IA - Clase 1

Introducción

Repaso de Matemáticas

Conceptos de Aprendizaje de Máquina

(ML – Machine Learning)

Repaso de Matemáticas

- Función vectorial: $f(x, y) = x^\top Ay + x^\top Bx - Cy + D$

con:

- $x \in \mathbb{R}^M$ (vector columna de dimensión M)
- $y \in \mathbb{R}^N$ (vector columna de dimensión N)
- $f : \mathbb{R}^M \times \mathbb{R}^N \to \mathbb{R}$ (escalar).

- Análisis de cada término

$x^\top Ay$

$x^\top$ es de tamaño $1 \times M$.

y es de tamaño $N \times 1$.

Para que el producto sea válido, A debe "conectar" M con N.

Es decir: $A \in \mathbb{R}^{M \times N}$.

Entonces:

$$(1 \times M)(M \times N)(N \times 1) = (1 \times 1) \quad (\text{escalar}).$$

Repaso de Matemáticas

- Análisis de cada término

$x^\top Bx$

$x^\top$ es $1 \times M$.

x es $M \times 1$.

Para que funcione, $B \in \mathbb{R}^{M \times M}$.

Entonces:

$$(1 \times M)(M \times M)(M \times 1) = (1 \times 1).$$

$-Cy$

y es $N \times 1$.

Para que dé un escalar, C debe ser $1 \times N$.

Entonces:

$$(1 \times N)(N \times 1) = (1 \times 1).$$

D

Como es una constante sumada a escalares, necesariamente debe ser un **escalar**.

$$D \in \mathbb{R}.$$

Repaso de Matemáticas

- Ejemplo numérico con M=2 N=3
  - Vectores:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

  - Matrices y escalar:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -1 \\ 0 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}, \quad D = 10$$

Cálculos paso a paso

1. $x^\top Ay = 150$
2. $x^\top Bx = 12$
3. $-Cy = -98$
4. $D = 10$

Resultado final

$$f(x, y) = 150 + 12 - 98 + 10 = 74$$

Repaso de Matemáticas

- Ejemplo numérico con M=2 N=3

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Paso 1.1: Ay

$$Ay = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 + 2 \cdot 4 + 3 \cdot 5 \\ 4 \cdot 3 + 5 \cdot 4 + 6 \cdot 5 \end{bmatrix} = \begin{bmatrix} 26 \\ 62 \end{bmatrix}$$

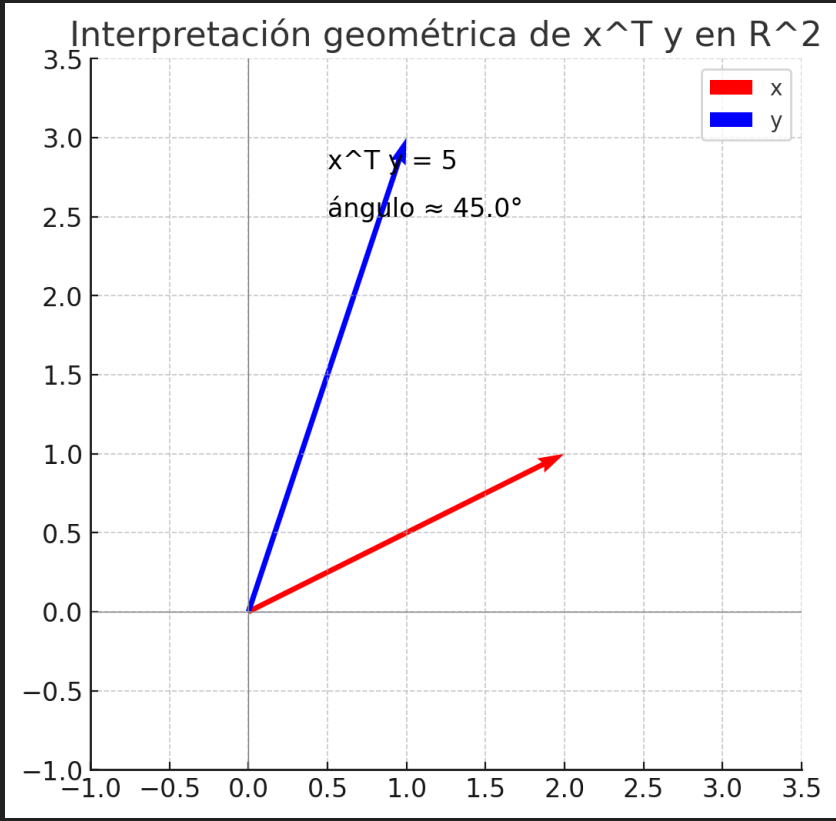
Paso 1.2: $x^\top (Ay)$

$$x^\top (Ay) = \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 26 \\ 62 \end{bmatrix} = 1 \cdot 26 + 2 \cdot 62 = 150$$

Repaso de Matemáticas

- Análisis de cada término
  - x^T
    - La transpuesta de un vector/matriz consiste en cambiar filas ↔ columnas.
    - Si x es un vector columna (M×1), entonces:
    - $x^T=[x1\ x2\cdots xM] \in R1\times M$
  - Ahora x^T es un vector fila.
  - Si multiplicáramos x·y directamente, no siempre tendría sentido porque ambos son vectores columna.
  - Pero al usar x^T y tenemos:
    - $(1\times M)(M\times 1)=(1\times 1)$,
    - es decir, un escalar: justamente el producto interno o producto punto de dos vectores.
  - x^Ty es el producto escalar entre x e y, mide qué tanto apuntan en la misma dirección. De hecho:
  - $x^Ty=\|x\|\|y\|\cos(\theta)$, donde θ es el ángulo entre x e y.

Repaso de Matemáticas



Cálculo exacto con $x = (2, 1), y = (1, 3)$

1. Producto escalar:

$$x^\top y = 2 \cdot 1 + 1 \cdot 3 = 5$$

2. Normas:

$$\|x\| = \sqrt{2^2 + 1^2} = \sqrt{5}, \quad \|y\| = \sqrt{1^2 + 3^2} = \sqrt{10}$$

3. Coseno del ángulo:

$$\cos(\theta) = \frac{5}{\sqrt{5}\sqrt{10}} = \frac{5}{\sqrt{50}} = \frac{5}{7.071} \approx 0.707$$

4. Ángulo:

$$\theta = \arccos(0.707) \approx 45^\circ$$

En 2D y 3D podemos visualizar los vectores y el ángulo.

En 4D o más, no podemos dibujar, pero el producto escalar y la fórmula del coseno nos siguen diciendo si los vectores son "cercanos" (ángulo pequeño), "ortogonales" (θ=90°) u "opuestos" (θ>90°).

Repaso de Matemáticas

- El vector es uno de los conceptos matemáticos más importantes en Inteligencia Artificial (IA) y en particular en Machine Learning (ML) y Deep Learning (DL).
- Representación de datos
  - En IA, todo se representa como vectores: una imagen en escala de grises 28×28 (como MNIST) se convierte en un vector de dimensión 784
  - Una palabra (ej. "gato") se representa como un vector en un espacio semántico (Word Embeddings como Word2Vec, GloVe, o BERT).
  - Una fila de una base de datos con variables (edad, peso, altura, ingresos) también se guarda como un vector de características (feature vector).
  - El vector es la forma estándar de pasar la información al modelo.

Repaso de Matemáticas

- Operaciones básicas en modelos
- Los algoritmos de IA dependen de operaciones con vectores:
  - Producto escalar ($x^T y$): mide similitud entre dos vectores (coseno del ángulo → usado en búsqueda semántica, recomendación).
  - Norma (tamaño, longitud o magnitud $\|x\|$): mide la magnitud de un vector → usado en regularización, distancias.
  - Distancia Euclidiana ($\|x-y\|$): mide cuán distintos son dos vectores → en KNN, clustering, embeddings.
- Redes neuronales
  - En redes neuronales profundas una capa densa hace :
    - $z=Wx+b$
      - x es el vector de entrada,
      - W es la matriz de pesos,
      - b es el vector de sesgo,
      - z es el nuevo vector de salida.
  - Cada neurona es un producto escalar entre el vector de pesos y el vector de entrada.

Repaso de Matemáticas

- Geometría de la IA
  - En espacios de alta dimensión, cada vector representa un punto.
  - Clasificar, agrupar o predecir significa separar y organizar vectores en ese espacio.
    - Ejemplo: un clasificador lineal busca un hiperplano (definido por un vector normal) que divida los vectores de dos clases.

- Interpretación semántica
  - En IA moderna (NLP *Procesamiento de Lenguaje Natural*, visión, audio):
    - Dos vectores cercanos → significan cosas similares (ej. "rey" y "reina").
    - Dos vectores ortogonales → no relacionados.
    - Dos vectores opuestos → significados contrarios.
  - Esto se usa en embeddings para búsquedas, traducción automática, chatbots, etc.

Repaso de Matemáticas
Analicemos un ejemplo

- NLP = IA aplicada al lenguaje humano.
  - Representa palabras/oraciones como vectores en espacios de alta dimensión.
  - Se aplica en traducción, chatbots, resumen, análisis de sentimiento, búsqueda semántica.
  - Los Transformers revolucionaron NLP → base de los LLM como ChatGPT.
- Ver ejemplo en Python en:
  - *IA_TP1A_ML_Aprendizaje _Máquina_R1.ipynb*

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Representaciones vectoriales de palabras, frases o documentos en un espacio de dimensión finita (R^d).
  - Intuición: convierten texto en números que capturan similitud semántica.
    - Ejemplo: “gato” → [0.2,−0.5,0.8,0.1]
    - “perro” → cercano a “gato” en el espacio.
  - Uso: entradas para modelos de NLP, búsqueda semántica, clasificación.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Por qué el vector “gato” → [0.2,−0.5,0.8,0.1] tiene 4 valores en vez de uno ?
  - Supongamos que representamos cada palabra con un único valor:
    - "gato" → 0.2
    - "perro" → 0.5
    - "avión" → 0.9
  - Problema: con 1 dimensión solo tenemos una recta, y no podemos capturar similitudes complejas.
  - Ejemplo: ¿dónde pongo "león"? ¿Más cerca de “perro” o de “avión”?
  - Con un solo número es imposible reflejar múltiples relaciones.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Con varias dimensiones (ejemplo 4D)
  - Al usar un vector en R^d cada coordenada puede capturar un aspecto diferente (aunque no interpretable directamente):
    - "gato" → [0.2,−0.5,0.8,0.1]
    - "perro" → [0.1,−0.6,0.7,0.0]
    - "avión" → [−0.9,0.2,−0.1,0.5]
  - “gato” y “perro” quedan cerca en el espacio (vectores similares), pero “avión” queda lejos.
  - No es que la primera coordenada sea “animal” y la segunda “tamaño”.
  - Más bien, el conjunto completo del vector es lo que captura el significado.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Dimensiones típicas
    - Word2Vec → 50 a 300 dimensiones.
    - GloVe → 100 a 300.
    - BERT / GPT → 768, 1024 o más.
  - Cuantas más dimensiones → más capacidad para representar relaciones semánticas finas, aunque también más costo de cómputo.
  - Imaginar que cada palabra es un punto en un espacio de alta dimensión.
    - Distancia pequeña → significados cercanos ("gato", "perro").
    - Distancia grande → significados lejanos ("gato", "avión").
    - Direcciones → relaciones semánticas ("rey - hombre + mujer ≈ reina").

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - Una palabra no se puede describir con un solo número porque el lenguaje tiene muchos matices.
  - Un embedding de varias dimensiones permite capturar múltiples aspectos de significado.
  - No miramos cada número por separado → lo importante es la posición relativa de los vectores en el espacio.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - ¿Cada elemento del vector representa una caracteristica (cercana o lejana)?
  - Cada elemento del vector es una coordenada.
  - En un embedding, la palabra se representa como un punto en un espacio de muchas dimensiones.
  - Cada número es una coordenada en ese espacio (igual que un punto en 2D tiene (x,y) pero aquí hay cientos de coordenadas).
  - Ejemplo en 2D (fácil de visualizar): (en 4D o más ya es imposible visualizarlos)
    - "gato" → (2, 1)
    - "perro" → (2.2, 1.1)
    - "avión" → (-3, 5)
    - “Gato” y “perro” quedan cerca en el plano, “avión” queda lejos.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Embeddings
  - ¿Cada coordenada = representa una característica?
  - No necesariamente.
    - No es que el primer número sea “animal”, el segundo “pelaje” o el tercero “doméstico”.
  - Los modelos no aprenden significados humanos directos, sino patrones estadísticos del lenguaje.
  - Lo que buscamos es la combinación de todas las coordenadas que ubica cada palabra en un lugar donde la distancia y la dirección reflejan similitud semántica.
  - Si solo tenemos 1 dimensión (una línea), solo podemos decir “más a la izquierda” o “más a la derecha”.
  - Con 2 dimensiones (un plano), ya podemos distinguir norte-sur y este-oeste.
  - Con 300 dimensiones, cada objeto (palabra) se ubica de forma que refleja muchos matices de similitud al mismo tiempo.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Transformers
  - Arquitectura de red neuronal (2017, “Attention is All You Need”) que revolucionó NLP.
  - Usa mecanismo de atención para procesar texto en paralelo y capturar relaciones de largo alcance.
  - Ejemplo: en “El gato duerme en la cama”, el modelo entiende que “cama” se relaciona más con “duerme” que con “El”.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- BERT (Bidirectional Encoder Representations from Transformers)
  - Modelo de lenguaje basado en la parte encoder del Transformer (Google, 2018).
  - Bidireccional: mira a la izquierda y a la derecha del token al mismo tiempo.
  - Ejemplo:
    - Frase: “El ___ duerme en la cama”
    - BERT puede predecir que la palabra faltante es “gato”.
    - Uso: clasificación, extracción de entidades, búsqueda semántica, Q&A.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Loss (Función de pérdida)
  - Medida de qué tan mal lo está haciendo el modelo.
  - Matemáticamente: diferencia entre predicción y valor real.
  - Ejemplo:
    - Predicción del modelo: “positivo” con 0.7 de confianza.
    - Etiqueta real: “positivo” (=1).
    - Loss (ej. cross-entropy) = número positivo → mientras más chico, mejor.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Backpropagation
  - Algoritmo para ajustar los pesos de la red.
  - Proceso:
    - Calcular la pérdida (loss).
    - Propagar hacia atrás el gradiente $(\frac{\partial \text{loss}}{\partial w})$.
    - Ajustar cada peso en dirección de menor error.
  - Ejemplo intuitivo:
    - como aprender a tirar flechas → cada vez que se falla al blanco se corrige el lanzamiento en la dirección contraria al error.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Pieza importante del aprendizaje en redes neuronales (incluyendo embeddings, Transformers, BERT, etc.).
  - Matemáticamente: el gradiente de una función f(w) respecto a sus parámetros w es un vector de derivadas parciales.

$$\nabla_{\mathbf{w}} f = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_n} \right]$$

  - En ML: mide cómo cambia la pérdida (loss) si ajusto un poquito cada peso.
  - El gradiente apunta en la dirección de máxima subida de la función de pérdida.
  - Para minimizar la pérdida, caminamos en la dirección opuesta al gradiente.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Ejemplo con embeddings. Supongamos un embedding simple en 2D para la palabra "gato":

$$\text{vec}(\text{"}gato\text{"}) = (0.2, 0.5)$$

  - Si el modelo predice mal, la pérdida $\mathcal{L}$ depende de esos valores.
  - El gradiente puede resultar en algo como:

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2} \right) = (-0.3, 0.1)$$

  - Si aumento w_1, la pérdida sube (porque el gradiente es negativo, conviene restar y hacerlo crecer).
  - Si aumento w_2, la pérdida baja (gradiente positivo, conviene restar y hacerlo más chico).

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Gradiente
  - Regla básica de descenso por gradiente: $w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \nabla \mathcal{L}$
  - Ejemplo numérico:
    - w=(0.2,0.5)
    - Gradiente = (−0.3,0.1)
    - Learning rate = η=0.1
    - W_{nuevo}=(0.2,0.5)−0.1·(−0.3,0.1)=(0.23,0.49)
  - El embedding de “gato” se mueve en el espacio vectorial para reducir la pérdida.
  - El gradiente dice cómo modificar cada peso para que el modelo se equivoque menos.
  - Como un “mapa de pendientes” que guía al entrenamiento.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Optimizador
  - Regla que usan los gradientes para actualizar los pesos.
  - Ejemplo:
    - Regla básica (SGD): $w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w}$
    - η es la tasa de aprend
    - η es un número positivo que controla la velocidad de aprendizaje del modelo.
    - Es un hiperparámetro → valores que decidimos antes del entrenamiento y que controlan cómo aprende el modelo.
    - Se define por diseño. En trabajos de NLP con Adam, se suele usar $\eta = 10^{-3}$(0.001) como valor inicial recomendado.
      - Puede ajustarse experimentalmente según el comportamiento del entrenamiento.
      - O usar usan técnicas como grid search, random search o Bayesian optimization para elegirlo automáticamente.

η Muy baja → lento.
η Muy alta → inestable.
η Óptima → balance entre velocidad y estabilidad.

Repaso de Matemáticas
Analicemos un ejemplo

- Terminología *(más adelante veremos cada uno de estos temas en detalle)*
- Adam (Adaptive Moment Estimation)
  - Un optimizador avanzado muy usado en Deep Learning.
  - Adapta automáticamente la tasa de aprendizaje para cada peso usando promedios de gradientes (momentum).
  - Intuición: en lugar de dar pasos iguales, da pasos “inteligentes” → más grandes en direcciones seguras, más chicos donde hay oscilaciones.
  - Entrena redes neuronales más rápido y estable que SGD puro.

Repaso de Matemáticas
Analicemos un ejemplo

- Paso1 - Entrada: Texto en lenguaje humano
  - "El gato duerme en la cama"
- Paso 2 - Tokenización
  - El texto se convierte en tokens (piezas de palabras).
  - Con tokenizador de sub-palabras (BPE, WordPiece).
  - Cada token se asocia a un índice en un vocabulario.
  - ["El", "gato", "duer", "me", "en", "la", "cama"]
- Paso 3 - Embeddings
  - Cada token se transforma en un vector en R^{d}.. Ejemplo (d=4)
  - vec("*gato*")=[0.23,−0.11,0.87,0.45]
  - La oración es ahora 1 matriz: cada fila es un token, cada columna una dimensión.

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Esto va al corazón de cómo un modelo de NLP aprende representaciones de palabras (embeddings).
- Embeddings (vectores de palabras)
  - Cuando transformamos un token en un vector en R^d, cada palabra queda representada como: vec("gato")=[w_1,w_2,...,w_d]
  - Los números w_i son los pesos que definen la posición de la palabra en el espacio vectorial.
- Entonces: ¿Cómo se definen esos pesos?
  - 2 grandes enfoques:
    - Embeddings preentrenados (no se entrena desde cero)
    - Embeddings entrenados junto con el modelo

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Embeddings preentrenados (no se entrena desde cero)
  - Se entrenan sobre grandes corpus de texto (Wikipedia, noticias, libros).
    - Ejemplos: Word2Vec, GloVe, FastText, BERT embeddings.
  - Cada palabra obtiene un vector que refleja similitud semántica.
    - “gato” y “perro” tendrán vectores cercanos.
    - “gato” y “avión” estarán lejos.
  - Estos pesos vienen “listos” y se usan en el modelo como tablas de búsqueda.
  - Los pesos ya están definidos por otro entrenamiento.(¿?)

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Embeddings entrenados junto con el modelo
  - En redes neuronales (ej. Transformer, LSTM), los embeddings son parámetros inicializados aleatoriamente.
  - Al entrenar la red, los pesos se ajustan mediante backpropagation.
  - Se aprende automáticamente qué valores de los vectores hacen que el modelo minimice la pérdida (loss).
  - Aquí el modelo define los pesos a medida que aprende la tarea.
- Cómo se almacenan?

$$E \in \mathbb{R}^{|V| \times d}$$

  - Se usa una matriz de embeddings:
    - $|V|$ = tamaño del vocabulario (cantidad de palabras/tokens).
    - d = dimensión de cada vector.
    - Cada fila E_i corresponde al embedding del token i.
  - Cuando se encuentra un token en el texto, se reemplaza por su fila correspondiente en *E.*

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Ejemplo: *(|V|=5,d=3)*

$$E = \begin{bmatrix} 0.2 & -0.1 & 0.9 \\ \text{(token 1: ”el”)} & & \\ -0.3 & 0.5 & 0.7 \\ \text{(token 2: ”gato”)} & & \\ 0.8 & 0.1 & -0.4 \\ \text{(token 3: ”duerme”)} & & \\ \cdots & & \end{bmatrix}$$

  - Los pesos no tienen un significado individual.
    - El conjunto completo de coordenadas es lo que captura relaciones semánticas y sintácticas.
    - Ejemplo: *vec("rey")−vec("hombre")+vec("mujer")≈vec("reina")*
  - Esto se debe a la forma en que el entrenamiento ajusta esos pesos en miles de dimensiones.
  - Los “pesos” de cada palabra = sus coordenadas en el espacio vectorial. Se obtienen de una matriz de embeddings.
  - Pueden venir preentrenados (Word2Vec, GloVe, BERT) o se aprenden durante el entrenamiento.
  - Capturan significados relativos (similitud, analogía), no un “atributo humano” en cada coordenada.

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- ¿ Quién define los pesos del entrenamiento ?
- Aquí tenemos el dilema del huevo y la gallina ...
  - ¿De dónde salen los pesos iniciales de los vectores si todavía no entrenamos?
  - ¿Y cómo sabe el modelo cómo ajustarlos?
- Inicialización de los pesos
  - Al crear una matriz de embeddings $E \in \mathbb{R}^{|V| \times d}$
  - Cada fila (embedding de una palabra) se inicializa con valores aleatorios pequeños (normalmente distribuciones gaussiana o uniforme).
  - Al principio los vectores no tienen ningún significado.
  - Ejemplo (para vocabulario de 5 palabras, dimensión 3):

$$E = \begin{bmatrix} 0.02 & -0.01 & 0.03 \\ -0.04 & 0.05 & -0.02 \\ 0.01 & 0.07 & -0.06 \\ \cdots & & \end{bmatrix}$$

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Proceso de entrenamiento
  - Aquí entra la “gallina”: el algoritmo de aprendizaje.
  - Se define una función de pérdida (loss).
  - Ejemplo: en un clasificador de sentimiento, penaliza si el modelo predice negativo cuando el texto era positivo.
  - Con cada ejemplo de entrenamiento, el modelo calcula:

$$\text{pérdida} = \mathcal{L}(predicción, etiqueta\ real)$$

  - Se aplica backpropagation:
  - La red calcula las derivadas *de la pérdida respecto a cada peso (incluidos los embeddings).*
  - El optimizador (ej. SGD, Adam) actualiza los pesos un poco en la dirección que reduce la pérdida:

$$w_{\text{nuevo}} = w_{\text{viejo}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w}$$

  - los vectores empiezan a moverse en el espacio hasta organizarse según la tarea.

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Resultado del aprendizaje, Después de muchas iteraciones:
  - Palabras que aparecen en contextos similares terminan con vectores cercanos.
  - Palabras opuestas (positivo ↔ negativo) se separan.
  - Relaciones semánticas aparecen “solas” porque ayudan a minimizar la pérdida.
- ¿Entonces, quién define los pesos ?
  - Al principio: nadie → son aleatorios.
  - Durante el entrenamiento: los ajusta el algoritmo de optimización guiado por los datos y la función de pérdida.
  - Al final: los pesos reflejan la estructura estadística del lenguaje en los datos que vio el modelo.
  - El “huevo” son los pesos aleatorios iniciales.
  - La “gallina” es el entrenamiento con datos que va corrigiendo y “dando forma” a esos pesos.

Repaso de Matemáticas
Analicemos el Paso 3
¿Primero el “Huevo o la Gallina” ?

- Nadie escribe los pesos de las palabras a mano.
- Los embeddings empiezan al azar.
- El entrenamiento supervisado o auto-supervisado los acomoda para capturar significados y relaciones.

Repaso de Matemáticas
Analicemos un ejemplo

- Paso 4 - Positional Encoding
  - Como los Transformers no saben de orden por sí mismos, se agrega un vector de posición a cada embedding.
  - Así, el modelo distingue "gato duerme" de "duerme gato".
- Paso 5 - Mecanismo de Atención
  - Cada token genera 3 vectores:
    - Q = Query
    - K = Key
    - V = Value

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

  - La atención mide cuánto “mira” un token a los otros
    - Si el token es "duerme", probablemente preste más atención a "gato". Si el token es "cama", prestará atención a "en la".
  - Esto permite capturar dependencias largas en la oración.

Repaso de Matemáticas
Analicemos un ejemplo

- Paso 6 - Capas de Transformer
  - Se aplican varias capas de atención + feed-forward (redes densas).
  - Cada capa produce una representación contextualizada de los tokens.
  - Ejemplo: el vector de "banco" será diferente si hablamos de sentarse o dinero.
- Paso 7 - Salida
  - En GPT (generación de texto):
    - El modelo predice el siguiente token con probabilidad:
      - $P(token_{t+1} | tokens\ anteriores)$
  - En BERT (comprensión de texto): el modelo puede clasificar la oración, rellenar una palabra faltante, responder preguntas, etc.
- Paso 8 - Resultado
  - El Transformer devuelve: texto generado (ej: ChatGPT).
  - Etiquetas de clasificación (ej: sentimiento).
  - Respuestas a preguntas (ej: BERT en Q&A).

Aprendizaje de Máquina
(ML – Machine Learning)

- El objetivo es aprender patrones y relaciones a partir de datos sin tener que programar explícitamente para cada tarea.

- No indicamos paso a paso qué hacer.

- Entrenamos un modelo que ajusta sus parámetros internos para predecir, clasificar o tomar decisiones.

Aprendizaje de Máquina
(ML – Machine Learning)

- El sistema aprende una función o relación a partir de ejemplos para luego generalizar a datos nuevos.

- Un modelo puede entenderse como una función.

Aprendizaje de Máquina
(ML – Machine Learning)

- ¿ Qué es un modelo ?
  - Representación matemática o computacional que captura la relación entre las entradas (caracteŕisticas, variables) y las salidas (predicciones o decisiones) a partir de datos de entrenamiento.

$\hat{y}$ $$\hat{y} = f(\mathbf{x}; \theta)$$

  - x: vector de entradas o variables independientes (ej. temperatura, humedad)
  - θ: parámetros del modelo (valores que se ajustan durante el entrenamiento).
  - f: estructura matemática que transforma las entradas en salidas.
  - Salida: predicción ($\hat{y}$) o decisión que el modelo genera.

Aprendizaje de Máquina (ML – Machine Learning)
Etapas

- Definición del problema
  - ¿Qué queremos predecir, clasificar o detectar?
  - Tipo de aprendizaje (supervisado, no supervisado, por refuerzo).

- Recolección de datos
  - Obtener datos relevantes y suficientes.
  - Fuentes: sensores, bases de datos, APIs, experimentos.

Aprendizaje de Máquina (ML – Machine Learning)
Etapas

- Preprocesamiento de datos
  - Limpieza de datos: manejo de valores faltantes, outliers, escalado.
  - Transformación de variables: normalización, codificación de categorías.

- Selección del modelo
  - Elegir un algoritmo adecuado: regresión, SVM, árboles, redes neuronales, etc.

Aprendizaje de Máquina (ML – Machine Learning)
Etapas

- Entrenamiento del modelo
  - Ajustar los parámetros θ para minimizar el error.
  - División en conjuntos de entrenamiento y validación.

- Evaluación del modelo
  - Medir el rendimiento en datos no vistos (test set).
  - Usar métricas adecuadas (accuracy, F1, MSE, etc.).

Aprendizaje de Máquina (ML – Machine Learning)
Etapas

- Ajuste y optimización
  - Mejorar el modelo ajustando hiperparámetros, cambiando arquitectura o más datos.

- Despliegue y monitoreo
  - Implementar el modelo en producción.
  - Supervisar su rendimiento y reentrenar si es necesario.

Aprendizaje de Máquina (ML – Machine Learning)
Representación Matemática

Representación matemática del proceso. El objetivo es aprender una función f a partir de ejemplos (xi,yi)

$$\hat{y}_i = f(\mathbf{x}_i; \theta)$$

θ es el mejor conjunto de parámetros que vamos a encontrar. Son los parámetros que hacen que el modelo funcione mejor según un criterio.*

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$$

$$\ell(y_i, \hat{y}_i)$$

Los parámetros θ son los valores que el algoritmo puede ajustar para aprender (en una regresión lineal, estos son los coeficientes de la recta). se ajustan minimizando un error global en el conjunto de datos

N es el número total de datos de entrenamiento, la cantidad de ejemplos que tenemos. 1/N es para obtener el error promedio, para que la cantidad total no dependa del nro de ejemplos

Es la función de pérdida o error para el ejemplo i.Mide qué tan diferente es la predicción del valor real.Puede ser el error cuadrático o la diferencia absoluta.

Aprendizaje de Máquina (ML – Machine Learning)
Representación Matemática

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$$

El operador arg min θ indica que buscamos los parámetros θ que minimizan ese error promedio.

Buscamos encontrar los parámetros θ* que hagan que la función f(x;θ)f prediga lo más cerca posible de los valores reales y, minimizando el error promedio sobre todos los datos.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida

- La función de pérdida (ℓ) es una fórmula matemática que mide qué tan mal está prediciendo nuestro modelo.
  - Si el valor predicho $\hat{y}_i$ está muy cerca del valor real y, la pérdida es pequeña. Si está muy lejos, la pérdida es grande.
- En muchos modelos clásicos (regresión lineal por ej.) la minimización de la función de pérdida se puede hacer:
  1. Derivando y
  2. buscando el punto donde la derivada es cero (corresponde a un mínimo - o un mínimo local),
  3. y obtengo un mínimo analítico o solución cerrada.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida

- Ejemplo de Función de pérdida (ℓ) para problemas de regresión, Error Cuadrático Medio (MSE):

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- N = número de datos
- y_i = valor real
- $\hat{y}_i = f(\mathbf{x}_i;\theta)$ = valor predicho por el modelo
- θ = parámetros del modelo

- El objetivo del entrenamiento es encontrar los parámetros θ que minimicen la función de pérdida.
  - "buscar los valores de θ que hacen que la función de pérdida sea lo más pequeña posible".

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta)$$

  - En cálculo, sabemos que para encontrar el mínimo de una función suave podemos derivar la función con respecto a sus parámetros (θ) e igualar la derivada a cero.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida

- Igualar la derivada a cero.
- Resolver para θ.

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0$$

- Si la pérdida tiene solución analítica
  - Podemos derivar y resolver directamente.

- Si la pérdida no tiene solución simple (como en redes neuronales)
  - Usamos métodos iterativos como descenso por gradiente, que ajusta θ en la dirección contraria al gradiente para acercarnos al mínimo.

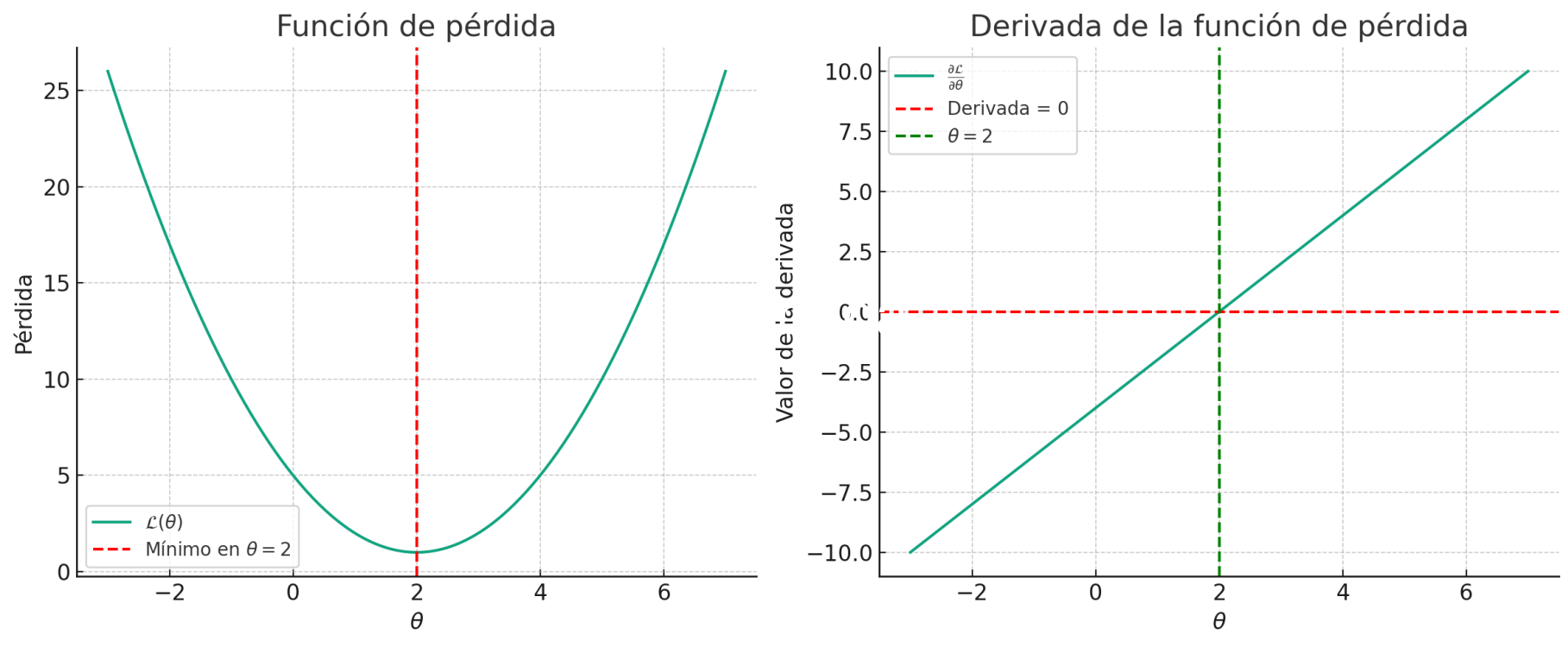
Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida

La función de pérdida mide el error.

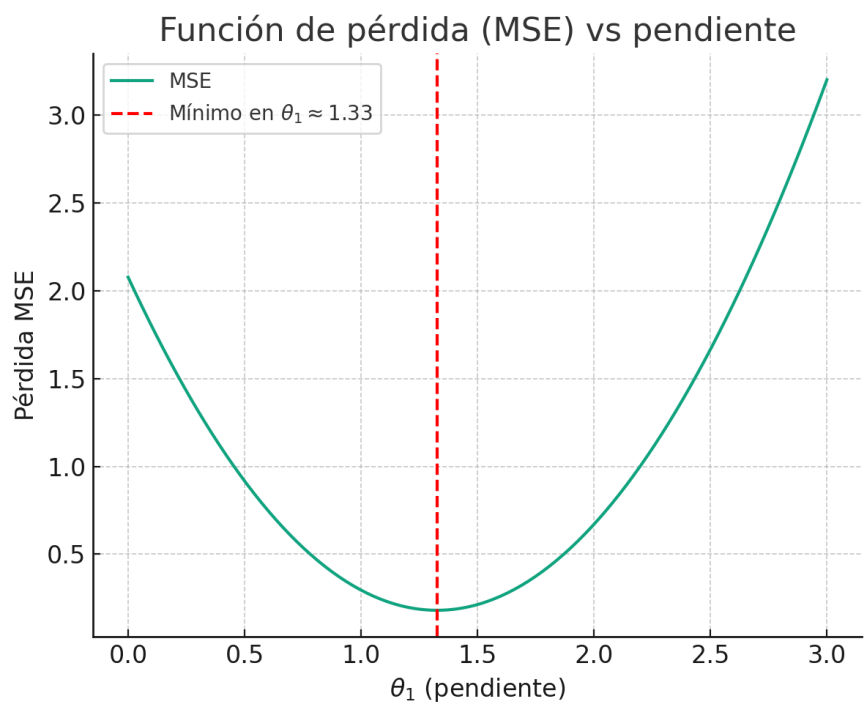
Minimizarla significa encontrar el mejor modelo.

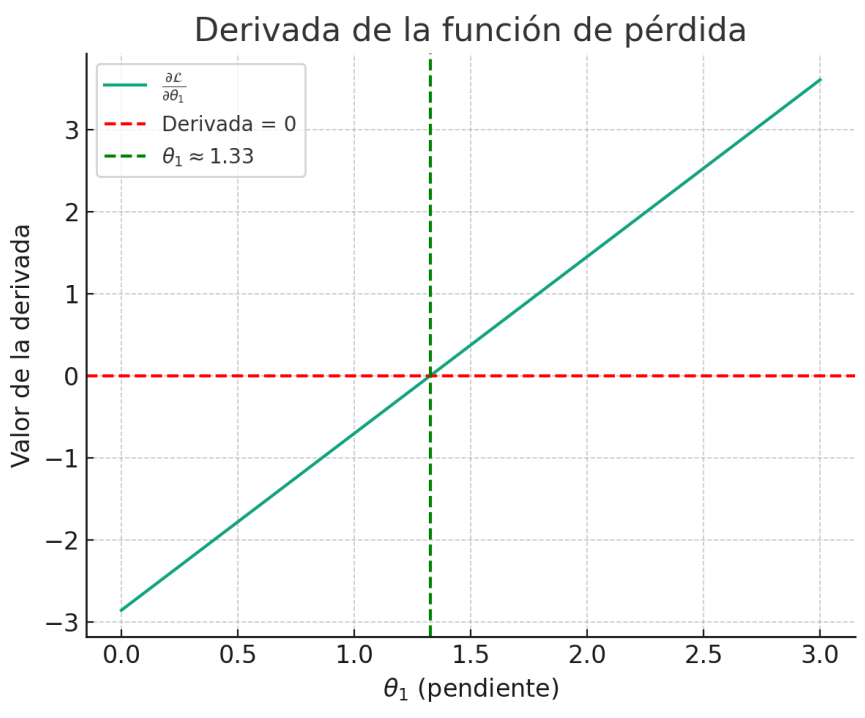
La derivada nos dice en qué dirección mover los parámetros para reducir ese error.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida

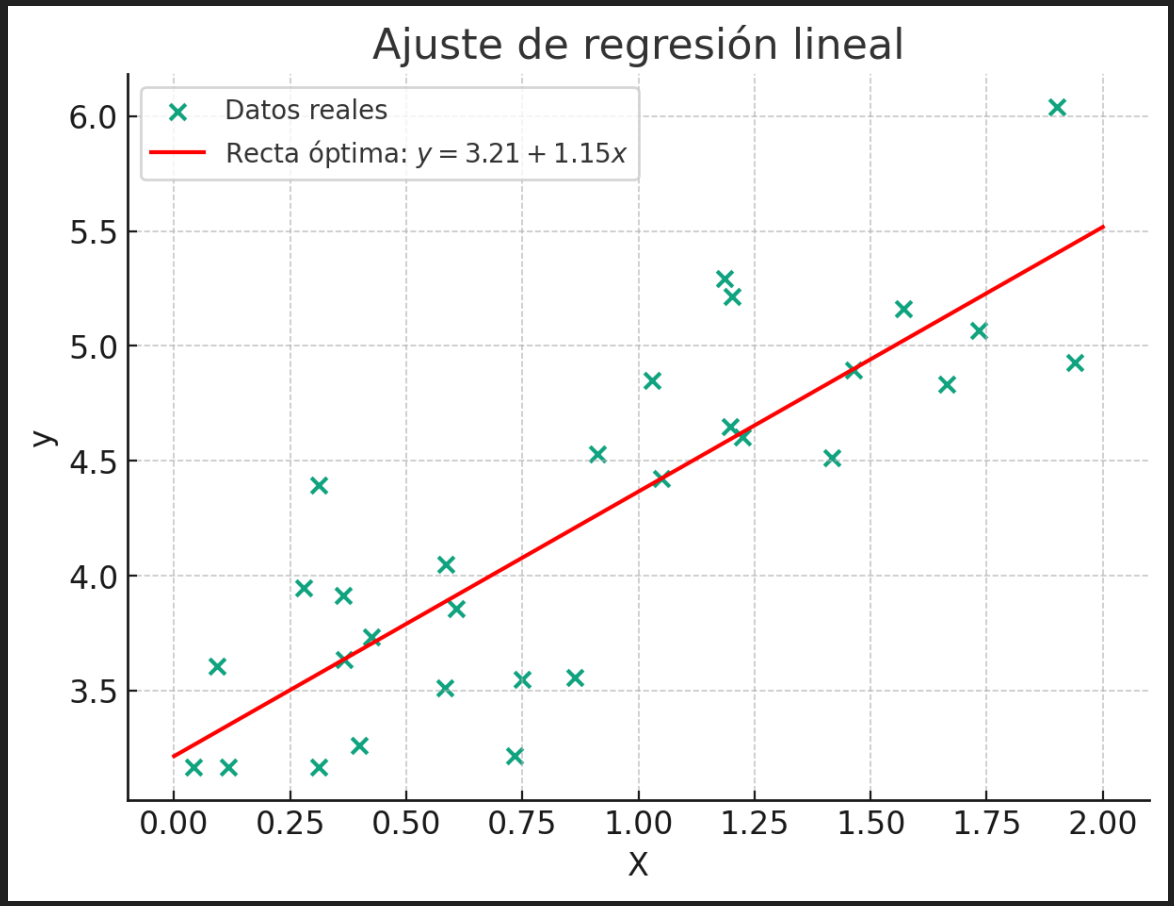


Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida





Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida



- Los puntos verdes son los datos reales.
- La línea roja es la ecuación que minimiza la función de pérdida (MSE), obtenida derivando y resolviendo.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Caso de Uso

- Simular el siguiente escenario:
  - En un invernadero el objetivo principal es mantener las condiciones óptimas para el cultivo, y el consumo energético es una variable derivada.
- Tenemos un rango óptimo de temperatura para la planta:
  - $T_{min} \leq T_{ideal} \leq T_{max}$
- El consumo energético depende de cuánta diferencia hay entre la temperatura medida y la temperatura ideal.
- El modelo de ML puede aprender a predecir cuánto tiempo y con qué potencia debe trabajar la climatización para mantener ese rango.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Caso de Uso

- Función objetivo
  - Si la temperatura medida Tm está fuera del rango óptimo, hay que corregirla.
  - Podemos definir la variable objetivo (lo que el modelo debe aprender) como la potencia necesaria para llevar Tm a Tideal:
  - $Pnecesaria = k \cdot |Tideal - Tm|$
  - k es un factor que representa la capacidad térmica del invernadero y la eficiencia del sistema.
- Función de pérdida
  - El modelo puede ser entrenado para minimizar el error entre la potencia predicha y la potencia real necesaria:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(P_i - \hat{P}_i \right)^2$$

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Caso de Uso

- Aplicación práctica
  - El modelo se entrena con datos reales de sensores y potencia aplicada.
- Una vez entrenado, el sistema puede:
  - Leer temperatura actual.
  - Calcular potencia necesaria para alcanzar la temperatura ideal.
  - Ajustar el calefactor o enfriador automáticamente.
- Esto permite que el criterio principal sea el bienestar de la planta y el consumo energético se optimice de forma secundaria, evitando daños por temperaturas fuera del rango óptimo.

Aprendizaje de Máquina (ML – Machine Learning)
Existe una formula predecida o sólo valores ?

- Fórmula predefinida (modelo físico o empírico)
  - Si conocemos las leyes físicas y parámetros del sistema, podemos usar una ecuación predecida.
  - En climatización de invernaderos, el cálculo de energía necesaria para cambiar la temperatura sigue una fórmula basada en calor específico:

$$Q = m \cdot c_p \cdot \Delta T$$

  - Q = energía necesaria (kJ)
  - m = masa de aire dentro del invernadero (kg)
  - c_p = calor específico del aire (~1.005 kJ/kg·°C)
  - ΔT = diferencia entre temperatura ideal y medida (°C)

- Esto nos da una predicción directa de energía sin necesidad de entrenar un modelo, pero requiere conocer dimensiones, pérdidas de calor, infiltraciones, eficiencia, etc.

Aprendizaje de Máquina (ML – Machine Learning)
Existe una formula predecida o sólo valores ?

- Sólo valores (modelo de ML puro)
  - Si no tenemos la fórmula exacta o el sistema es demasiado complejo, usamos Machine Learning para aprender la relación directamente de los datos:
  - Entradas: temperatura medida, humedad, radiación solar, etc.
  - Salida: potencia o tiempo de calefacción requerido.
  - El modelo aprende una función aproximada $\hat{P}$ = *f(sensores)* que minimiza la función de pérdida.
- Aquí no partimos de una ecuación física, sino que el modelo aprende la función a partir de ejemplos.
- Lo ideal puede ser combinar ambos (modelo híbrido físico–ML) para mejorar precisión y robustez.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Cómo se deriva?

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(P_i - \hat{P}_i\right)^2$$

- Buscamos el conjunto de parámetros θ que minimizan el error medio cuadrático entre la potencia real y la predicha: $\theta^* = \arg\min_{\theta} \mathcal{L}(\theta)$

- Modelo lineal elegido: $\hat{P}_i = \theta_0 + \theta_1 T_i$

  - T_i es la temperatura medida por el sensor.
  - θ_0 y θ_1 son los parámetros a aprender.

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Cómo se deriva?

- Sustitución en la función de pérdida:

$$\mathcal{L}(\theta_0, \theta_1) = \frac{1}{N} \sum_{i=1}^{N} [P_i - (\theta_0 + \theta_1 T_i)]^2$$

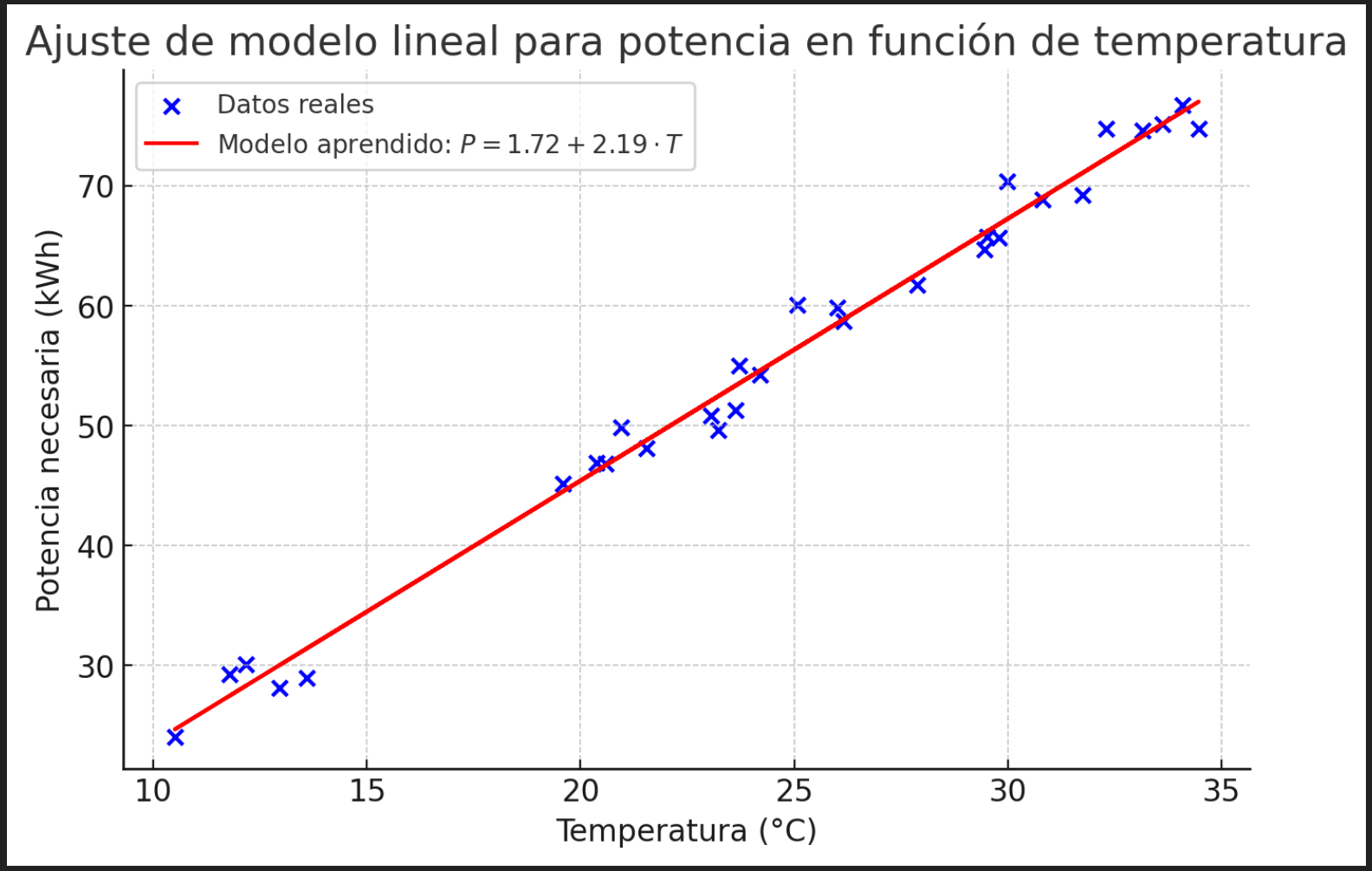
- Derivadas parciales para encontrar el mínimo:
  - Con respecto a θ_0:

$$\frac{\partial \mathcal{L}}{\partial \theta_0} = -\frac{2}{N} \sum_{i=1}^{N} [P_i - (\theta_0 + \theta_1 T_i)] = 0$$

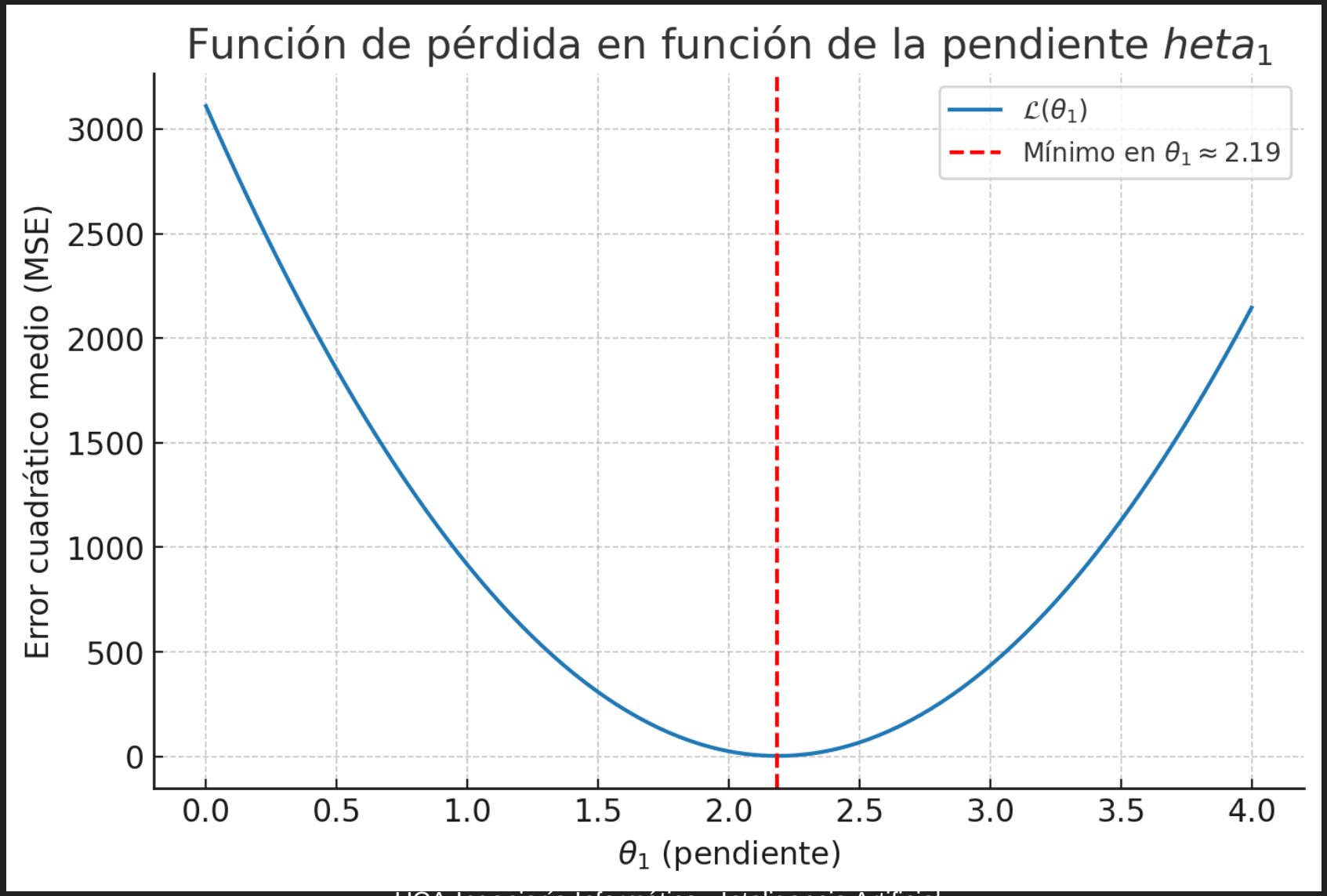
  - Con respecto a θ_1:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -\frac{2}{N} \sum_{i=1}^{N} T_i [P_i - (\theta_0 + \theta_1 T_i)] = 0$$

Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Cómo se deriva?



Aprendizaje de Máquina (ML – Machine Learning)
Función de Pérdida – Cómo se deriva?



Aprendizaje de Máquina
Función de Pérdida – De dónde salen los valores 1.72 y 2.19 ?

- Esos valores son aproximados.
- Salen del proceso de entrenamiento del modelo.
- Calculan la pendiente θ_1 y el intercepto θ_0 de la recta que mejor se ajusta a los datos (T_i,P_i) por mínimos cuadrados.

$$\theta_1 = \frac{\sum_{i=1}^{N}(T_i - \bar{T})(P_i - \bar{P})}{\sum_{i=1}^{N}(T_i - \bar{T})^2}$$

$$\theta_0 = \bar{P} - \theta_1 \cdot \bar{T}$$

Aprendizaje de Máquina
Función de Pérdida – De dónde salen los valores 1.72 y 2.19 ?

- Esos valores son aproximados.
- Salen del proceso de entrenamiento del modelo.
- Calculan la pendiente θ_1 y el intercepto θ_0 de la recta que mejor se ajusta a los datos (T_i,P_i) por mínimos cuadrados.

$$\hat{P}$$

$$\theta_1 = \frac{\sum_{i=1}^{N}(T_i - \bar{T})(P_i - \bar{P})}{\sum_{i=1}^{N}(T_i - \bar{T})^2}$$

$$\theta_0 = \bar{P} - \theta_1 \cdot \bar{T}$$

Aprendizaje de Máquina

Diferencia entre las fórmulas clásicas y las librerías automáticas de machine learning (ejemplo scikit-learn).

- Método manual (mínimos cuadrados)
  - Se aplican las fórmulas clásicas de estadística para calcular la recta que mejor se ajusta a los datos.
  - Calculamos el promedio de las temperaturas y el promedio de las potencias reales.
  - Medimos cómo varían juntas (covarianza) y cuánto varía la temperatura por sí sola (varianza)
  - Deducimos la pendiente y el punto de inicio de la recta (intercepto).
- Método útil para aprender porque muestra paso a paso cómo se calcula la recta a partir de los datos.
- Sólo funciona bien con una variable (regresión simple). No escala fácilmente a muchos datos o variables.

Aprendizaje de Máquina

Diferencia entre las fórmulas clásicas y las librerías automáticas de machine learning (ejemplo scikit-learn).

- Método scikit-learn (con LinearRegression)
  - Al usar scikit-learn el algoritmo usa herramientas del álgebra lineal para encontrar la mejor recta.

- Este método representa los datos como matrices.
- Técnica matemática llamada ecuación normal para resolver el problema de forma general.
- Puede trabajar con muchas variables, muchos datos y ajustarse a distintos tipos de modelos.
- Valida internamente los datos, detecta errores, optimiza el cálculo, etc.

Aprendizaje de Máquina

Diferencia entre las fórmulas clásicas y las librerías automáticas de machine learning (ejemplo scikit-learn).

- Regresión lineal multivariable (*ecuación normal, permite calcular directamente los parámetros del modelo sin usar iteraciones, como en el descenso por gradiente).*

$$\theta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

$\mathbf{X}$ es la matriz de entradas de tamaño $N \times (n+1)$

$\mathbf{y}$ es el vector columna de salidas reales, de tamaño $N \times 1$

$\boldsymbol{\theta}$ es el vector columna de parámetros del modelo, de tamaño $(n+1) \times 1$

$\mathbf{X}^\top$ representa la transpuesta de la matriz $\mathbf{X}$

$(\mathbf{X}^\top \mathbf{X})^{-1}$ es la inversa del producto matricial, si $\mathbf{X}^\top \mathbf{X}$ es no singular

Aprendizaje de Máquina

Diferencia entre las fórmulas clásicas y las librerías automáticas de machine learning (ejemplo scikit-learn).

- Resumen:
  - Resultado final:
    - En el caso univariable, los resultados son exactamente iguales (misma pendiente e intercepto).
  - Escalabilidad: La solución manual es limitada a una variable. scikit-learn escala bien a muchas variables.
  - Optimización: Para miles de muestras y muchas variables, scikit-learn puede usar algoritmos más eficientes (ej. QR, SVD o Gradiente).
  - Comodidad: scikit-learn maneja internamente validación, manejo de NaNs, matrices, etc.

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- Objetivo:
  - Predecir la geometría óptima de un perfil estructural en steel frame (espesor, sección, etc.) para resistir una determinada presión de viento, sabiendo otras variables del diseño (altura, ancho, separación entre montantes, etc.).

Entrada	Presión + altura + zona
Salida	Espesor e, tipo de perfil C100C100, etc.
Tipo de problema	Ingeniería asistida, Modelo ML Inverso (regresión supervisada)

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- Queremos aprender la función que resuelva el problema.
- Queremos un modelo que, dado el vector de condiciones x (carga de viento y geometría del panel), prediga la geometría óptima del perfil: espesor e (regresión) y/o tipo de sección c (clasificación).
  - *f(P,h,w,s,z)→espesor del perfil e*
    - P: presión de viento (N/m^2)
    - h: altura del panel
    - w: ancho del panel
    - s: separación entre montantes (ej. 0.40 m)
    - z: zona o categoría (puede codificarse)
  - Predecir:
    - e: espesor del perfil (en mm)
    - tipo de perfil si planteamos una clasificación (C90, C100, C150, etc.)

Aprendizaje de Máquina (ML – Machine Learning)
Caso de Uso – Steel Frame

- ¿Cómo obtener los datos para entrenar?
  - Hay 3 opciones:
    - Simulación con software de cálculo: generar combinaciones (pressión, altura, ancho) y registrar qué espesor cumple.
    - Normativas (CIRSOC 303, AISI, etc.): usar tablas de diseño estructural como dataset base.
    - Diseños reales existentes.

$$\mathbf{x} = [P,\ h,\ w,\ s,\ z]$$