

Inteligencia Artificial

Machine Learning - (Aprendizaje Automático)

Introducción



Machine Learning

“The field of study that gives computers the ability to learn without being explicitly programmed.”

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210–229.

“A computer program is said to learn from **experience E** with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Machine Learning

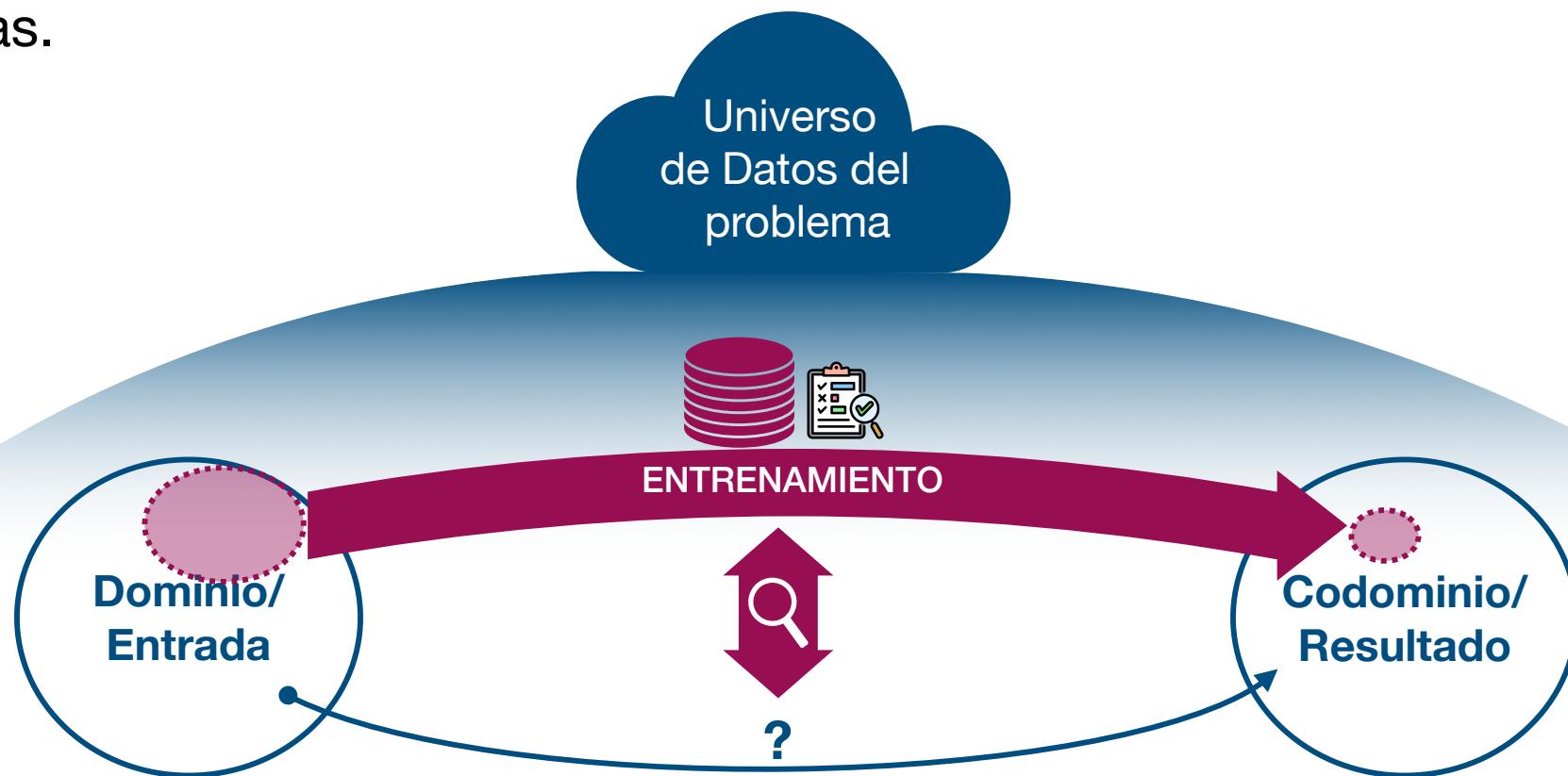
En otras palabras podemos definir el aprendizaje como “**capacidad de mejorar en una tarea a través de la práctica**”.

Esto plantea dos preguntas fundamentales: **¿cómo sabe la computadora si está mejorando?** y **¿cómo sabe qué debe cambiar para mejorar?**. Existen diferentes respuestas posibles, que dan lugar a distintos tipos de aprendizaje automático:

- **Aprendizaje supervisado**: El algoritmo recibe un conjunto de entrenamiento con las respuestas correctas y aprende a generalizar para responder correctamente a nuevas entradas.
- **Aprendizaje no supervisado**: No se proporcionan respuestas correctas. El algoritmo busca similitudes entre los datos para agruparlos.
- **Aprendizaje por refuerzo (reinforcing)**: El algoritmo sabe si su respuesta fue correcta o no, pero no cómo mejorarla. Aprende explorando distintas opciones.
- **Aprendizaje evolutivo**: La evolución biológica puede interpretarse como un proceso de aprendizaje (aptitud de adaptación). Podemos modelar este proceso en una computadora usando el concepto de aptitud, que representa una puntuación que indica qué tan buena es una solución actual.

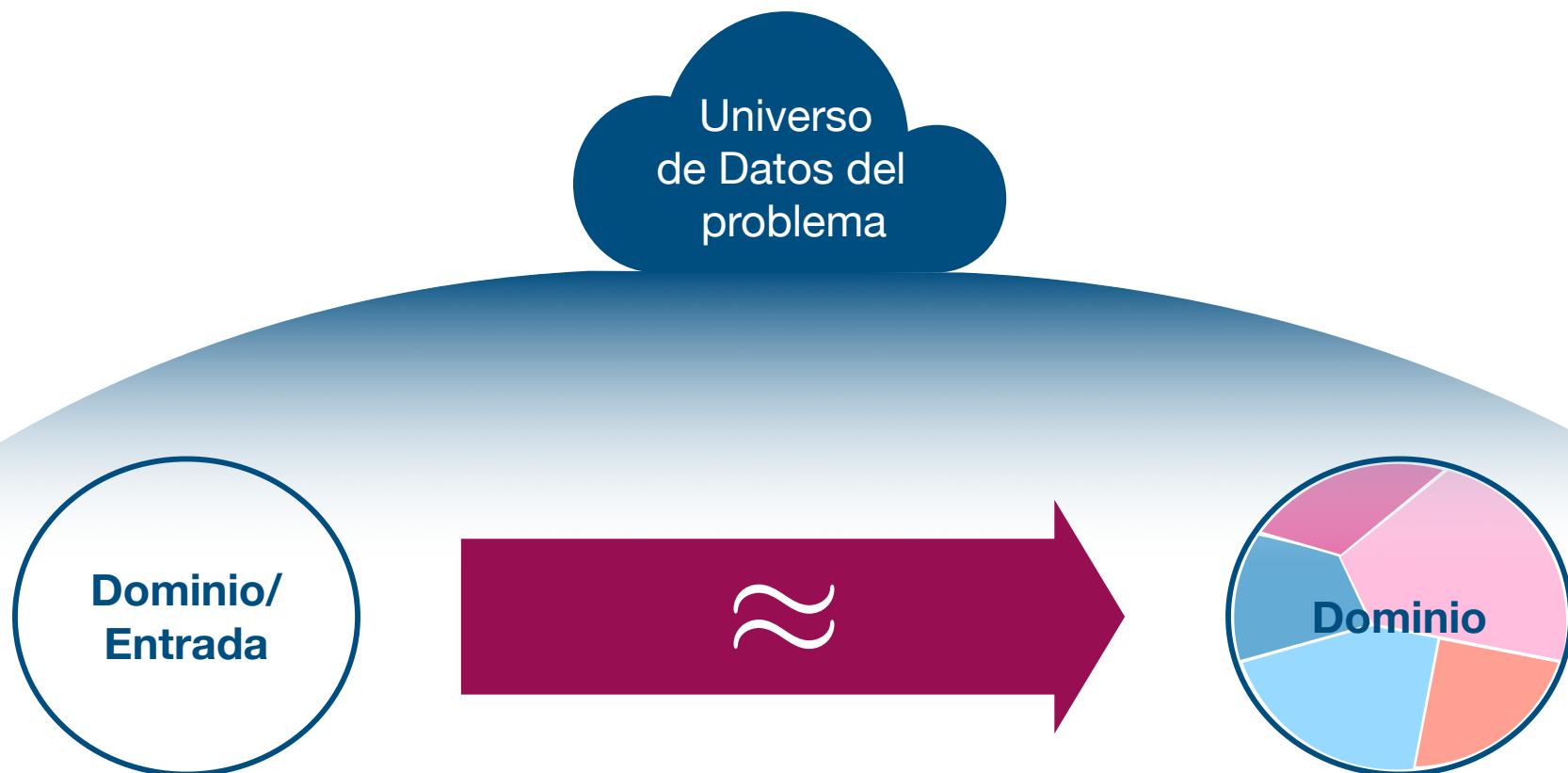
Machine Learning - Aprendizaje Supervisado

El algoritmo recibe un conjunto de entrenamiento **con las respuestas correctas** y aprende a generalizar para responder correctamente a nuevas entradas.



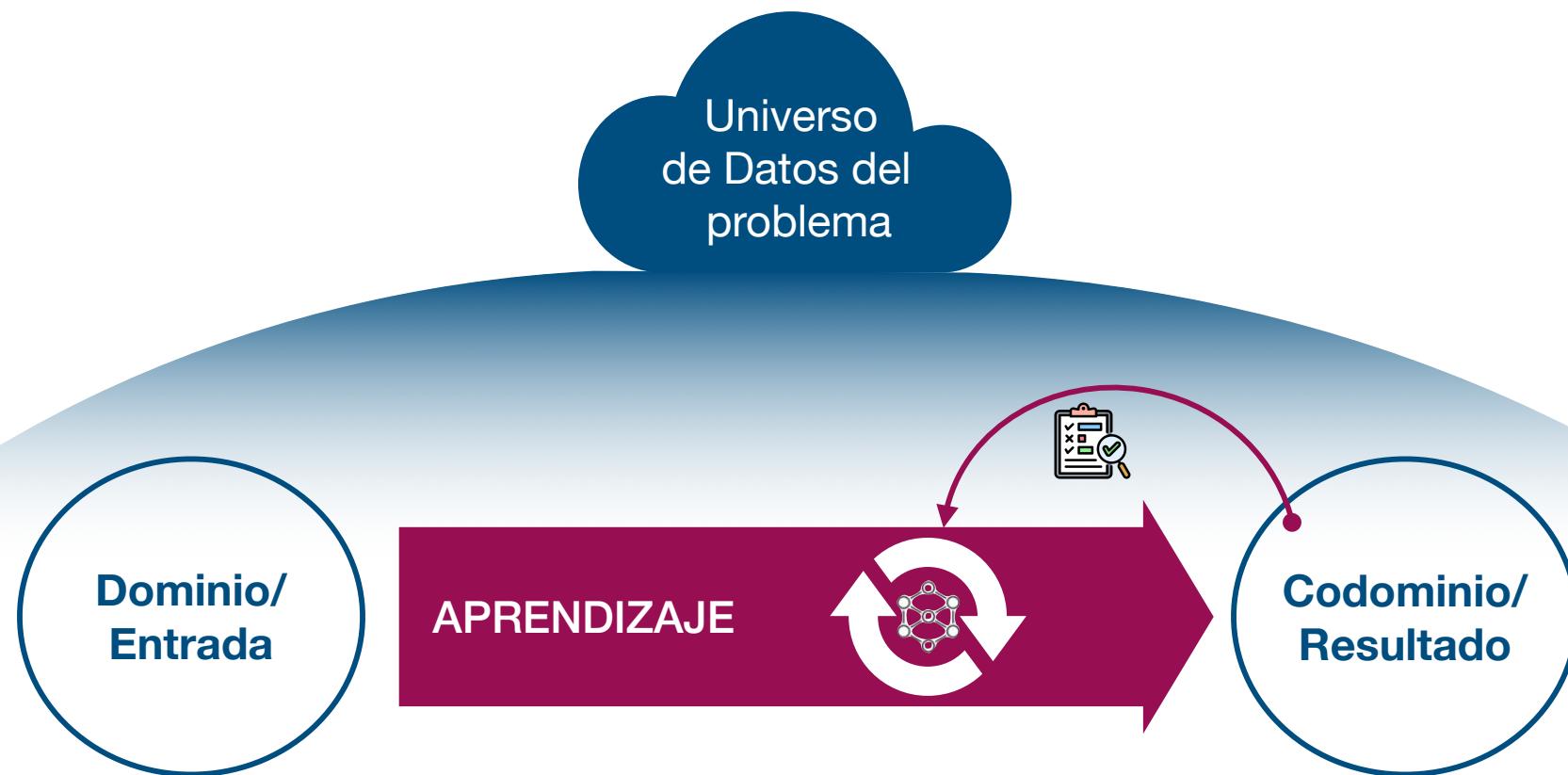
Machine Learning - Aprendizaje No Supervisado

No se proporcionan respuestas correctas. El algoritmo busca similitudes entre los datos para agruparlos



Machine Learning - Aprendizaje por refuerzo (reinforcing)

El algoritmo sabe si su respuesta fue correcta o no, pero no cómo mejorarla. Aprende explorando distintas opciones.



Machine Learning - ¿ Cuándo utilizarlo ?

Aunque el uso de ML ha crecido exponencialmente, su aplicación se **concentra en contextos con grandes volúmenes de datos** para extraer información relevante

- ML es útil en problemas que requieren la definición de un **número muy grande de reglas**, o de reglas muy complejas
- ML puede dar buenas soluciones a problemas complejos, para los que no conocemos una buena solución algorítmica
- Un sistema de ML puede adaptarse a ambientes cambiantes entrenándolo periódicamente con nuevos datos
- ML puede ayudar a las personas a ganar entendimiento sobre problemas difíciles
- Puede ayudar a analizar y extraer conclusiones a partir de grandes cantidades de datos

Machine Learning - Qué esperamos como resultado

Tipo de Aprendizaje	Tipo de Resultado Esperado	Ejemplo
Aprendizaje Supervisado	Clasificación	Predecir categorías. Ej: detectar si un email es spam o no.
	Regresión	Predecir valores numéricos. Ej: precio de una casa.
Aprendizaje No Supervisado	Agrupamiento (Clustering)	Descubrir grupos. Ej: segmentación de clientes por comportamiento.
	Detección de anomalías	Identificar valores atípicos. Ej: fraude bancario.
	Reducción de dimensionalidad	Simplificar datos. Ej: visualización de datos de alta dimensión.
	Reglas de asociación	Encontrar relaciones frecuentes. Ej: productos comprados juntos.
Aprendizaje por Refuerzo	Secuencias óptimas de acciones (reglas)	Tomar decisiones en ambientes dinámicos. Ej: robots o videojuegos.
Aprendizaje Auto-supervisado	Representaciones de datos	Aprender estructuras internas sin etiquetas explícitas. Ej: embeddings.
Aprendizaje Semi-supervisado	Clasificación / Regresión	Mezcla de datos etiquetados y no etiquetados. Mejora generalización.

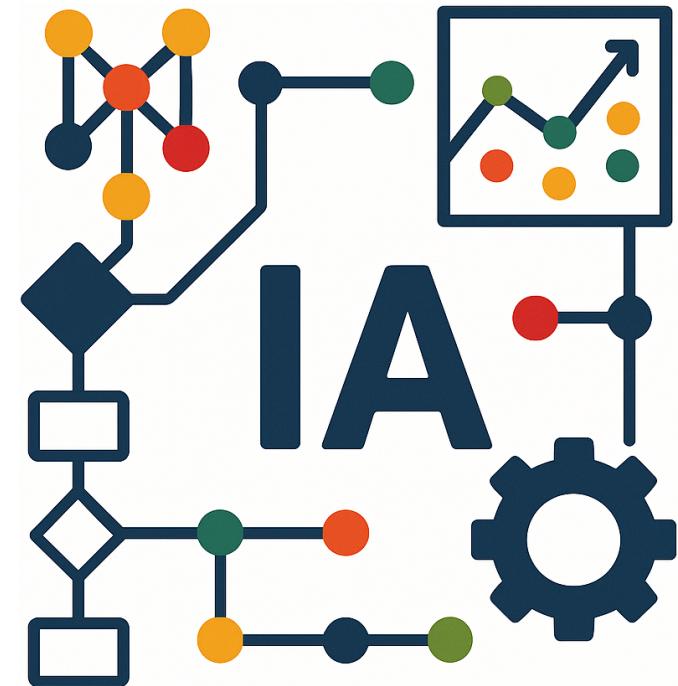
Machine Learning - Problemas y desventajas

Tipo de Problema	Problema / Desventaja	Descripción	Ejemplos
Datos	Insuficiencia de datos	ML requiere grandes volúmenes de datos diversos para generalizar bien.	Sobreajuste, mala precisión.
	Datos sesgados o desequilibrados	Si los datos no representan todas las clases o contextos, el modelo será parcial o injusto.	Discriminación algorítmica.
	Mala calidad de datos	Datos erróneos, incompletos afectan negativamente el aprendizaje.	Predicciones erróneas, comportamiento impredecible.
	Dificultad para etiquetar datos	El etiquetado supervisado puede ser costoso o lento.	Retrasa el entrenamiento y limita la precisión.
Modelado	Sobreajuste (overfitting)	El modelo se ajusta demasiado a los datos de entrenamiento y falla en generalizar.	Buen rendimiento en entrenamiento, pobre en producción.
	Subajuste (underfitting)	El modelo es demasiado simple y no captura la complejidad del problema.	Baja precisión general.
	Alta complejidad computacional	Modelos avanzados como redes neuronales profundas requieren mucha computación y energía.	Costos elevados de entrenamiento y mantenimiento.
	Interpretabilidad limitada	Muchos modelos son cajas negras difíciles de explicar.	Problemas de confianza, auditoría y cumplimiento normativo.
Ética y Sociedad	Discriminación y sesgo	El modelo puede reproducir o amplificar sesgos presentes en los datos.	Injusticia en decisiones
	Falta de transparencia	No siempre se entiende por qué el modelo toma una decisión.	Dificultad para explicar decisiones ante usuarios o reguladores.
	Riesgo de automatización	Puede reemplazar tareas humanas sin considerar impacto social o laboral.	Pérdida de empleos, brechas digitales.
Implementación	Requiere infraestructura	Se necesita hardware especializado, almacenamiento, y despliegue robusto.	Costos de infraestructura.
	Mantenimiento constante	Los modelos necesitan ser actualizados con nuevos datos.	Degrado del rendimiento con el tiempo.
	Riesgo de sobreconfianza	Creer que el modelo siempre tiene razón puede llevar a malas decisiones automatizadas.	Falta de supervisión humana.

Inteligencia Artificial

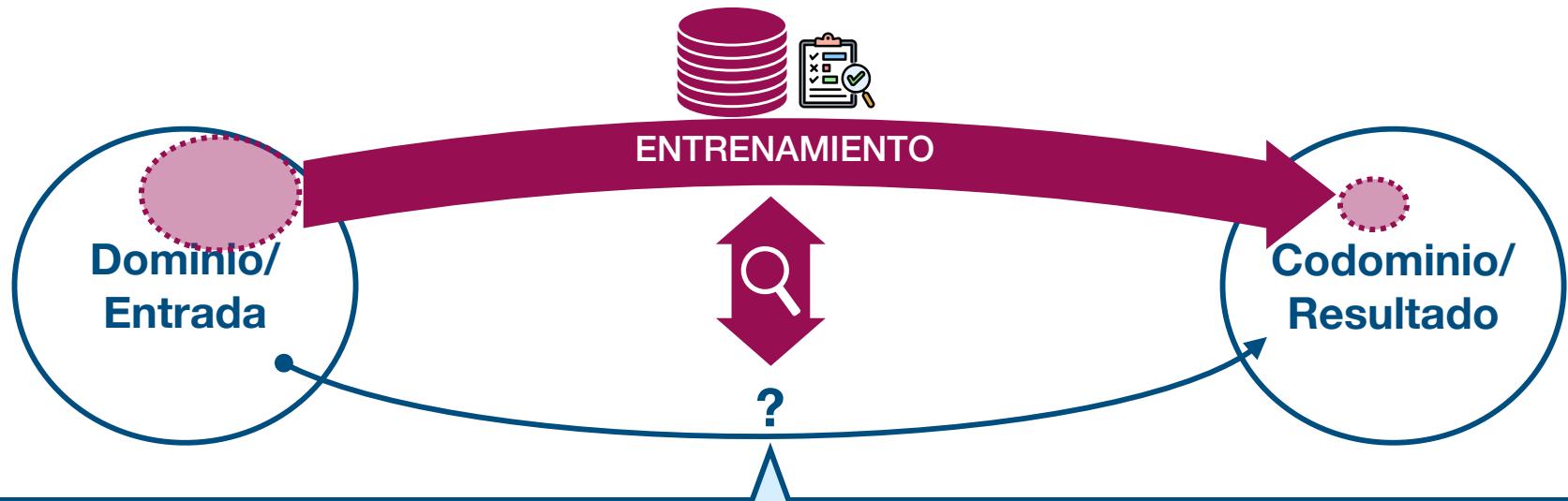
Machine Learning - (Aprendizaje Automático)

Aprendizaje Supervisado



Machine Learning - Aprendizaje Supervisado

El algoritmo recibe un conjunto de entrenamiento **con las respuestas correctas** y aprende a generalizar para responder correctamente a nuevas entradas.



De manera general, los problemas se dividen en dos categorías, de acuerdo a la salida esperada del algoritmo:

- Problemas de **clasificación**: La salida del algoritmo es un valor tomado de un conjunto de finito de valores
- Problemas de **predicción (regresión)**: La salida es un número entero o real

ML Supervisado - Desafíos del aprendizaje

El objetivo del aprendizaje automático es conseguir **un modelo que se ajuste lo mejor posible** (aprenda) a los datos de entrenamiento. La **calidad** del modelo aprendido será determinante para la certeza de las respuestas cuando lo utilicemos para clasificar o predecir datos ante nuevas consultas (no estudiadas durante el entrenamiento).

Los problemas que conllevan los extremos del ajuste impactan negativamente en la calidad del modelo:

- **Subajuste (Underfitting)**: modelo es demasiado simple para captar los patrones de los datos.
- **Sobreajuste (Overfitting)**: cuando el modelo aprende excesivamente de los datos de entrenamiento, incluyendo el ruido o las particularidades específicas, y por eso no generaliza bien a nuevos dato.

ML Supervisado - ¿Cómo medimos la calidad?

Según el tipo de problema a resolver (clasificación o regresión), se emplean distintas métricas que permiten evaluar la calidad del modelo. Estas métricas son fundamentales durante el desarrollo, ya que orientan el ajuste del modelo hacia una solución adecuada.

Por ejemplo, en problemas de **clasiﬁcación** se utilizan métricas como la **precisión** (accuracy), la **sensibilidad** (recall) o **F1-Score** que tiene en cuenta ambas anteriores.

En cambio, en problemas de **regresión** (predicción), se aplican métricas como el **error absoluto medio** (MAE) o el **error cuadrático medio** (MSE), que indican qué tan lejos están las predicciones de los valores reales

ML Supervisado - Proceso general de desarrollo e implementación

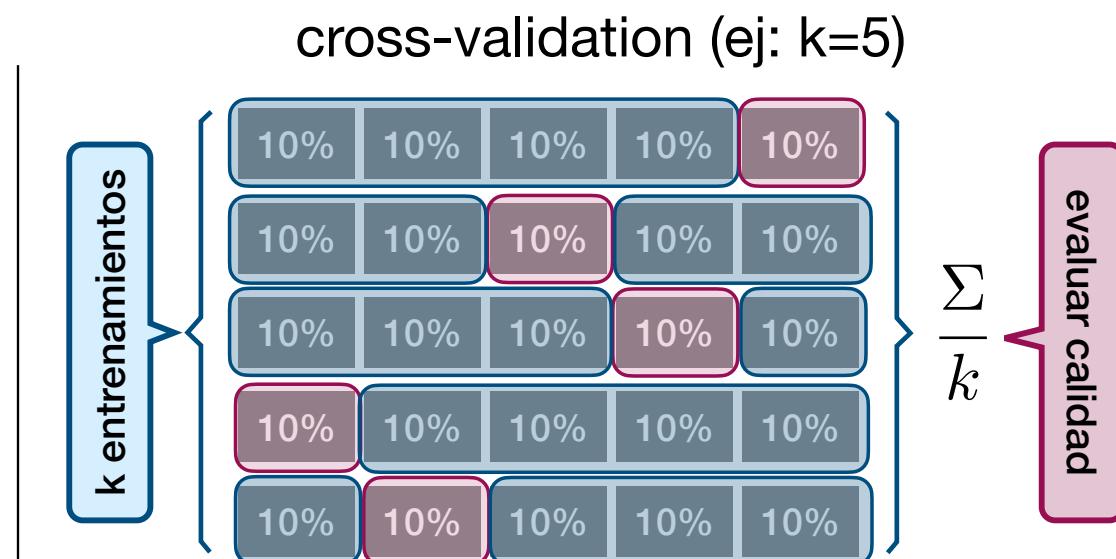
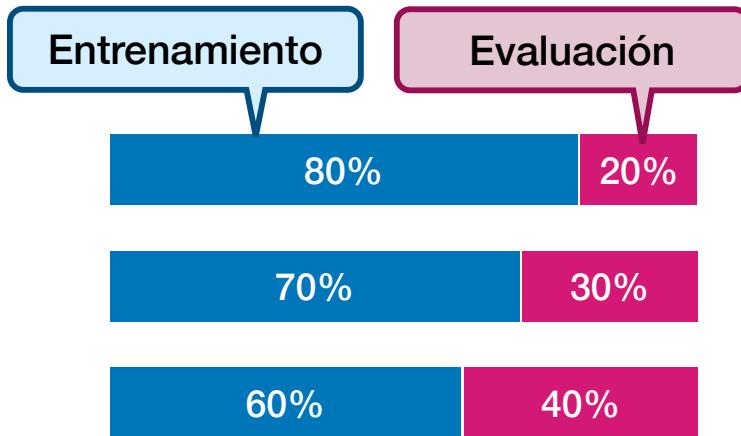
Para aplicar las métricas que guían el desarrollo del modelo, y dado que este se entrena a partir de un conjunto de datos, es necesario **particionar** dicho conjunto en diferentes subconjuntos, destinados a las **distintas etapas** del proceso: uno para el **aprendizaje (entrenamiento)** y otro para la **evaluación (prueba)**.

La elección de cómo seleccionar estas particiones y sus elementos tiene un alto impacto en el proceso de desarrollo. Aunque en general depende fuertemente de cada problema a resolver y el tipo de modelo, se destina un porcentaje mayor de los datos a la etapa de entrenamiento.

ML Supervisado - Partición de los datos durante el desarrollo

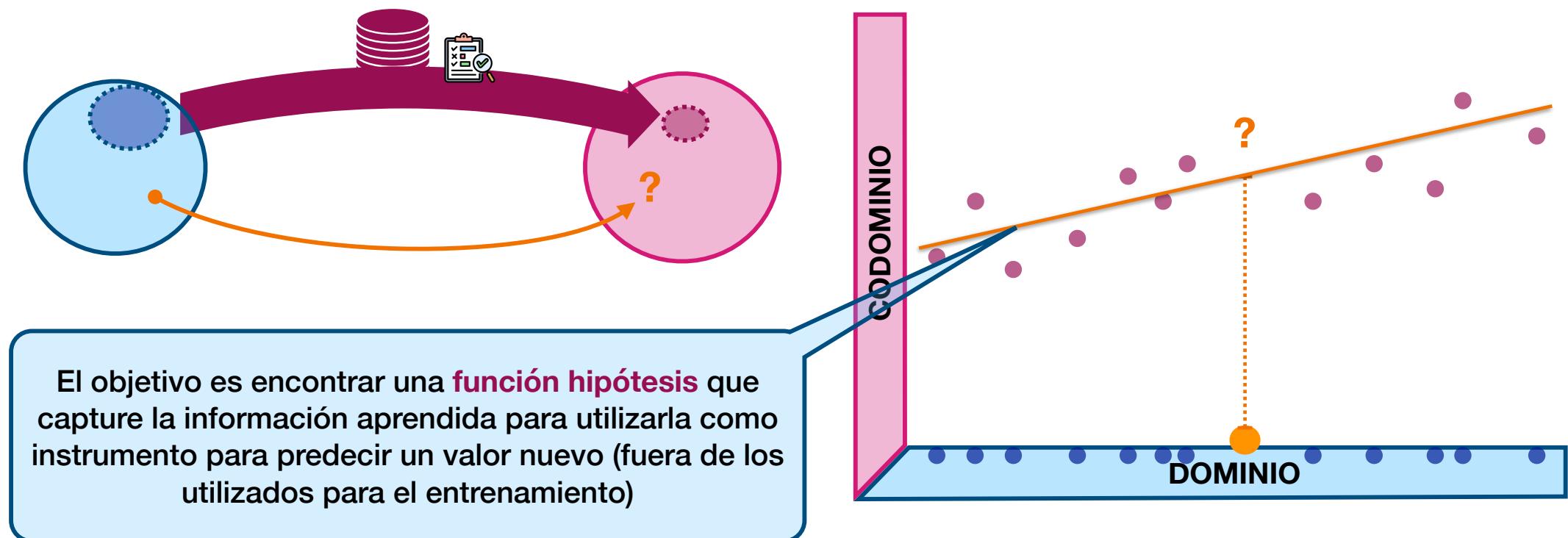
Una práctica común es dividir los datos en un 80 % para entrenamiento y un 20 % para evaluación, aunque también se utilizan otras proporciones como 70/30 o 60/40, dependiendo del tamaño y la calidad del conjunto de datos disponible.

Cuando se dispone de **pocos datos** o para fortalecer la evaluación, se utiliza una técnica llamada **validación cruzada (cross-validation)**. La variante más simple (**k-fold cross-validation**), propone dividir los datos en **k subconjuntos (folds)**. Se entrena el modelo **k** veces utilizando **k-1** folds para entrenamiento y el restante para prueba. Al final, se promedian los resultados.



ML Supervisado - Regresión (predicción) Lineal (RL)

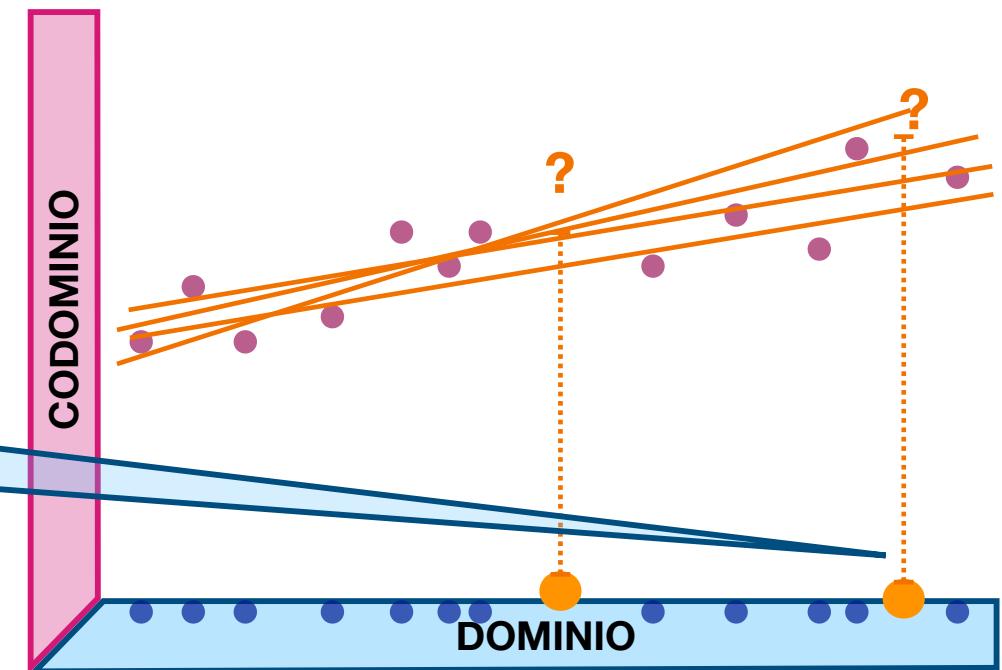
Comenzaremos el abordaje de técnicas de ML Supervisado explorando una técnica simple de predicción. Esto es, dado un sub-conjunto de datos que representan cierta información (entrada y resultado), nuestro objetivo es aprender de este sub-conjunto para poder predecir el resultado para cualquier otro dato de entrada.



ML Supervisado - Regresión Lineal (RL)

Pueden existir varias **funciones lineales** que capturen la información, lo que buscamos es la **más precisa**. Es decir, aquella que minimice el error entre los datos (puntos) del aprendizaje y la función.

La función a encontrar debe poder predecir de la mejor manera posible cualquier otro dato que no forme parte del entrenamiento



ML Supervisado - Regresión Lineal (RL)

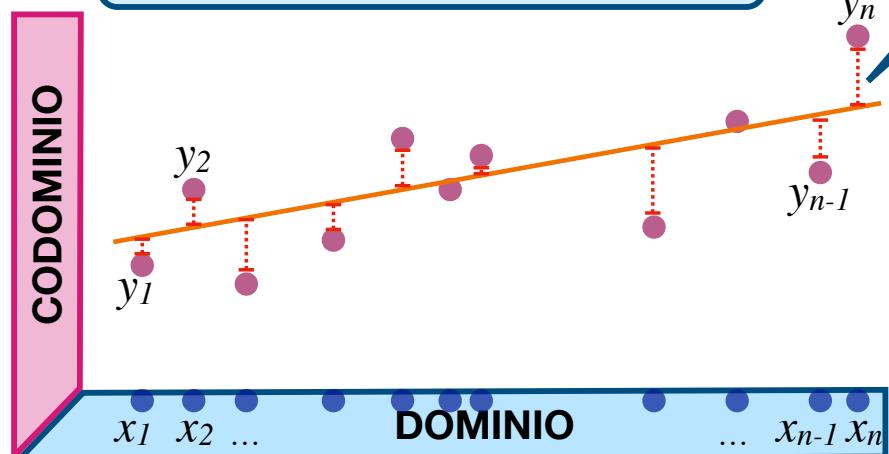
Una métrica posible para alcanzar el objetivo es el **Error Mínimo Cuadrado (MSE)** que computa **el promedio de los errores** entre los puntos de entrenamiento y la función hipótesis (h) a encontrar. Recordemos que la forma normal de una función lineal es:

$$h(x) = \theta_0 + \theta_1 x$$

θ_0 y θ_1 son los **parámetros** del modelo a aprender

Podemos calcular al diferencia como:
 $(y_i - h(x_i))$

$$\text{MSE} = \frac{\cancel{+} + \cancel{+} + \cancel{+} + \cancel{+} + \cancel{+} + \cancel{+}}{n}$$



Reemplazando h por su definición y tomando el cuadrado obtenemos :
 $(y_i - (\theta_0 + \theta_1 x_i))^2$

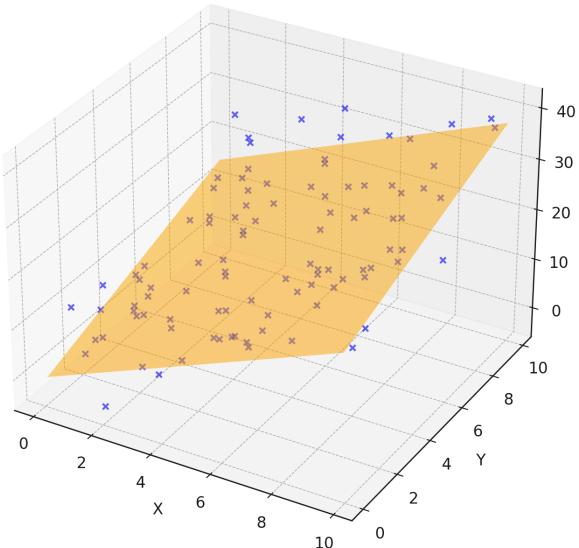
$$\frac{\sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2}{n}$$

ML Supervisado - RL - Generalización

Los modelos a aprender y predecir pueden tener más de una dimensión (características a predecir)

Ejemplo podrían estar en el espacio
(en caso de 2 características)

$$h(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



Para el caso general de m características

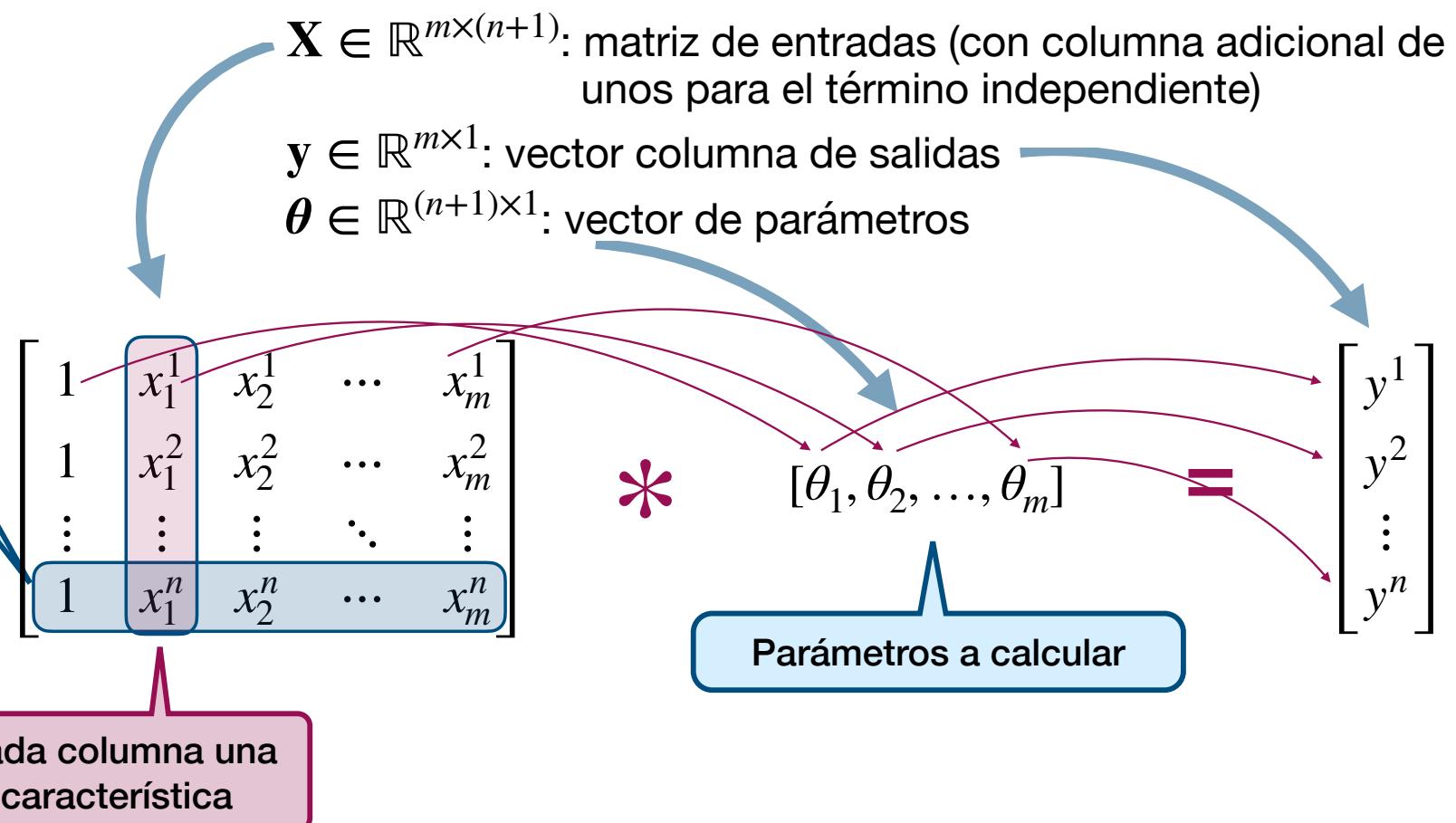
$$h(x_1, \dots, x_m) = \theta_0 + \theta_1 x_1 + \dots + \theta_m x_m$$

MSE

$$\frac{\sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_1^i + \dots + \theta_m x_m^i))^2}{n}$$

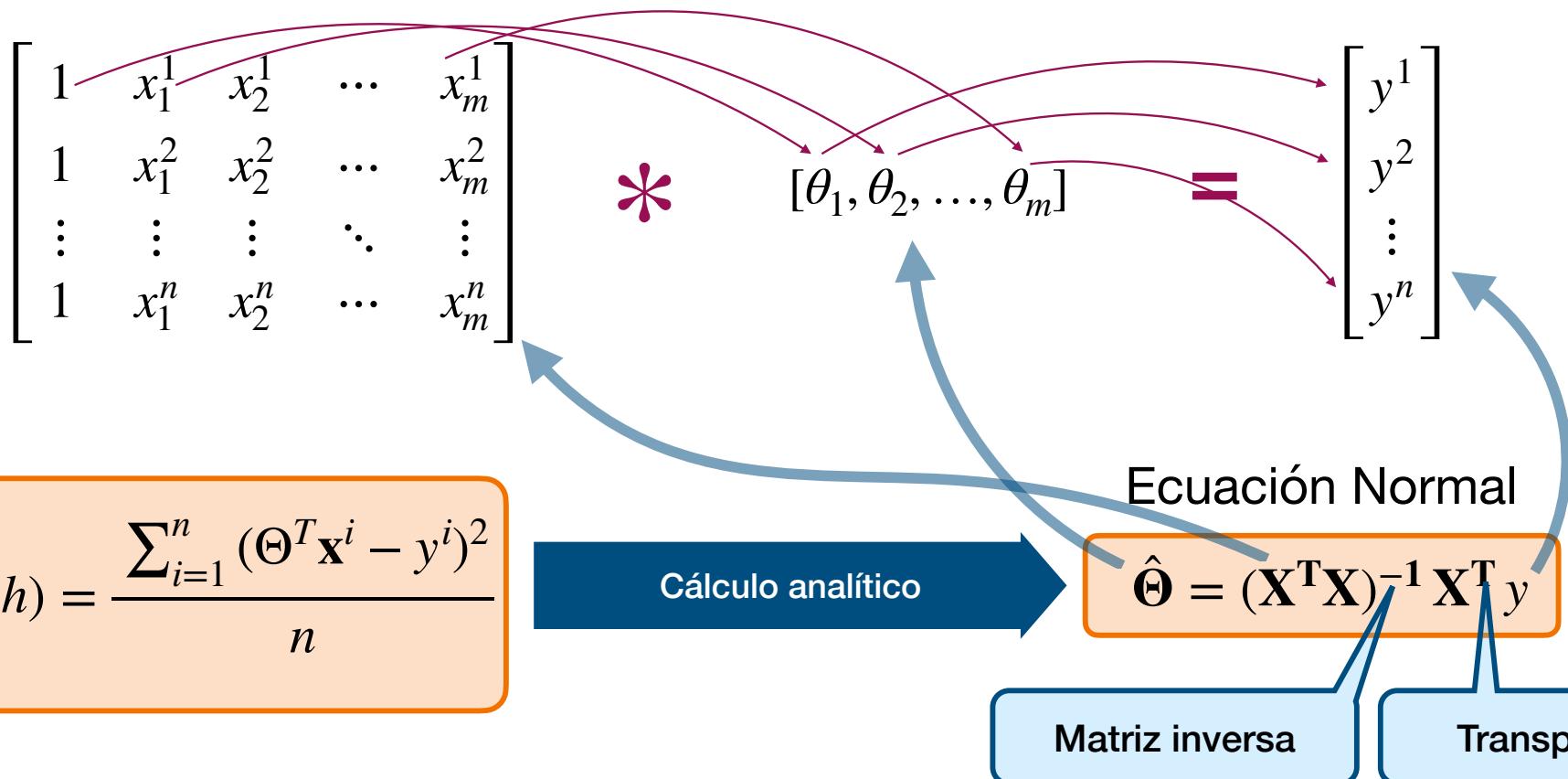
ML Supervisado - RL - Notación Matricial

Si tenemos un conjunto de datos de entrenamiento de n instancias de m características, anotamos:



ML Supervisado - RL - Ecuación Normal y MSE

Con esta representación matricial, utilizando la definición de MSE, podemos computar de manera analítica (propiedades de álgebra) el vector de parámetros que ajusta la función **hipótesis** a obtener.



ML Supervisado - RL - Ejemplo

$$\hat{\Theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y$$

x1	x2	y
1	2	10
2	1	12
3	4	20
4	3	22
5	5	30

$$X = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 3 & 4 \\ 1 & 4 & 3 \\ 1 & 5 & 5 \end{bmatrix}, \quad y = \begin{bmatrix} 10 \\ 12 \\ 20 \\ 22 \\ 30 \end{bmatrix}$$

Para calcular la **inversa** debe ser una matriz cuadrada y el **determinante** distinto de cero.

$$X^{-1} = \frac{1}{\det(X)} * \text{adj}(X)$$

DEBE coincidir la cantidad de **filas** con la cantidad de **columnas** de los multiplicandos

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{bmatrix} \quad \sum *$$

$$X^T X = \begin{bmatrix} 5 & 15 & 15 \\ 15 & 55 & 49 \\ 15 & 49 & 55 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} 1.2 & -0.1667 & -0.1667 \\ -0.1667 & 0.2778 & -0.2222 \\ -0.1667 & -0.2222 & 0.2778 \end{bmatrix}$$

$$\hat{\Theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} \quad h(x_1, x_2) = 2 + 3x_1 + 1x_2$$

ML Supervisado - RL - Costo computacional

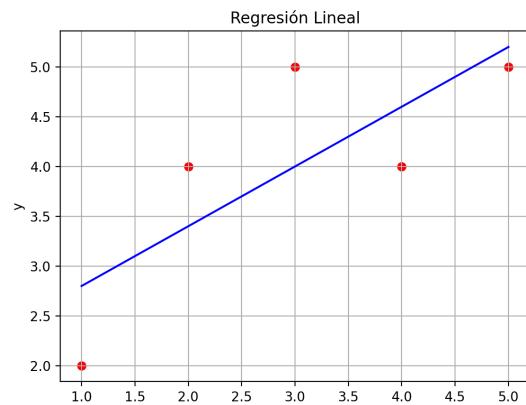
Como se puede observar, el costo computacional no es simple. Los algoritmos para calcular la ecuación normal para n instancias teniendo en cuenta m características es:

$$\mathcal{O}(nm^2 + m^3)$$

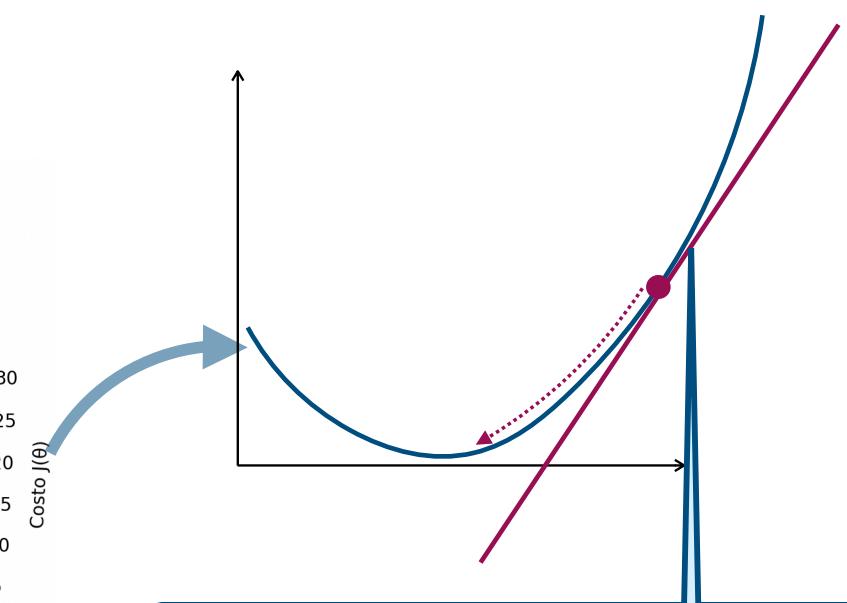
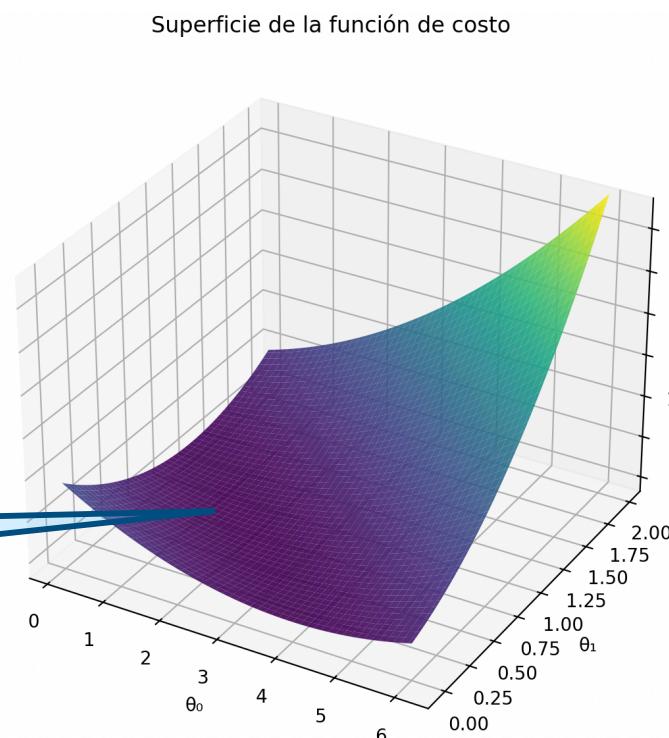
Recordemos que la precisión de nuestro modelo se encuentra altamente relacionado con la cantidad de datos de entrenamiento, por ejemplo, si tenemos 100.000 instancias (un ejemplo de muy simple), aún con pocas características, deja de ser una opción viable. Además, del tiempo, debemos mantener las matrices en memoria para poder operar con ellas.

ML Supervisado - RL - Cómputo con gradientes

Una alternativa es utilizar las propiedades de la función de costo para guiar la búsqueda de los parámetros.



Para regresión lineal, el espacio de pesos es una función convexa con un único mínimo

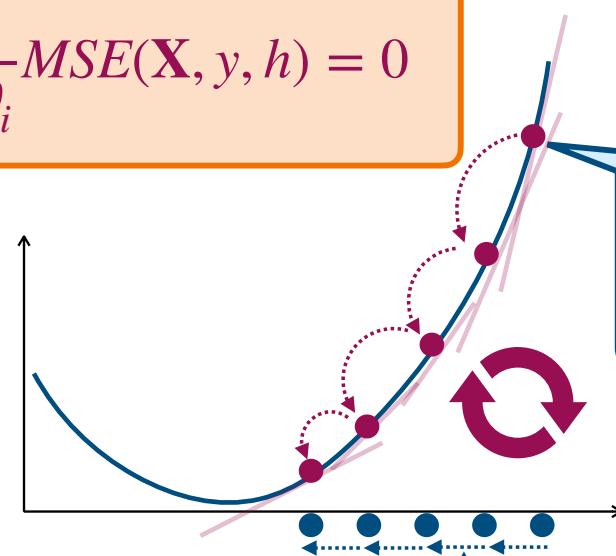


Podemos utilizar la derivada de la función de costo para encontrar el mínimo, es decir, los **parámetros** de nuestra función de hipótesis

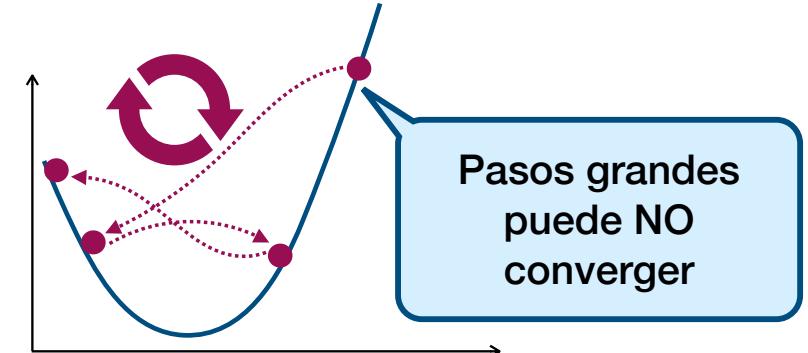
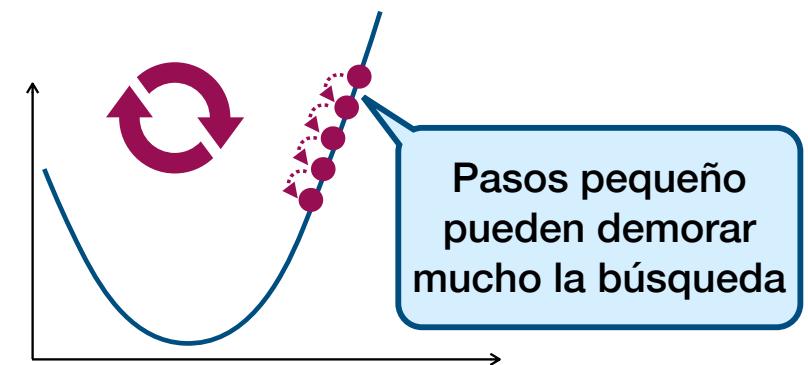
ML Supervisado - RL - Cómputo con gradientes

La idea es ir recorriendo iterativamente la derivada de la función de costo en búsqueda del mínimo. Esto es, los valores de los parámetros que mejor ajustan nuestra función de hipótesis, **cuando la derivada de la función de error es cero.**

$$\frac{\partial}{\partial \theta_i} MSE(\mathbf{X}, y, h) = 0$$



Vamos buscando de a **pasos** en búsqueda del mínimo
(similar “*hill climbing*” pero con mínimo en vez de máximo)



ML Supervisado - RL - Repaso de reglas de derivación

Constante	$\frac{\partial}{\partial x} k = 0$
Constante por función	$\frac{\partial}{\partial x} k f(x) = k \frac{\partial}{\partial x} f(x)$
Regla de la suma	$\frac{\partial}{\partial x} f(x) + g(x) = \frac{\partial}{\partial x} f(x) + \frac{\partial}{\partial x} g(x)$
Regla de la potencia	$\frac{\partial}{\partial x} x^k = k x^{k-1} \quad (r \neq 0)$
Regla de la cadena	$\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \frac{\partial}{\partial x} g(x)$

ML Supervisado - RL - Cómputo con gradientes

Comencemos con un modelo simple de una sola característica: $\theta_0 1 + \theta_1 x$

$$\frac{\partial}{\partial \theta_j} MSE(\mathbf{x}, y, h_{\theta}) = 0$$

$$\begin{aligned}\frac{\partial}{\partial \theta_j} MSE(x, y, h_{\theta}) &= \frac{\partial}{\partial \theta_j} (h_{\theta}(x) - y)^2 \\ &= \frac{\partial}{\partial \theta_j} ((\theta_0 1 + \theta_1 x) - y)^2 \\ &= 2((\theta_0 1 + \theta_1 x) - y) * \frac{\partial}{\partial \theta_j} ((\theta_0 1 + \theta_1 x) - y) \\ &= 2(h_{\theta}(x)) - y) * \frac{\partial}{\partial \theta_j} ((\theta_0 1 + \theta_1 x) - y)\end{aligned}$$

ML Supervisado - RL - Cómputo con gradientes

$$\frac{\partial}{\partial \theta_j} MSE(x, y, h_{\theta}) = 2(h_{\theta}(x)) - y) * \frac{\partial}{\partial \theta_j} ((\theta_0 1 + \theta_1 x) - y)$$

$$\frac{\partial}{\partial \theta_0} MSE(x, y, h_{\theta}) = 2(h_{\theta}(x)) - y) * 1$$

$$\frac{\partial}{\partial \theta_1} MSE(x, y, h_{\theta}) = 2(h_{\theta}(x)) - y) * x$$

Para movernos en la dirección contraria al gradiente debemos restar el gradiente a los pesos actuales:

- $\theta_0 = \theta_0 - \eta * 2(h_{\theta}(x)) - y)$
- $\theta_1 = \theta_1 - \eta * 2(h_{\theta}(x)) - y) * x$

η es la tasa de aprendizaje (learning rate), que permite regular la velocidad del aprendizaje

Notar que las actualizaciones son intuitivas, si $h_{\theta}(x) = \theta_0 + \theta_1 x > y$ debemos:

- Reducir θ_0
- Si x es positivo reducir θ_1 ; si x es negativo agrandar θ_1

ML Supervisado - RL - Multiples gradientes (características)

$$\nabla_{\theta} MSE(\mathbf{X}, y, h_{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} MSE(\mathbf{X}, y, h_{\theta}) \\ \frac{\partial}{\partial \theta_1} MSE(\mathbf{X}, y, h_{\theta}) \\ \dots \\ \frac{\partial}{\partial \theta_n} MSE(\mathbf{X}, y, h_{\theta}) \end{pmatrix} = \begin{pmatrix} 2 \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i) * 1 \\ 2 \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i) * x_{i,1} \\ \dots \\ 2 \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i) * x_{i,n} \end{pmatrix} = 2\mathbf{X}^T(\mathbf{X}\theta - y)$$

Para simplificar la lectura omitimos la división por n

Para movernos en la dirección contraria al gradiente:

$$\theta = \theta - \eta \nabla_{\theta} MSE(\mathbf{X}, y, h_{\theta}) = \theta - \eta 2\mathbf{X}^T(\mathbf{X}\theta - y)$$

Este método utiliza todo \mathbf{X} para entrenar, con los cuales con un conjunto grande de entrenamiento puede tardar mucho. Funciona bien en cuanto el número de características, es mucho más rápido que la ecuación normal.

ML Supervisado - RL - Estructura del Algoritmo

```
# Parámetros iniciales
theta = np.zeros((2, 1)) # θ₀ y θ₁
eta = 0.01                # tasa de aprendizaje
n_iter = 1000              # número de iteraciones

for iteration in range(n_iter):
    gradients = (1/m) * X_b.T @ (X_b @ theta - y) # derivadas parciales
    theta = theta - eta * gradients
```

Cantidad de instancias
(Datos de Entrenamiento)

ML Supervisado - RL - Cómputo estocástico con gradientes

Una desventaja del aprendizaje por gradientes, es que se utilizan **todos** los datos de entrenamiento para la búsqueda de los parámetros del modelo.

$$\theta = \theta - \eta \nabla_{\theta} MSE(\mathbf{X}, y, h_{\theta}) = \theta - \eta 2\mathbf{X}^T(\mathbf{X}\theta - y)$$

Una alternativa es el **aprendizaje estocástico por gradientes** utiliza un subconjunto elegido de forma aleatoria de los datos para la búsqueda.

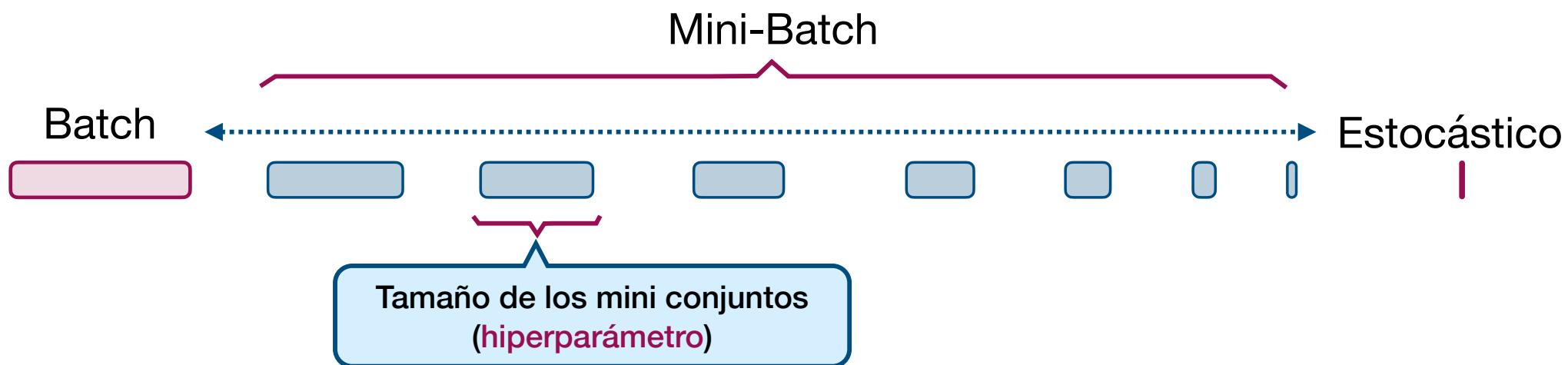
- La principal ventaja que ofrece el cómputo estocástico con gradientes es que **no se necesitan cargar todas las instancias** para realizar la búsqueda.
- La desventaja es que **puede tardar más tiempo** en converger a un buen mínimo local y no necesariamente siempre encontramos el valor óptimo para todo el conjunto de datos.
- Existen técnicas para ayudar la convergencia como **scheduling learning rate (Momentum)**, que ajusta la tasa de aprendizaje a medida que se acerca a un mínimo, similar a la técnica de **simulated annealing**.

ML Supervisado - RL - Cómputo estocástico con gradientes (**mini-batch**)

Una alternativa que pretende tomar lo mejor de las técnicas anteriores es la de **mini-batch**.

La idea es parecida que aprendizaje por gradientes, pero en vez de tomar una sola instancia se toman conjuntos chicos de forma aleatoria.

- Es menos errática que el aprendizaje estocástico.
- No utiliza todo el conjunto de entrenamiento.



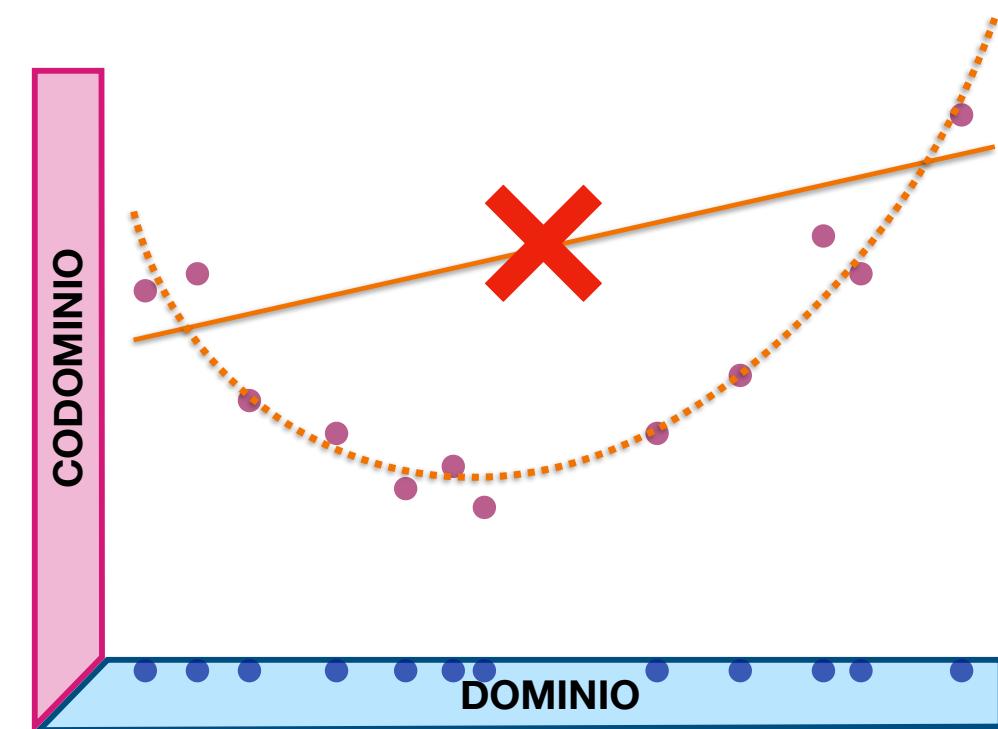
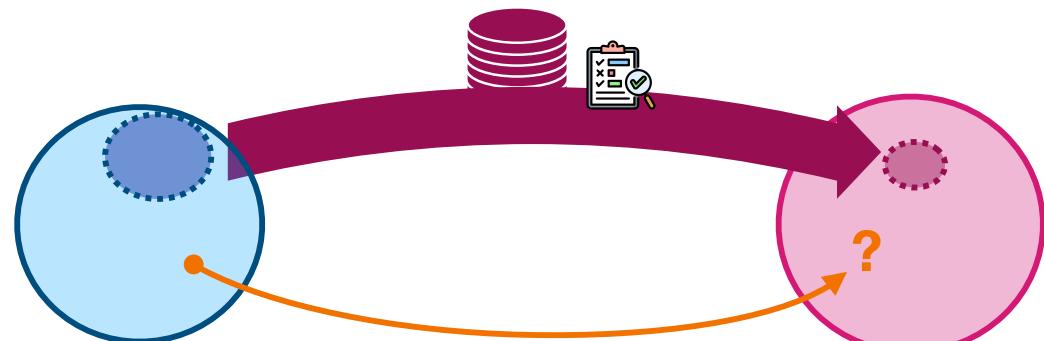
ML Supervisado - RL - Comparación



Convergencia	Inmediata	Gradual directa (poca varianza entre iteraciones)	Gradual (varianza media entre iteraciones)	Gradual (alta varianza entre iteraciones)
Hiperparámetros	Ninguno	Tasa de aprendizaje	Tasa de aprendizaje, tamaño de batch, etc.	Tasa de aprendizaje, etc.
Uso de datos de entrenamiento	exhaustivo	exhaustivo	reducido (ajustable)	mínimo
Comportamiento con muchas características	Lento	Rápido	Rápido	Rápido
Comportamiento con muchas instancias	Rápido	Lento	Rápido	Rápido

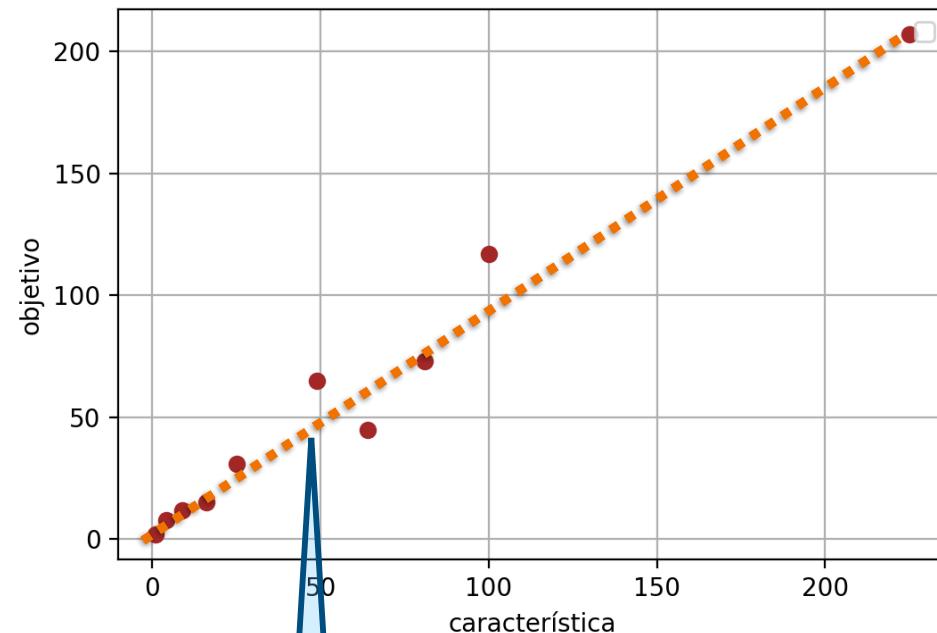
ML Supervisado - Regresión Polinomial

¿ Qué sucede si la función hipótesis a encontrar no tiene el comportamiento de una función lineal ?



ML Supervisado - Regresión Polinomial

Objetivo (y)	Característica (x)	x^2
65	-7	49
12	-3	9
8	2	4
2	1	1
8	2	4
15	4	16
31	5	25
45	7	49
73	9	81
117	10	100
207	15	225



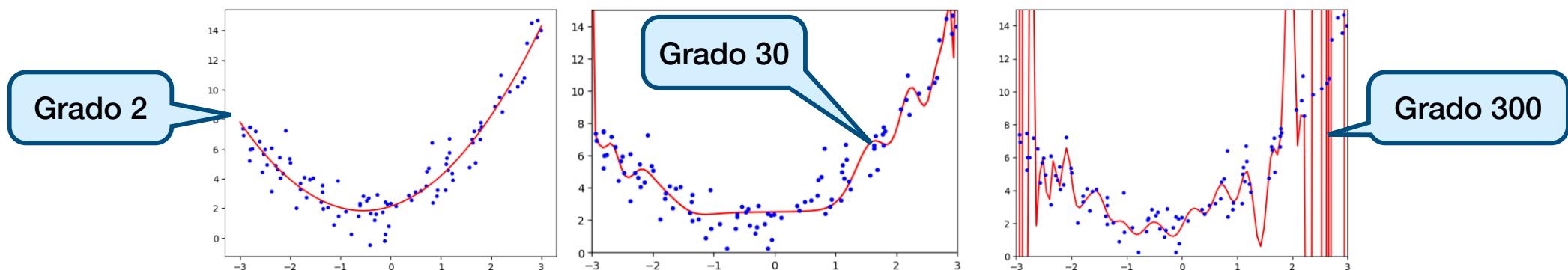
Teniendo en cuenta la característica agregada, el modelo parece ajustarse a una función hipótesis cuadrática. Ahora el modelo se ajusta por una función lineal.

ML Supervisado - Regresión Polinomial

La idea para poder ajustar modelos polinomiales es agregar más datos (características) con el valor de las características originales elevados al polinomio correspondiente. Luego buscamos, mediante **la técnica de regresión lineal el modelo** que mejor ajuste estos datos.

Si hay más de una característica propia del modelo a ajustar, **se agregan combinaciones** de las mismas. Por ejemplo, para 2 caract. y grado 3, agregan $a^2, a^3, b^2, b^3, ab, a^2b, ab^2$.

De manera general para n características y grado d , se agregan $\frac{(n + d)!}{d!n!}$ características adicionales. Se debe **tener en cuenta el problema la explosión de la cantidad de características** a incorporar.

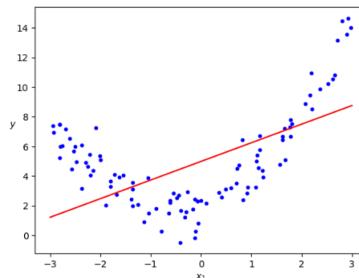


ML Supervisado - Underfitting y Overfitting

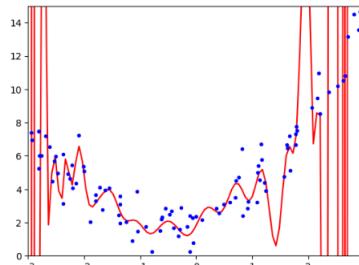
Como podemos apreciar, la elección de polinomio para el modelo a ajustar son parte de los **hiperparámetros**. La elección de los mismos puede derivar en la obtención de modelos ajustados serios problemas de predicción.

Se denomina **underfitting** cuando el modelo obtenido es demasiado simple y no logra aprender los patrones importantes de los datos.

Se denomina **overfitting** cuando el modelo obtenido aprende demasiados detalles y el ruido del conjunto de entrenamiento.



Underfitting: la función aprendida no refleja el comportamiento de los datos de entrenamiento

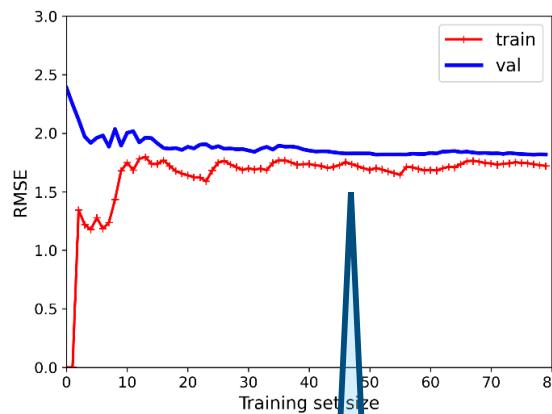


Overfitting: la función se ajusta muy bien a los datos de entrenamiento, pero no va a funcionar bien para nuevos datos

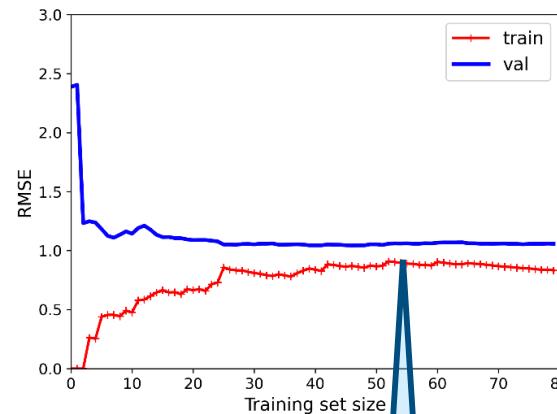
ML Supervisado - Curvas de aprendizaje

Una manera de evaluar la calidad de nuestra función hipótesis (modelo aprendido) es mediante la denominadas **curvas de aprendizaje**.

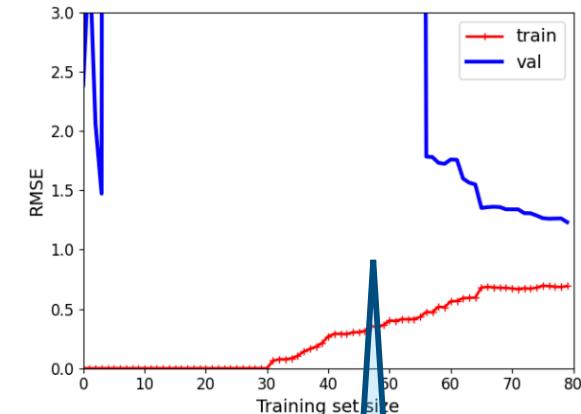
Estas **curvas representan gráficamente el error mínimo cuadrado (RMSE)** contrastándolo entre el **conjunto de datos de entrenamiento y el de validación**.



Si ambas curvas convergen en un error alto, estamos en presencia de underfitting. Generaliza de acuerdo al aprendizaje pero con un alto grado de error



Si ambas curvas convergen en un error moderado/bajo, estamos en ajustando correctamente el modelo



Si las curvas presentan mucha distancia entre ellas, estamos en presencia de overfitting. Se ajusta muy bien a los datos del entrenamiento pero no generaliza

ML Supervisado - SESGO vs VARIANZA

Un resultado teórico importante de la estadística y el aprendizaje automático es que el error de generalización de un modelo puede expresarse como la suma de tres tipos de errores muy diferentes:

Sesgo (Bias): Esta parte del error de generalización se debe a suposiciones incorrectas, como asumir que los datos son lineales cuando en realidad son cuadráticos. Un modelo con alto sesgo probablemente presentará underfitting en los datos de entrenamiento.

Varianza (Variance): Esta parte se debe a la excesiva sensibilidad del modelo ante pequeñas variaciones en los datos de entrenamiento. Un modelo con muchos grados de libertad seguramente tendrá alta varianza y, por lo tanto, overfitting.

Error irreducible (Irreducible error): Esta parte se debe al ruido inherente de los datos. La única manera de reducir este error es sanitizar los datos (por ejemplo, eliminando valores atípicos).

En general, aumentar la complejidad de un modelo suele incrementar su varianza y reducir su sesgo. Reducir la complejidad tiende a aumentar el sesgo y disminuir la varianza.

ML Supervisado - Regularización

Una buena manera de reducir el **overfitting** es mediante la **regularización** del modelo, es decir, imponerle restricciones. Una forma sencilla de regularizar un modelo polinómico es reducir el número de grados del polinomio. La regularización se logra normalmente restringiendo los pesos del modelo:

Ridge Regression: Penaliza la suma de los cuadrados de los pesos del modelo.

Fórmula de penalización: $cost(h_{\theta}) = MSE(h_{\theta}) + \alpha \sum_i \theta_i^2$

Los pesos grandes se reducen hacia cero, pero nunca se vuelven exactamente cero. No elimina variables irrelevantes, solo las reduce.

Lasso Regression (Least Absolute Shrinkage and Selection Operator) : Penaliza la suma de los valores absolutos de los pesos.

Fórmula de penalización: $cost(h_{\theta}) = MSE(h_{\theta}) + \alpha \sum_i |\theta_i|$

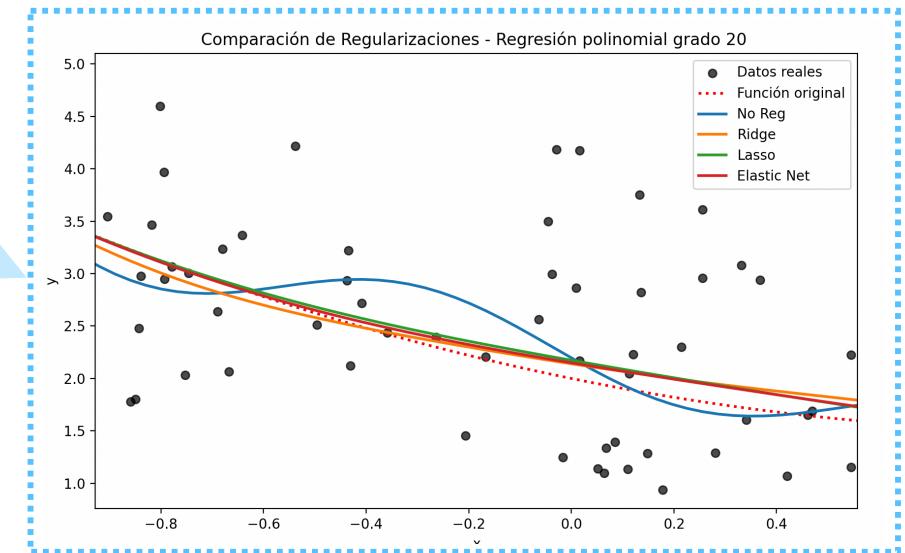
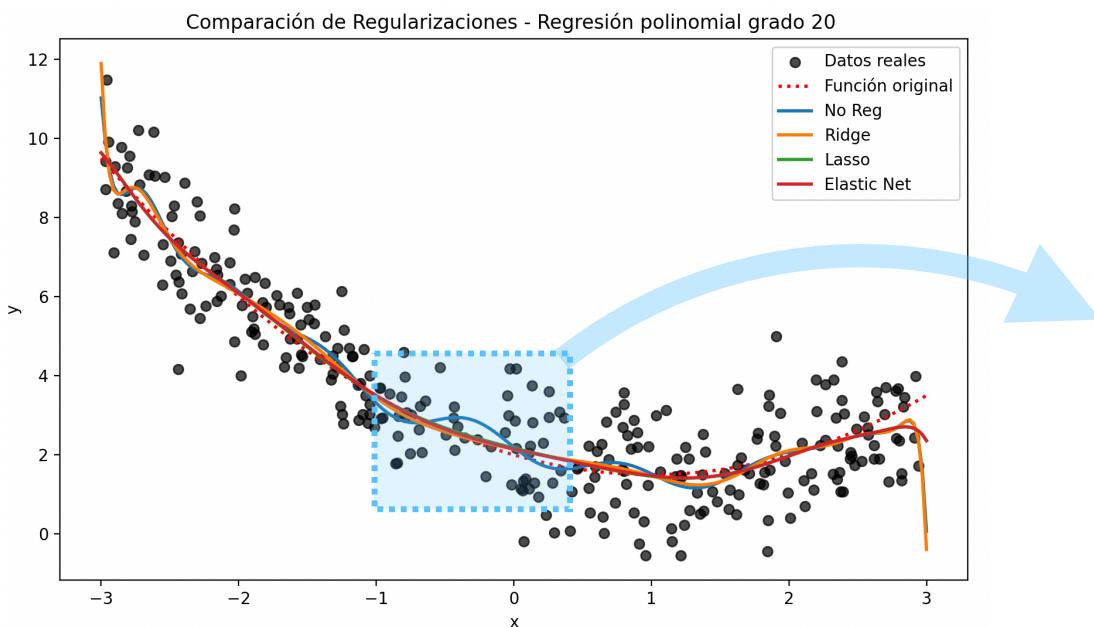
Tiende a reducir algunos pesos exactamente a cero, quita la variable asociada del modelo. Hace selección automática de variables y simplifica el modelo. Menos estable cuando hay muchas variables correlacionadas.

ML Supervisado - Regularización

Una técnica de **regularización** que combina **Ridge** y **Lasso** es **Elastic Net Regression**:

Fórmula de penalización: $cost(h_\theta) = MSE(h_\theta) + r\alpha \sum_{i=1}^n |\theta_i| + (1 - r)\alpha \sum_{i=1}^n \theta_i^2$

La primera componente de la fórmula (como Lasso) elimina variables irrelevantes. La segunda (como Ridge) maneja bien la multicolinealidad. Funciona bien cuando hay muchas variables correlacionadas y cuando queremos al mismo tiempo regularizar y seleccionar variables.



ML Supervisado - Regularización - Uso

- Casi siempre es preferible usar algún grado de regularización
- **Ridge** suele ser una buena opción por defecto
- Si se sospecha que pocas características son útiles, se prefiere **Lasso** o **Elastic Net**, ya que reducen los pesos de las características innecesarias a cero.
- **Elastic Net** suele ser preferible respecto Lasso. Lasso NO suele funcionar bien cuando el número de características es mayor al número de instancias de entrenamiento, o cuando hay una correlación fuerte entre algunas características
- De todos modos, como es típico en ML, en la práctica usualmente se pueden **probar distintas alternativas** experimentalmente, y elegir la que funcione mejor para el problema.

ML Supervisado - Finalización Temprana (Early Stopping)

- Cuando el entrenamiento es iterativo (Estocástico por Gradiente), el modelo que da mejores resultados en el conjunto de validación *puede* aparecer en **iteraciones intermedias** durante el entrenamiento.
- Esto se debe a que en las últimas iteraciones se suele hacer **overfitting** sobre el conjunto de entrenamiento.
- **Early stopping** consiste en guardar el modelo con mejores resultados contrastados con el conjunto de datos de validación.

