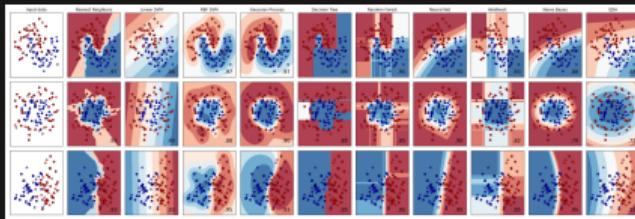


Modelos de Regresión

Aprendizaje Automatico



Marco Teran
EAFIT

2025

Contenido

- 1 Objetivos de aprendizaje
- 2 Introducción
- 3 Regresión Lineal: Fundamentos
- 4 Formalización Matemática
 - ¿Cómo encuentra los mejores parámetros?
 - Descenso del Gradiente
 - Regularización
- 5 Regresión Logística
 - Métricas de evaluación
- 6 Extensiones Multiclasificación
- 7 Problemas Prácticos
- 8 Comparación y Aplicaciones
- 9 Conclusiones

Objetivos de aprendizaj

Objetivos de aprendizaje

- **Dominar fundamentos matemáticos** de regresión lineal y logística
- **Comprender analogías de ingeniería**: control, señales y termodinámica
- **Implementar soluciones robustas**: ecuación normal, SVD y gradient descent
- **Aplicar regularización**: Ridge, LASSO y Elastic Net
- **Diagnosticar problemas comunes**: multicolinealidad, mal condicionamiento, separación perfecta
- **Optimizar para producción**: estabilidad numérica y eficiencia computacional

Filosofía central

Balancear rigurosidad matemática, interpretabilidad y eficiencia computacional

Regresión: Predicción y Clasificación

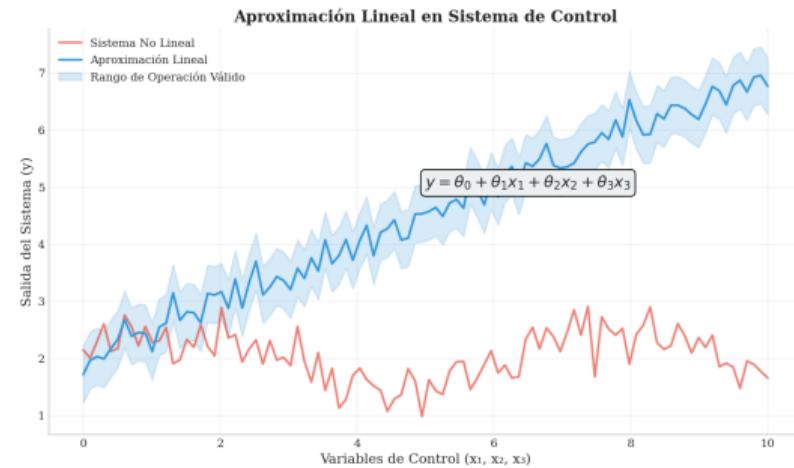
¿Qué aprenderemos hoy?

■ Regresión Lineal: Predecir valores numéricos

- Precio de una casa
- Temperatura mañana
- Ventas del próximo mes

■ Regresión Logística: Predecir categorías

- Email: ¿spam o no spam?
- Paciente: ¿sano o enfermo?
- Cliente: ¿comprará o no?



Objetivo

Entender cómo las máquinas aprenden a predecir desde los datos

Regresión Lineal: Fundamentos

Contexto histórico: De astronomía a machine learning

- **1805:** Legendre publica método de mínimos cuadrados
- **1795:** Gauss afirma uso previo para órbita de Ceres
- **Teorema de Gauss-Markov:** OLS es BLUE (*Best Linear Unbiased Estimator*)
- **Siglo XXI:** Base de sistemas inteligentes modernos

Contexto histórico: De astronomía a machine learning

¿Por qué sigue siendo relevante?

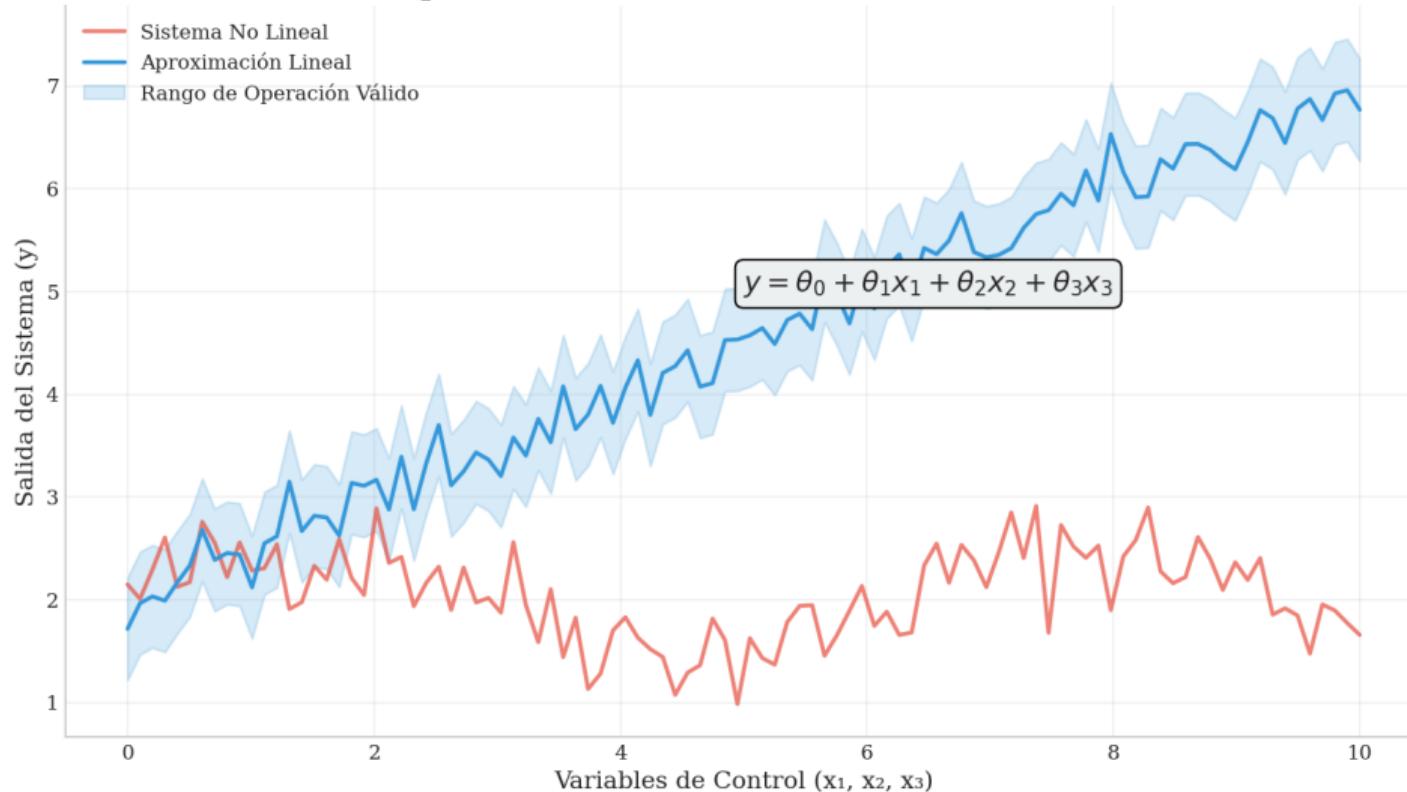
Convexidad global, solución cerrada, interpretabilidad directa

Aplicación moderna

Predicción de precios inmobiliarios: cada parámetro = impacto marginal

Analogía de sistemas de control

Aproximación Lineal en Sistema de Control



Analogía de sistemas de control

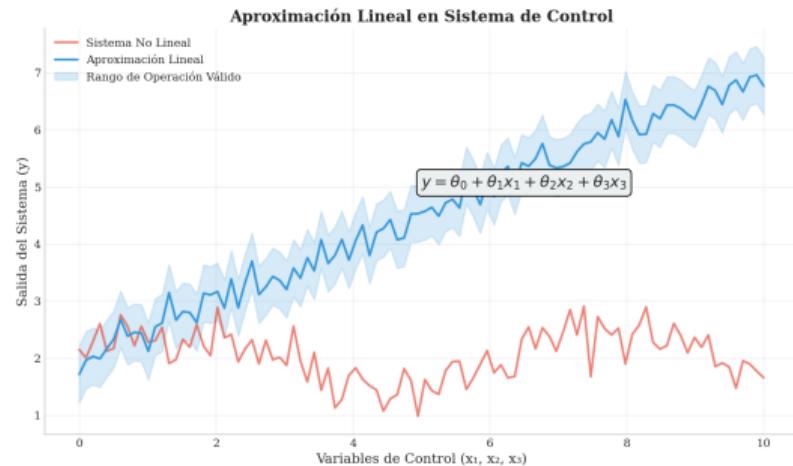
Reactor químico en estado estacionario:

- Temperatura salida: y
- Flujo refrigerante: x_1
- Presión sistema: x_2
- Concentración: x_3

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Interpretación:

- θ_i : sensibilidad del sistema
- Válido en rango operativo limitado
- Aproximación lineal local

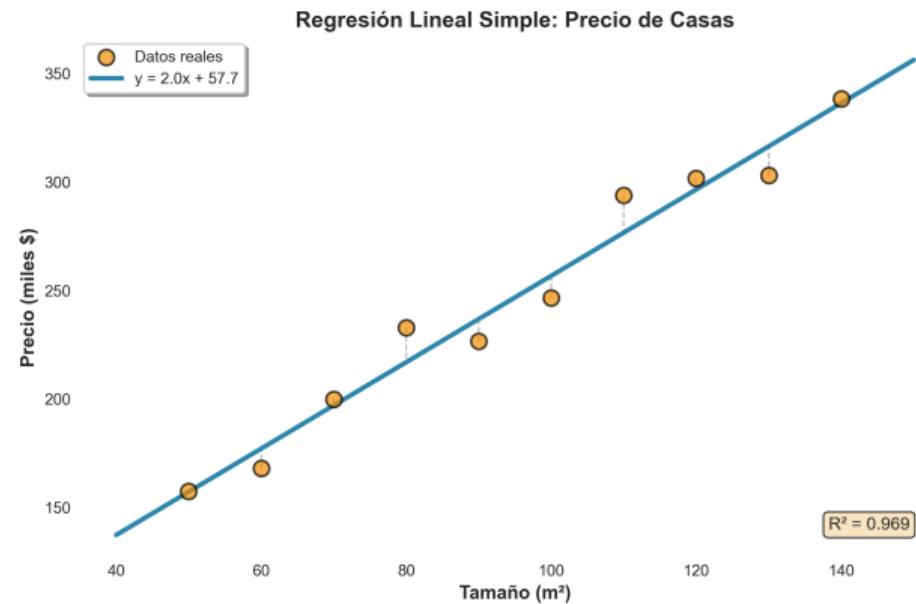


La idea básica

Queremos encontrar una línea que mejor se ajuste a nuestros datos

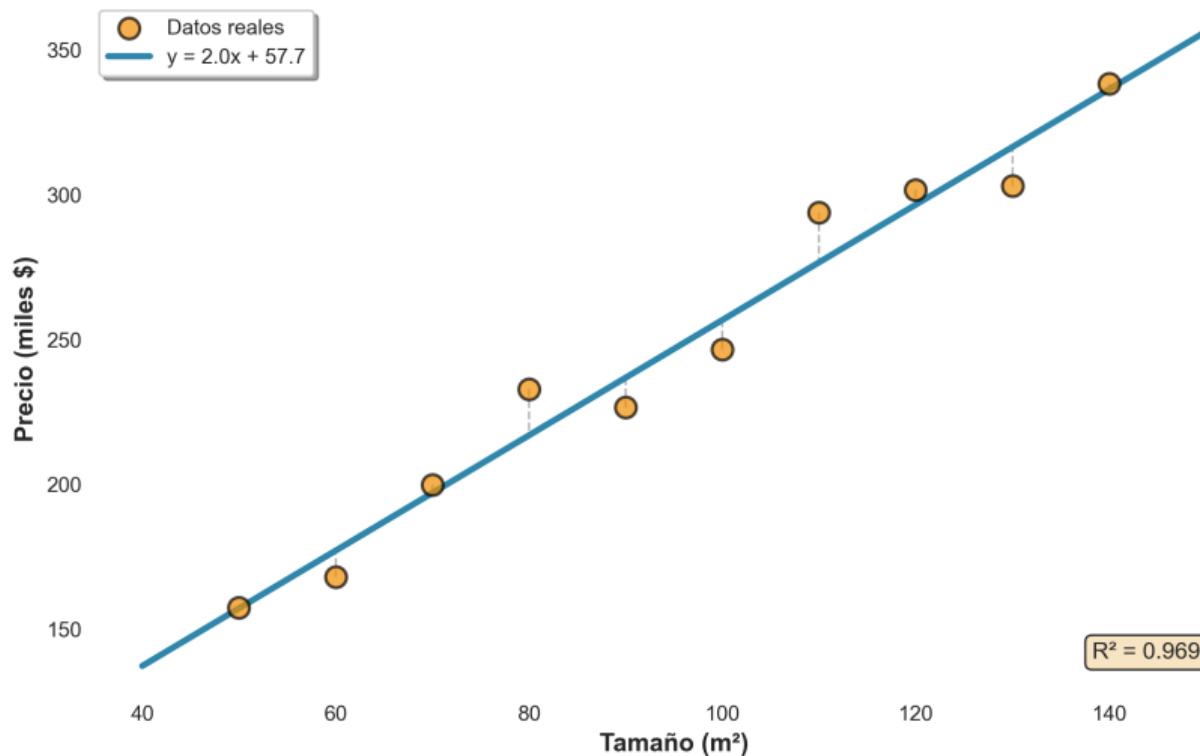
Ejemplo: Precio de casas

- Entrada (x): Tamaño en m^2
- Salida (y): Precio en miles \$
- Objetivo: Predecir precio



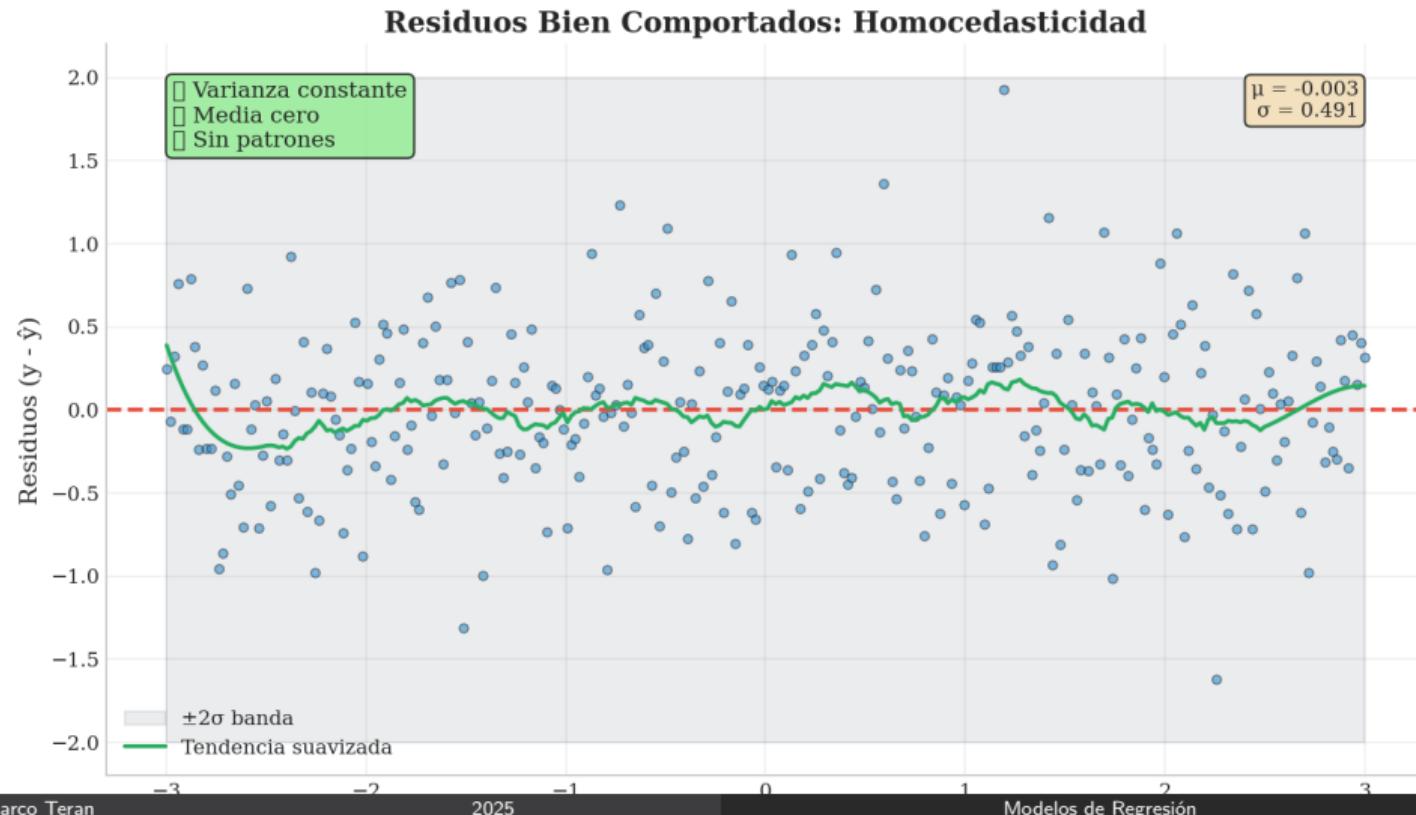
La idea básica

Regresión Lineal Simple: Precio de Casas



¿Cómo encuentra la mejor línea?

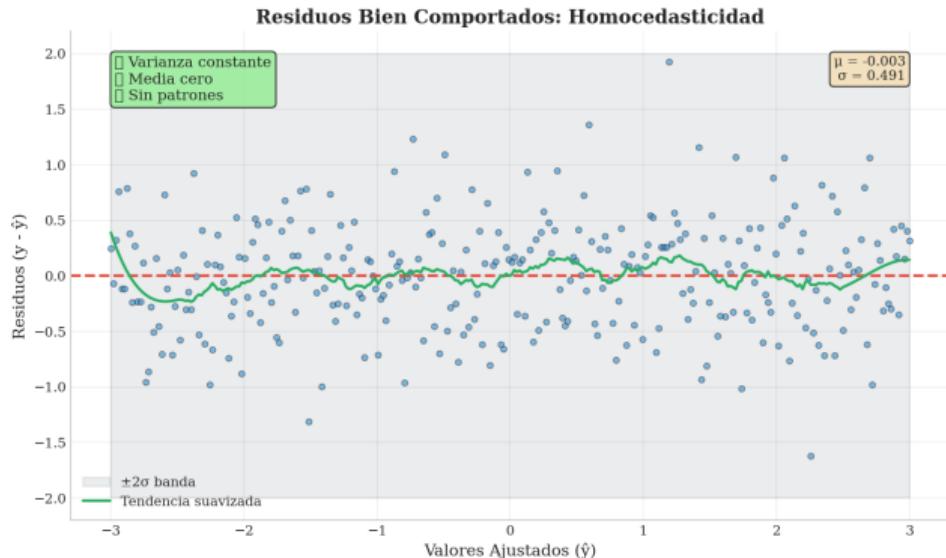
El problema: Tenemos muchos puntos, ¿cuál es la mejor línea?



¿Cómo encuentra la mejor línea?

Estrategia: Minimizar errores

- 1 Medir error de cada punto
- 2 Sumar todos los errores
- 3 Encontrar línea con menor error total



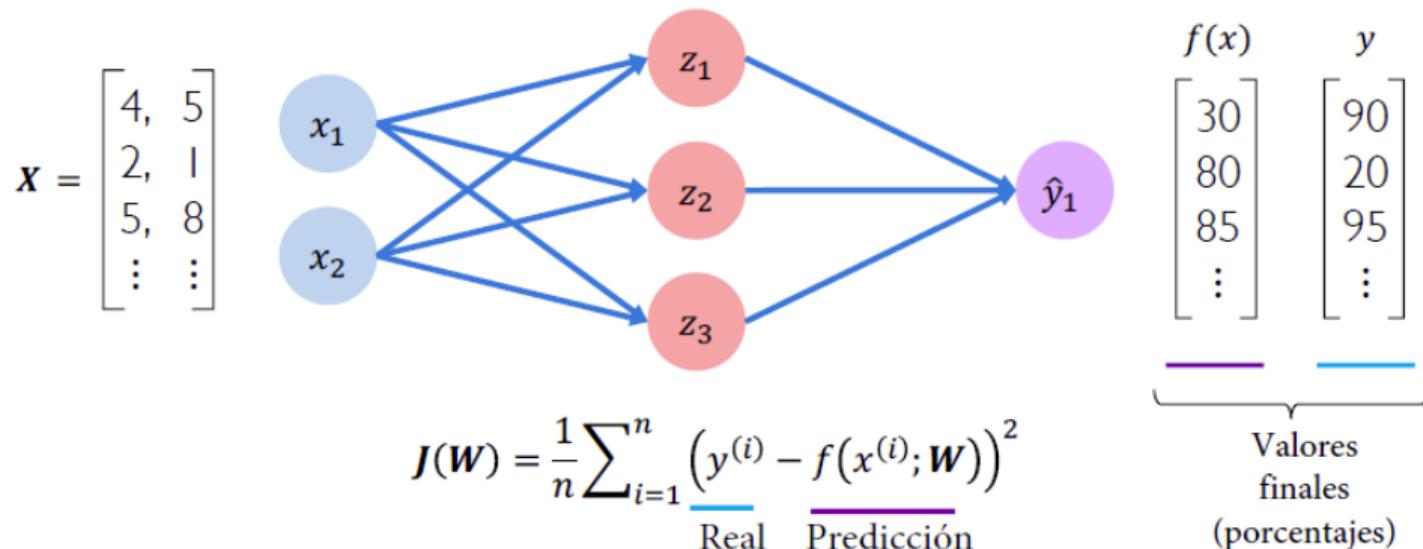
Error = Diferencia entre valor real y predicción

$$\text{Error} = y - \hat{y}$$

Perdida del Error cuadrático medio

Mean Squared Error Loss

La pérdida media de error al cuadrado se puede usar con modelos de regresión que producen números reales continuos



```
loss = tf.reduce_mean( tf.square(tf.subtract(model.y, model.pred)) )
```

La función de costo

¿Cómo medimos qué tan mal está nuestra predicción?

Función de Costo (MSE)

$$J = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

Optimización de la pérdida

Queremos encontrar los parámetros del modelo lineal que **minimicen la pérdida**

$$\theta^* = \operatorname{argmín}_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(f(x^{(i)}; \theta), y^{(i)})$$

$$\theta^* = \operatorname{argmín}_{\theta} J(\theta)$$

Optimización de la pérdida

Queremos encontrar los parámetros del modelo lineal que **minimicen la pérdida**

$$\theta^* = \operatorname{argmín}_{\theta} \frac{1}{n} \sum_{i=1}^n \ell(f(x^{(i)}; \theta), y^{(i)})$$

$$\theta^* = \operatorname{argmín}_{\theta} J(\theta)$$

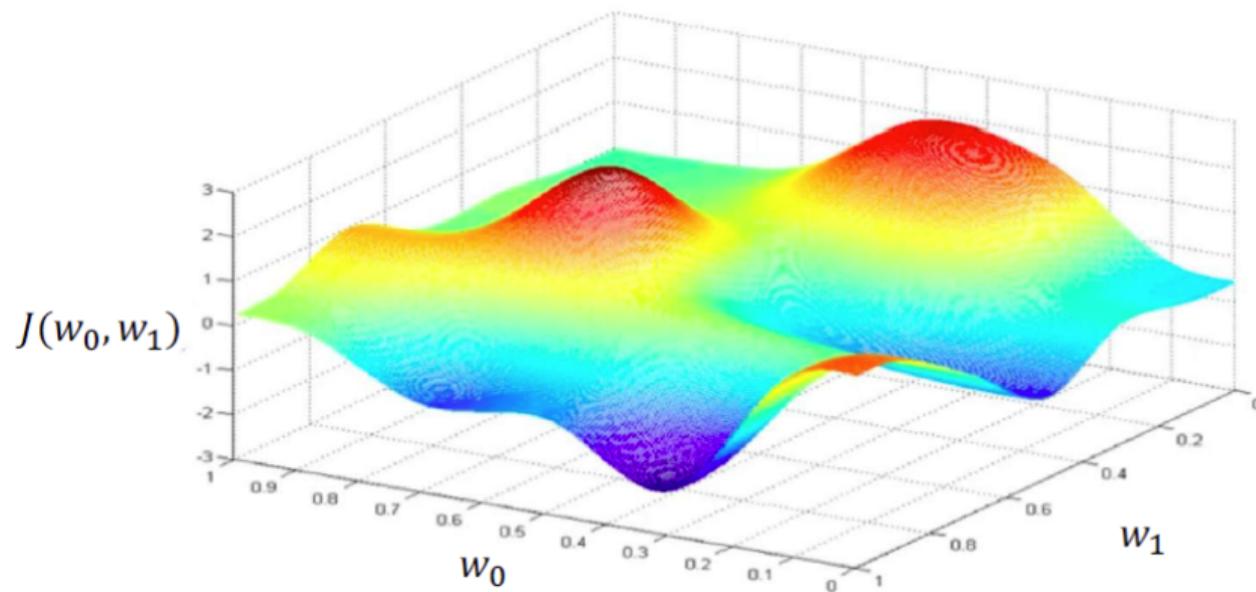
Recuerda:

$$\theta = \{\theta^{(0)}, \theta^{(1)}, \dots, \theta^{(p)}\}$$

donde cada $\theta^{(j)}$ es un coeficiente del modelo lineal.

Optimización de la pérdida

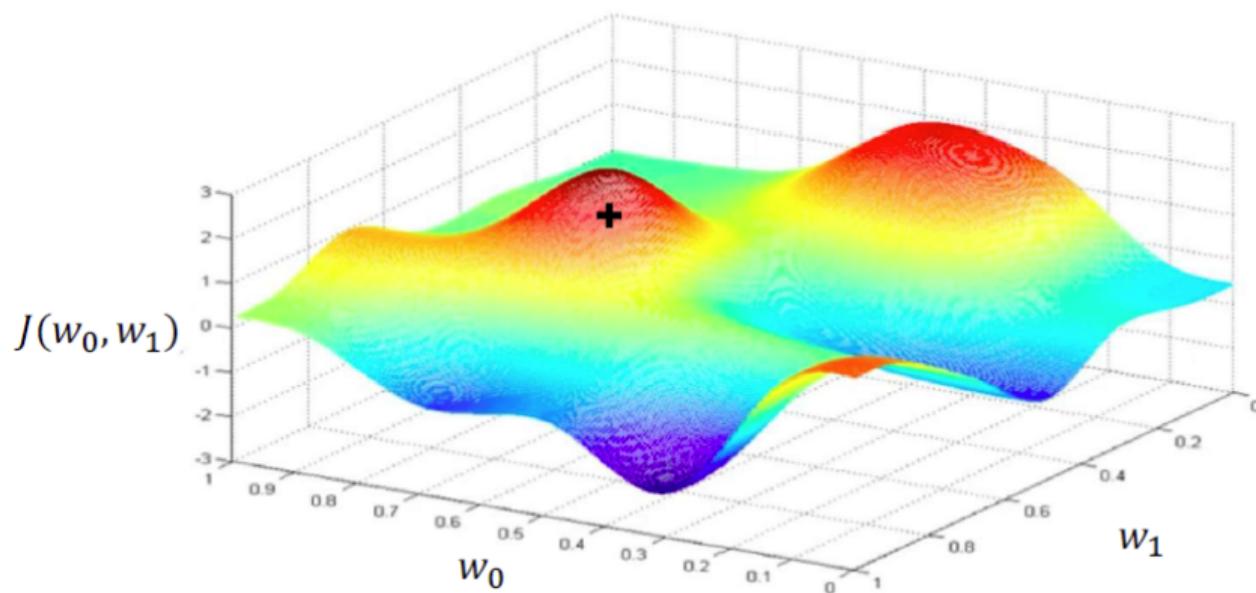
$$\theta^* = \operatorname{arg\!min}_{\theta} J(\theta)$$



Recuerda: la pérdida es una función de los parámetros θ .

Optimización de la perdida

Escoge al azar una inicial (w_0, w_1)



Formalización Matemática

Definición formal del modelo

Notación:

- Dataset: $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$
- Features: $x^{(i)} \in \mathbb{R}^n$, objetivo: $y^{(i)} \in \mathbb{R}$
- Hipótesis: $y^{(i)} = h_{\theta}(x) = \theta^T x = \sum_{j=0}^n \theta_j x_j$
- Convención: $x_0 = 1$ (término de sesgo)

Definición formal del modelo

Problema de optimización:

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^{n+1}} J(\theta)$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Formulación matricial y ecuación normal

Notación matricial:

- Matriz diseño: $X \in \mathbb{R}^{m \times (n+1)}$
- Vector objetivo: $y \in \mathbb{R}^m$
- Predicciones: $\hat{y} = X\theta$

Formulación matricial y ecuación normal

$$J(\theta) = \frac{1}{2m} \|X\theta - y\|_2^2$$

Ecuación normal:

$$\nabla_{\theta} J = 0 \Rightarrow X^T X \theta = X^T y$$

Solución analítica

Si $X^T X$ es invertible: $\theta^* = (X^T X)^{-1} X^T y$

Condiciones de invertibilidad

Para que $X^T X$ sea invertible:

- **Necesario:** $m \geq n + 1$ (más datos que parámetros)
- **Suficiente:** columnas de X linealmente independientes

Condiciones de invertibilidad

Casos problemáticos

- Multicolinealidad perfecta
- Datos insuficientes: $m < n + 1$
- Features constantes (excepto intercept)

Condiciones de invertibilidad

Solución: Descomposición SVD

$$X = U\Sigma V^T$$

$$X^+ = V\Sigma^+U^T \quad (\text{pseudoinversa})$$

Ejemplo numérico simple

Datos de entrenamiento (3 casas):

Casa	Tamaño (m ²)	Precio (miles \$)
1	50	150
2	80	200
3	110	250

Ejemplo numérico simple

Datos de entrenamiento (3 casas):

Casa	Tamaño (m ²)	Precio (miles \$)
1	50	150
2	80	200
3	110	250

Modelo: precio = $\theta_0 + \theta_1 \cdot \text{tamaño}$

Si probamos: $\theta_0 = 100$, $\theta_1 = 1.5$

Ejemplo numérico simple

Datos de entrenamiento (3 casas):

Casa	Tamaño (m ²)	Precio (miles \$)
1	50	150
2	80	200
3	110	250

Modelo: precio = $\theta_0 + \theta_1 \cdot \text{tamaño}$

Si probamos: $\theta_0 = 100$, $\theta_1 = 1.5$

- **Casa 1:** Predicción = $100 + 1.5(50) = 175$ (Real: 150) Error = 25
- **Casa 2:** Predicción = $100 + 1.5(80) = 220$ (Real: 200) Error = 20
- **Casa 3:** Predicción = $100 + 1.5(110) = 265$ (Real: 250) Error = 15

Ejemplo numérico simple

Modelo: precio = $\theta_0 + \theta_1 \cdot \text{tamaño}$

Si probamos: $\theta_0 = 100$, $\theta_1 = 1.5$

- Casa 1: Predicción = $100 + 1.5(50) = 175$ (Real: 150) Error = 25
- Casa 2: Predicción = $100 + 1.5(80) = 220$ (Real: 200) Error = 20
- Casa 3: Predicción = $100 + 1.5(110) = 265$ (Real: 250) Error = 15

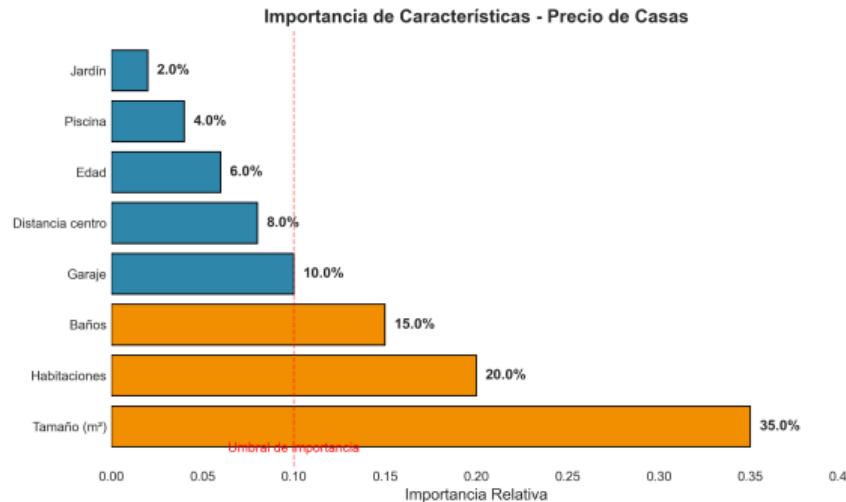
$$\text{MSE} = (25^2 + 20^2 + 15^2)/3 = 416.67$$

Múltiples características

En la vida real, usamos múltiples variables

Ejemplo: Precio de casa depende de:

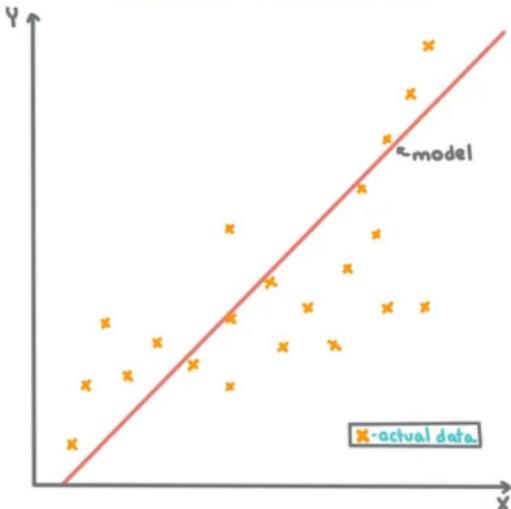
- x_1 : Tamaño (m^2)
- x_2 : Número de habitaciones
- x_3 : Edad de la casa
- x_4 : Distancia al centro



Modelo con múltiples características

$$\text{precio} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

LINEAR REGRESSION



HOW IT WORKS

establishes a relationship between X and Y.

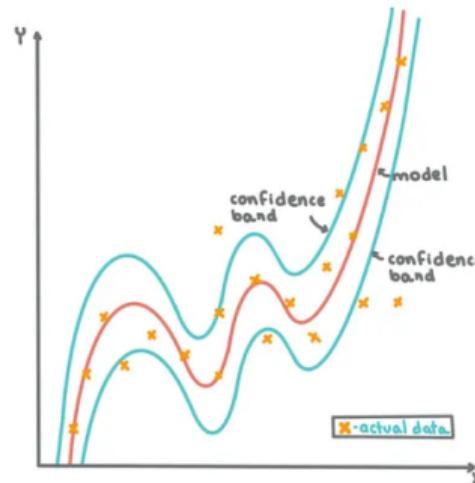
EXAMPLE

$$Y = b + mX$$

GOAL

optimize the slope (m) to reduce loss

POLYNOMIAL REGRESSION



HOW IT WORKS

type of linear regression where Y is modeled as an n^{th} degree polynomial of X.

EXAMPLE

$$Y = b + m_1X^1 + m_2X^2 + \dots + m_nX^n$$

GOAL

optimize the coefficients $[m_1, m_2, \dots, m_n]$ to reduce loss.

@DASANI_DECODED

**¿Cómo encuentra los mejores
parámetros?**

¿Cómo encuentra los mejores parámetros?

Dos métodos principales:

1. Ecuación Normal

- Solución matemática directa
- Un solo cálculo
- Rápido para pocos datos
- Puede fallar con muchas variables

2. Descenso del Gradiente

- Mejora iterativa
- Como bajar una montaña
- Funciona con muchos datos
- Necesita más tiempo

Analogía

Ecuación Normal = GPS (te da la ruta directa)

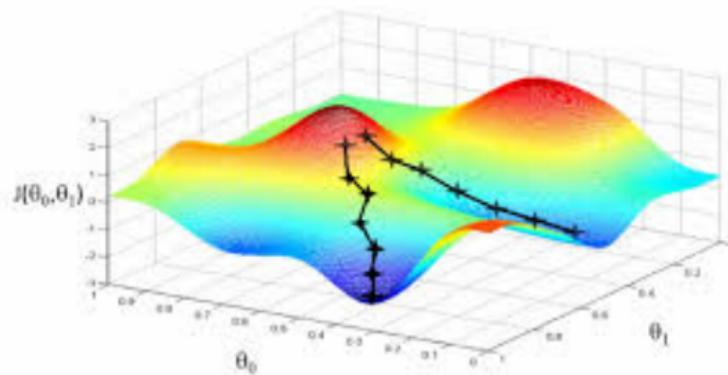
Gradiente = Seguir la pendiente más empinada cuesta abajo

Descenso del Gradiente

¿Qué es el Descenso del Gradiente?

Definición

El **descenso del gradiente** es un *algoritmo iterativo de optimización* utilizado para encontrar el mínimo de una función. Se basa en mover los parámetros θ en la dirección opuesta al gradiente de la función de error, ya que el gradiente indica la dirección de mayor incremento.



Descenso del Gradiente - Paso a paso

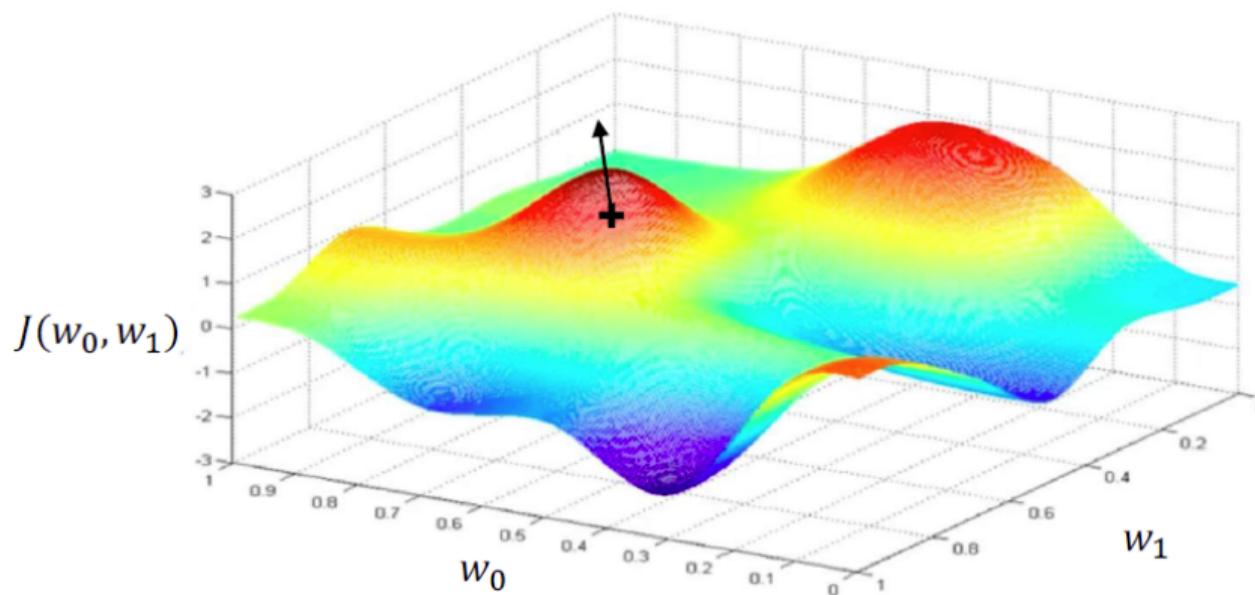
Algoritmo iterativo para encontrar el mínimo:

- 1 Inicializar:** Empezar con valores aleatorios para θ
- 2 Calcular error:** Evaluar la función de costo
- 3 Calcular dirección:** Obtener el gradiente $\nabla J(\theta)$
- 4 Dar un paso:** Actualizar θ en sentido opuesto al gradiente
- 5 Repetir:** Hasta alcanzar el mínimo (convergencia)

Optimización de la pérdida

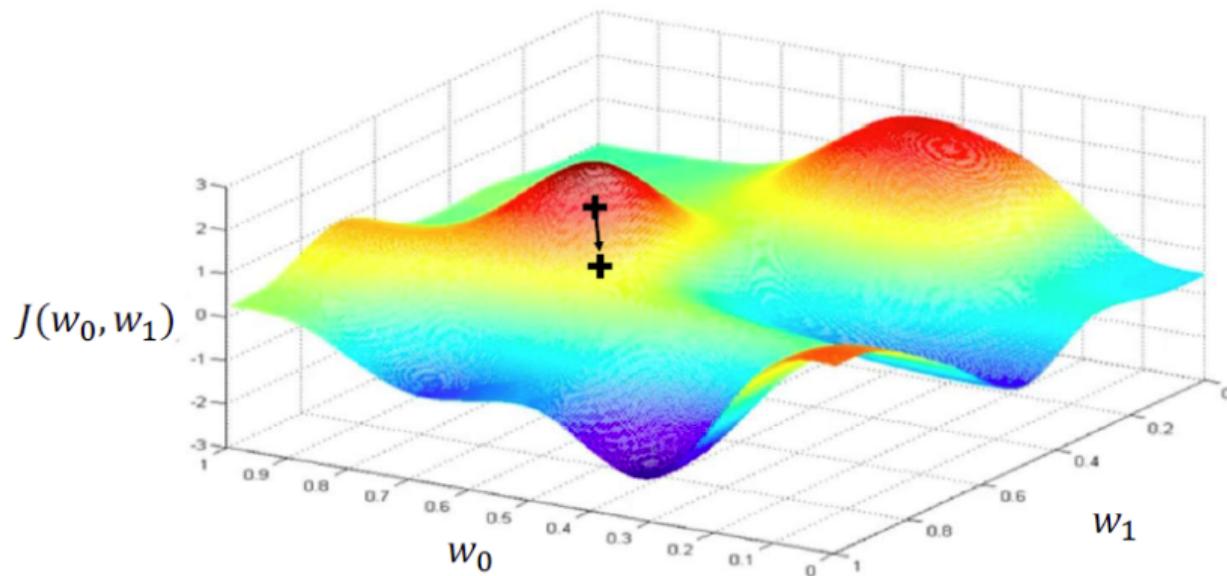
Cálculo del gradiente:

$$\frac{\partial J(\theta)}{\partial \theta}$$



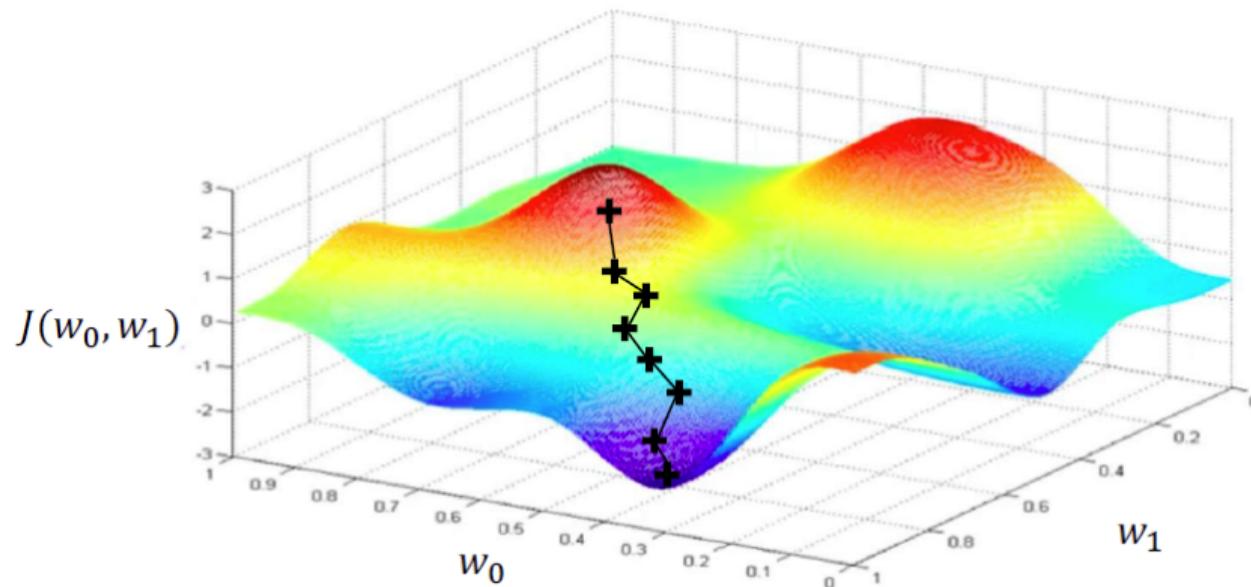
Optimización de la perdida

De un pequeño paso en dirección opuesta al gradiente



Gradiente descendente

Repita hasta la convergencia



Gradiente descendente

Algoritmo:

- Iniciar los parámetros al azar $\theta \sim \mathcal{N}(0, \sigma^2)$

```
 weights = tf.random_normal(shape, stddev=sigma)
```

- Bucle hasta la convergencia:

- Calcular el gradiente $\frac{\partial J(\theta)}{\partial \theta}$

```
 grads = tf.gradients(ys=loss, xs=weights)
```

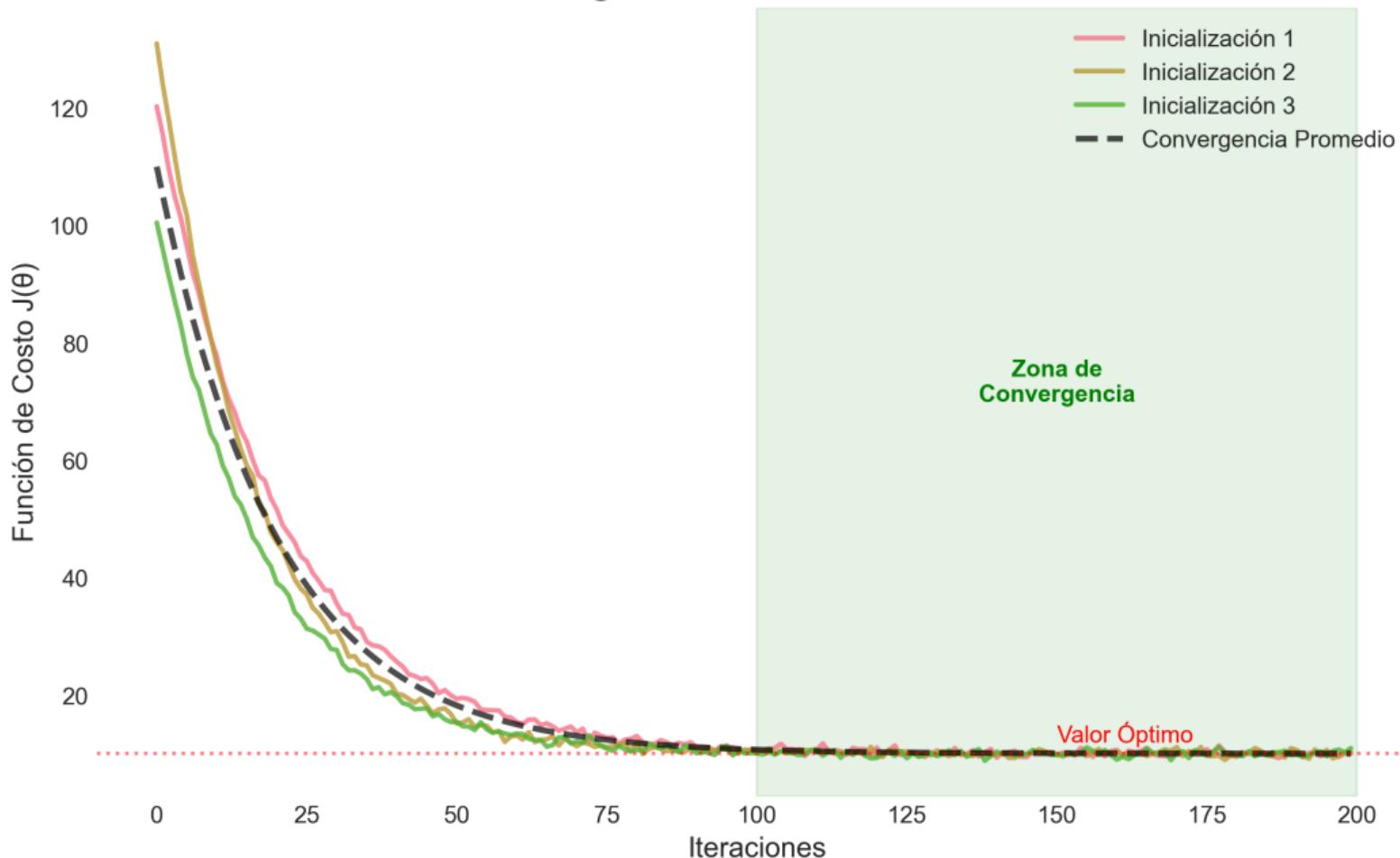
- Actualizar los parámetros:

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

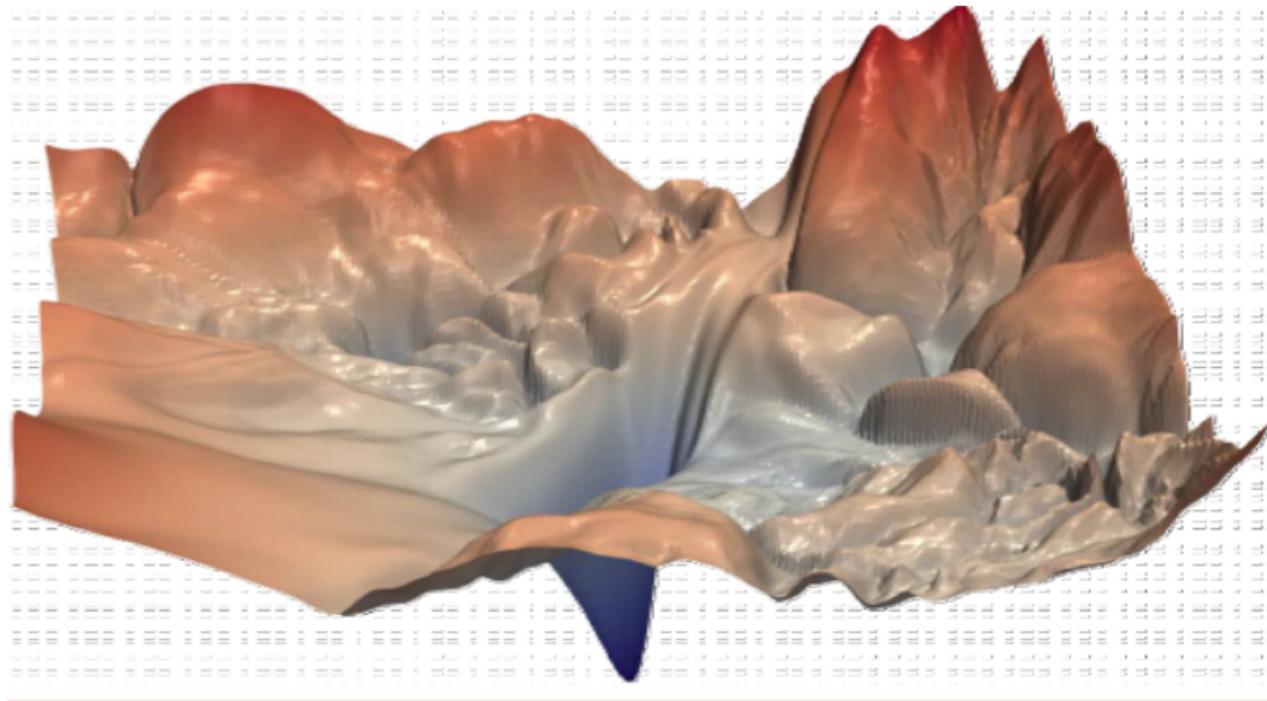
```
 weights_new = weights.assign(weights - lr * grads)
```

- Devolver los parámetros óptimos θ^*

Convergencia del Gradient Descent



El entrenamiento de redes neuronales es complejo



"Visualizando el paisaje de pérdida de las redes neuronales". Diciembre de 2017.

Las funciones de pérdida pueden ser difíciles de optimizar

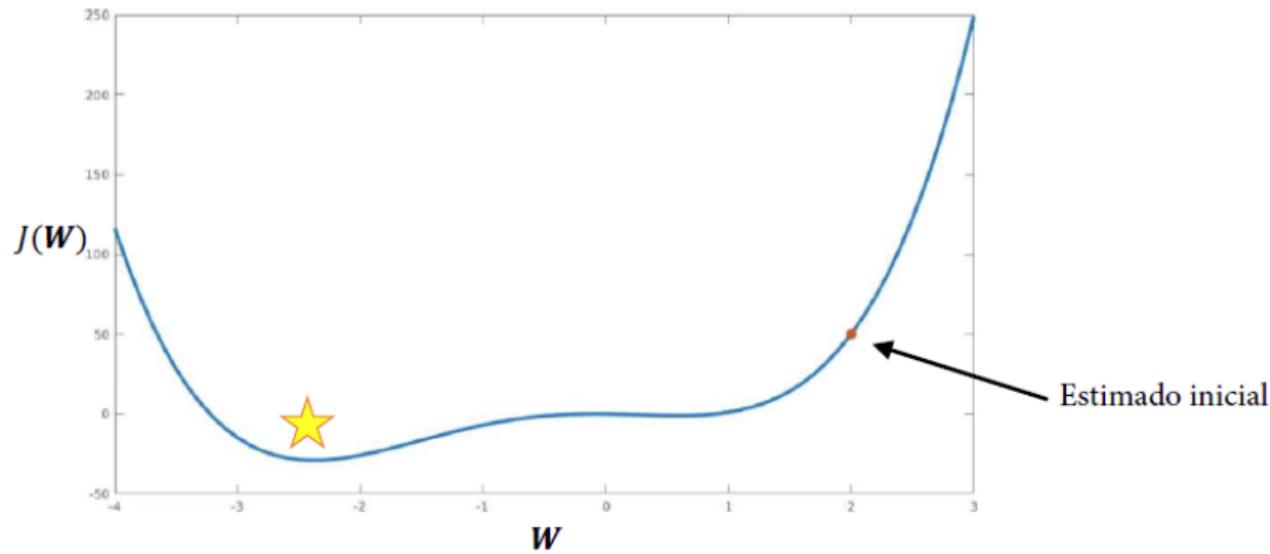
Recuerda: La optimización se realiza mediante **descenso del gradiente**:

$$\theta \leftarrow \theta - \eta \frac{\partial J(\theta)}{\partial \theta}$$

Pregunta clave: ¿Cómo elegir la *tasa de aprendizaje* (η)?

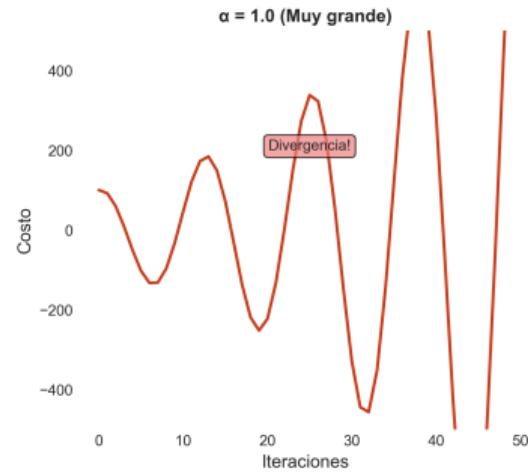
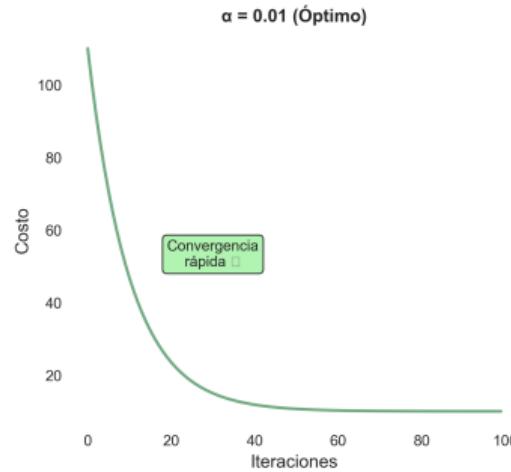
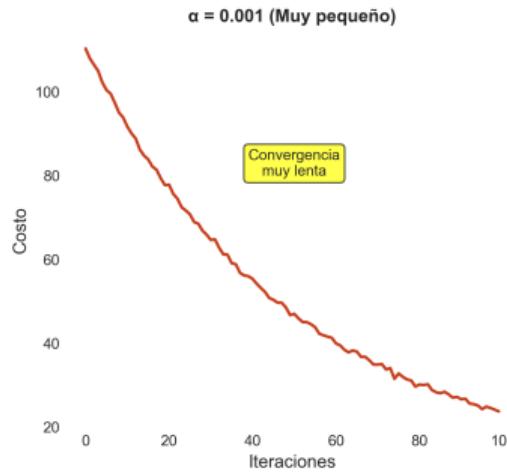
Ajuste de la tasa de aprendizaje

Recuerde: Una pequeña tasa de aprendizaje converge lentamente y se atasca en falsos mínimos locales



Learning Rate: ¿Qué tan rápido aprendemos?

Efecto del Learning Rate en la Convergencia



Learning Rate: ¿Qué tan rápido aprendemos?

α muy pequeño:

- Convergencia muy lenta
- Muchas iteraciones
- Costoso computacionalmente

α muy grande:

- Puede diverger
- Salta el mínimo
- Nunca converge

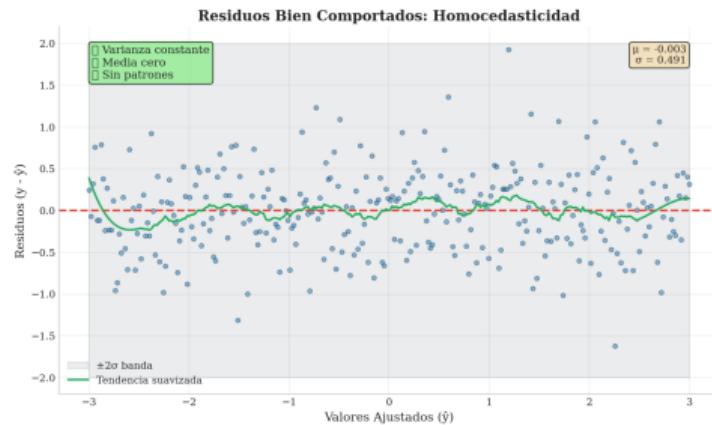
Regla práctica

Probar con $\alpha = 0.01$, luego ajustar según resultados

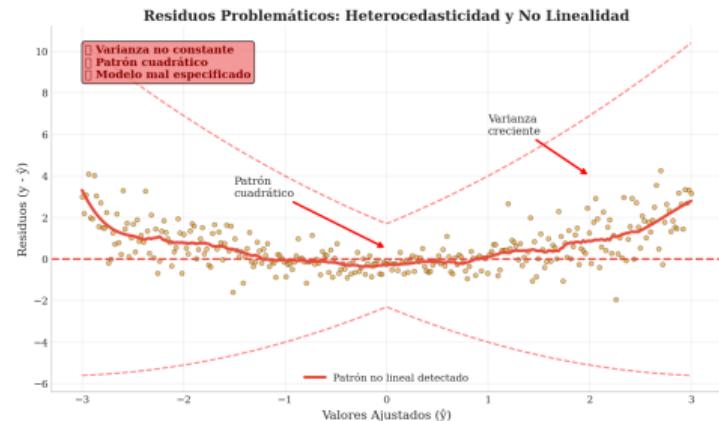
¿El modelo está aprendiendo bien?

Diagnóstico visual de residuos:

Modelo Bueno



Modelo con Problemas

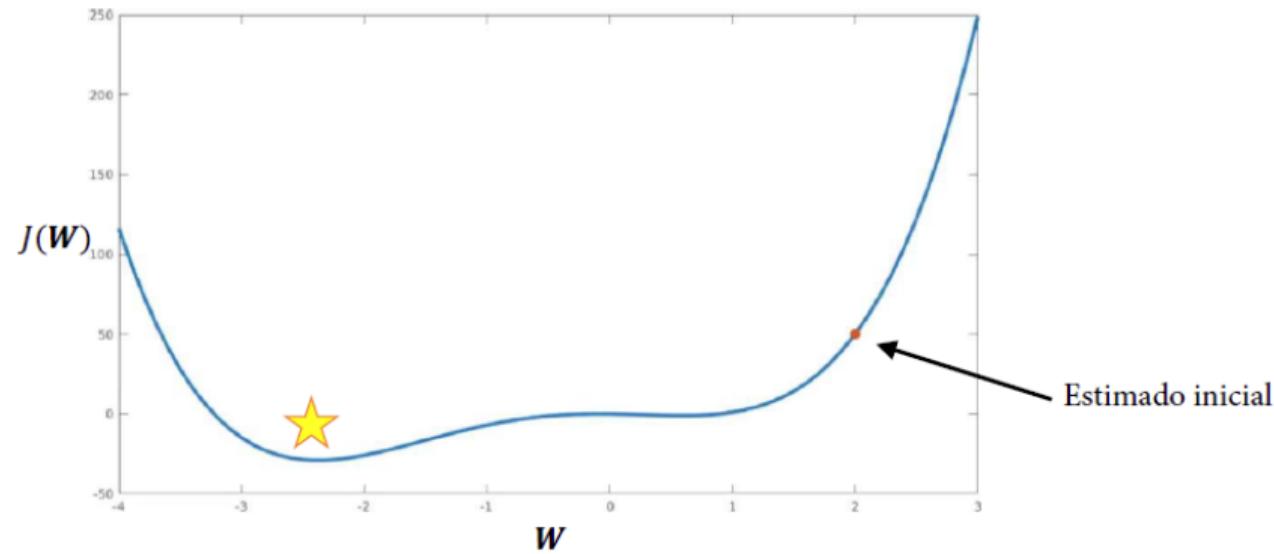


- Errores aleatorios
- Sin patrones
- Centrados en cero

- Patrón visible
- Relación no lineal
- Necesita más features

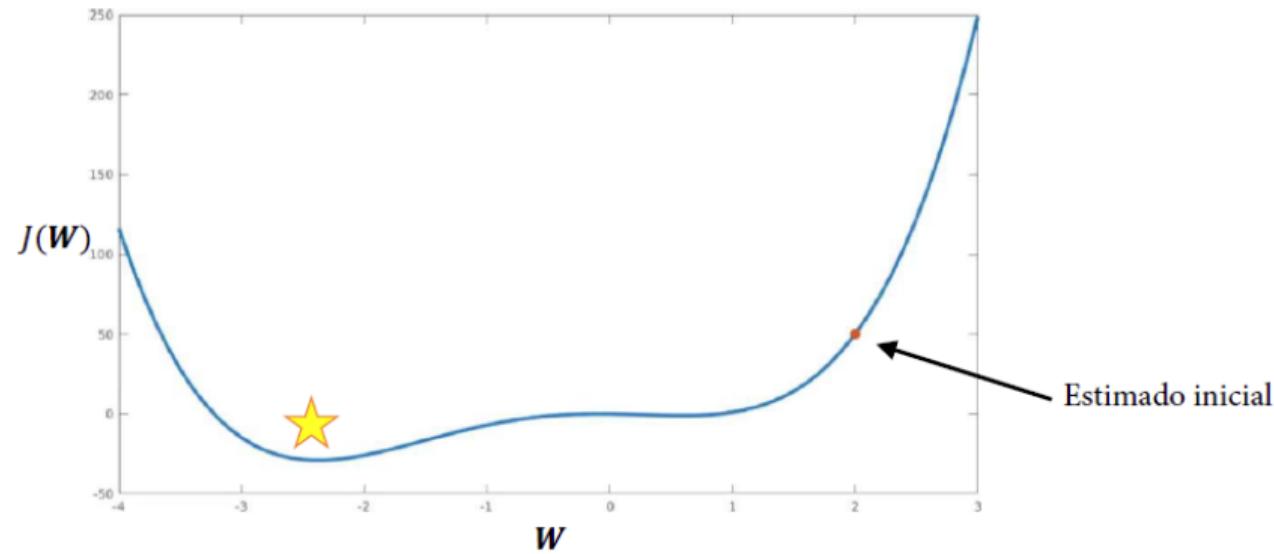
Ajuste de la tasa de aprendizaje

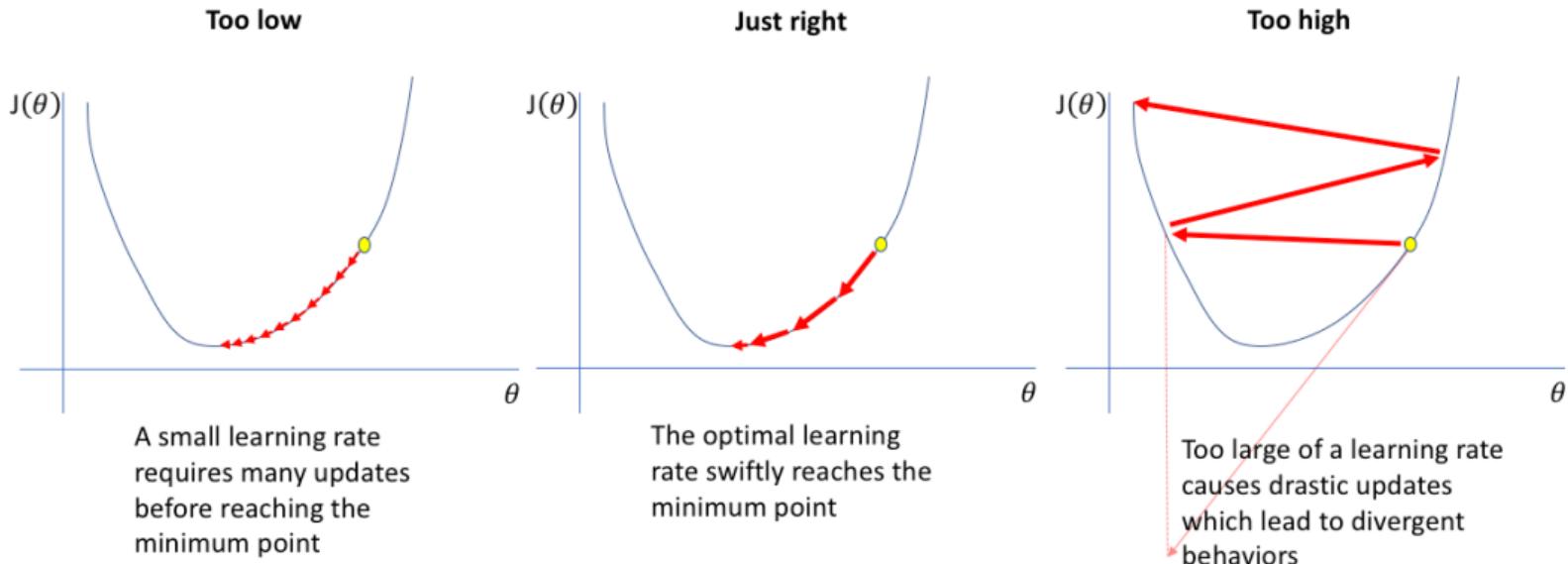
Recuerde: Las grandes tasas de aprendizaje se sobreponen, se vuelven inestables y divergen

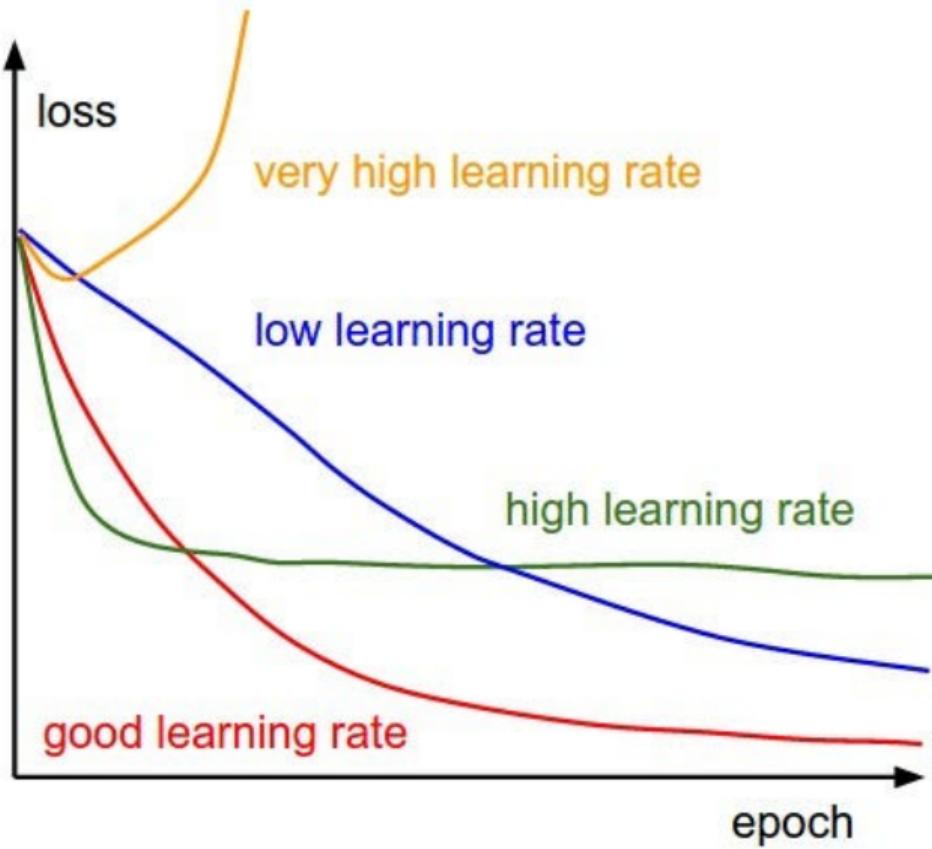


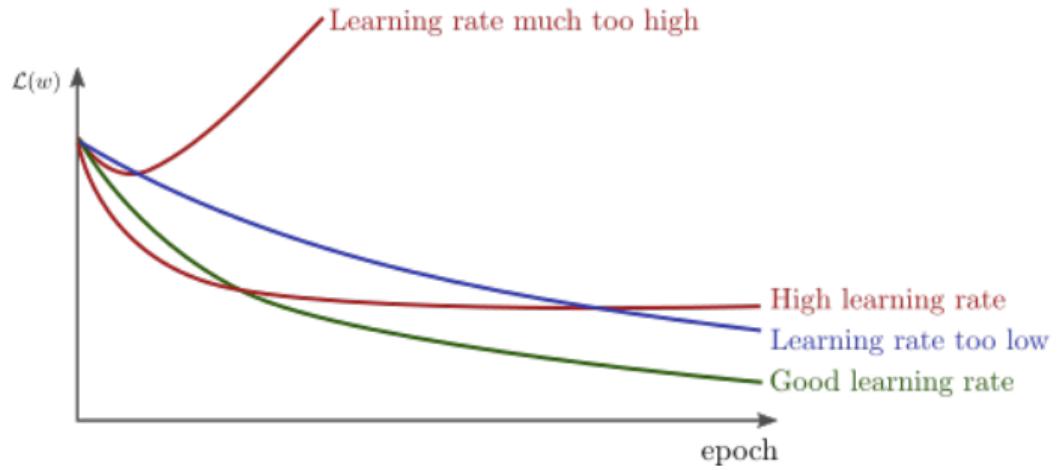
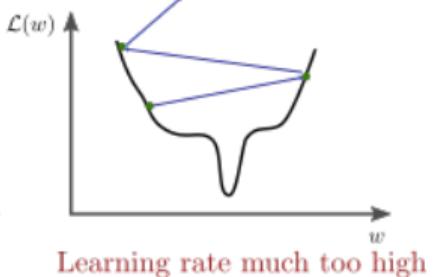
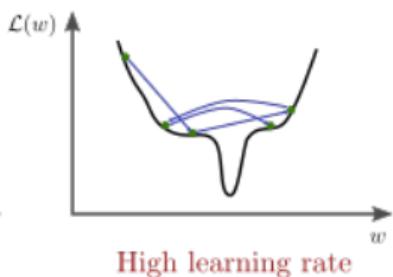
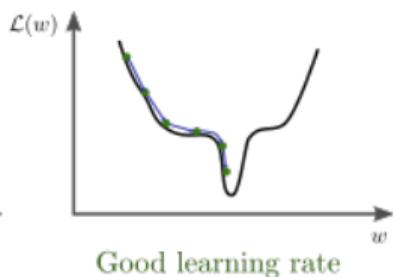
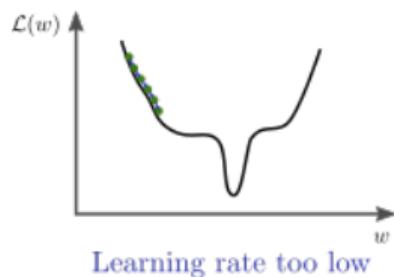
Ajuste de la tasa de aprendizaje

Recuerde: Las tasas de aprendizaje estables convergen sin problemas y evitan los mínimos locales









¿Cómo se puede hacer frente a esto?

- **Idea 1:** Intentar muchas tasas de aprendizaje diferentes y ver cuál funciona "bien".
- **Idea 2:** ¡Haz algo más inteligente! Diseñar una tasa de aprendizaje adaptativo que se adapte al paisaje

Tasas de aprendizaje adaptativas

- Las tasas de aprendizaje ya no son fijas
- Pueden hacerse más grandes o más pequeñas dependiendo de:
 - de cuán grande sea el gradiente
 - lo rápido que se está aprendiendo
 - tamaño de pesos particulares
 - etc...

Algoritmos de tasas de aprendizaje adaptativas

- Momentum
- Adagrad
- Adadelta
- RMSProp

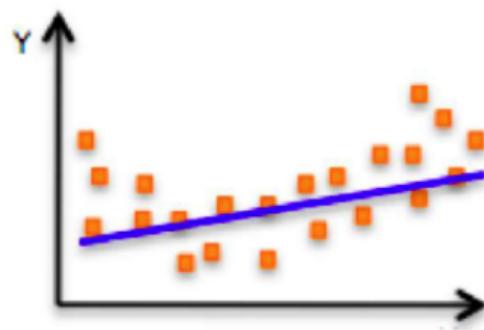


Detalles adicionales:

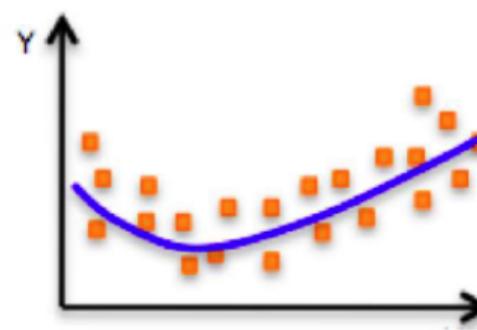
<http://ruder.io/optimizing-gradient-descent/>

Overfitting: cuando memorizamos en vez de aprender

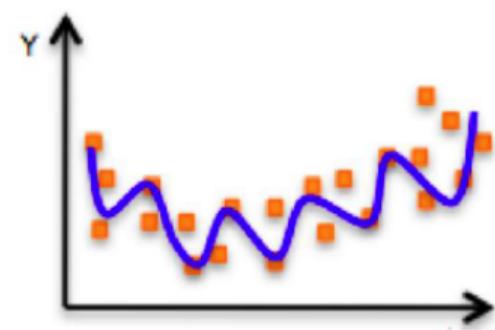
- Modelo **demasiado complejo** para la cantidad de datos.
- **Memoriza** el conjunto de entrenamiento.
- **No generaliza** bien a datos nuevos.



Underfitting
El modelo no tiene capacidad
para aprender completamente
los datos

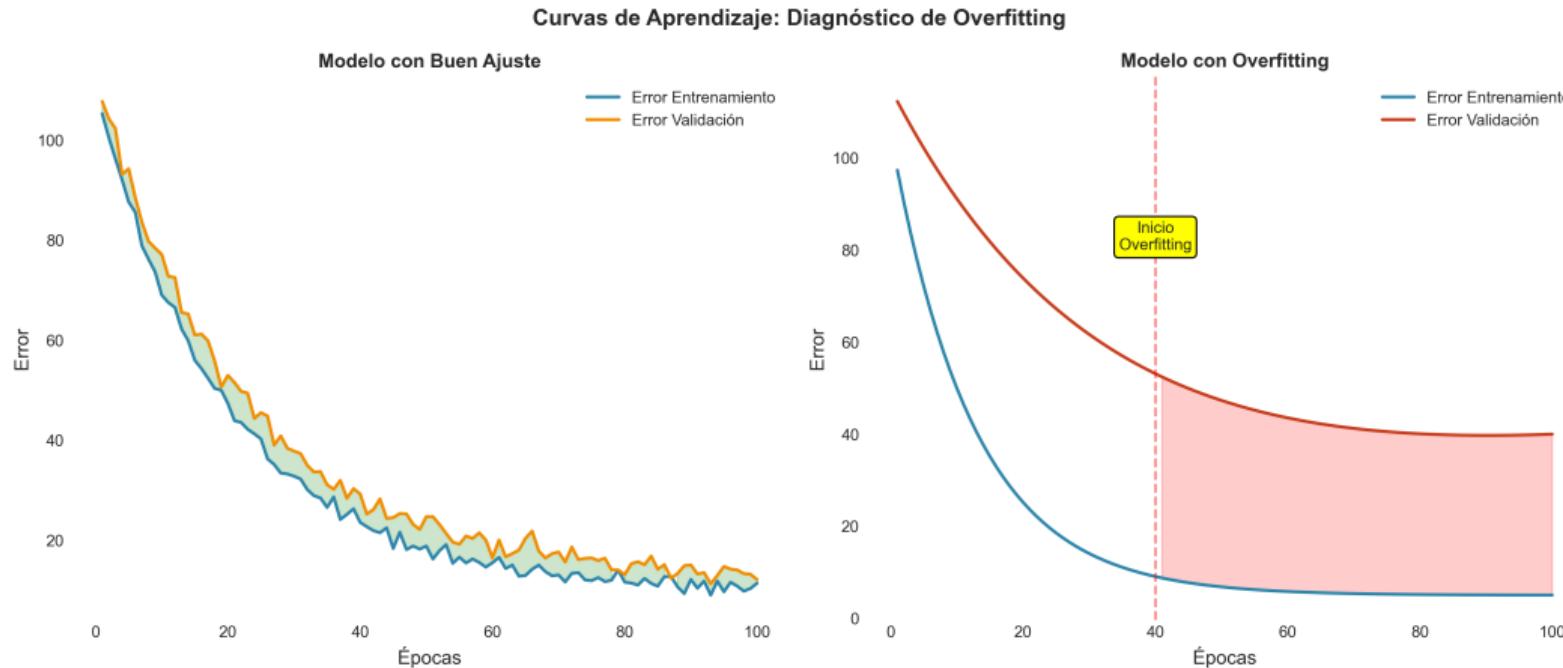


← Ajuste ideal →



Overfitting
Demasiado complejo, parámetros
adicionales, no se generaliza bien

Overfitting: curvas de aprendizaje

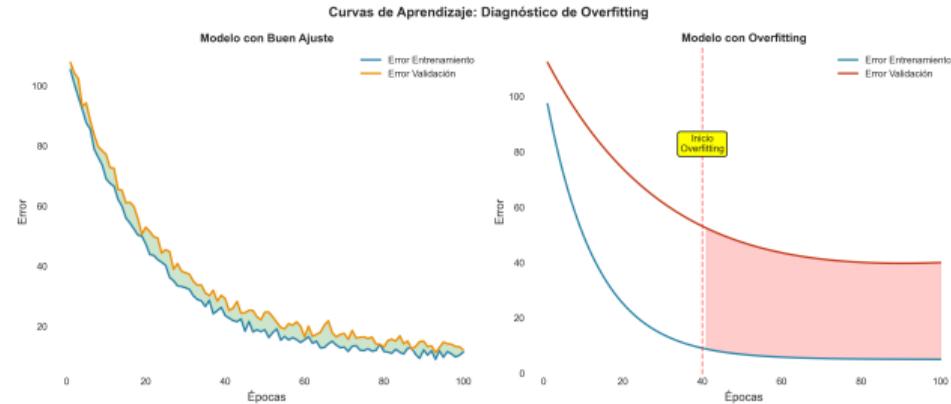


- **Brecha alta** entre error de entrenamiento y validación \Rightarrow sobreajuste.
- Soluciones típicas: *regularización*, más datos, *early stopping*, reducir complejidad.

Overfitting: Cuando memorizamos en vez de aprender

Síntomas:

- Error training: muy bajo
- Error test: muy alto
- Curva muy compleja



Analogía

Como estudiante que memoriza respuestas sin entender conceptos

Regularización: Evitando el overfitting

Regularización

- **¿Qué es?** Técnica que limita nuestro problema de optimización para no incentivar la generación de modelos complejos
- **¿Por qué lo necesitamos?** Mejorar la generalización de nuestro modelo sobre datos no vistos

Regularización: Evitando el overfitting

Idea: Penalizar modelos muy complejos

Función de Costo con Regularización

$$J = \text{Error} + \lambda \cdot \text{Complejidad}$$

Tipos de regularización:

- **Ridge (L2):** Reduce magnitud de parámetros
- **LASSO (L1):** Pone algunos parámetros en cero
- **Elastic Net:** Combina ambos

λ pequeño

Modelo más flexible, riesgo de overfitting

λ grande

Modelo más simple, riesgo de underfitting

Ridge Regression (L2)

Función de costo modificada:

$$J_{Ridge}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Nota: No penalizamos θ_0 (bias)

Solución analítica:

$$\theta^* = (X^T X + \lambda I_n)^{-1} X^T y$$

Propiedades de Ridge

- Siempre tiene solución única
- Sesga estimadores pero reduce varianza
- Equivalente a MAP con prior Gaussiano

LASSO (L1)

LASSO: Promueve sparsidad

$$J_{LASSO}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n |\theta_j|$$

- No tiene solución cerrada
- Requiere coordinate descent o proximal methods
- Selección automática de features

Elastic Net

Elastic Net: Combina L1 + L2

$$J_{EN}(\theta) = J_{MSE} + \lambda_1 \sum_{j=1}^n |\theta_j| + \frac{\lambda_2}{2} \sum_{j=1}^n \theta_j^2$$

Cuándo usar cada uno

Ridge: multicolinealidad | LASSO: selección features | Elastic Net: ambos

Regresión Logística: Clasificación

De predicción a clasificación

Problema: ¿Y si queremos predecir categorías?

Regresión Lineal:

- Predice valores continuos
- Rango: $(-\infty, +\infty)$
- Ej: precio = \$250,000

Regresión Logística:

- Predice probabilidades
- Rango: $[0, 1]$
- Ej: $P(\text{spam}) = 0.85$

La solución: Función Sísmoide

Transforma cualquier valor en una probabilidad entre 0 y 1

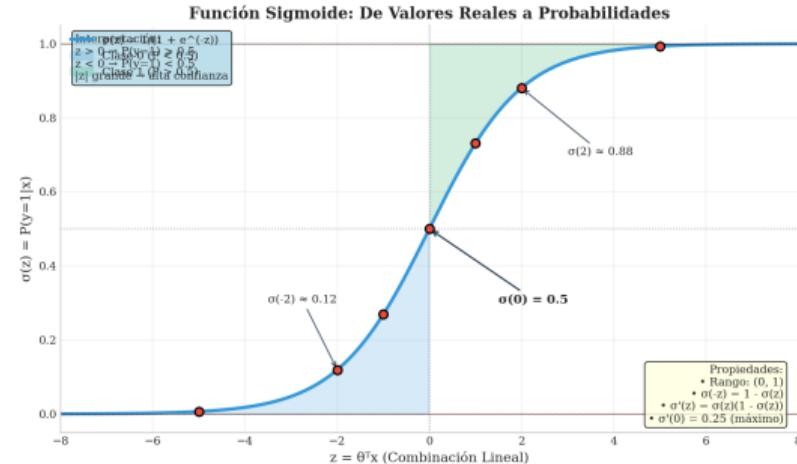
La función sigmoide

Fórmula

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Propiedades:

- Siempre entre 0 y 1
- Suave y diferenciable
- $\sigma(0) = 0.5$
- Simétrica



La función sigmoide

Interpretación

- $z > 0$: Probabilidad > 50% (Clase 1)
- $z < 0$: Probabilidad < 50% (Clase 0)
- $|z|$ grande: Mayor certeza

De regresión a clasificación

Motivación: Predecir probabilidades $P(y = 1|x) \in [0, 1]$

Función sigmoide:

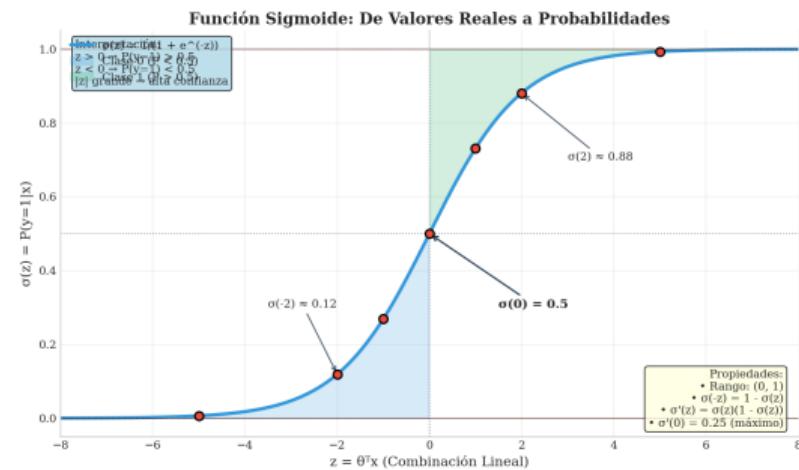
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

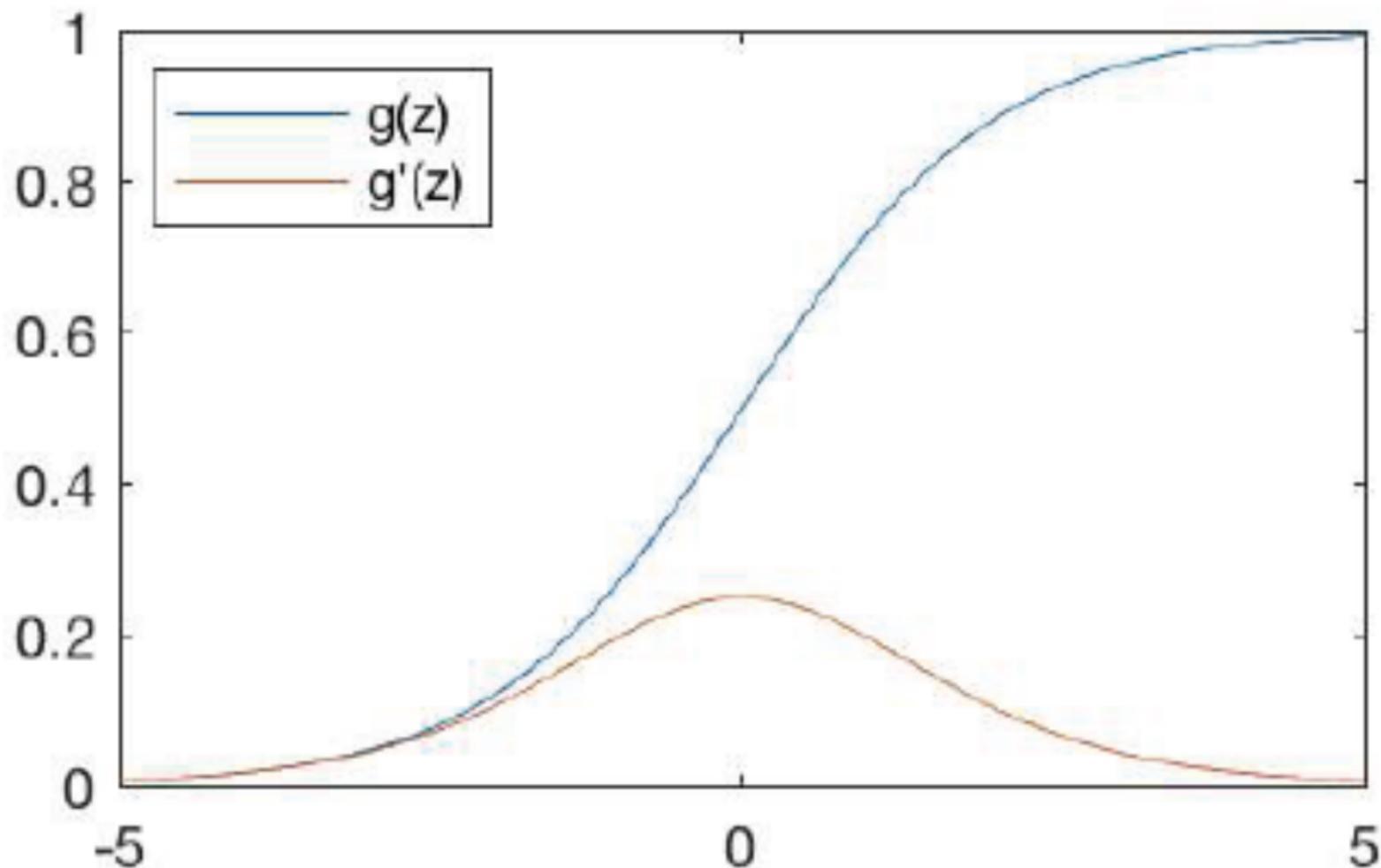
Modelo:

$$P(y = 1|x; \theta) = h_{\theta}(x) = \sigma(\theta^T x)$$

Propiedades:

- $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- $\sigma(-z) = 1 - \sigma(z)$
- Log-odds lineales





Ejemplo: Clasificación de emails

Objetivo: Predecir si un email es spam **Features:**

- x_1 : Número de palabras en mayúsculas
- x_2 : Contiene la palabra "GRATIS"
- x_3 : Número de signos de exclamación

Ejemplo: Clasificación de emails

Objetivo: Predecir si un email es spam **Features:**

- x_1 : Número de palabras en mayúsculas
- x_2 : Contiene la palabra "GRATIS"
- x_3 : Número de signos de exclamación

Modelo:

$$z = -2 + 0.5x_1 + 3x_2 + 0.8x_3$$

$$P(\text{spam}) = \frac{1}{1 + e^{-z}}$$

Ejemplo: Clasificación de emails

Objetivo: Predecir si un email es spam **Features:**

- x_1 : Número de palabras en mayúsculas
- x_2 : Contiene la palabra "GRATIS"
- x_3 : Número de signos de exclamación

Modelo:

$$z = -2 + 0.5x_1 + 3x_2 + 0.8x_3$$

$$P(\text{spam}) = \frac{1}{1 + e^{-z}}$$

Ejemplo de predicción:

- Email con: 5 mayúsculas, tiene "GRATIS", 2 exclamaciones
- $z = -2 + 0.5(5) + 3(1) + 0.8(2) = 5.1$
- $P(\text{spam}) = \frac{1}{1 + e^{-5.1}} = 0.994 \rightarrow \textbf{99.4 \% spam!}$

Frontera de decisión

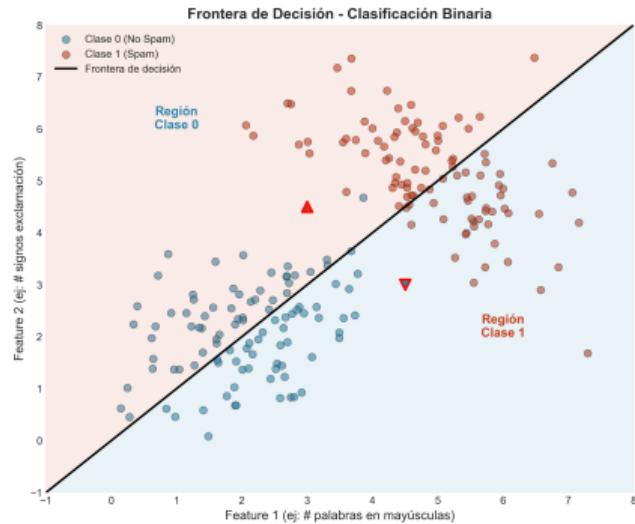
¿Dónde el modelo decide entre una clase y otra?

Frontera de decisión:

- Donde $P = 0.5$
- Equivale a $z = 0$
- Es una línea (o plano)

En 2D:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$



Interpretación

Los puntos de un lado son clase 0, del otro lado son clase 1

Función de costo para clasificación

No podemos usar MSE (no es convexa para clasificación)

Cross-Entropy (Log Loss)

$$J = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log(p^{(i)}) + (1 - y^{(i)}) \log(1 - p^{(i)}) \right]$$

Interpretación intuitiva:

- Si $y = 1$ y predecimos $p \approx 1$: Costo bajo
- Si $y = 1$ y predecimos $p \approx 0$: Costo alto
- Si $y = 0$ y predecimos $p \approx 0$: Costo bajo
- Si $y = 0$ y predecimos $p \approx 1$: Costo alto

Penalización

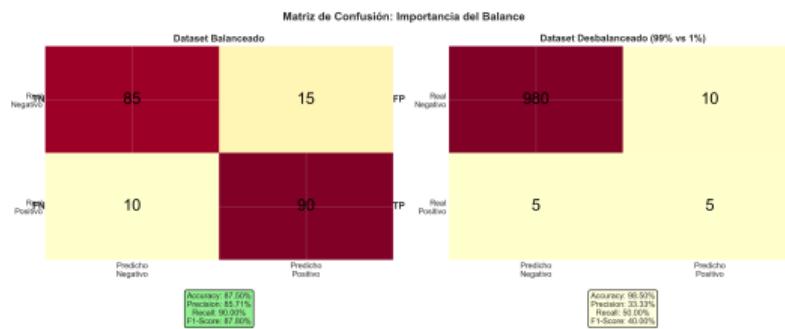
Penaliza fuertemente predicciones incorrectas con alta confianza

Métricas de evaluación

Métricas de evaluación

¿Cómo sabemos si nuestro clasificador es bueno?

Matriz de Confusión:



Métricas clave:

- **Accuracy:** % total correcto
- **Precision:** De los que predije positivo, ¿cuántos lo eran?
- **Recall:** De los positivos reales, ¿cuántos detecté?
- **F1-Score:** Balance entre precision y recall

Ejemplo médico

Detectar enfermedad: Preferimos falsos positivos que falsos negativos

Curva ROC

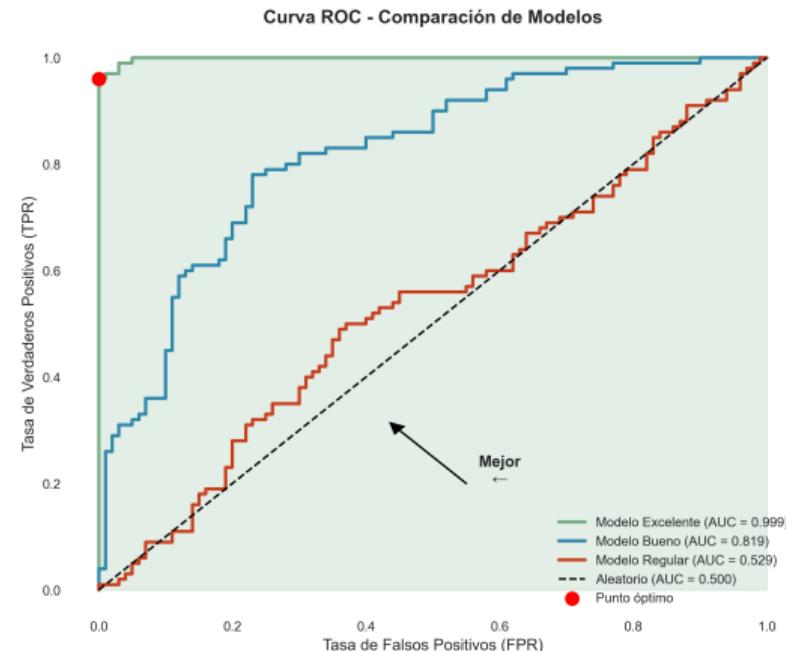
ROC: Receiver Operating Characteristic

¿Qué muestra?:

- Trade-off entre sensibilidad y especificidad
- Rendimiento para todos los umbrales
- AUC: Área bajo la curva

Interpretación AUC:

- 0.5 = Aleatorio
- 0.7-0.8 = Aceptable
- 0.8-0.9 = Bueno
- > 0.9 = Excelente



Extensiones Multiclasé

Clasificación multiclas

¿Y si tenemos más de 2 clases? Ejemplo: Clasificar dígitos (0-9)

Estrategia 1: One-vs-All

- Entrenar K clasificadores
- Cada uno: clase k vs resto
- Elegir el más confiado

Estrategia 2: Softmax

- Generalización de sigmoide
- Salida: K probabilidades
- Suman a 1

Función Softmax

$$P(y = k) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Softmax: Generalización a K clases

Función softmax:

$$P(y = k|x; \Theta) = \frac{e^{\theta_k^T x}}{\sum_{j=1}^K e^{\theta_j^T x}}$$

- Parámetros: $\Theta = [\theta_1, \dots, \theta_K]$
- Generaliza sigmoide (K=2)
- Salida: distribución de probabilidad

Cross-entropy multiclasa:

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{k=1}^K 1\{y^{(i)} = k\} \log P(y^{(i)} = k|x^{(i)})$$

Estabilidad numérica

Usar trick: $\log \text{softmax}(z) = z - \log \sum_j e^{z_j}$

Problemas Prácticos

Separación perfecta en clasificación

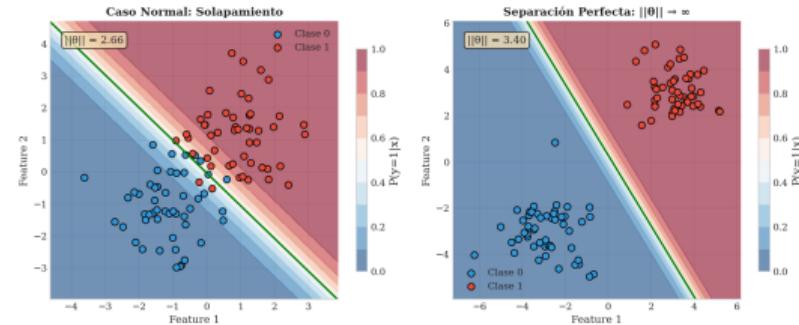
Problema: Clases linealmente separables $\Rightarrow \|\theta\| \rightarrow \infty$

Síntomas:

- Parámetros explotan
- Probabilidades 0 o 1
- Gradientes vanishing

Detección:

- Monitorear $\|\theta\|$
- Verificar $h_\theta \approx 0$ o 1



Solución

Regularización L2 garantiza solución finita

Maximum Likelihood y Cross-Entropy

Likelihood:

$$L(\theta) = \prod_{i=1}^m h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

Función de costo (negative log-likelihood):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

Gradiente (sorprendentemente simple):

$$\nabla_\theta J = \frac{1}{m} X^T (h_\theta(X) - y)$$

Misma forma que regresión lineal

Diferencia clave: $h_\theta(x) = \sigma(\theta^T x)$ vs. $h_\theta(x) = \theta^T x$

¿Cuándo usar cada modelo?

Regresión Lineal vs Logística

Aspecto	Lineal	Logística
Output	Valor continuo	Probabilidad [0,1]
Uso	Predicción	Clasificación
Función	$y = \theta^T x$	$p = \sigma(\theta^T x)$
Costo	MSE	Cross-entropy
Ejemplo	Precio casa	Spam/No spam

Cuándo usar cada modelo

Usa Regresión Lineal cuando

- Predices cantidades numéricas continuas
- Necesitas valor exacto
- Relación aproximadamente lineal
- Interpretabilidad crucial
- Alta dimensión, muestra limitada
- Necesita solución cerrada

Limitaciones:

- Asume linealidad
- Sensible a outliers
- Requiere feature engineering

Usa Regresión Logística cuando

- Clasificas categorías binarias/multiclas
- Salida probabilística calibrada
- Interpretación odds ratios
- Umbrales de decisión ajustables
- Separación lineal suficiente

Limitaciones:

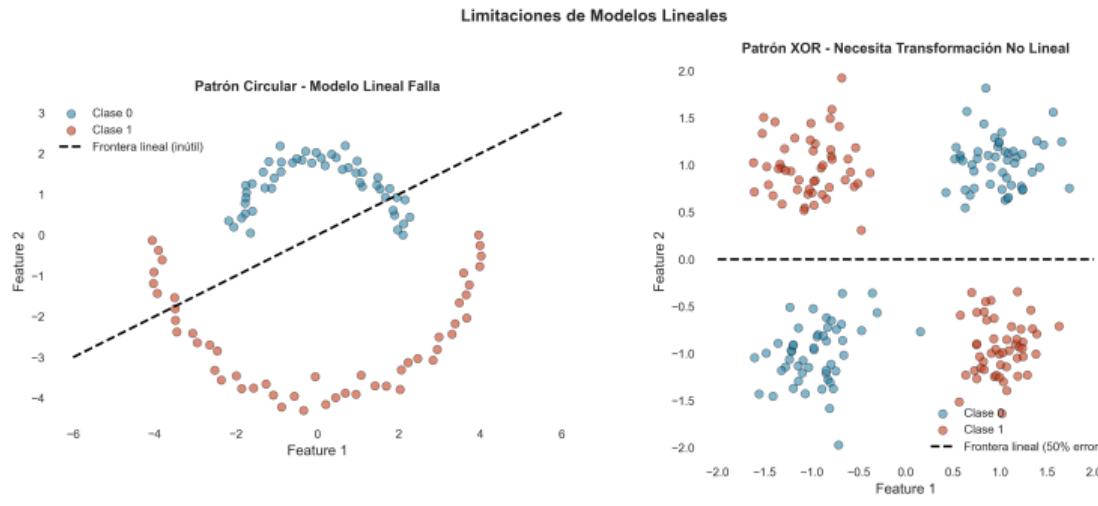
- Frontera de decisión lineal
- Problemas con separación perfecta
- Sensible a desbalance

Limitaciones de ambos modelos

Asumen relaciones lineales:

- No capturan patrones complejos
- Necesitan feature engineering manual
- Mal rendimiento con relaciones no lineales

Limitaciones de ambos modelos



Datos con patrón circular

Soluciones

- Añadir términos polinómicos (x^2 , $x_1 \cdot x_2$)
- Usar modelos más complejos (árboles, neural networks)

Aplicaciones en la vida real

Regresión Lineal:

- **Finanzas:** Predicción de precios
- **Economía:** Demanda vs precio
- **Medicina:** Dosis vs respuesta
- **Marketing:** Inversión vs ventas
- **Ingeniería:** Consumo energético

Regresión Logística:

- **Medicina:** Diagnóstico (sano/enfermo)
- **Banca:** Aprobación de créditos
- **Marketing:** Cliente comprará/no
- **Seguridad:** Fraude/legítimo
- **HR:** Contratación sí/no

Ventaja común

Ambos modelos son interpretables: podemos entender qué variables importan más

Resumen y Conclusiones

Lo que aprendimos

Conceptos clave:

- Función de costo
- Descenso del gradiente
- Overfitting y regularización
- Métricas de evaluación

Habilidades prácticas:

- Elegir el modelo correcto
- Diagnosticar problemas
- Interpretar resultados
- Mejorar rendimiento

Mensaje clave

La regresión lineal y logística son los fundamentos del Machine Learning. Dominarlos es esencial para entender modelos más complejos.

Próximos pasos

Estos conceptos se extienden a redes neuronales, donde cada neurona es como una pequeña regresión logística

Preguntas de repaso

- 1 ¿Cuál es la diferencia principal entre regresión lineal y logística?
- 2 ¿Por qué usamos MSE en regresión lineal?
- 3 ¿Qué hace la función sigmoide?
- 4 ¿Qué es el overfitting y cómo lo prevenimos?
- 5 ¿Cuándo es accuracy una mala métrica?
- 6 ¿Qué significa un AUC de 0.5? ¿Y de 0.95?
- 7 ¿Cómo elegimos el learning rate?
- 8 ¿Qué nos dicen los parámetros θ en el modelo?

Literatura recomendada

Libros fundamentales:

- **Pattern Recognition and Machine Learning** - Bishop
 - Capítulos 3 y 4: Modelos lineales
- **The Elements of Statistical Learning** - Hastie, Tibshirani
 - Capítulo 3: Regresión lineal
 - Capítulo 4: Clasificación lineal
- **Introduction to Statistical Learning** - James et al.
 - Versión más accesible del anterior

Recursos online:

- Curso de Andrew Ng (Coursera)
- Fast.ai - Practical Deep Learning
- Scikit-learn documentation

¡Muchas gracias por su atención!

¿Preguntas?



Contacto: Marco Teran
webpage: marcoteran.github.io/
e-mail: mtteranl@eafit.edu.co