5. Aprendizaje Automático (Machine Learning)

5.1 Introducción y Cambio de Paradigma

Hasta ahora (Sistemas Expertos), programábamos la IA diciéndole explícitamente *qué hacer* (reglas `SI-ENTONCES`). En el Aprendizaje Automático, cambiamos el enfoque: **la máquina aprende las reglas a partir de los datos**.

Un programa aprende si mejora su rendimiento (**P**) en una tarea (**T**) basándose en la experiencia (**E**).

- **Ejemplo (Damas)**:
  - **T (Tarea)**: Jugar a las damas.
  - **E (Experiencia)**: Jugar miles de partidas contra sí mismo.
  - **P (Rendimiento)**: % de partidas ganadas.

5.2 Tipos de Aprendizaje

No todos los problemas son iguales. Según la información que tengamos, clasificamos el aprendizaje en:

5.2.1 Aprendizaje Supervisado

Tenemos los "exámenes resueltos". Entregamos a la máquina datos de entrada (x) junto con su respuesta correcta o etiqueta (y). El objetivo es aprender la relación para predecir y en datos nuevos.

Se divide en dos subtipos clave:

1. **Regresión**: Predice un **valor numérico continuo** (infinitas posibilidades).
   - *Ejemplo*: Predecir el precio de una casa según sus metros cuadrados.
2. **Clasificación**: Predice una **clase o etiqueta discreta** (categorías fijas).
   - *Ejemplo*: ¿Es este tumor maligno o benigno? (0 o 1).
   - *Nota*: Esto es lo que hacía el Perceptrón (clasificar).

5.2.2 Aprendizaje No Supervisado

Solo tenemos los datos (x), sin respuestas. La máquina debe encontrar **estructuras o patrones** por sí sola.

- *Ejemplo (Clustering)*: Agrupar noticias similares en Google News o segmentar clientes por comportamiento .

5.3 Regresión Lineal: El Modelo Base

Es el "Hola Mundo" del Machine Learning. Busca dibujar la **línea recta** que mejor se ajusta a una nube de puntos.

5.3.1 La Hipótesis (h_θ)

Es la fórmula matemática que usamos para predecir. En regresión lineal simple (una variable), es la ecuación de una recta:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

- θ_0: El punto de corte (Bias).
- θ_1: La pendiente (Peso).
- *Conexión*: ¿Te suena? Es idéntico a la parte interna de una neurona: $w \cdot x + b$.

5.3.2 La Función de Coste ($J(\theta)$)

Para aprender, necesitamos medir cuánto nos equivocamos. Usamos el Error Cuadrático Medio (MSE):

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Básicamente: "Calcula la diferencia entre lo que predijiste y el valor real, elévalo al cuadrado (para que sea siempre positivo) y haz la media".
- **Objetivo**: Encontrar los θ (pesos) que hagan que esta J sea **mínima** (cero error idealmente).

5.3.3 Generalización: Regresión Lineal Multivariable

El modelo básico ($h_\theta(x) = \theta_0 + \theta_1 x$) solo sirve si hay una única característica de entrada. Pero el mundo es complejo.

- **Concepto:** si tenemos n características ($x_1, x_2, \ldots, x_n$), la fórmula se espande
- **Nueva Hipótesis:**

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- **Vectorización:** Para que el ordenador lo calcule rápido, esto se expresa como una multiplicación de matrices: $h_\theta(x) = \theta^T x$.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{metros} \\ \text{habitaciones} \end{bmatrix}$$

$$\text{Si } \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}, \text{entonces } \theta^T = [\theta_0, \theta_1, \theta_2].$$

$$\theta^T x = [\theta_0, \theta_1, \theta_2] \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

5.3.4 Adaptación a Curvas: Regresión Polinómica

A veces, los datos no siguen una línea recta, sino una curva.

- **El Truco:** no necesitamos inventar un algoritmo nuevo. Podemos "engañar" a al regresión lineal creando **nuevas características artificiales** elevando las originales al cuadrado, al cubo, etc.
- **Hipótesis:**

$$\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2$$

- **Peligro:** Al añadir términos polinómicos ($x^2, x^3 \ldots$), la línea se vuelve muy flexible y curva. Esto mejora el ajuste, pero si nos pasamos, causaremos **Overfitting** (esto se explica más adelante).

Imagina que quieres predecir cuánto vas a gastar en electricidad (y) basándote en la temperatura de la calle (x).

- **Intento con Regresión Lineal (Línea Recta):**
  - La máquina pensaría: *"Cuanto más calor hace, más gastas"* (Línea subiendo) O BIEN *"Cuanto más calor hace, menos gastas"* (Línea bajando).
  - **¿Por qué falla?** Porque la realidad es más compleja:
    - Si hace mucho frío (0ºC), gastas mucho (calefacción).
    - Si hace una temperatura media (20ºC), gastas poco (ni frío ni calor).
    - Si hace mucho calor (40ºC), vuelves a gastar mucho (aire acondicionado).
  - **Forma de los datos**: Tienen forma de **"U"** (una parábola).
- **Solución (Regresión Polinómica):**
  - Al añadir el término al cuadrado (x^2), la fórmula puede curvarse y crear esa "U".
  - **Conclusión**: El modelo aprende que el gasto sube en *ambos extremos* de la temperatura.

5.4 Regresión Logística (Para Clasificación)

5.4.1 La hipótesis

*(Importante: Aunque se llame "regresión", sirve para **clasificar**).* Es el modelo matemático que conecta la Regresión Lineal con las Neuronas Artificiales.

- **El Problema**: La regresión lineal predice valores continuos (puede dar 500, -20, 1000). Pero si queremos clasificar (0 o 1, Tumor benigno o maligno), necesitamos que la salida esté acotada entre 0 y 1.
- **La Solución (Función Sigmoide):** Envolvemos la hipótesis lineal dentro de una función logística o sigmoide ($g(z)$).

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

- **Interpretación**: La salida se convierte en una **probabilidad**.
  - Si $h_\theta(x) = 0.7$, significa "Hay un 70% de probabilidad de que sea clase 1".
- **Frontera de Decisión**: Es la línea (o curva) que separa la zona donde predecimos 0 de la zona donde predecimos 1. Se produce cuando la probabilidad es exactamente 0.5.

5.4.2 Función de Coste

Se define en dos partes:

- Si la respuesta real es 1 ($y = 1$):

$$Coste = -\log(h_\theta(x))$$

  - *Interpretación*: Si la máquina predice 1 (acierta), el coste es 0. Si predice 0 (falla estrepitosamente), el coste tiende a infinito (castigo enorme).
- Si la respuesta real es 0 ($y = 0$):

$$Coste = -\log(1 - h_\theta(x))$$

  - *Interpretación*: Si predice 0 (acierta), coste 0. Si predice 1 (falla), coste infinito.

Para no escribir dos fórmulas, las combinamos en una sola ecuación elegante:

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

- *Nota*: Solo uno de los dos términos se activa a la vez (porque y es 0 o 1, anulando la otra parte).

5.4.3 Clasificación Binaria vs. Multiclase

La Regresión Logística está diseñada por naturaleza para responder "Sí/No" (Binaria). ¿Pero qué pasa si quiero clasificar correos en "Trabajo", "Amigos" y "Spam" (3 clases)?

A. Clasificación Binaria (2 Clases)

- **Datos**: $y \in \{0, 1\}$.
- **Frontera de Decisión**: Una sola línea (o curva) que separa el espacio en dos zonas.
- **Modelo**: Una sola hipótesis $h_\theta(x)$.

B. Clasificación Multiclase (Más de 2 grupos)

- **Datos**: $y \in \{0, 1, 2, \dots, n\}$.
- **Estrategia**: Usamos la técnica **"Uno contra Todos" (One-vs-All)**.

¿Cómo funciona "Uno contra Todos"?
En lugar de entrenar un solo clasificador complejo, entrenamos varios clasificadores binarios:

1. **Transformamos el problema**:
   - Imagina que tienes 3 clases: Triángulos, Cuadrados y Círculos.
   - Entrenas el **Clasificador 1**: ¿Es un Triángulo o NO? (Juntas cuadrados y círculos en el grupo "NO").
   - Entrenas el **Clasificador 2**: ¿Es un Cuadrado o NO?
   - Entrenas el **Clasificador 3**: ¿Es un Círculo o NO?
2. **Predicción**:
   - Cuando llega un dato nuevo, lo pasas por los 3 clasificadores.
   - Cada uno te dará una probabilidad (ej. C1: 20%, C2: 80%, C3: 10%).
   - **Eliges la clase con la probabilidad más alta** $(\max h_\theta^{(i)}(x))$.

5.4.4 Ejemplo

Quieres que la IA decida si un email entrante es "Spam" o "Correo Normal".

- **Intento con Regresión Lineal:**
  - Le das al sistema el número de veces que aparece la palabra "Oferta".
  - La regresión lineal te podría devolver un valor como 150 o -20.
  - **¿Por qué falla?** ¿Qué significa que un correo sea "150 de Spam"? No tiene sentido. Tú necesitas una probabilidad entre 0 y 1.
- **Solución (Regresión Logística):**
  - La función **Sigmoide** "aplasta" cualquier número que salga para que esté siempre entre **0 y 1**.
  - **Resultado**: Te dice *"La probabilidad de que esto sea Spam es 0.95"*.
  - **Decisión**: Como 0.95 > 0.5 (umbral), lo manda a la carpeta de Spam.

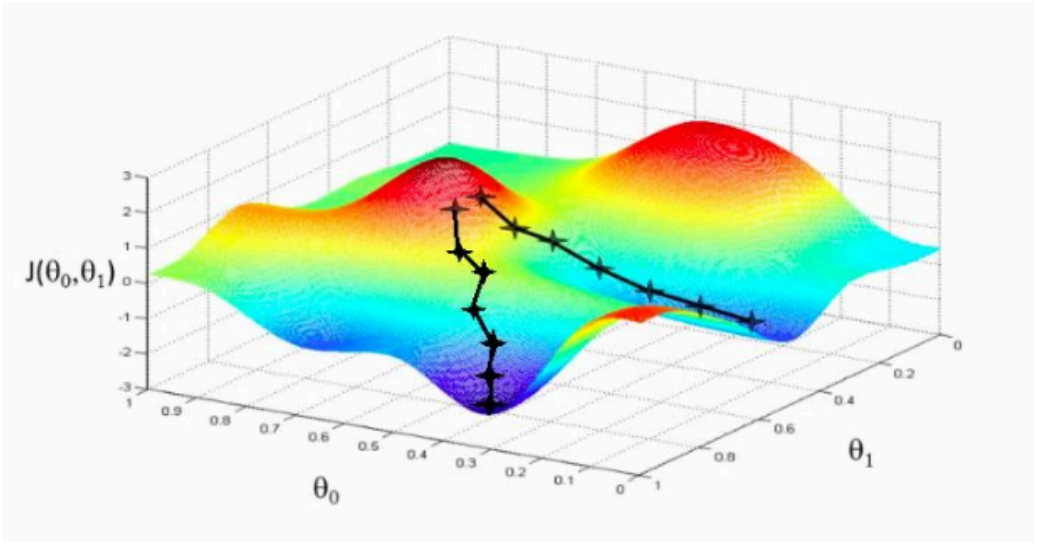
5.5 El Motor de Aprendizaje: Descenso del Gradiente

¿Cómo encuentra la máquina los mejores θ sin probar a lo loco? Usando un algoritmo de optimización iterativo, similar a "bajar una montaña a ciegas tanteando el terreno".

Algoritmo: Repetir hasta converger:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

- **Derivada ($\frac{\partial}{\partial \theta}$)**: Nos dice la pendiente. Si es positiva, debemos bajar (restar); si es negativa, subir. Nos da la **dirección** correcta.
- **Tasa de Aprendizaje (α)**: Es el **tamaño del paso**.
  - Si α es muy pequeño: El aprendizaje es lentísimo (pasitos de bebé).
  - Si α es muy grande: Podemos "saltarnos" el mínimo y nunca converger (pasos de gigante) 8.
- *Conexión*: Este es el mismo principio matemático que usaba el Perceptrón para actualizar sus pesos ($w_{nuevo} = w_{viejo} + \Delta w$).



5.6 Problemas Fundamentales: Sesgo vs. Varianza

Cuando entrenamos un modelo, podemos fallar por dos extremos opuestos:

5.6.1 Underfitting (Subajuste / Alto Sesgo)

El modelo es **demasiado simple** para explicar los datos.

- *Ejemplo*: Intentar ajustar una línea recta a datos que forman una curva parabólica.
- *Síntoma*: Tiene mucho error tanto en el entrenamiento como en la prueba. No aprende bien.

5.6.2 Overfitting (Sobreajuste / Alta Varianza)

El modelo es **demasiado complejo**. Se aprende de memoria los datos de entrenamiento, incluido el ruido.

- *Ejemplo*: Un polinomio de grado 10 que pasa por *todos* los puntos haciendo zig-zag.
- *Síntoma*: Error bajísimo en entrenamiento, pero altísimo en datos nuevos (falla al generalizar)9.

5.7 Solución: Regularización

¿Cómo evitamos el Overfitting si queremos usar modelos complejos (muchas variables)?

La **Regularización** consiste en "castigar" al modelo si usa pesos (θ) muy grandes. Modificamos la Función de Coste añadiendo un término extra:

$$J(\theta) = \text{Error Original} + \lambda \sum \theta_j^2$$

- λ **(Lambda)**: Es el parámetro de regularización.
  - Si λ es alto: Obligamos a que los pesos θ sean muy pequeños (casi cero), simplificando el modelo (evita overfitting).
  - Si λ es cero: Volvemos a la regresión normal (riesgo de overfitting)10.