- **Negativos Verdaderos (TN):** predijimos que no, y no tienen la enfermedad.
- **Falsos Positivos (FP):** predijimos que sí, pero en realidad no tienen la enfermedad (también conocido como "error tipo I").
- **Falsos Negativos (FN):** predijimos que no, pero en realidad tienen la enfermedad. (también conocido como "error de tipo II").

— 01.11 Confusion matrix.

		predicho		
		NO	SI	*tot*
real	NO	TB = 50	FP = 10	*60*
	SI	FN = 5	TP = 100	*105*
	tot	*55*	*110*	

Ratio	**Fórmula**	**Resultado**
Accuracy Precision	*(TP + TN) / total*	¿Con qué frecuencia es correcto el clasificador?
Clasificación errónea	*(FP + FN) / total*	¿Con qué frecuencia está mal?
Sensitivity o Recall	*(Tp) / actual YES*	¿Con qué frecuencia predice SÍ cuando es SÍ?
Falsos Positivos	*(FP) / actual NO*	¿Con qué frecuencia predice SÍ cuando es NO?
Negativos Verdaderos	*(TN) / actual NO*	¿Con qué frecuencia predice NO cuando es NO?
Precisión	*(TP) / predicciones SÍ*	¿Con qué frecuencia es correcta la predicción SÍ?

— 01.12

Convolutional Networks.

(Redes convolucionales)

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Al poder trabajar con matrices bidimensionales son muy efectivas para tareas relacionadas con la visión artificial (tanto clasificación como detección).

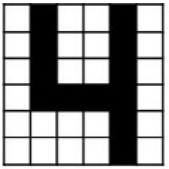
El objetivo de la aplicación de un filtro (o kernel) convolucional es la transformación de los datos de forma que ciertas características de la imagen (en función del filtro aplicado) se vuelvan más predominantes, obteniendo como resultado un conjunto de datos menos sensibles a pequeñas modificaciones de los datos de entrada.

La generalización desde el punto de vista matemático para la convolución:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} * \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ x_{21} & y_{22} & \cdots & y_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x\,(m-i)\,(n-j)\,y\,(1+i)\,(1+j)$$

— 01.12 Convolutional Networks.

Imagen original | Matriz | Filtro | Resultado



0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0
0	0	0	0	1	0
0	0	0	0	1	0

1	0	1
0	1	0
1	0	1

=

1	2	2	1
2	3	3	2
1	2	3	1
1	2	3	2

x | y | r

Paso 1

0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0
0	0	0	0	1	0
0	0	0	0	1	0

1	0	1
0	1	0
1	0	1

=

1			

$$r_{(1,1)} = x_{(1,1)} * y_{(1,1)} + x_{(1,2)} * y_{(1,2)} + x_{(1,3)} * y_{(1,3)} + x_{(2,1)} * y_{(2,1)} + x_{(2,2)} * y_{(2,2)} + x_{(2,3)} * y_{(2,3)} + x_{(3,1)} * y(3,1) + x_{(3,2)} * y_{(3,2)} + x_{(3,3)} * y_{(3,3)}$$

$$r_{(1,1)} = 0*1 + 1*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*1 + 1*0 + 0*1 = 1$$

x | y | r

Paso 2

0	1	0	0	1	0
0	1	0	0	1	0
0	1	0	0	1	0
0	1	1	1	1	0
0	0	0	0	1	0
0	0	0	0	1	0

1	0	1
0	1	0
1	0	1

=

1	2		

$$r_{(2,1)} = x_{(1,2)} * y_{(1,1)} + x_{(1,3)} * y_{(1,2)} + x_{(1,4)} * y_{(1,3)} + x_{(2,2)} * y_{(2,1)} + x_{(2,3)} * y_{(2,2)} + x_{(2,4)} * y_{(2,3)} + x_{(3,2)} * y_{(3,1)} + x_{(3,3)} * y_{(3,2)} + x_{(3,4)} * y_{(3,3)}$$

$$r_{(1,1)} = 1*1 + 0*0 + 0*1 + 1*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*1 = 2$$

— 01.13

Decision tree.

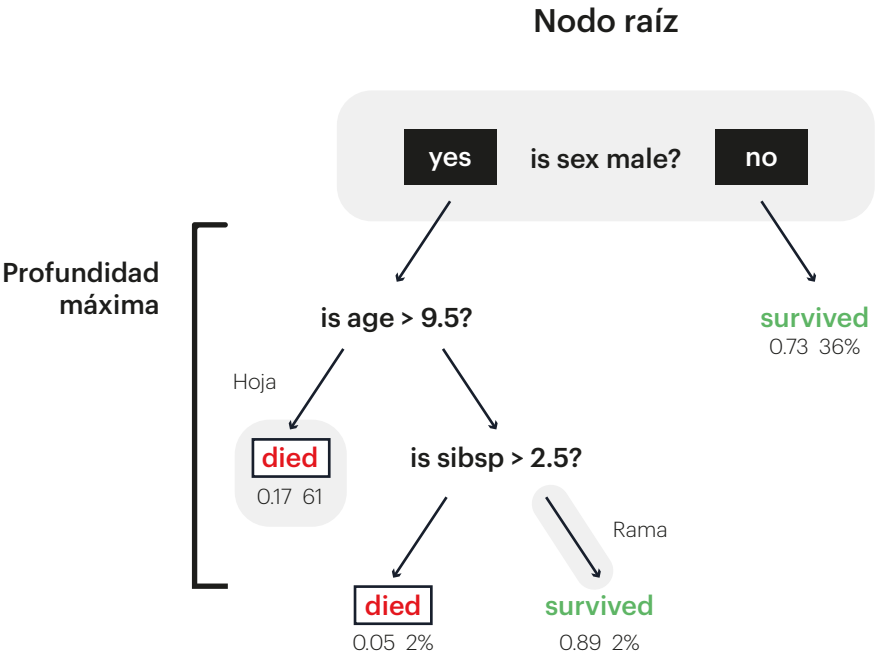
(Árbol de decisión)

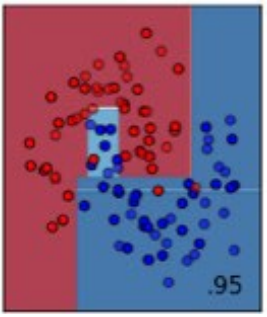
Un árbol de decisión se trata de una estructura jerárquica construida de arriba abajo compuesta por nodos, ramas y hojas. El nodo superior representa la raíz del árbol y, cuando un nodo interno no se divide en más ramas, se convierte en una hoja. Además, el número de niveles que hay entre la raíz y la hoja más alejada, se conoce como profundidad máxima.

Los árboles de decisión se utilizan de distintas formas en una amplia variedad de campos. En el caso del aprendizaje automático, se trata de un algoritmo empleado en problemas tanto de clasificación como de regresión. En ambas situaciones, los datos se utilizan para establecer las fronteras de decisión durante el proceso de entrenamiento. Una vez creado el modelo, las nuevas observaciones atraviesan el árbol desde la raíz hasta que llegan a una hoja, cuyo valor será la predicción del algoritmo.

En la imagen se puede ver la representación en 2D de las fronteras de decisión de un árbol utilizado para un problema de clasificación binaria.

— 01.13 Decision tree.





Salida de un modelo de clasificación binaria.

— 01.14

Deep Learning.

(Aprendizaje profundo)

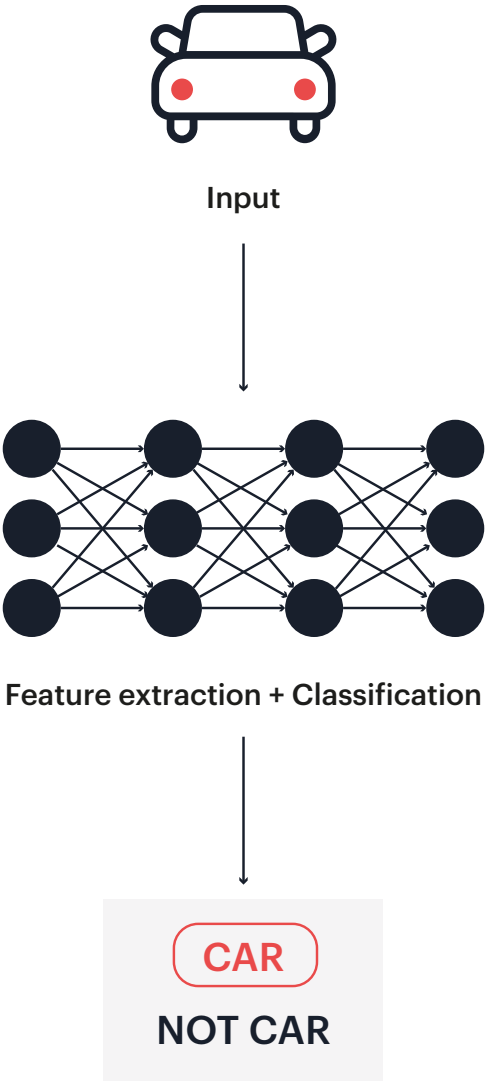
El Deep Learning o Aprendizaje Profundo es un conjunto de algoritmos que buscan reproducir los mismos resultados que el cerebro humano.

Los algoritmos siguen una lógica de procesos por capas que simulan el funcionamiento básico del cerebro a través de las neuronas. En el Deep Learning a esas neuronas se les conoce como “capas”.

Al igual que aprende nuestro cerebro cuando nos enfrentamos a aprender algo nuevo, cómo hablar, montar en bici, etc; los algoritmos buscan esta imitación aprendiendo a reconocer patrones de repetición, palabras concretas, comportamientos frecuentes, de forma que son capaces de dar respuesta de forma automática a datos de entrada, al igual que nuestro cerebro da respuesta a cualquier input.

Por ejemplo, entrenando un modelo con una base de ejemplos de imágenes de marcas de coche. Cuando el modelo recibe el input de una imagen de un coche de manera automática, es capaz de dar respuestas ante si es o no un coche, llegando incluso a devolver la marca y modelo, en caso de haber entrenado el modelo con un conjunto clasificado de marcas y modelos de coches.

— 01.14 Deep Learning.



— 01.15

Dimensionality Reduction.

(Reducción de la dimensionalidad)

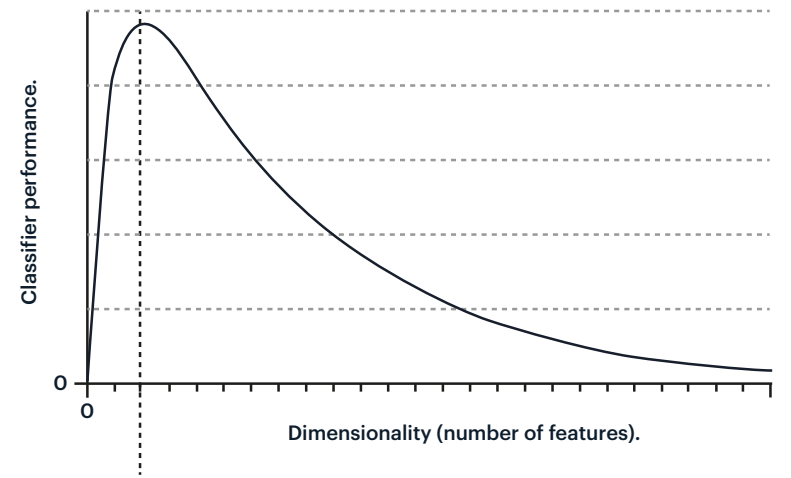
La reducción de la dimensionalidad es el proceso de reducir el número de variables aleatorias consideradas mediante la obtención de un conjunto de variables principales.

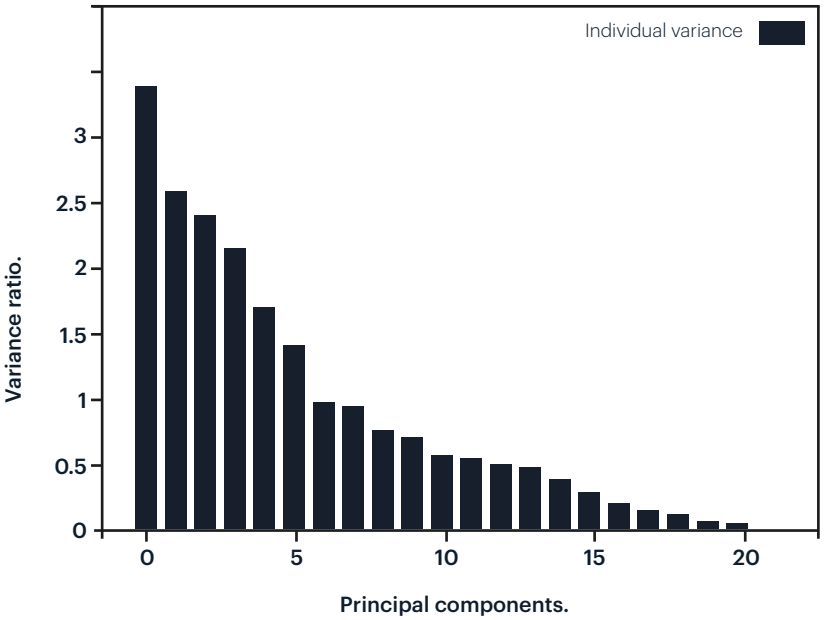
Se refiere al proceso de convertir un conjunto de datos que tienen grandes dimensiones en datos con dimensiones menores, asegurando que transmite información similar de manera concisa. Algunos de los beneficios de la reducción de dimensionalidad son:

- Ayuda en la compresión de datos y reduce el espacio de almacenamiento requerido.
- Ajusta el tiempo requerido para realizar los mismos cálculos.
- Se encarga de la multicolinealidad que mejora el rendimiento del modelo, eliminando funciones redundantes.
- Reducir las dimensiones de los datos a 2D o 3D, puede permitirnos trazar y visualizar con precisión.
- También es útil para eliminar el ruido y, como resultado, podemos mejorar el rendimiento de los modelos.

Entre las técnicas más importantes tenemos: PCA (Análisis del componente principal), LDA (Análisis discriminante lineal), SVD (Descomposición en valores singulares).

— 01.15 Dimensionality Reduction.





— 01.16

Epoch.

(Época)

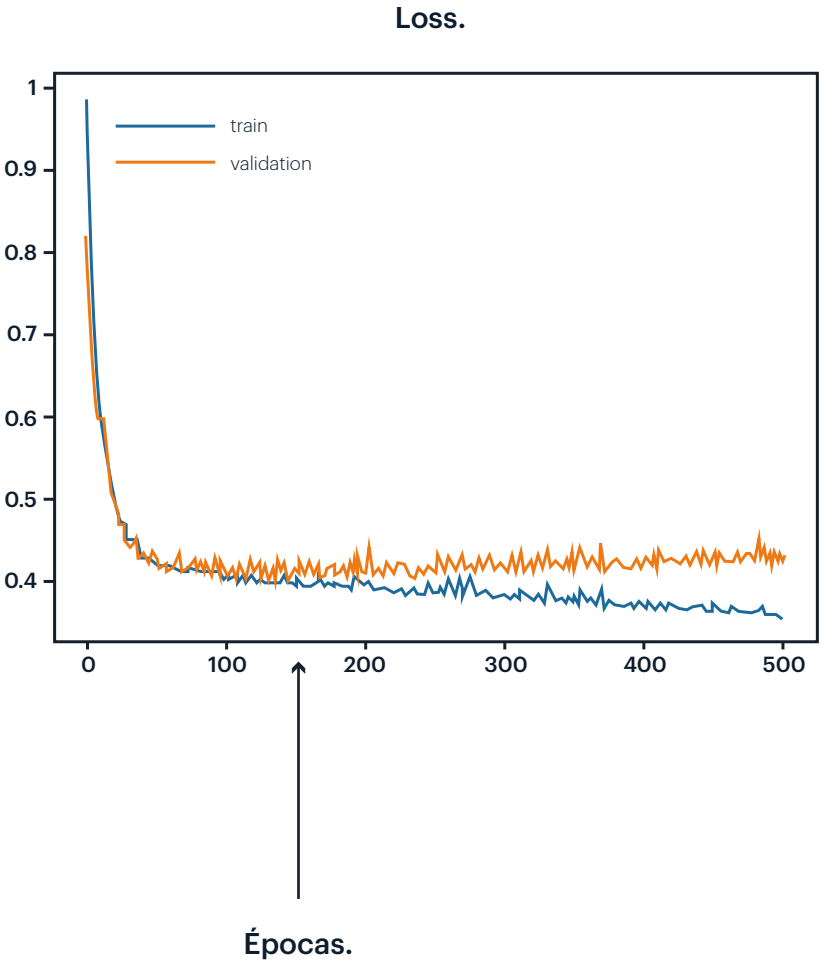
El número de épocas es un hiperparámetro que define el número de veces que el algoritmo de aprendizaje funcionará en todo el conjunto de datos de entrenamiento.

Una época significa que cada muestra en el conjunto de datos de entrenamiento ha tenido la oportunidad de actualizar los parámetros internos del modelo. Se compone de uno o más lotes (batch).

El número de épocas es tradicionalmente grande, a menudo cientos o miles, lo que permite que el algoritmo de aprendizaje se ejecute hasta que el error del modelo se haya minimizado lo suficiente. Puede ver ejemplos del número de épocas establecidos en 10, 100, 500, 1000 y más.

Es común crear gráficos de líneas que muestran épocas a lo largo del eje X como tiempo y el error o habilidad del modelo en el eje Y. Estas se llaman curvas de aprendizaje. Nos ayudan a diagnosticar si el modelo ha superado el aprendizaje o no, y si se ajusta adecuadamente al conjunto de datos de entrenamiento.

— 01.16 Epoch.


Loss.
train
validation
1
0.9
0.8
0.7
0.6
0.5
0.4
0
100
200
300
400
500
Épocas.

— 01.17

Explainability.

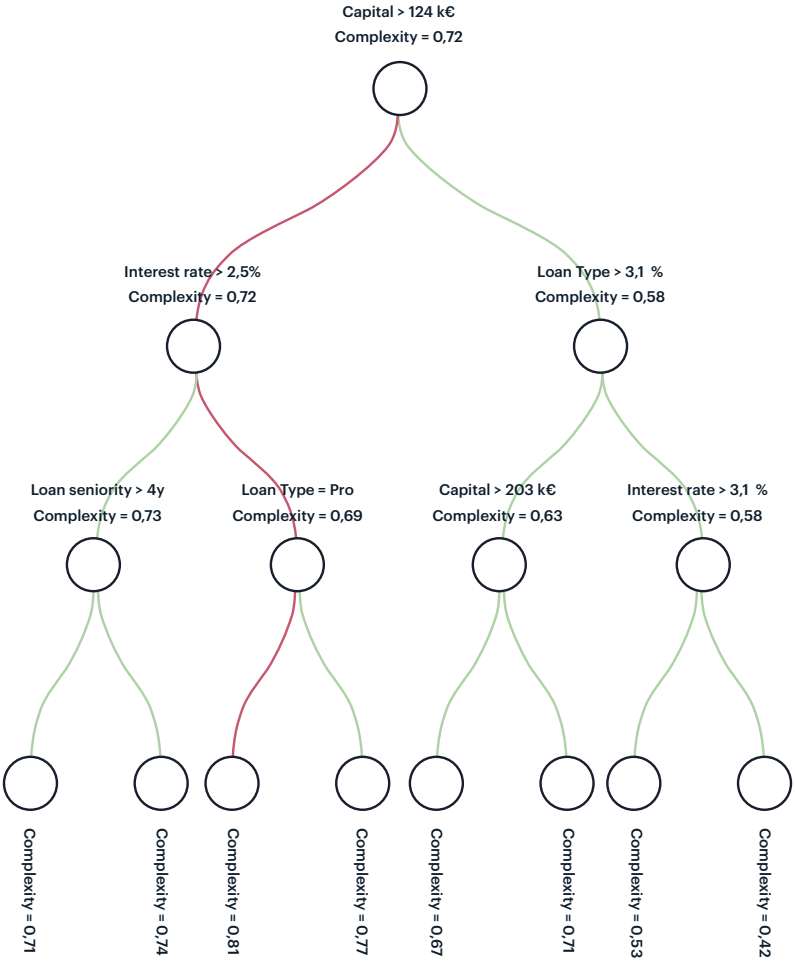
(Explicabilidad)

La explicabilidad es una disciplina del aprendizaje automático que surge con el objetivo de descifrar los mecanismos internos de la toma de decisiones de los modelos y que facilita la comprensión.

A menudo, al funcionamiento interno de los modelos se le tilda de "caja negra" o "caja mágica", ya que a pesar de proporcionar unos buenos resultados, no se conocen los caminos seguidos hasta llegar a ellos. Por lo tanto, la explicabilidad se encarga, mediante un conjunto de técnicas de modelaje, de mantener dichos resultados mientras se proporciona información sobre cómo funcionan dichas "cajas". Se trata de una rama que cobra vital importancia con el incremento de procesos autónomos y el auge del Deep Learning. En la imagen podemos ver cómo se explota la explicabilidad de un árbol de decisión.

En ocasiones, se intercambia el término explicabilidad con interpretabilidad. Sin embargo, este último se entiende como la capacidad de discernir la relación causal entre una entrada y una salida. Es decir, a partir de observaciones empíricas, interpretar cuál es el comportamiento del modelo sin necesariamente saber cómo funcionan procesos internos.

— 01.17 Explainability.



prediction 0,81 = **0,65** (trainset mean complexity) **+ 0,07** (gain from Capital) **- 0,03** (loss from Interest rate) **+ 0,12** (gain from loan type)

— 01.18

Exploratory Data Analysis.

(Análisis Exploratorio de Datos)

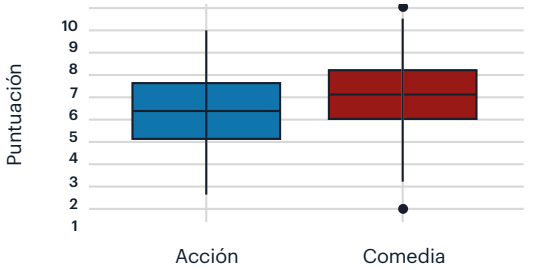
Es una parte fundamental del proceso de creación de un modelo de Machine Learning, en la que el científico de datos tiene que estudiar las características estadísticas tanto de las variables explicativas como de la variable objetivo, así como de las relaciones entre ellas.

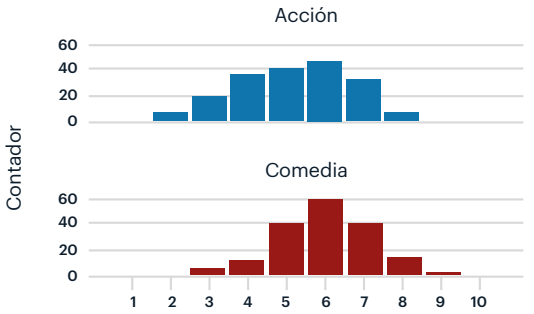
Un EDA bien hecho nos permite comprender en un primer acercamiento cómo se comportan nuestras variables, qué distribución tienen, cuál es la correlación entre ellas, si hay colinealidades, etc. A veces permite, incluso, comprender la naturaleza de la relación entre nuestras características y las de la variable objetivo, dando a conocer si es lineal, cuadrática, logarítmica...

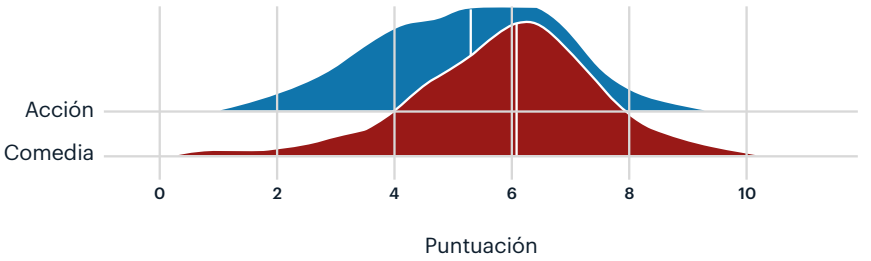
Normalmente se construye en base a tablas y diagramas: percentiles, medias, varianzas, diagramas de caja, distribuciones, mapas de calor, etc.

¿Las comedias obtienen mayores calificaciones que las películas de acción?

Muestra de 400 películas extraídas de IMDB.







— 01.19

Feature.

(Atributo o Característica)

Cuando atajamos un problema de aprendizaje automático, al final estamos intentando modelar una función o una respuesta de una variable que denominamos "objetivo" en base a otra serie de variables que denominamos "atributos". Si logramos esto, también podremos predecir cómo se va a comportar nuestra variable objetivo en función de un conjunto de atributos.

En un caso sencillo, podemos intentar predecir el precio de un piso en función de sus propiedades, como lo son la superficie, orientación, altura, aislamiento, código postal, etc. En este ejemplo, el precio es nuestra variable objetivo y sus propiedades son los atributos que nos permiten asignar dicho precio en función de los mismos.

— 01.19 Feature.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.60	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10
10	0.22489	12.5	7.87	0.0	0.524	6.377	94.3	6.3467	5.0	311.0	15.2	392.52	20.45
11	0.11747	12.5	7.87	0.0	0.524	6.009	82.9	6.2267	5.0	311.0	15.2	396.90	13.27
12	0.09378	12.5	7.87	0.0	0.524	5.889	39.0	5.4509	5.0	311.0	15.2	390.50	15.71
13	0.62976	0.0	8.14	0.0	0.538	5.949	61.8	4.7075	4.0	307.0	21.0	396.90	8.26
14	0.63796	0.0	8.14	0.0	0.538	6.096	84.5	4.4619	4.0	307.0	21.0	380.02	10.26
15	0.62739	0.0	8.14	0.0	0.538	5.834	56.5	4.4986	4.0	307.0	21.0	395.62	8.47

— 01.20

Feature Engineering.

(Ingeniería de atributos)

Estamos haciendo ingeniería de atributos cuando efectuamos transformaciones sobre nuestras variables explicativas, generando otras nuevas de forma que ayudan al modelo a capturar de forma más precisa la relación entre nuestros atributos y la variable objetivo que estamos intentando modelar.

Estas transformaciones pueden ser de diversa índole (por ejemplo matemática, pero hay otras) y muchas veces se dice que es más arte que ciencia, por lo que el conocimiento del ámbito de aplicación nos es de gran ayuda para llegar a una precisión satisfactoria.

Un ejemplo sencillo sería si intentamos modelar la afluencia a un museo en nuestra ciudad y disponemos de los datos históricos de las visitas por fecha. Entonces podemos generar un atributo sintético que se llame "fin_de_semana" con posibles valores 0 o 1, de forma que nuestro modelo es capaz de capturar la contribución de este hecho en el número de visitas.

— 01.20 Feature Engineering.

FECHA	AFLUENCIA	FIN DE SEMANA
01/01/2020	215	0
02/01/2020	231	0
03/01/2020	292	1
04/01/2020	295	1
05/01/2020	246	0
06/01/2020	288	0
07/01/2020	253	0
08/01/2020	235	0
09/01/2020	261	0
10/01/2020	273	1
11/01/2020	278	1
12/01/2020	265	0

“

Si se requieren 200 años para lograr la inteligencia artificial, y finalmente hay un libro de texto que explica cómo se hace, la parte más difícil de ese libro de texto será en la que se explica por qué la gente no lo pensó hace 200 años.

John McCarthy.
Padre de la IA, fue de los primeros en acuñar el término y creador del lenguaje LISP.

— 01.21

Generative adversarial networks (GAN).

(Redes generativas adversarias)

Las redes generativas adversarias se asocian habitualmente a procesos de creación de contenido. Su objetivo es generar un resultado nuevo en base a unos parámetros de entrada y a partir del conocimiento adquirido. Se aplica a cualquier tipo de contenido, ya sea imágenes, audio, texto...

Consiste principalmente en enfrentar dos redes de neuronas. La primera generará nuevos candidatos, mientras que la segunda discrimina si el contenido obtenido es o no válido en función de su conocimiento.

El objetivo de la red generadora será, por lo tanto, engañar a la red discriminadora incrementando el ratio de error en la evaluación (al ser capaz de hacer que la red discriminadora acepte su contenido).

En cada paso, ambas redes aprenden mutuamente, permitiendo mejorar de forma continua su aprendizaje en lo que se denomina juego de suma cero, ya que el error/acierto de una de las redes se compensará con el resultado de la otra red.

— 01.21 Generative adversarial networks (GAN).

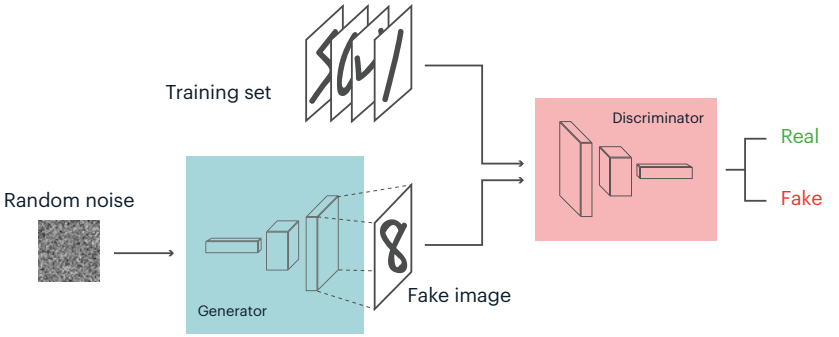


Ilustración de las habilidades de GAN por Ian Goodfellow y coautores.

— 01.22

GPT-2.

GPT-2 es un modelo generativo de lenguaje creado por OpenAI. Esto significa que genera texto al predecir palabra por palabra, qué palabra sigue en una secuencia en función de lo que el modelo ha aprendido previamente.

GPT-2 fue entrenado con 40 GB de texto extraido de Internet. En su versión más completa define más de 1.500 millones de parámetros. Esto lo hace realmente disruptivo comparado con aproximaciones anteriores.

Tras su lanzamiento, GPT-2 generó mucha publicidad y atención en gran parte debido a los ejemplos que lo acompañaron, el más famoso un boletín de noticias que documenta el descubrimiento de unicornios de habla inglesa en los Andes. También ha sido utilizado en otras aplicaciones como en el juego AI Dungeon.

GPT-2 ha despertado la inquietud en sus creadores y en la comunidad en general sobre la aplicación de estas técnicas generativas avanzadas para generación de contenido fake y su uso indebido, por ejemplo, en la creación de discursos políticos falsos.

— 01.22 GPT-2.


OpenAI
SYSTEM PROMPT (HUMAN-WRITTEN)
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)
The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.
Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.
While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."
Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.
While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."

— 01.23

Gradient Descent.

(Descenso del gradiente)

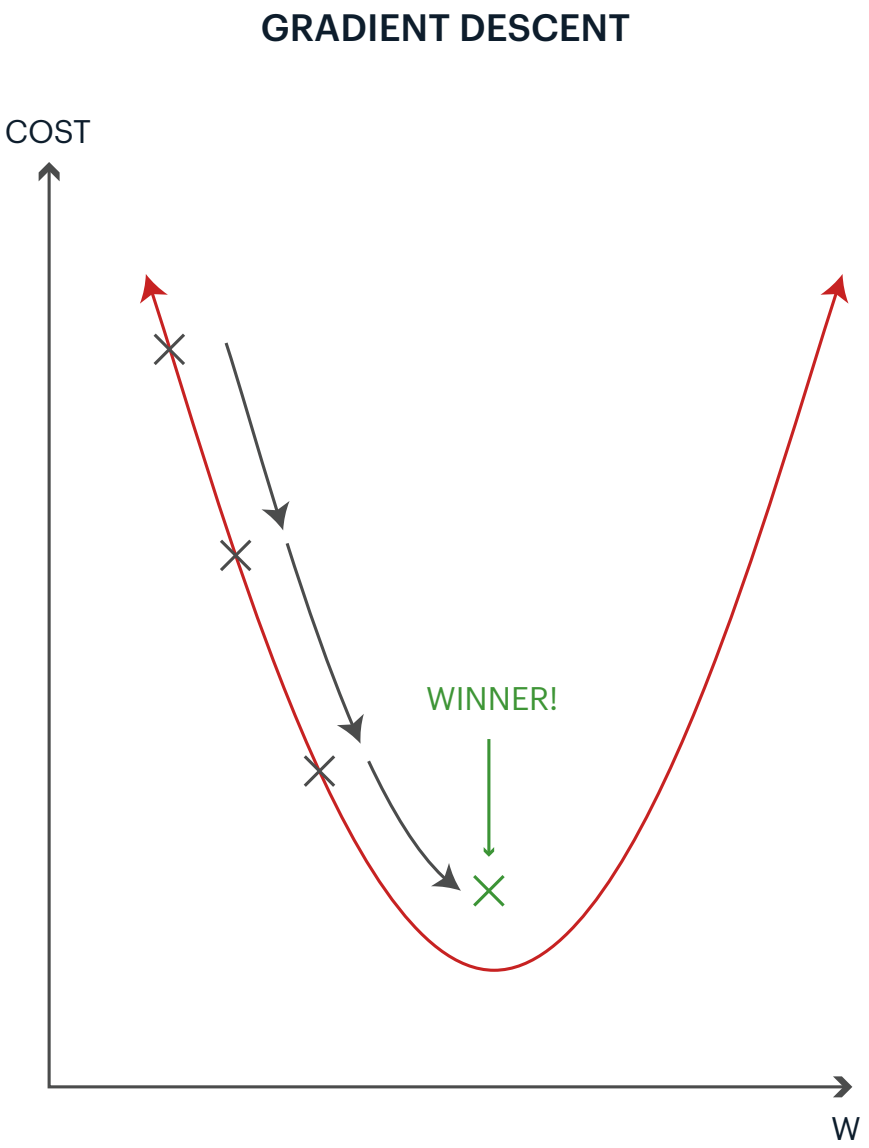
El descenso de gradiente es un algoritmo de optimización utilizado para minimizar algunas funciones moviéndose iterativamente en la dirección del descenso más pronunciado según lo definido por el negativo del gradiente.

En el aprendizaje automático, utilizamos el descenso de gradiente para actualizar los parámetros de nuestro modelo. Los parámetros se refieren a coeficientes en regresión lineal y pesos en redes neuronales.

Se usa mejor cuando los parámetros no pueden calcularse analíticamente (por ejemplo, usando álgebra lineal) y deben buscarse mediante un algoritmo de optimización.

Hay tres variantes de descenso de gradiente, que difieren en la cantidad de datos que usamos para calcular el gradiente de la función objetivo. Dependiendo de la cantidad de datos, hacemos una compensación entre la precisión de la actualización de parámetros y el tiempo que lleva realizar una actualización.

— 01.23 Gradient Descent.



— 01.24

Hyperparameters.

(Hiperparámetros)

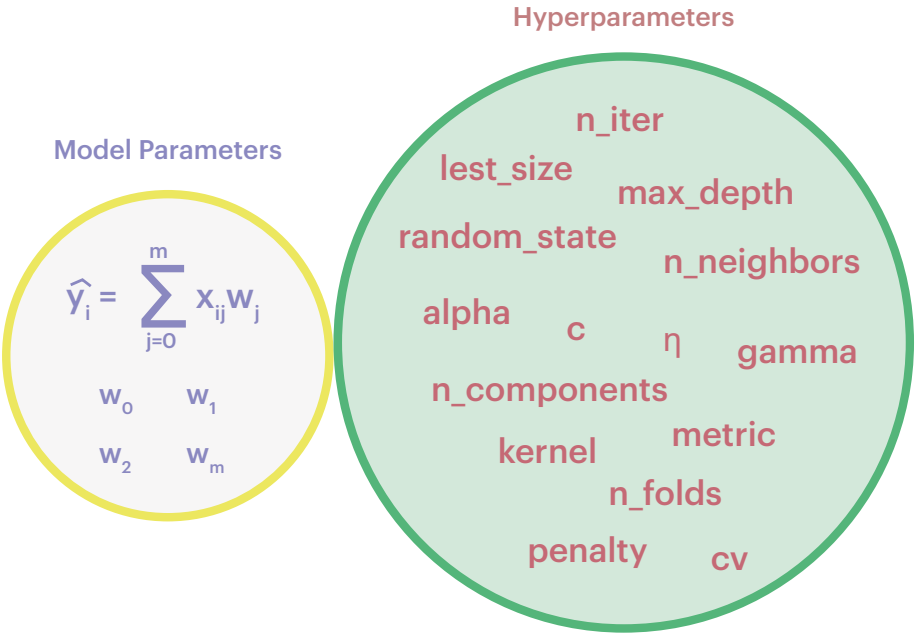
Los hiperparametros de un modelo de aprendizaje automático conforman una configuración cuyo valor no puede ser estimado por los datos y tiene que ser establecido antes del proceso de entrenamiento.

Hay hiperparámetros comunes como el número de iteraciones o la partición porcentual de los datos en entrenamiento, validación y test, y hay otros que son distintos para cada modelo. y en una red neuronal, el número de capas y neuronas son ejemplos significativos de hiperparámetros.

Es importante comprender la diferencia entre los hiperparámetros y los propios parámetros de un modelo. Los primeros son utilizados para configurar el proceso de entrenamiento que va a optimizar el valor de los segundos (“aprendidos” a partir de los datos) para que el desempeño del modelo sea el mejor posible. Este proceso se le conoce como fine-tunning o reajuste y consiste en probar distintas configuraciones de hiperparámetros para que el valor de los parámetros sea el que genere mejor desempeño del modelo. Un ejemplo de parámetros son los pesos w y el sesgo b de una función de aproximación en una red neuronal y el número de capas y neuronas son ejemplos significativos de hiperparámetros..

La configuración de hiperparámetros suele ser establecida por el usuario aunque en ocasiones se utilizan heurísticas para automatizar el proceso.

— 01.24 Hyperparameters.



— 01.25

Learning rate.

(Ratio o tasa de aprendizaje)

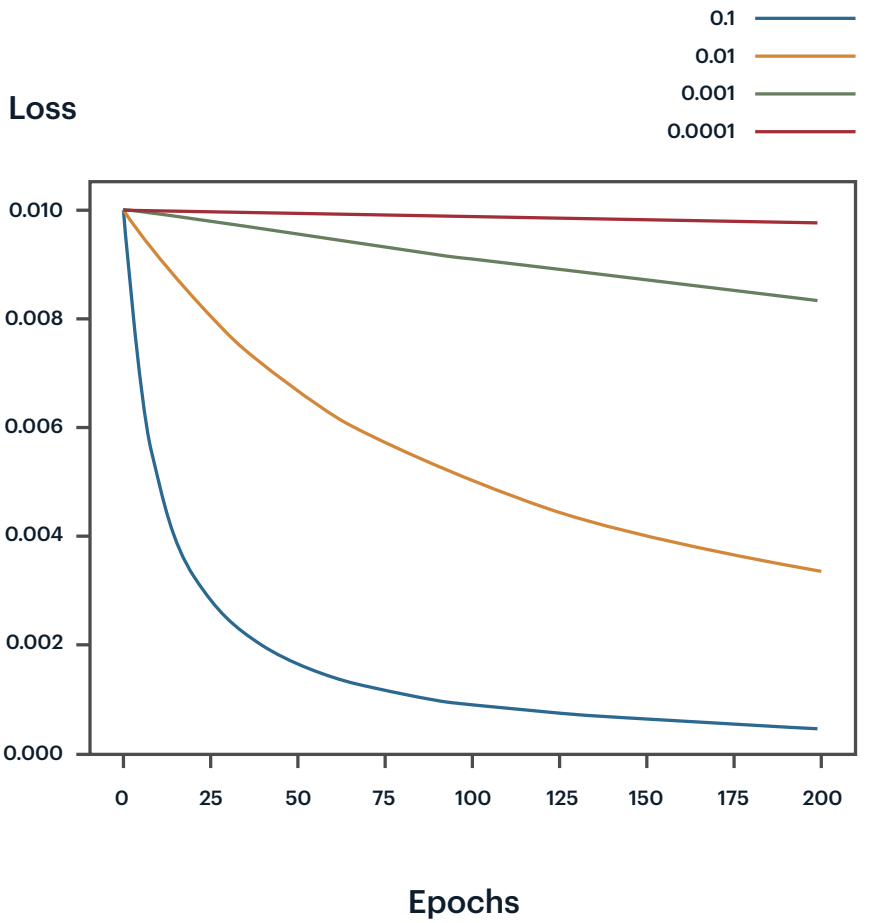
El ratio o tasa de aprendizaje es uno de los hiperparámetros más importantes de las redes neuronales y se encarga de actualizar los modelos durante el proceso de backpropagation en respuesta al gradiente de error resultante del descenso de gradientes.

Se trata de un valor escalar positivo que durante cada iteración del proceso de entrenamiento (epoch), actualiza el valor de los pesos w y sesgos b en la dirección que minimiza la pérdida/loss (maximiza el desempeño del modelo). Normalmente, su valor es relativamente pequeño y oscila entre 0.0 y 1.0.

La tasa de aprendizaje controla cómo de rápido un modelo intenta ajustarse a los datos en cuestión y, por lo tanto, con qué velocidad "aprende". De este modo, para un mismo problema, valores más pequeños supondrán actualizaciones más suaves en el modelo y requerirán un mayor número de épocas de entrenamiento con el riesgo de estancarse. Por el contrario, tasas superiores harán modificaciones más severas y necesitarán menos iteraciones. Sin embargo, valores elevados pueden conducir a que el modelo converja en una solución sub-óptima.

La selección del ratio de aprendizaje es una tarea crítica en el proceso de refinamiento de una red neuronal. Para ello, se suele graficar el error a lo largo de las iteraciones de entrenamiento para cada una de las tasas comparadas.

— 01.25 Learning rate.



— 01.26

Long Short-Term Memory (LSTM).

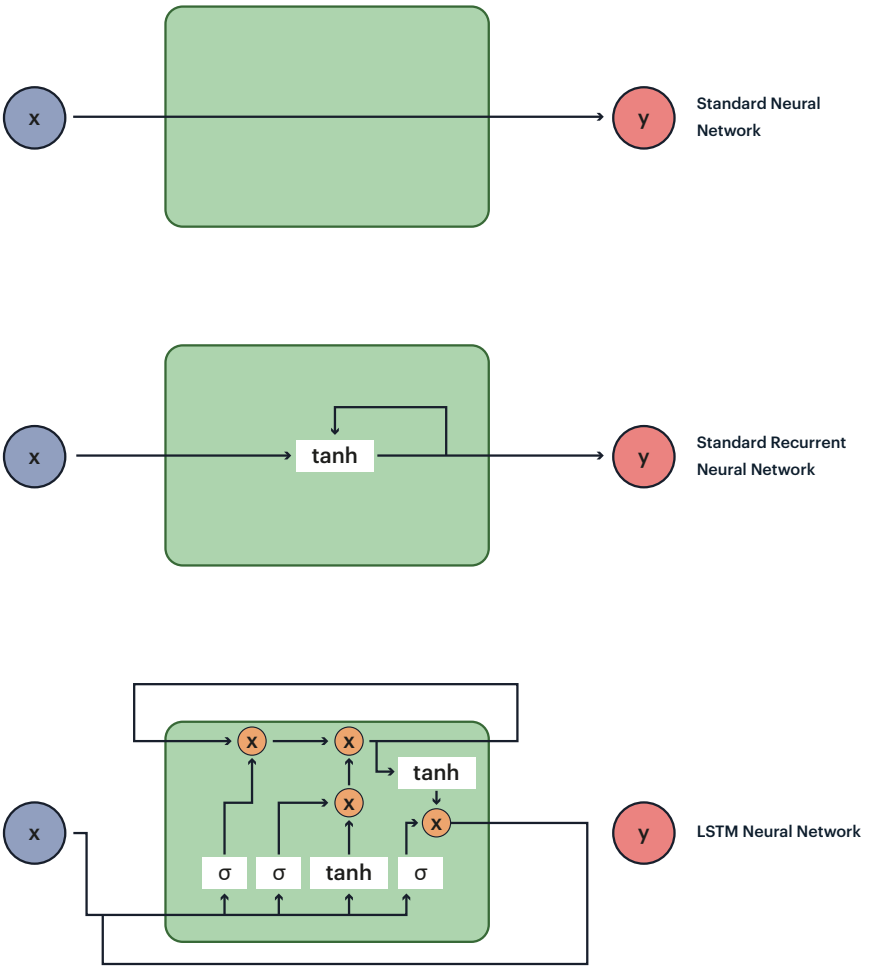
(Memoria a corto largo plazo)

Las redes LSTM son un tipo de red neuronal recurrente (Recurrent Neural Network - RNN).

Las redes RNN se caracterizan por poseer una sistema de 'persistencia' o memoria que les permite 'recordar' la información procesada en ciclos anteriores, al introducir bucles dentro del diagrama de red, generando de esta forma un estado 'oculto'. Dada su estructura esta memoria se diluye rápidamente, siendo principalmente efectiva a corto plazo.
La idea detrás de las redes LSTM para lograr una memoria a un plazo mayor, es incorporar un conjunto de puertas que permiten proteger, eliminar o añadir información al estado de la neurona en función de su relevancia, incrementando el tiempo/ciclos que los datos en esta memoria siguen teniendo efecto sobre el resultado obtenido por la neurona.

Estas redes están muy relacionados a problemas de procesado de lenguaje natural, modelado de lenguaje, traducción...

— 01.26 Long Short-Term Memory (LSTM).



Esquema simplificado de la estructura en los distintos tipos de redes de neuronas.

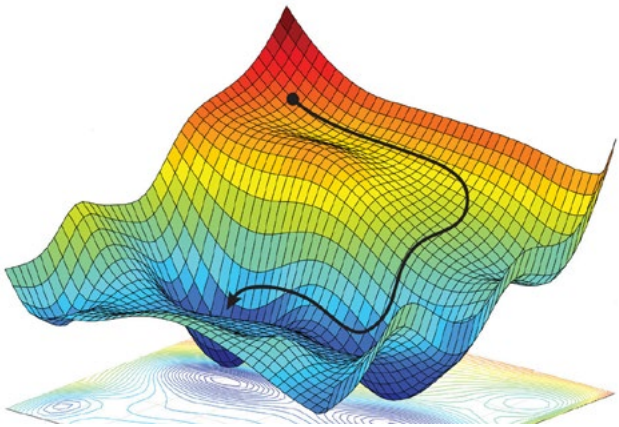
— 01.27

Loss.

(Pérdida)

En aprendizaje automático, la función de pérdida (loss function) se aplica sobre un conjunto de variables para dar lugar a un valor numérico indicativo de la calidad del ajuste del modelo utilizado, con respecto a los valores reales cuyo comportamiento se pretende modelar.

En un problema de aprendizaje automático se busca minimizar la función de pérdida de forma gradual (en diferentes iteraciones) hasta conseguir el menor error posible (mejor ajuste) dentro de unos determinados márgenes que permitan que el modelo sea generalizable (evitando el overfitting).



— 01.27 Loss.

Algunas funciones típicas para calcular la pérdida son las siguientes:

- **Error cuadrático medio (MSE):** media de las diferencias cuadradas entre los valores observados y los predichos. Se utiliza frecuentemente en problemas de regresión.
- **Error medio absoluto (MAE):** media de los errores absolutos entre los valores observados y los predichos. Se utiliza habitualmente en problemas de regresión.
- **Pérdida SVN:** mide la calidad de una clasificación multiclase comparando las sumas de los scores de las muestras categorizadas correctamente en comparación con las incorrectas aplicando un margen de seguridad.
- **Pérdida de entropía cruzada:** mide el rendimiento de modelos de clasificación cuyos resultados ofrecen valores probabilísticos entre 0 y 1. Esta función incrementa su valor en cuanto la probabilidad predicha diverge de la etiqueta original.

— 01.28

MLOps.

MLOps, o Machine Learning Operations, hace referencia a un conjunto de técnicas enfocadas a asegurar la robustez en tiempo de despliegue y operación de modelos de Machine Learning.

A pesar de poder encontrar diferentes aplicaciones concretas para esta definición, hallamos un consenso cuando hablamos de cuatro características que tiene que cumplir un modelo de Machine Learning para ser apto para producción:

- **Reproducible:** tenemos que poder exportar y ejecutar el código en entornos distintos del de desarrollo. Para ello, nos apoyamos en entornos aislados y contenedores.
- **Trazable:** si nuestro modelo hace algo raro, necesitamos poder localizar exactamente cómo se entrenó para poder depurarlo. Esto implica conocer tanto los datos de test y entrenamiento como los parámetros de entrada, con herramientas como MLFlow.
- **Colaborativo:** los desarrollos de Machine Learning son a menudo complejos y requieren de la participación de varios científicos de datos. Debemos no solo facilitar sino alentar esta faceta, mediante el uso de repositorios comunes y control de versiones.
- **Adaptativo:** la naturaleza mutable de la propia realidad hace que los modelos predictivos se desactualicen debido a cambios en las condiciones. Por tanto, hemos de prever procesos de monitorización y reentrenamiendo de nuestros modelos para mantener su vigencia.

— 01.28 MLOps.

— 01.29

MNIST.

MNIST se refiere a una conocida base de datos de dígitos manuscritos que contiene 60.000 imágenes de entrenamiento y 10.000 de test. Cada imagen es un dígito del 0 al 9 escrito por un humano con un tamaño de 28x28 píxeles y cada píxel con un valor de una escala de grises entre 0 y 255.

Este conjunto de datos es público y es usado como un referencia canónica en el mundo del Machine Learning a la hora de comprobar cómo de bueno es un algoritmo en el problema de reconocimiento de dígitos. Multitud de papers científicos usan MNIST como datos de ejemplo para plantear nuevos enfoques y técnicas.

Ahora mismo el algoritmo que mejor ha resuelto este problema es uno compuesto por varias redes convolucionales cooperando que obtiene un error del 0,17%, que es realmente bajo.

MNIST fue creado en 1998 por Yann LeCun, Corinna Cortes y Christopher J.C. Burges. Yann LeCun es uno de los padres de la visión por computador y del Deep Learning. En 2018 ganó el prestigioso premio Turing y actualmente es el científico jefe de IA en Facebook.

— 01.29 MNIST.

Yann LeCun.

— 01.30

Neural Networks.

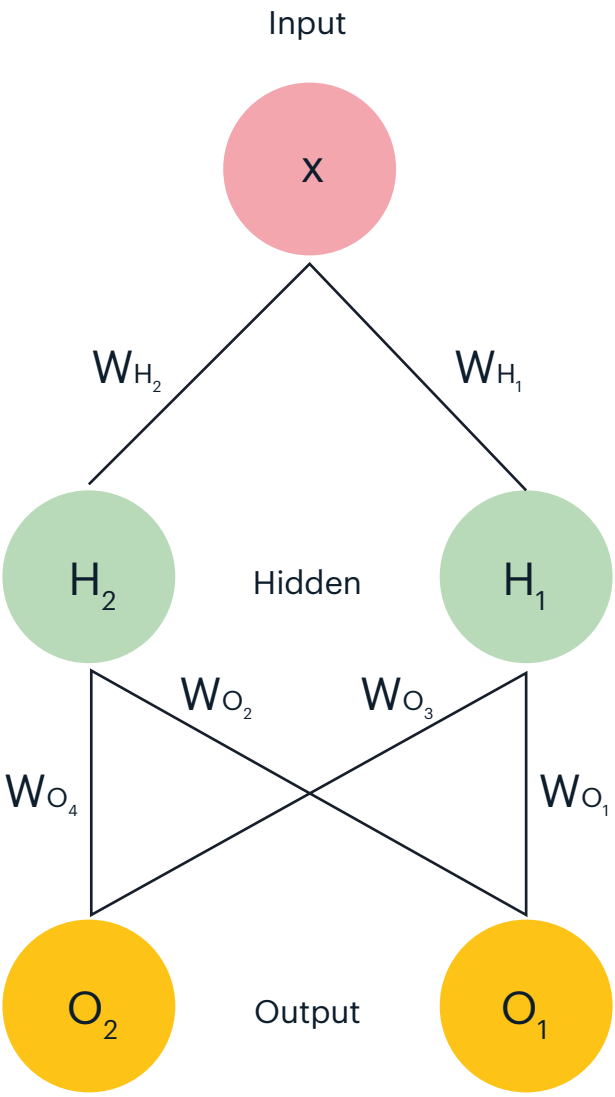
(Redes neuronales)

Las redes neuronales son una clase de algoritmos de aprendizaje automático utilizados para modelar patrones complejos en conjuntos de datos utilizando múltiples capas ocultas y funciones de activación no lineal.

Una red neuronal toma una entrada, la pasa a través de múltiples capas de neuronas ocultas (funciones más pequeñas con coeficientes únicos que deben aprenderse) y genera una predicción que representa la entrada combinada de todas las neuronas.

Las redes neuronales se entrenan de forma iterativa utilizando técnicas de optimización como el descenso de gradiente. Después de cada ciclo de entrenamiento, se calcula una métrica de error basada en la diferencia entre la predicción y el objetivo. Las derivadas de esta métrica de error se calculan y se propagan a través de la red utilizando una técnica llamada retropropagación. Los coeficientes (pesos) de cada neurona se ajustan en función de cuánto contribuyeron al error total. Este proceso se repite iterativamente hasta que el error de red cae por debajo de un umbral aceptable.

— 01.30 Neural Networks.



“

Creo que la gente necesita comprender que el aprendizaje profundo está haciendo que muchas cosas, entre bastidores, sean mucho mejores. El aprendizaje profundo ya está funcionando desde hace años en las búsquedas de Google.

Geoffrey Hinton.
Investigador de IA en Google Brain y profesor de la Universidad de Toronto.

— 01.31

NLP.

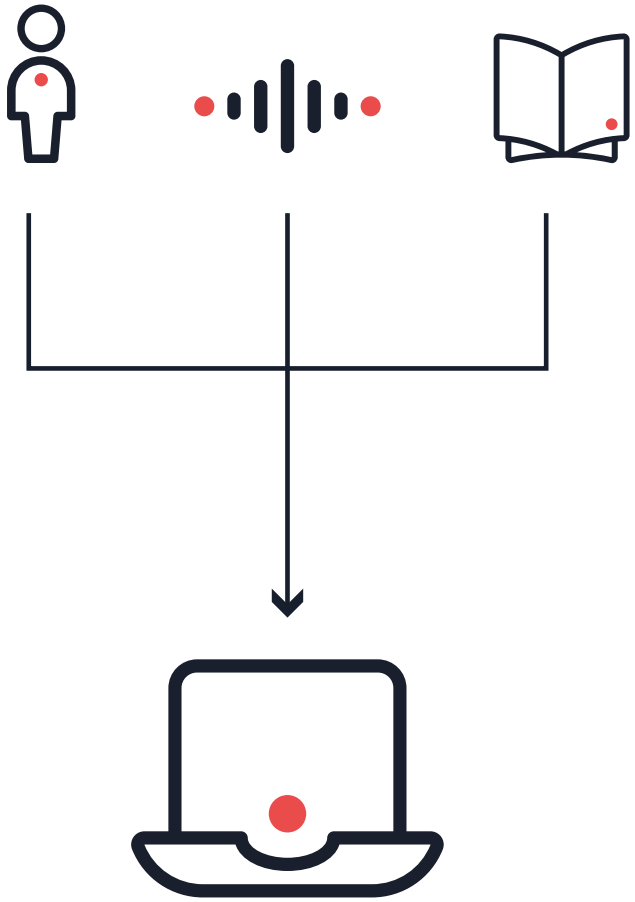
(Procesado de Lenguaje Natural)

Procesado de Lenguaje Natural es un término muy amplio que abarca todas las técnicas relacionadas con el procesamiento de comunicaciones humanas, tanto lenguaje oral como escrito.

Tradicionalmente, el análisis NLP estaba basado en reglas lexicográficas. Con el auge del Machine Learning, se pueden combinar con nuevas herramientas de IA como Deep Learning. Entre ellas, podemos destacar las redes LSTM.

Las aplicaciones prácticas de NLP son muchas y han experimentado un crecimiento espectacular gracias a las nuevas técnicas de Machine Learning. Hay muchas aplicaciones de NLP, entre las que se pueden destacar: traducción de textos, voz a texto, texto a voz, extracción de entidades, clasificación, análisis de sentimiento, y de emociones, e incluso, bots conversacionales y asistentes virtuales como Alexa o Siri.

— 01.31 NLP.



— 01.32

Overfitting.

(Sobreajuste)

Un modelo de aprendizaje automático se ajusta durante el proceso de entrenamiento cuando es capaz de captar las tendencias subyacentes y los patrones presentes en los datos. Entonces, decimos que el modelo "ha aprendido" a generalizar dichos patrones sobre nuevas observaciones.

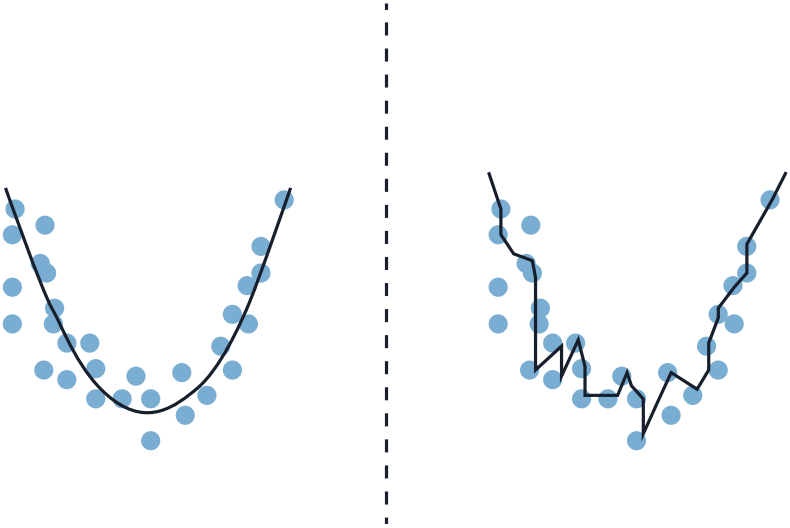
Por lo tanto, dado un set de datos dividido en entrenamiento y validación, un modelo está sobreajustado cuando "aprende" patrones que se dan de forma aleatoria y están presentes solo en el entrenamiento. Es decir, es demasiado sensible a los datos observados durante el proceso de ajuste y fracasará a la hora de generalizar sobre el set de validación. En otras palabras, tendrá un error bajo durante el proceso de entrenamiento, pero el desempeño a la hora de inferir, será pobre.

Los procedimientos más comunes para solucionar el sobreajuste de un modelo son:
•Incluir regulación: ayuda a generalizar.
•Aumentar el volumen de datos: ayuda a reducir la sensibilidad del modelo a los patrones encontrados en las muestra de entrenamiento.

El sobreajuste está muy relacionado con el concepto de varianza. Cuando un modelo está sobreajustado, tendrá mayor varianza y, por lo tanto, un error mayor al generalizar que al entrenar.

— 01.32 Overfitting.

Ejemplo de la función de regresión aprendida por dos modelos después del entrenamiento (a), (b).



(a) Se trata de un modelo adecuadamente ajustado y (b) de un modelo sobreajustado.

— 01.33

Perceptron.

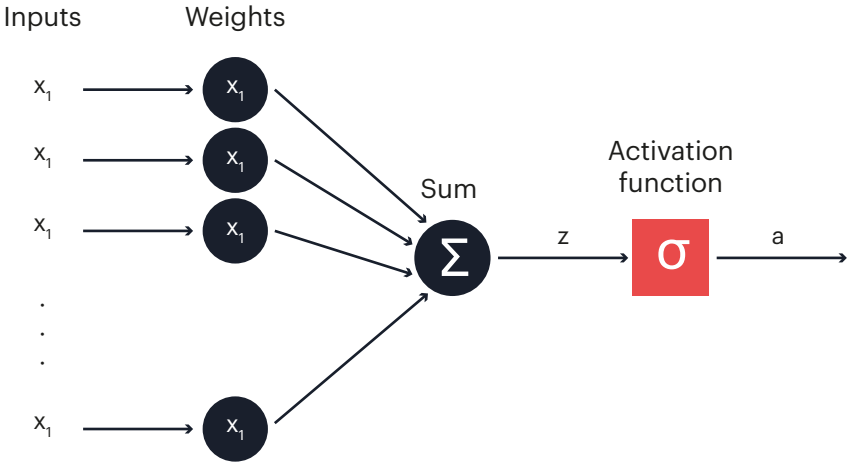
(Perceptrón)

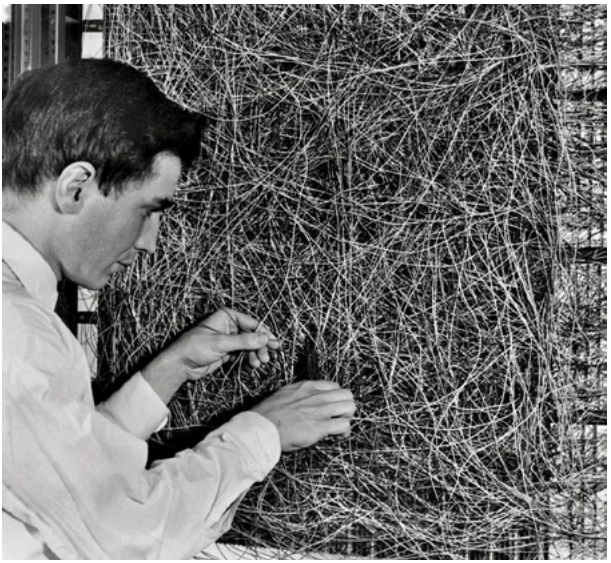
El perceptrón es el antecesor de las redes neuronales, el concepto fue desarrollado en la década de 1950 por Frank Rosenblatt. Inicialmente se vio como un gran avance en el campo de la Inteligencia Artificial, pero luego demostró tener limitaciones serias, suprimiendo el interés en las redes neuronales durante años.

Se trataría de un sistema (ya sea hardware o software) que toma uno o más valores de entrada, ejecuta una función en la suma ponderada de las entradas y calcula un único valor de salida. La función suele ser no lineal, como ReLU, sigmoide o tanh.

En esencia, un perceptrón podría verse como un nodo de las redes neuronales profundas modernas, es decir, una red neuronal profunda que consta de múltiples perceptrones conectados, además de un algoritmo de backpropagation para introducir retroalimentación.

— 01.33 Perceptron.





Frank Rosenblatt

— 01.34

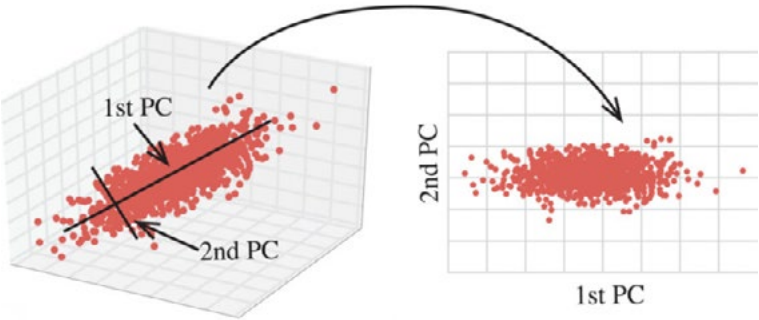
Principal Component Analysis (PCA).

(Análisis del Componente Principal)

El análisis del componente principal es un método de regresión lineal que encuentra un nuevo conjunto de ejes que están mejor alineados con los datos. El primer eje representa un porcentaje elevado de varianza de los datos. El segundo, un porcentaje menor y así sucesivamente.

Al proyectar nuestros datos sobre los nuevos ejes, podemos escoger los primeros k ejes que explican un determinado porcentaje de varianza V. Si $k < n$, entonces conseguimos reducir la dimensionalidad del dataset (el número de atributos), explicando un porcentaje de varianza V. Siempre hay una pequeña pérdida de información excepto si V=100%, entonces significa que no perdemos información.

— 01.34 Principal Component Analysis (PCA).



— 01.35

Random Forest.

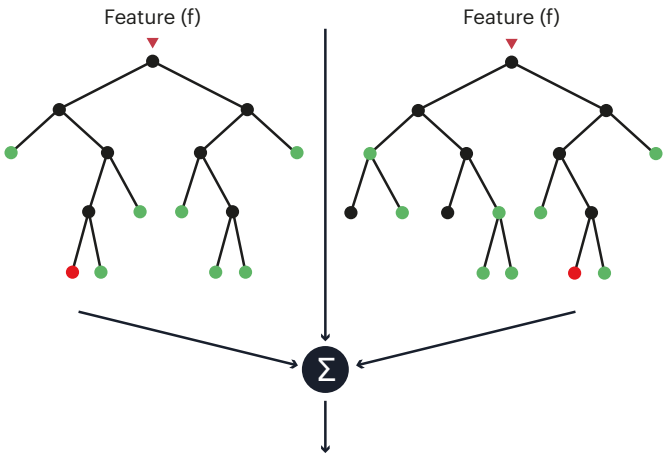
(Bosques Aleatorios)

Los bosques aleatorios son un método de aprendizaje para la clasificación, la regresión y otras tareas que operan construyendo una multitud de árboles de decisión generando la clase (clasificación) o predicción media (regresión) de los árboles individuales.

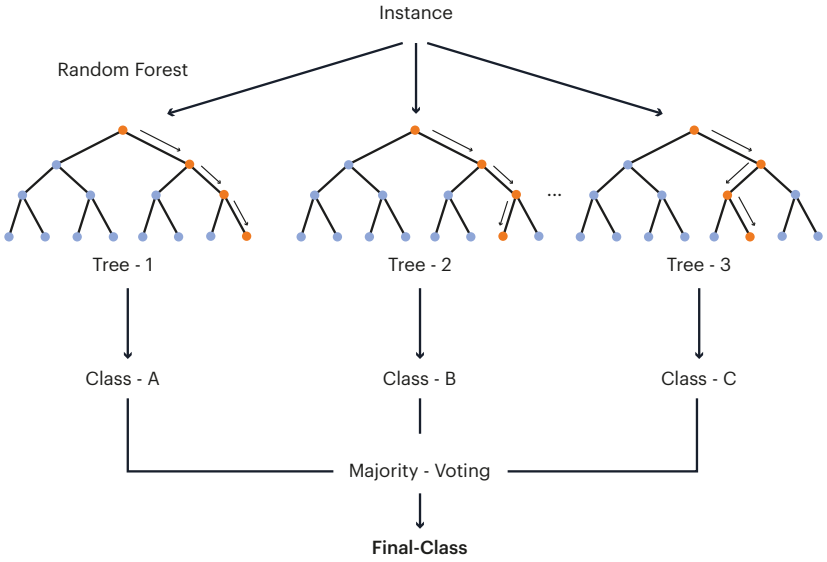
Mientras, los árboles de decisión pueden sufrir una varianza alta, los bosques de decisión aleatorios corrigen el hábito de sobreajustarse a su conjunto de entrenamiento.

Random Forest es una extensión del (bagging) que, además de construir árboles en base a múltiples muestras de sus datos de entrenamiento, también limita las características que se pueden usar para construir los árboles, lo que obliga a los árboles a ser diferentes. Esto, a su vez, puede elevar el rendimiento.

— 01.35 Random Forest.



Random Forest Simplified



— 01.36

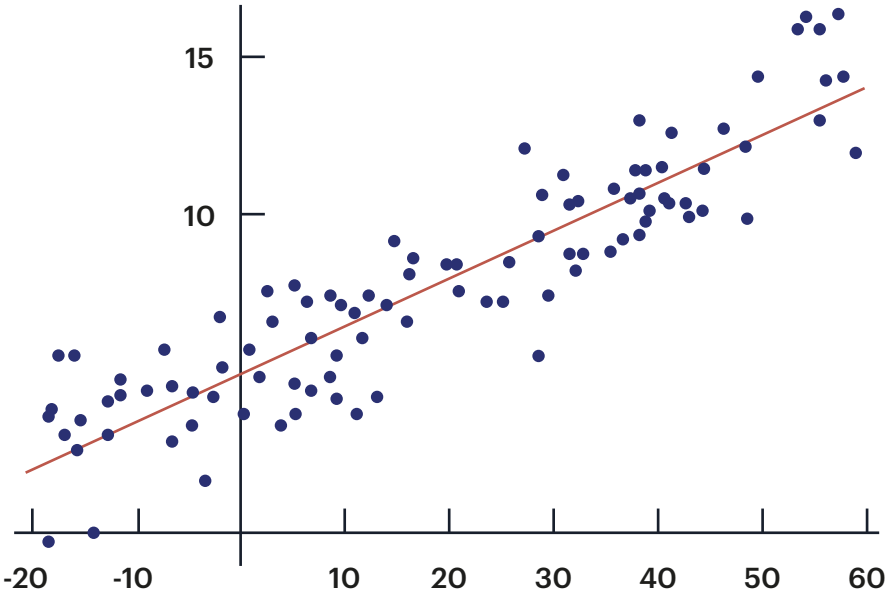
Regression.

(Regresión)

Los problemas de regresión buscan modelar el comportamiento de una variable cuantitativa (variable objetivo) en función de otras variables predictoras (componentes o features) que pueden ser cuantitativas o cualitativas con el objetivo habitual de realizar predicciones o estimaciones.
Existen varios algoritmos para poder resolver este tipo de problemas. Entre ellos, destacan los siguientes:

- **Regresión lineal:** incluye una serie de coeficientes y un término independiente que, aplicados sobre los valores de las variables componentes, permiten generar aproximadamente el valor de la variable objetivo. Requiere el uso de variables cuantitativas y destaca por su facilidad de interpretación.
- **Árboles de decisión:** la muestra se particiona según la profundidad del árbol y se promedian los valores de la variable objetivo en los nodos hoja. Su interpretación es sencilla y funciona tanto con variables componentes cualitativas como cuantitativas.
- **Random Forest y Gradient-Boosted Trees:** estos modelos, disponibles también en problemas de clasificación, pueden utilizarse en regresión para conseguir un mejor ajuste de la variable objetivo aunque dificultando la interpretación de los modelos.
- **Redes neuronales:** permiten modelar la variable objetivo mediante el cálculo de coeficientes por retropropagación. El uso de múltiples capas intermedias aumentará la complejidad del modelo pudiendo mejorar el ajuste de las predicciones.

— 01.36 Regression.



— 01.37

Regression metrics.

(Métricas de Regresión)

En un procedimiento de validación de un modelo de regresión suelen utilizarse las siguientes métricas para evaluar la calidad del ajuste:

- **RMSE (root mean squared error):** es un estimador de la calidad del ajuste de la regresión en el que se mide el promedio de las desviaciones al cuadrado (error cuadrático medio) sobre el que se realiza la raíz cuadrada. Este indicador tiene la característica de proporcionar resultados en las mismas unidades utilizadas por la variable objetivo. Cuanto menor sea el valor de esta métrica, mejor será el ajuste realizado, siendo 0 el valor óptimo que implicaría un perfecto ajuste del modelo de regresión sobre el comportamiento de la variable objetivo.
- **Coeficiente R^2:** este indicador, también conocido como coeficiente de determinación, representa el porcentaje de variación entre la variable predecida y las variables componentes del modelo. Ofrece valores comprendidos entre 0 y 1, representando el valor 1 (100%) un ajuste perfecto. A diferencia del RMSE, los resultados de este indicador son independientes de la escala de unidades utilizada.

— 01.37 Regression metrics.

$$\mathrm{RMSE} = \sqrt{\Sigma \frac{(Y_{\mathrm{pred}} - Y_{\mathrm{ref}})^2}{N}}$$

$$R^2 = 1 \frac{SS_{\mathrm{RES}}}{SS_{\mathrm{TOT}}} = 1 \frac{\Sigma_i (y_i - \hat{y}_i)^2}{\Sigma_i (y_i - \overline{y}_i)^2}$$

— 01.38

Reinforcement Learning.

(Aprendizaje por refuerzo)

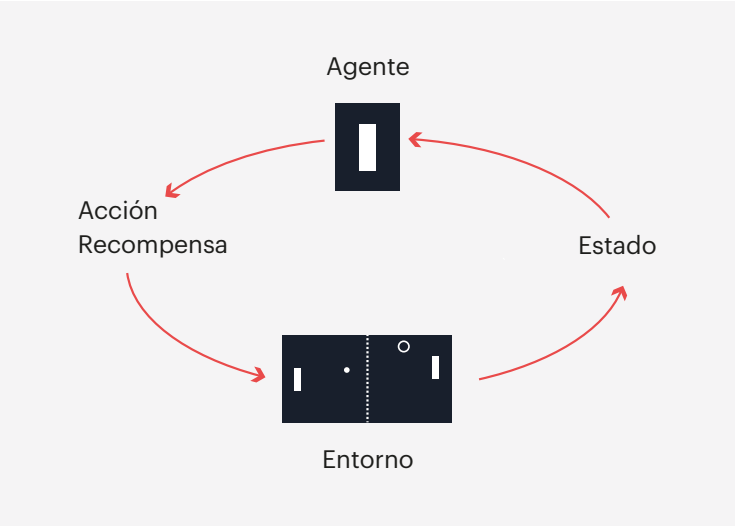
El aprendizaje por refuerzo es uno de los tres tipos en los que se suelen agrupar los tipos de aprendizaje (junto a supervisado y no supervisado).

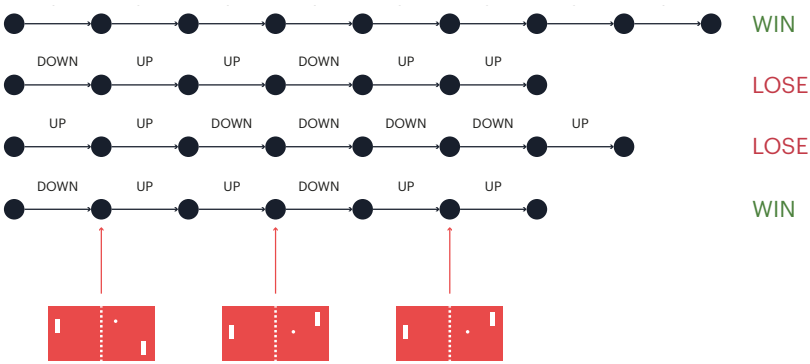
Se diferencia de los otros tipos en que trata algoritmos orientados a objetivos. El algoritmo debe aprender cómo lograr un objetivo complejo o uno a largo plazo a través de varios pasos.

Se definen los siguientes conceptos:

- **Agente:** el elemento que ejecuta las acciones.
- **Policy:** la estrategia que decide qué acciones se ejecutan en base a un estado.
- **Entorno:** el mundo en que se mueve el agente.
- **Recompensa:** la medida sobre la que se decide la bondad de una acción.
- **Estado:** la situación del entorno en un momento específico.
- **Acción:** una de los distintos actos que el agente puede ejecutar.

— 01.38 Reinforcement Learning.





— 01.39

Rectified Linear Unit (ReLU).

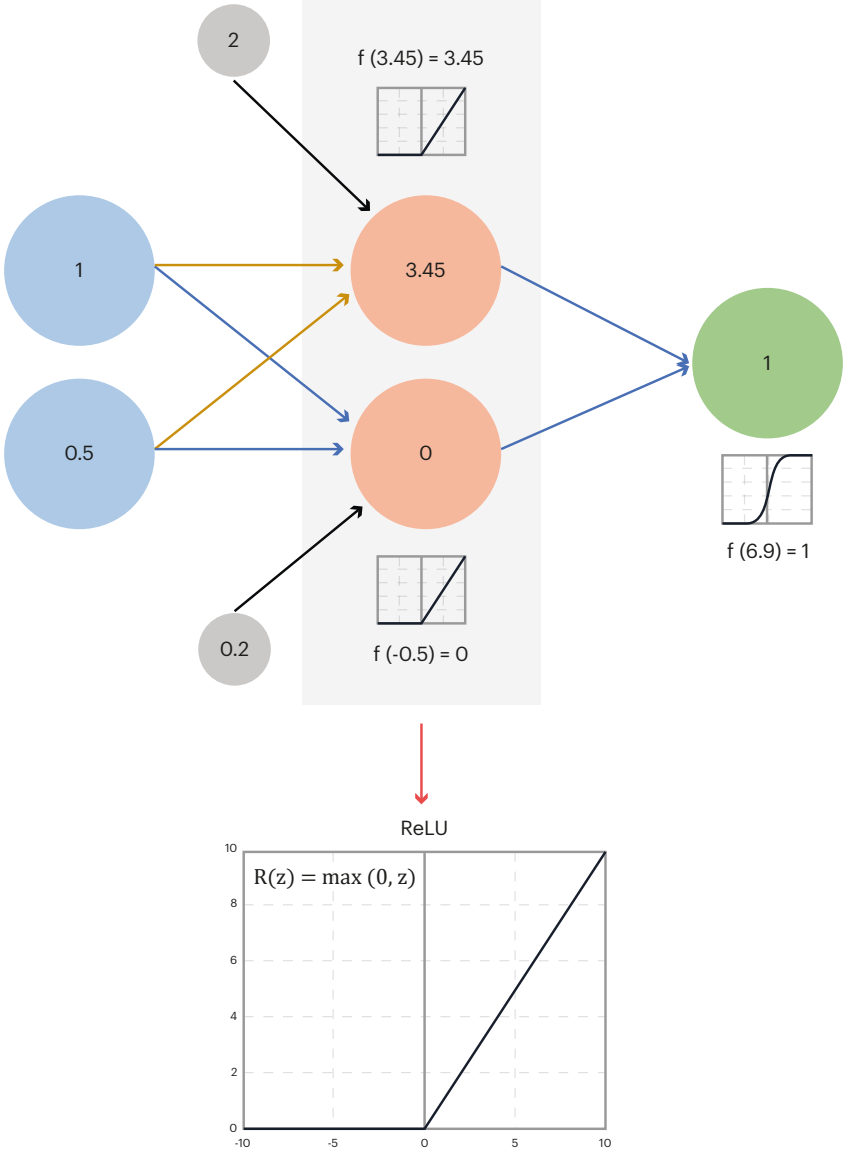
La función ReLU (unidad lineal rectificadora) es una función que recientemente se ha utilizado en redes neuronales, especialmente en las capas intermedias. La razón es que esta es una función muy simple de calcular: aplana la respuesta a todos los valores negativos a 0, mientras deja todo sin cambios para valores iguales o mayores que 0.

Esta simplicidad, combinada con el hecho de reducir drásticamente el problema del vanishing gradient (desvanecimiento de gradiente), la convierte en una función particularmente atractiva en las capas intermedias, donde la cantidad de pasajes y cálculos es importante.

El cálculo de la derivada es, de hecho, muy simple: para todos los valores negativos es igual a 0, mientras que para los positivos es igual a 1. Sin embargo, en el punto angular en el origen, la derivada no está definida pero, por convención, sigue siendo 0.

$$f(x) = x+ = \max(0, x),$$

— 01.39 Rectified Linear Unit (ReLU).



— 01.40

Sigmoid function.

(Función sigmoidea)

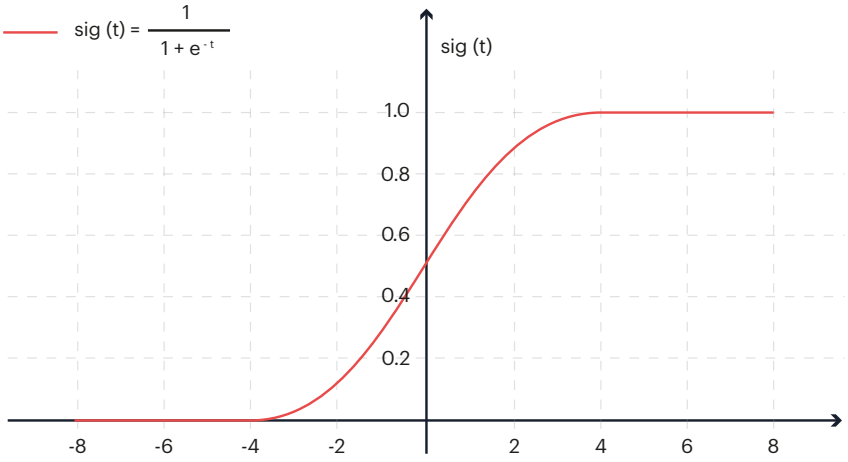
Es una función que transforma cualquier valor de menos infinito a infinito en otro comprendido entre 0 y 1.

$$f(t) = \frac{1}{1 + e^{-t}}$$

Su uso en Machine Learning es extendido, por ejemplo, en el algoritmo de clasificación conocido como regresión logística, en el que transforma un valor en su probabilidad de pertenecer a la clase 0 o 1.

También lo podemos ver ocasionalmente como función de activación en una red neuronal.

— 01.40 Sigmoid function.



“

La inteligencia artificial es la nueva electricidad.

Andrew Ng.
Investigador de IA en la Universidad de Stanford y unos de los fundadores de Google Brain.

— 01.41

Singularity.

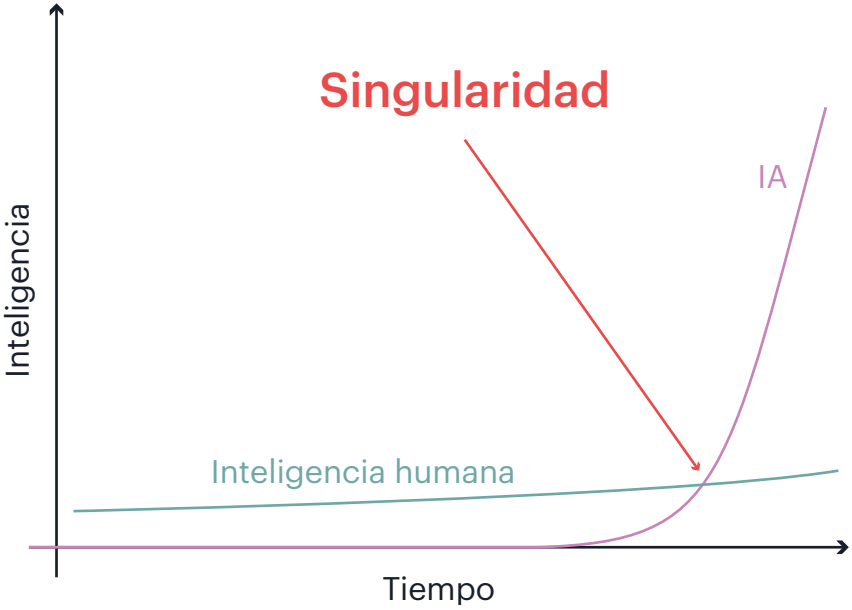
(Singularidad)

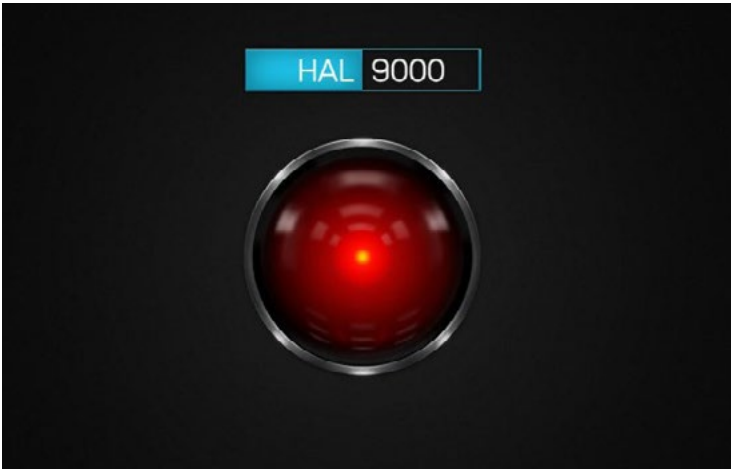
La singularidad se refiere al momento en que el desarrollo y la evolución de las capacidades cognitivas de la inteligencia artificial superarán a la inteligencia humana.

Hay discusión sobre si este momento realmente llegará y cuándo sucederá. Si esto sucediera, se habría creado una "superinteligencia artificial" que seguiría el paradigma de la inteligencia artificial general, es decir, una inteligencia artificial capaz de aprender y adaptarse a diferentes problemas y escenarios, como hacemos los humanos.

Algunos autores y figuras públicas, como Stephen Hawking y Elon Musk, han expresado su preocupación de que la inteligencia artificial general pueda provocar la extinción del ser humano. Las consecuencias de la singularidad y su potencial beneficio o daño para los humanos son un tema continuo de debate y de inspiración para la literatura y el cine.

— 01.41 Singularity.





— 01.42

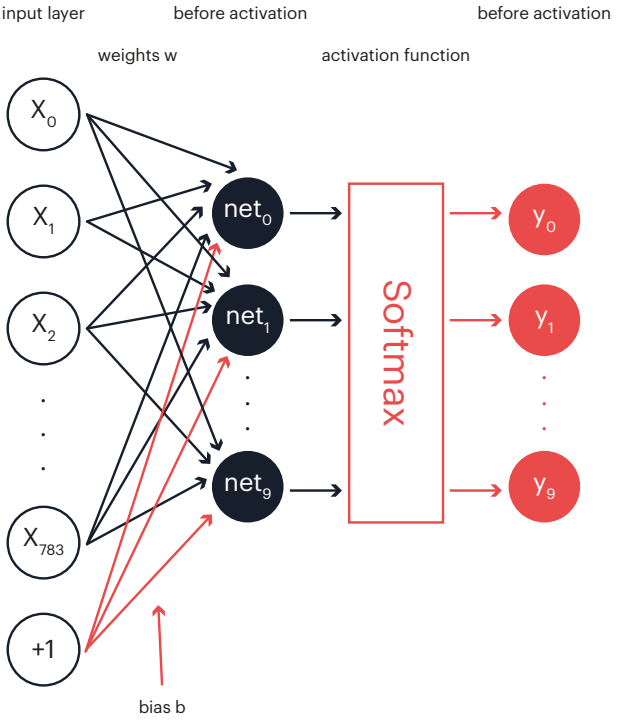
Softmax.

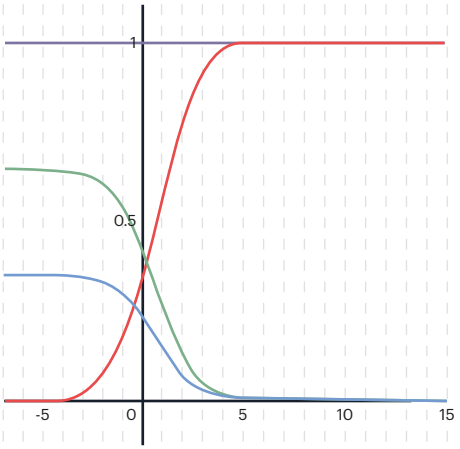
La función Softmax calcula la distribución de probabilidades del evento en “n” diferentes eventos. En general, esta función calculará las probabilidades del intervalo (0,1) de cada clase objetivo sobre todas las clases objetivo posibles. Más tarde, las probabilidades calculadas serán útiles para determinar la clase objetivo para las entradas dadas.

En Machine Learning la función Softmax se implementa a través de una capa de red neuronal justo antes de la capa de resultado. La capa de Softmax debe tener la misma cantidad de nodos que la capa de resultado.

$$P(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_x}}$$

— 01.42 Softmax.





— 01.43

Supervised/unsupervised learning.

(Aprendizaje supervisado/no supervisado)

Los algoritmos de Machine Learning se agrupan en tres grandes categorías: supervisado, sin supervisión o por refuerzo.

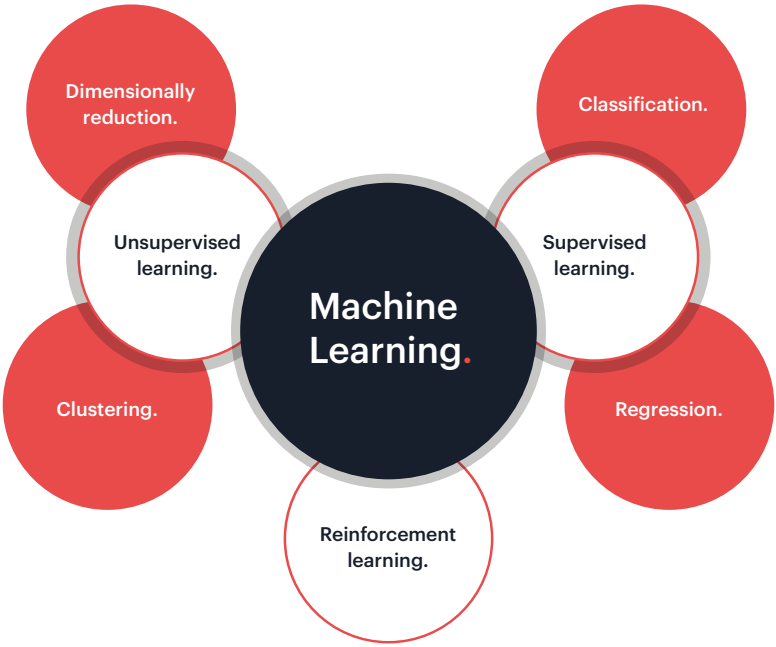
Un algoritmo supervisado analiza los datos de entrenamiento etiquetados y produce una función inferida, que puede usarse para mapear nuevos ejemplos. Algunos ejemplos de algoritmos supervisados serían los algoritmos de regresión o los árboles de decisión.

Los algoritmos no supervisados utilizan información que no está clasificada ni etiquetada y buscan patrones o tendencias en la estructura de los datos. Los más comunes de este tipo serían los algoritmos de clustering, aunque existen otros como los algoritmos de Análisis del Componente Principal (PCA).

Los algoritmos por refuerzo siguen un enfoque diferente basado en un modelo de recompensa.

Existe un cuarto grupo de algoritmos llamados los semi-supervisados, que se aplican cuando solo una parte del conjunto de datos está etiquetado y es necesario combinar técnicas supervisadas y no supervisadas.

— 01.43 Supervised/unsupervised learning.



— 01.44

SVM (Support Vector Machine).

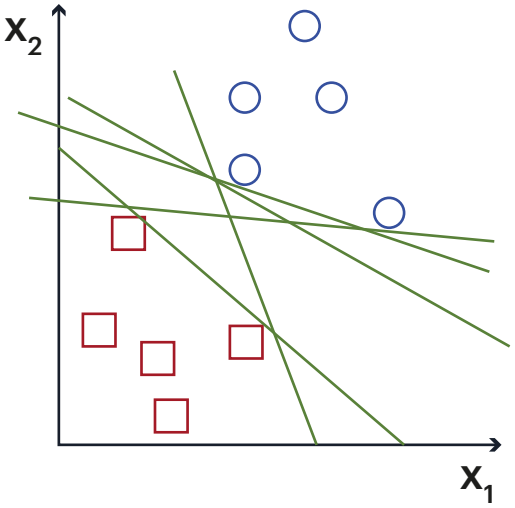
(Máquinas de Soporte Vectorial)

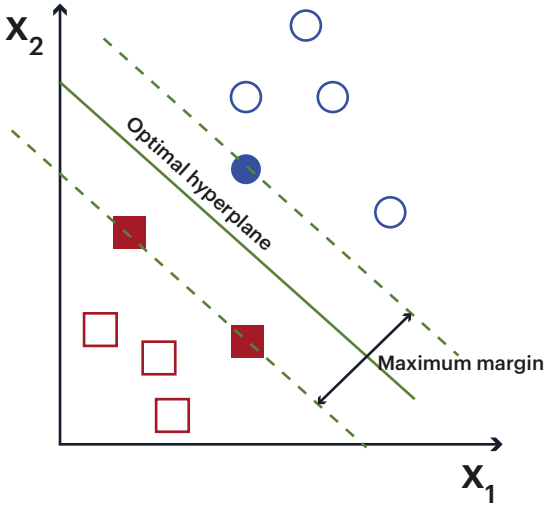
Las máquinas de vectores de soporte (SVM) son un conjunto de métodos de aprendizaje supervisado utilizados para clasificación, regresión y detección de valores atípicos (outliers).

Un SVM selecciona un hiperplano particular que no solo separa los puntos en las dos clases, sino que lo hace de una manera que maximiza el margen: la distancia entre el hiperplano y los puntos más cercanos del conjunto de entrenamiento.

Una ventaja de elegir el hiperplano de separación, para tener el mayor margen posible, es que puede haber puntos más cercanos al hiperplano en el conjunto de datos completo que en el de entrenamiento. Si es así tenemos más posibilidades que estos puntos se clasifiquen correctamente a que elegir el hiperplano en el conjunto de entrenamiento.

— 01.44 SVM (Support Vector Machine).





— 01.45

TensorFlow.

Tensorflow es la biblioteca de código abierto desarrollada por Google para llevar a cabo proyectos de Machine Learning.

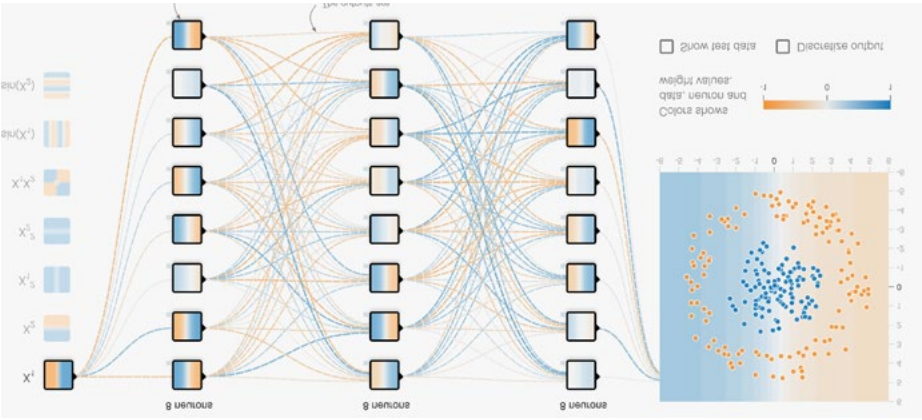
TensorFlow fue creada por el equipo de Google Brain y liberada en 2015 bajo licencia Apache 2.0. Hoy en día es una de las herramientas más extendidas en el mundo del Machine Learning, en particular para la construcción de redes de neuronas.

Aunque TensorFlow se usa principalmente en el área del Machine Learning, también se puede usar para otro tipo de algoritmos que requieran tareas de cálculo numérico mediante grafos de datos.

Existen otras alternativas a TensorFlow en el mercado como PyTorch de Facebook y MXNet de Amazon.

— 01.45 TensorFlow.

TensorFlow



— 01.46

Time series analysis.

(Análisis de series temporales)

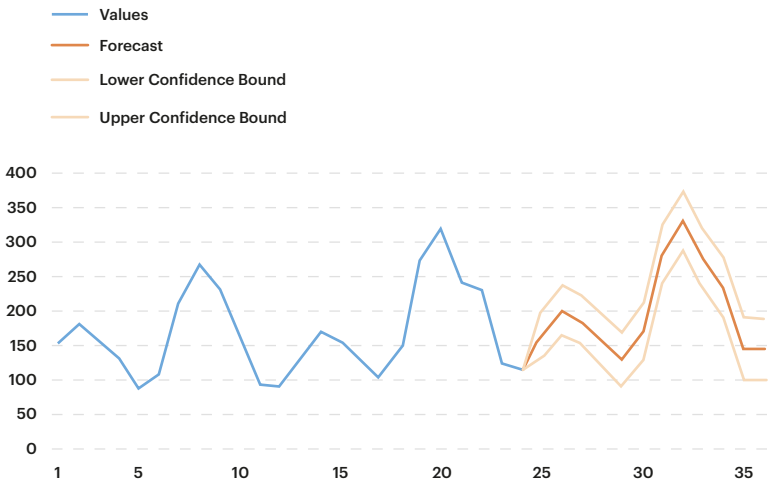
Una serie temporal es una sucesión de datos ordenados en el tiempo, cuyo análisis e interpretación puede dar lugar a la realización de predicciones sobre su comportamiento futuro.

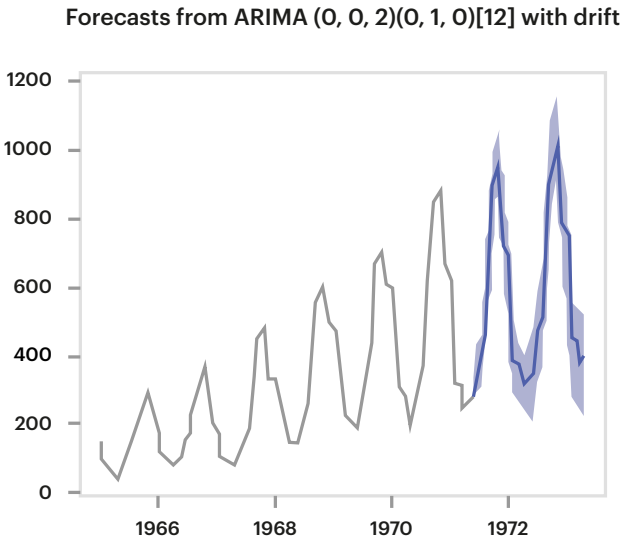
La descomposición de una serie temporal permite analizar la tendencia (propensión de los datos a aumentar o disminuir a largo plazo), estacionalidad (fluctuaciones periódicas que se repiten de forma cíclica) y el componente aleatorio (fluctuaciones impredecibles o no controladas).

Existen varios métodos para modelar una serie temporal con el fin de realizar predicciones a futuro sobre su comportamiento. Entre ellos, destacan el modelo ARIMA, exponential smoothing o los basados en redes neuronales. Estos modelos permiten generar observaciones futuras en base al histórico, incluyendo un intervalo de confianza estadística sobre estos valores.

Los modelos predictivos implementados, utilizando este tipo de algoritmos, suelen validarse de forma similar a un problema de regresión, particionando en este caso la muestra de forma ordenada en el tiempo y comparando los valores reales más recientes con los predichos por el modelo.

— 01.46 Time series analysis.





— 01.47

Training Set.

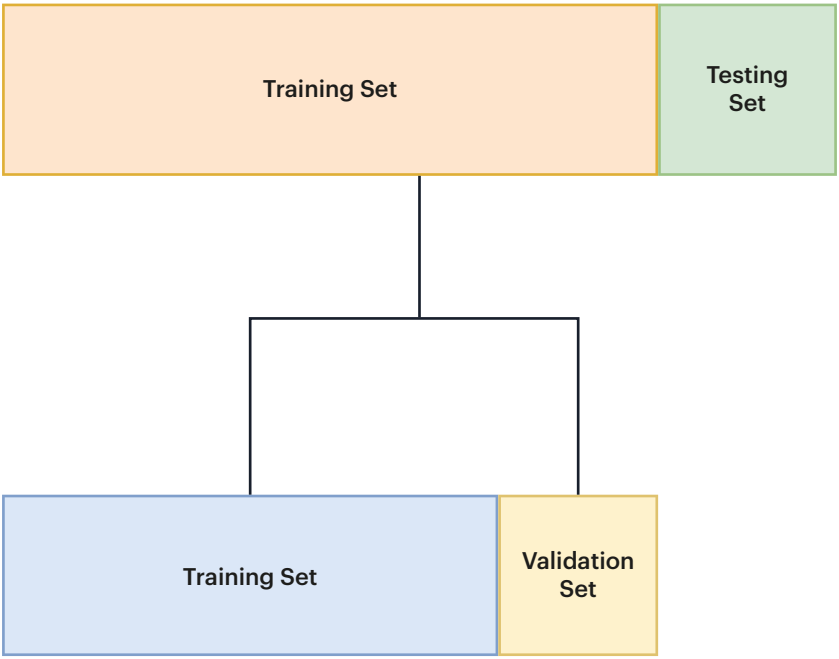
(Conjunto de Entrenamiento)

En Machine Learning, conjunto de entrenamiento son los datos con los que los algoritmos se autoajustan (“entrenan”) para acercarse a su objetivo. Por ejemplo, en un sistema de reconocimiento de gatos en imágenes, serán un montón de imágenes, cada una etiquetada correspondientemente.

Una vez entrenado el sistema, su funcionamiento se comprueba con otro conjunto de datos conocido como conjunto de test. Si un sistema se entrena solamente con un conjunto de entrenamiento, y no se comprueba su funcionamiento con un conjunto de test, se puede llegar a lo que se conoce como “overfitting” o sobreentrenamiento, donde los sistemas se ajustan perfectamente al conjunto de entrenamiento, pero no son capaces de resolver correctamente cuando se encuentran datos nuevos.

Si hay más de un algoritmo a evaluar o hay que hacer ajustes de hiperparámetros, se puede usar una parte del conjunto de entrenamiento como conjunto de validación, de forma que los distintos algoritmos o hiperparámetros se ajustan con una sección del conjunto de entrenamiento y las distintas opciones se comparan entre sí usando el conjunto de validación.

— 01.47 Training Set.



— 01.48

Transfer Learning.

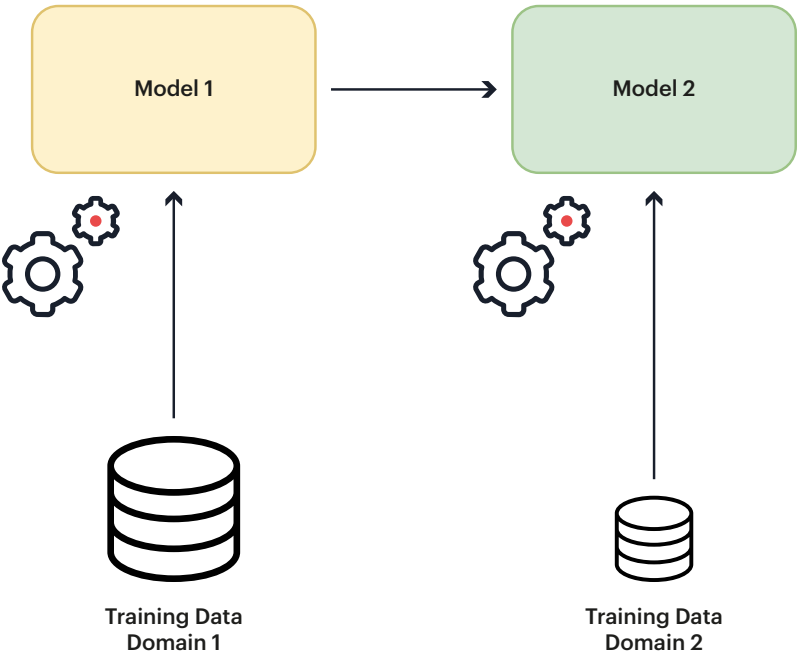
(Aprendizaje por Transferencia)

El aprendizaje por transferencia es una técnica para modelos de Deep Learning que consiste en reutilizar un modelo ya entrenado como punto de partida para otro modelo de un dominio diferente pero similar. Por ejemplo, usar un modelo de detección de objetos en imágenes como punto de partida para un modelo de detección de gatos en imágenes.

El objetivo de las técnicas de aprendizaje por transferencia es reaprovechar las primeras capas de modelos de redes neuronales, los cuales suelen ser capaces de extraer atributos más genéricos para ahorrar tiempo y costes.

En este tipo de técnicas son en las que se basan los productos que ofrecen modelos ya entrenados, como pueden ser Google Cloud AutoML o los modelos integrados de AWS Sagemaker. Es útil también cuando se dispone de un modelo más generalista y un dominio concreto en el que no se disponen de muchos datos de entrenamiento.

— 01.48 Transfer Learning.



— 01.49

Underfitting.

(Subajuste)

Un modelo de aprendizaje automático se ajusta durante el proceso de entrenamiento cuando es capaz de captar las tendencias subyacentes y los patrones presentes en los datos. Entonces, decimos que el modelo "ha aprendido" a generalizar dichos patrones sobre nuevas observaciones.

Por lo tanto, dado un set de datos dividido en entrenamiento y validación, un modelo está subajustado cuando es incapaz de "aprender" cuáles son los patrones presentes en las observaciones del entrenamiento y cuyo desempeño será pobre tanto durante el proceso de ajuste como a la hora de generalizar inferencias para el set de validación.

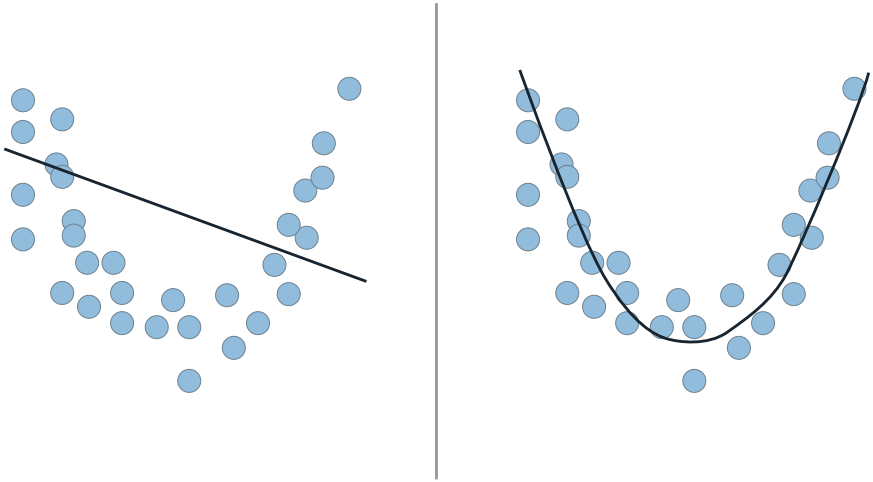
Los procedimientos más comunes para solucionar el subajuste de un modelo son:

- **Añadir variables:** el modelo tendrá más facilidades para captar los patrones.
- **Aumentar la complejidad de los modelos:** ampliar el número de capas y neuronas en el caso del Deep Learning, por ejemplo.
- **Alargar el proceso de entrenamiento:** el tiempo de ajuste varía según el modelo y los datos en cuestión.

Cuando un modelo está subajustado, tendrá mayor sesgo y por lo tanto, un mayor error de base cuando intente generalizar.

— 01.49 Underfitting.

Ejemplo de la función de regresión aprendida por dos modelos después del entrenamiento (a), (b).



(a) Se trata de un modelo subajustado y (b) de un modelo que se ajusta adecuadamente.

— 01.50

Variance.

(Varianza)

En el ámbito del aprendizaje automático, la varianza se trata de forma informal como el tipo de error que se percibe debido a la sensibilidad del modelo durante el proceso de entrenamiento. Por ejemplo, si durante el entrenamiento se registra un error del 15% y en set de validación el modelo presenta un error del 17%, pensamos en ese 2% de diferencia como la varianza del algoritmo.

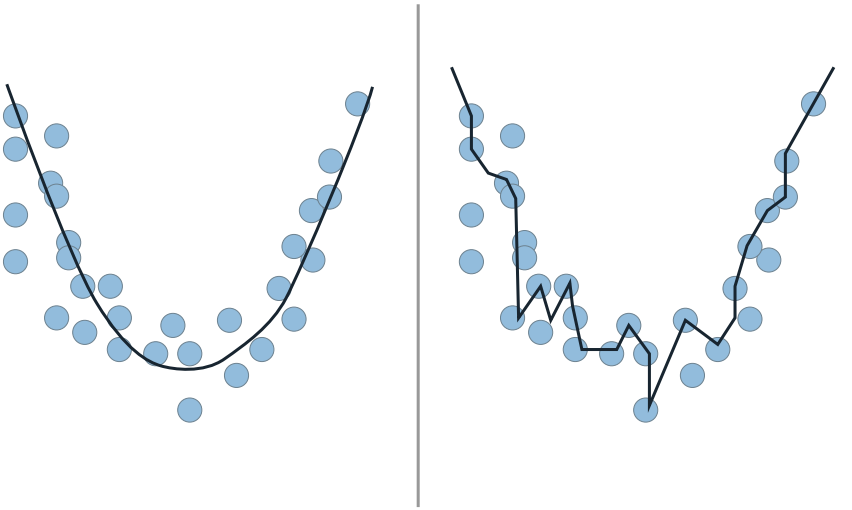
Desde otro punto de vista, el algoritmo trata de aproximar una función que capte patrones y tendencias a partir de un set de datos de entrenamiento concreto, por lo que cabe esperar que el modelo resultante tenga cierta varianza. Sin embargo, si cambiásemos las observaciones durante el proceso de aprendizaje, nos encontraríamos con modelos que estimarían una función similar (caso ideal) y otros cuyo resultado sería distinto (varianza alta).

De este modo, los modelos con una varianza alta prestarán mucha atención a los datos de entrenamiento y no serán capaces de generalizar sus predicciones sobre observaciones que no haya visto antes (b) y, por lo tanto, incurrirán en Overfitting.

El campo de la estadística se refiere más formalmente a la varianza en Machine Learning como a la variabilidad de las predicciones para una observación dada, cuyo valor es indicativo de la dispersión de los datos.

— 01.50 Variance.

Ejemplo de la función de regresión aprendida por dos modelos después del entrenamiento (a), (b).



(a) Se trata de un modelo con varianza equilibrada y (b) de un modelo con varianza alta.

“

La inteligencia artificial está sin duda viviendo una etapa dorada. Todos interaccionamos con sistemas de inteligencia artificial en nuestro día a día, todos los días. La AI ya impregna y enriquece nuestra vida cotidiana.

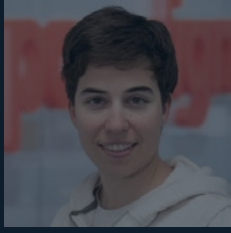
Nuria Oliver.
Investigadora en IA y una de las pioneras de la IA en España.

Autores.

Alberto Grande.

Innovation.

Alberto Grande es Ingeniero Informático por la UPM, amante de la tecnología y todo lo que la rodea. Está especialmente interesado en desarrollo (principalmente Java), arquitecturas distribuidas y escalables y sistemas Cloud. Piensa que en software la calidad debe estar por encima de todo y nunca debería ser negociable. Actualmente trabaja como Arquitecto Software en Paradigma.



Beatriz Blanco.

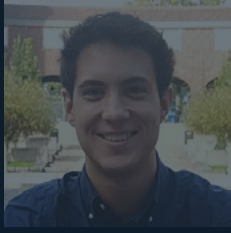
Data Engineering.

Big Data & AI Cloud Architect en Paradigma Digital. Durante toda mi trayectoria profesional he estado en contacto con proyectos en los que se construían modelos analíticos para todo tipo de sectores. Lo que me más me ha gustado siempre es combinar mi background tecnológico con mi experiencia en el mundo analítico trabajando con matemáticos y data scientist, buscando siempre obtener el mejor modelo matemático, pero bajo un proceso óptimo con una arquitectura robusta que permita desplegar los modelos en sistemas productivos.

Carlos Navarro.

Data Engineering.

Ingeniero de Telecomunicaciones por la UPM, Carlos Navarro lleva trabajando en Paradigma más de 10 años. En ese tiempo ha participado en proyectos, proyectos de tecnología semántica, de medición de influencia y proyectos europeos como MixedEmotions. También ha ejercido de arquitecto software en proyectos Big Data y con plataformas basadas en microservicios.



Francisco Rodes.

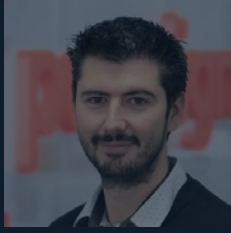
Data Science.

Soy Ingeniero Industrial convencido del valor que aporta la Transformación Digital a los negocios. Me defino como data-driven, curioso y con mentalidad emprendedora que afronta la resolución de problemas apoyándose en técnicas de analítica avanzada, tecnologías Cloud y Machine Learning. Actualmente trabajo como Data Scientist en Paradigma Digital.

José Manuel Carral.

Data Engineering.

Ingeniero en Informática. Comencé a interesarme por la tecnología cuando estrené mi primer ordenador en el año 2000. Actualmente trabajo en Paradigma Digital como ingeniero de datos y tengo experiencia en proyectos de Big Data, Business Intelligence y Machine Learning. Me encanta viajar, leer, ver cine y aprender cada día algo nuevo.



Juan Iglesias.

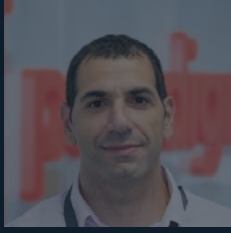
Data Science.

Ingeniero de telecomunicación por la UAM, mi carrera profesional ha estado muy ligada al ámbito de las telecomunicaciones en diferentes vertientes, con especial foco en la detección de fraude de bypass de llamadas. En ello he trabajado como analista de datos, preventa y liderando un pequeño equipo. Además, he desarrollado modelos predictivos para detección de fraude usando técnicas de Machine Learning.

Manuel Zaforas.

Innovation.

Manuel Zaforas es Ingeniero Superior en Informática por la UPM. Actualmente trabaja en Paradigma como líder de la iniciativa de innovación en AI y Big Data ayudando a diferentes compañías a aplicar tecnologías innovadoras en sus negocios. Está interesado en Big Data, AI, Data Science, Machine Learning, Deep Learning, HPC, IoT, Cloud, NoSQL, NewSQL, Python y Agile.



Marco Russo.

Data Science.

Apasionado de Finanza, Marketing e IT, es consultor y especialista en Digital Data Analytics a nivel internacional trabajando para diferentes sectores industriales. Además, desde hace 6 años compagina su trabajo con la formación in-company y en diferentes escuelas de negocios y Cámara de Comercio, realizando módulos y cursos de Analytics, DataViz, CRO y Tag Manager. Cuando no piensa en los datos, además de compaginar su tiempo con la familia, puedes encontrarlo escalando montañas en la sierra de Madrid con su bici de carretera o en una cancha de basket.

Conclusión:

“Como tecnóloga, veo cómo la IA y la cuarta revolución industrial van a impactar en cada aspecto de la vida de las personas.”

Fei-Fei Li, Investigadora de IA en Google en el área de computer vision y profesora en la Universidad de Stanford.

Sin duda, la explosión en la aplicación de las técnicas de Machine Learning ha cambiado la historia de la computación y de la inteligencia artificial en particular.

En este ebook hemos recopilado los 50 términos fundamentales para conocer y entender el funcionamiento de estas técnicas y dar el primer paso para profundizar en esta área tan apasionante y que tanto impacto está teniendo.

Think Big.

V.1.0 - Mayo de 2020

info@paradigmadigital.com

eBooks
Paradigma