Optimización en las Redes Neuronales

Este tema presenta los conceptos básicos de la optimización en las redes neuronales, explorando la importancia de los optimizadores en el entrenamiento de modelos de redes neuronales. Aprenderemos sobre algoritmos de optimización como Gradiente Descendente y Adam, así como sus variantes y aplicaciones.

El Papel Fundamental de la Optimización

Aprendizaje Automático

Los modelos de aprendizaje automático aprenden patrones a partir de datos, mejorando su rendimiento con el tiempo sin programación explícita para cada tarea.

Redes Neuronales

Inspiradas en el cerebro humano, las redes neuronales consisten en capas de neuronas artificiales que permiten el aprendizaje de relaciones complejas en los datos.

Optimización

La optimización ajusta los parámetros de un modelo para minimizar una función de coste, mejorando su precisión y capacidad predictiva.

Optimizadores: Guías del Aprendizaje

1 Optimizador Adam

Combina las ventajas de RMSProp y Momentum, adaptando dinámicamente la tasa de aprendizaje de cada parámetro.

2 Gradiente Descendente

Un algoritmo clásico que utiliza el gradiente para encontrar el mínimo de una función de coste.

3 RMSProp

Ajusta la tasa de aprendizaje de forma adaptativa para cada parámetro, ideal para problemas con gradientes ruidosos.

4 Momentum

Acelera el gradiente descendente acumulando gradientes pasados para moverse más rápido en direcciones consistentes.

Gradiente Descendente: Una Base Sólida

Definición

El gradiente es un vector que indica la dirección de mayor incremento de una función. En optimización, nos movemos en la dirección opuesta para minimizar la función de coste.

Funciones de Coste

Miden el error entre las predicciones del modelo y los valores reales. El objetivo del entrenamiento es minimizar este error.

Variantes

Existen variantes como Batch Gradient Descent, Stochastic Gradient Descent y Mini-batch Gradient Descent, cada una con sus ventajas y desventajas.

Gradiente Descendente:

El Gradiente es sinónimo de pendiente o inclinación, y matemáticamente recibe el nombre de derivada.

Para una función de coste $J(\theta)$, la actualización de los parámetros θ se realiza mediante:

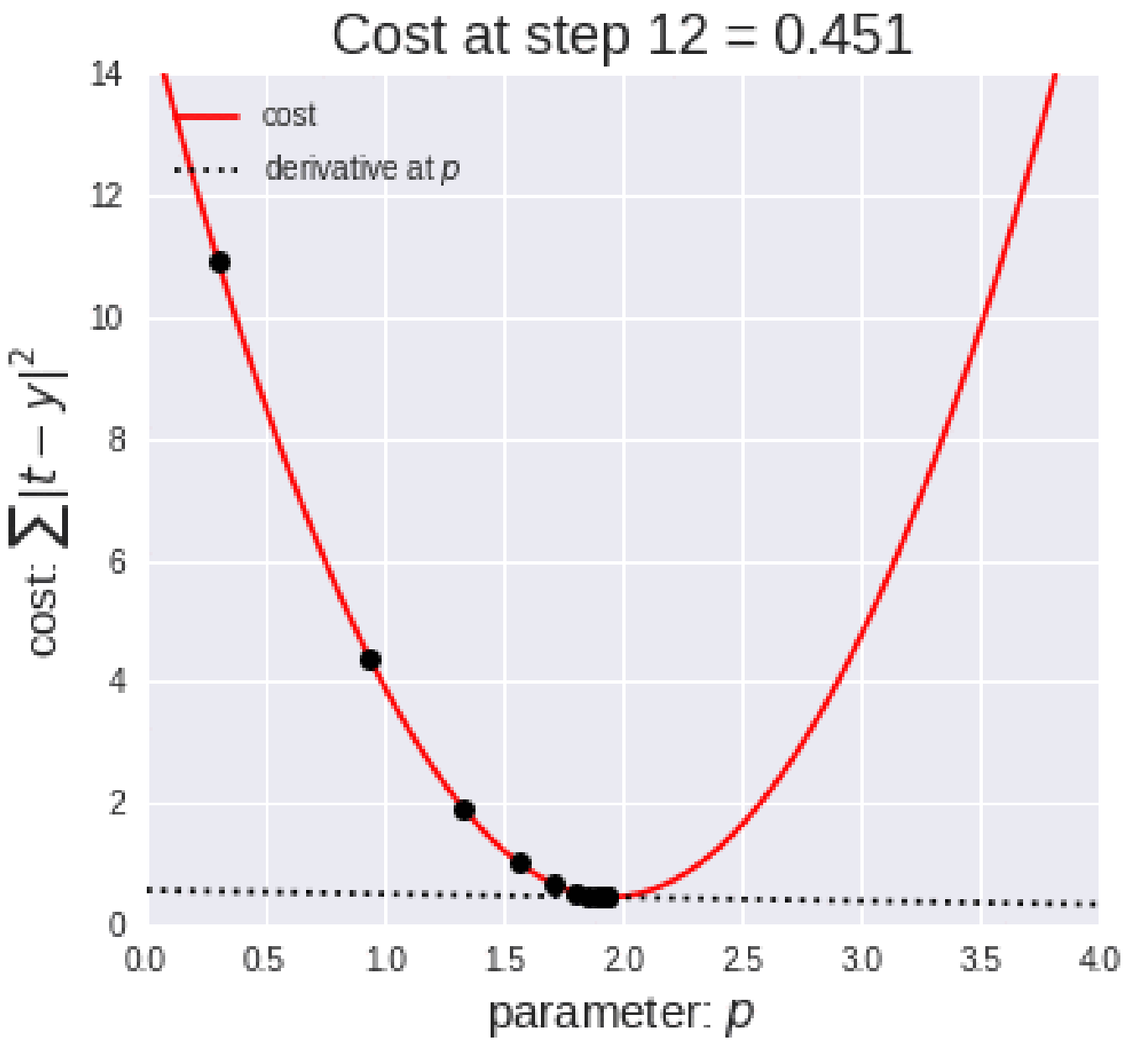
$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t)$$

donde:

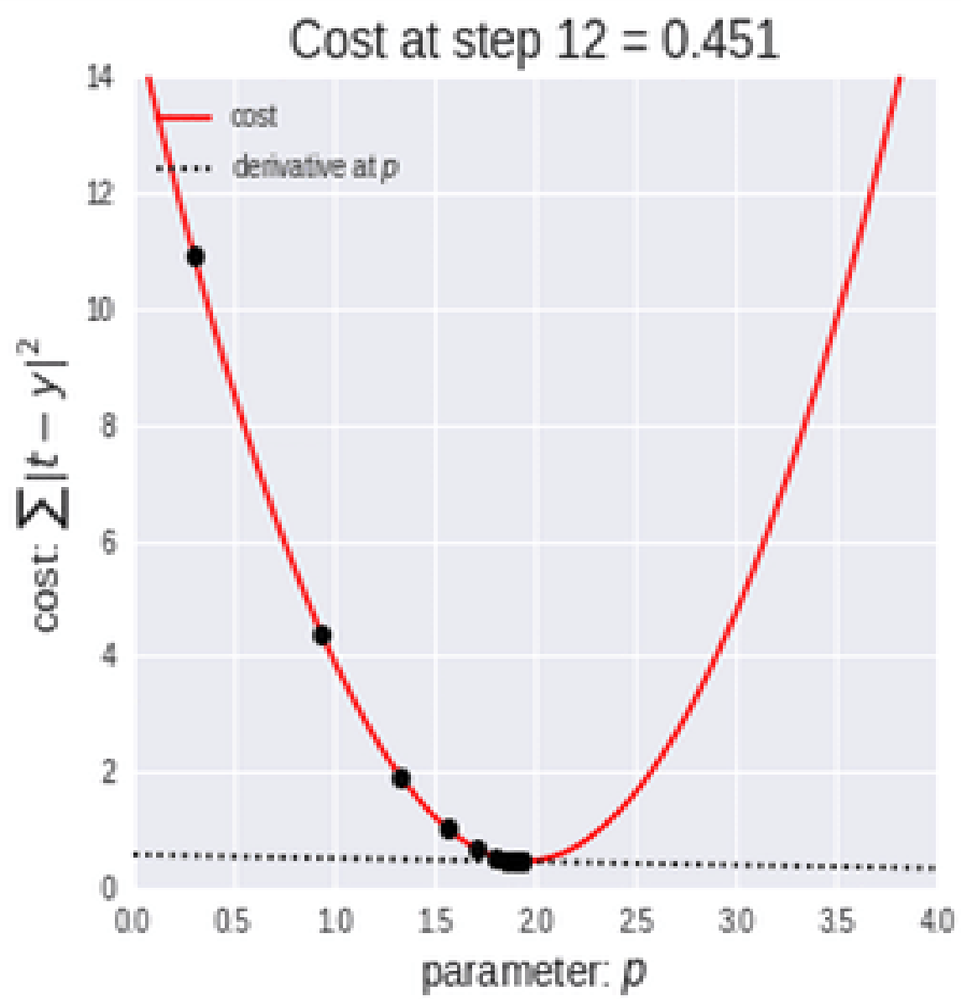
- α es la tasa de aprendizaje,
- $\nabla J(\theta_t)$ es el gradiente de la función de coste respecto a los parámetros en el paso t.

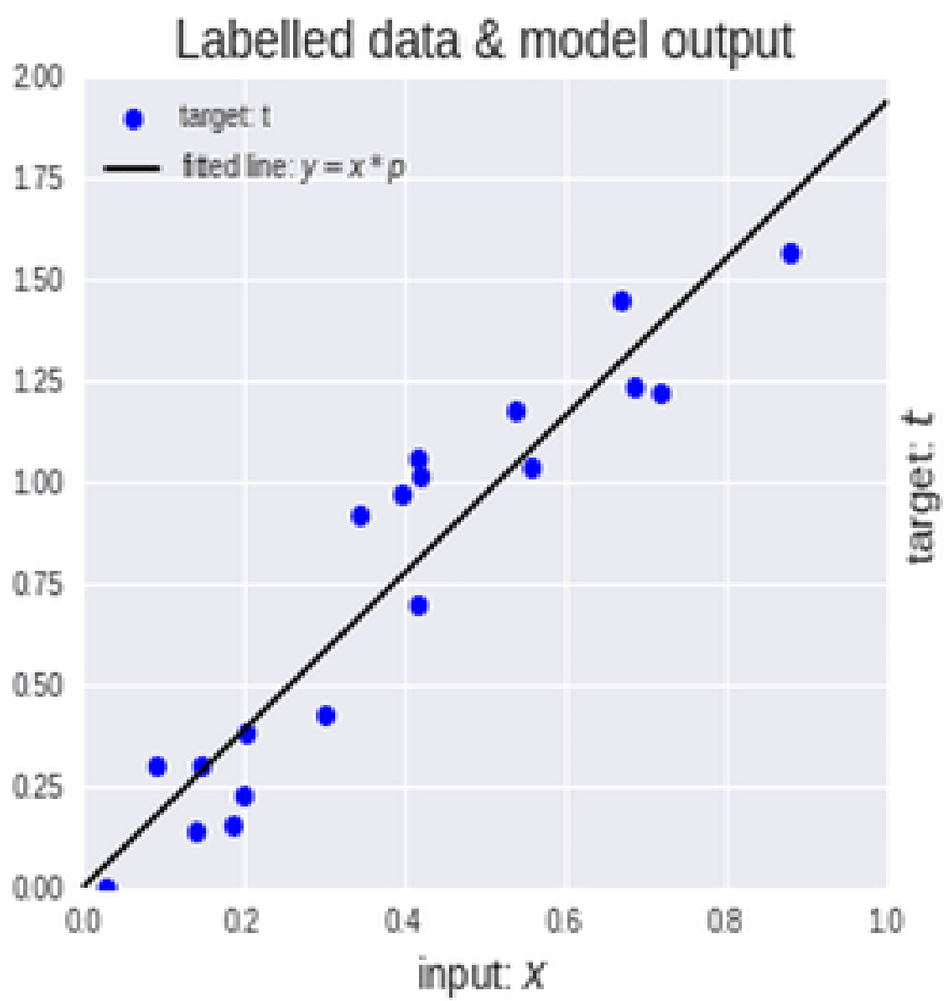
En forma expandida para múltiples parámetros:

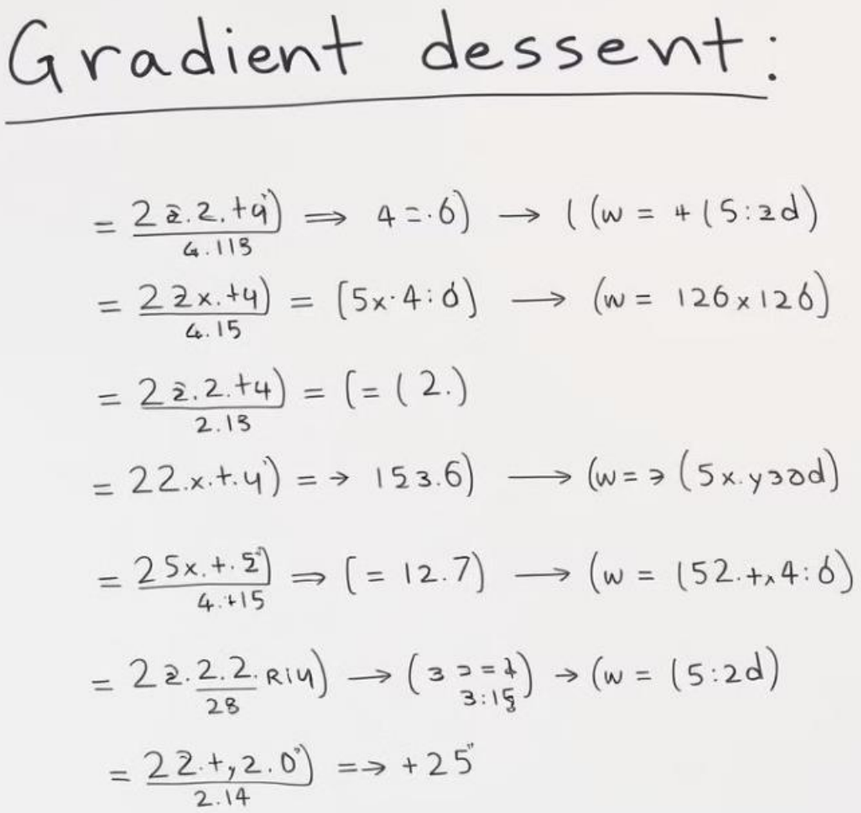
$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$



Gradiente Descendente:







Ejercicio Práctico: Gradiente Descendente

Paso 1

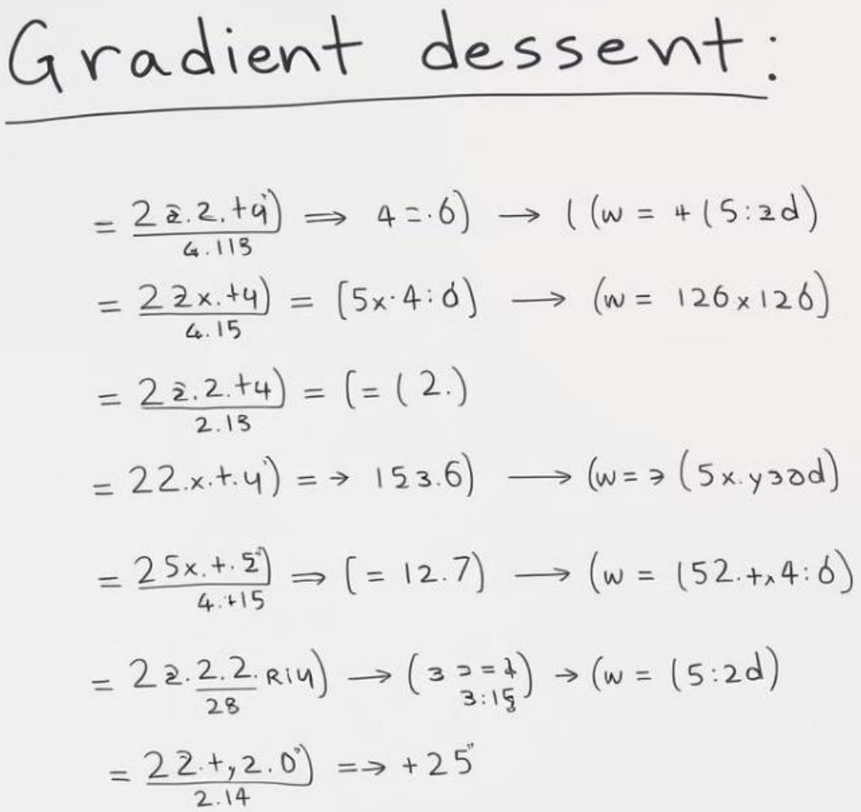
Calcular el gradiente de la función de coste en el punto actual.

Paso 2

Multiplicar el gradiente por la tasa de aprendizaje.

Paso 3

Restar el resultado al punto actual para obtener el nuevo punto.



Ejercicio Práctico: Gradiente Descendente

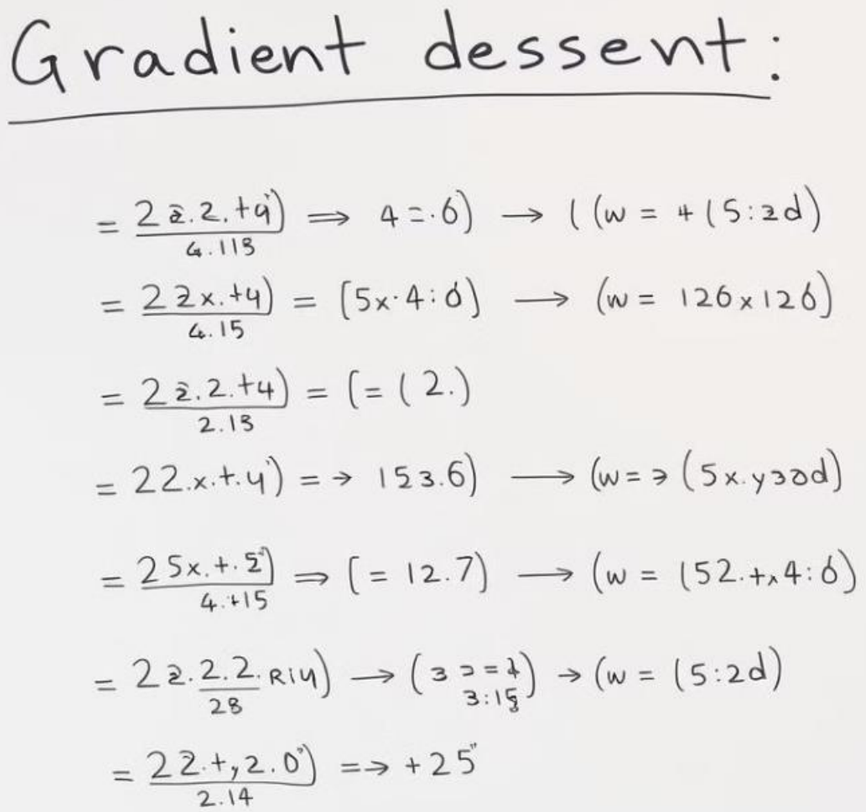
Minimizar la función $f(x) = x^2$ *Con:* ***X0***= 5,

α = 0.1, ***α*** = 0.25, ***α*** = 0.4, ***α*** = 1

Minimizar la función $f(w) = (w-3)^2$ *Con:* ***w0***= 0, ***α*** = 0.1

Iteración	w	$f(w)$	$f'(w)$	w_{nuevo}
1	0	9	-6	$0 - 0.1(-6) = 0.6$
2	0.6	5.76	-4.8	$0.6 - 0.1(-4.8) = 1.08$
3	1.08	3.6864	-3.84	$1.08 - 0.1(-3.84) = 1.464$
4	1.464	2.3594	-3.072	$1.464 - 0.1(-3.072) = 1.771$

Minimizar $f(x, y) = 3x^2 + 2xy + y^2$.



Ejercicio Práctico: Neurona con Gradiente Descendente

1 Definimos la neurona

Supongamos que tenemos una sola neurona con:

- Una entrada x
- Un peso w
- Una salida predicha $y^$
- Un valor real y
- Un valor de bias b

La función de la neurona es:

$$y^ = w \cdot x + b$$

Queremos que $y^$ se aproxime a y, y para ello minimizaremos la función de costo.

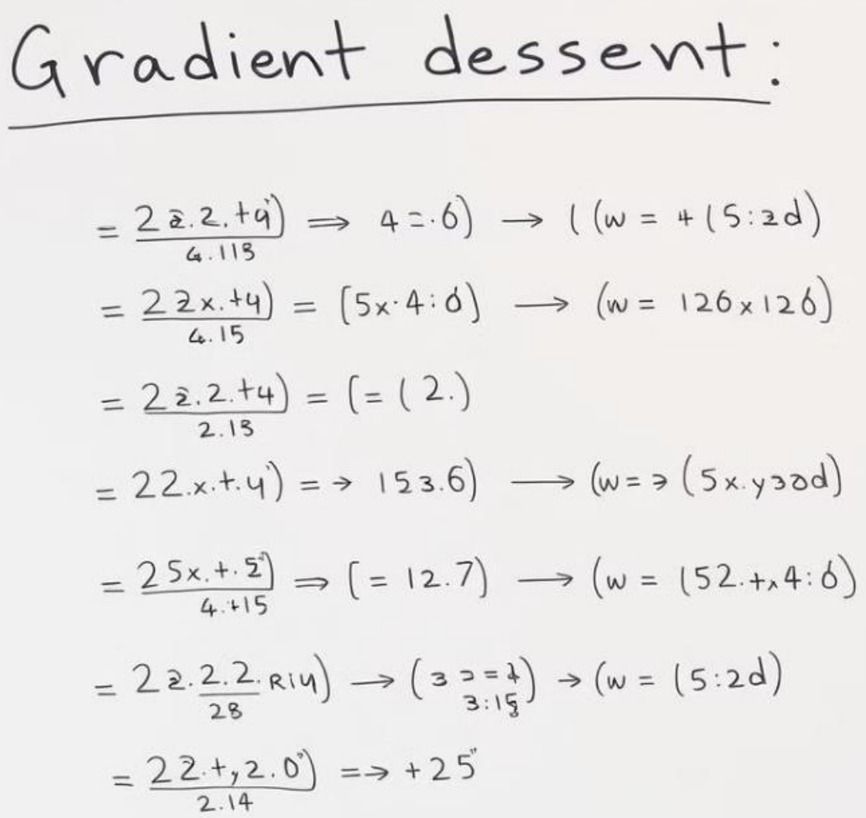
2 Definimos la función de costo

Usaremos el error cuadrático medio (MSE):

$$C(w) = \frac{1}{2}(\hat{y} - y)^2$$

Sustituyendo: $y^ = w \cdot x + b$

$$C(w) = \frac{1}{2}(w * x + b - y)$$



Ejercicio Práctico: Neurona con Gradiente Descendente

3 Aplicamos el Gradiente Descendente

La regla de actualización del peso es:

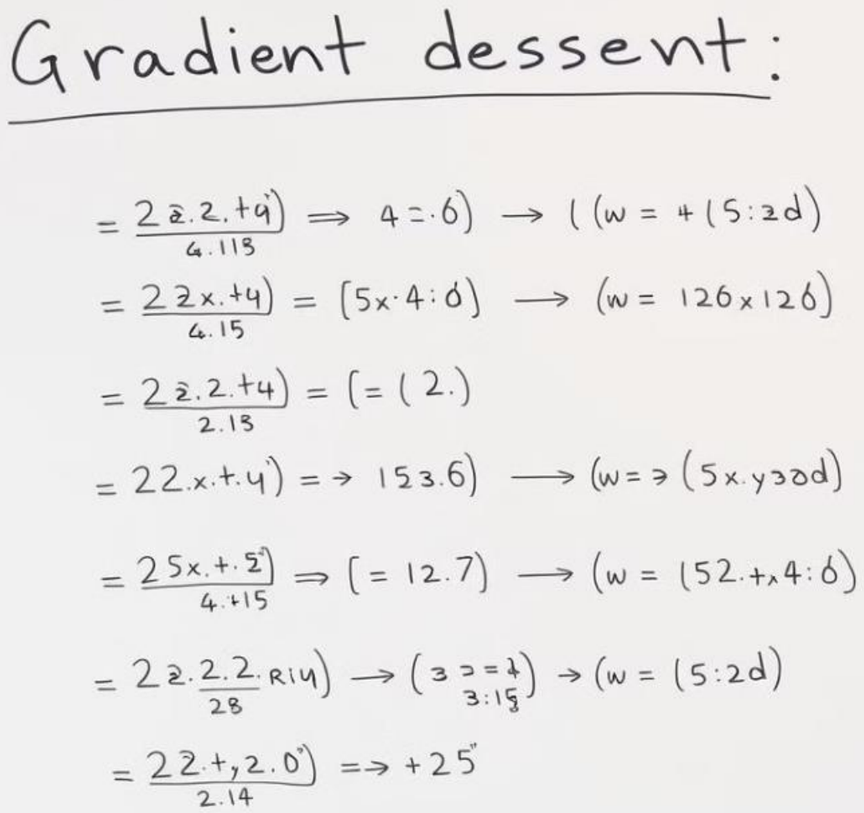
$$w_{\text{nuevo}} = w_{\text{viejo}} - \alpha \cdot \frac{dC}{dw}$$

Calculamos la derivada de la función de costo:

$$\frac{dC}{dw} = x(w * x + b - y)$$

Sustituyéndolo en la ecuación de actualización:

$$w_{nuevo} = w - \alpha * x(w * x + b - y)$$



Ejercicio Práctico: Neurona con Gradiente Descendente

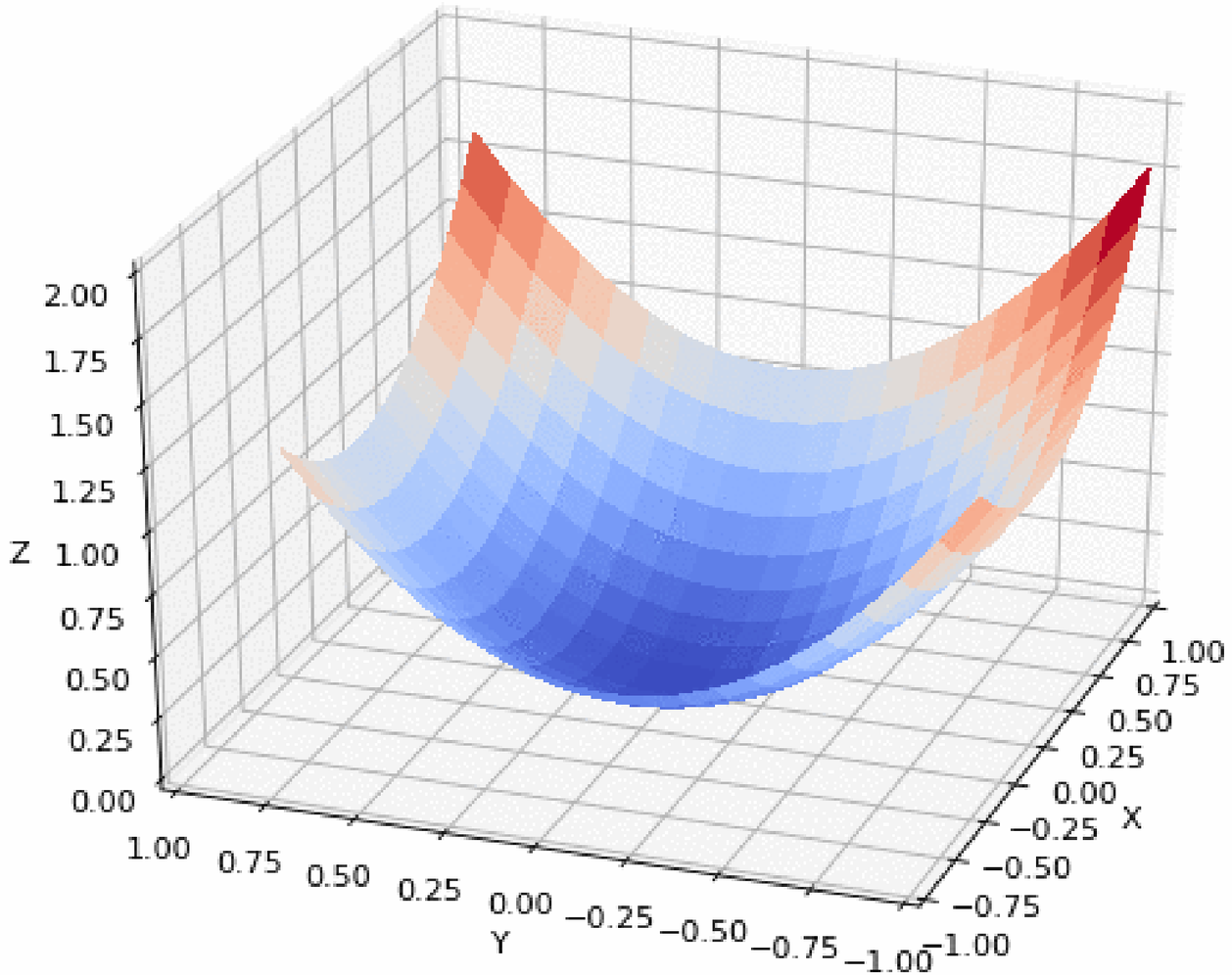
4 Realizamos las iteraciones (Ejemplo numérico)

Supongamos los siguientes valores iniciales:

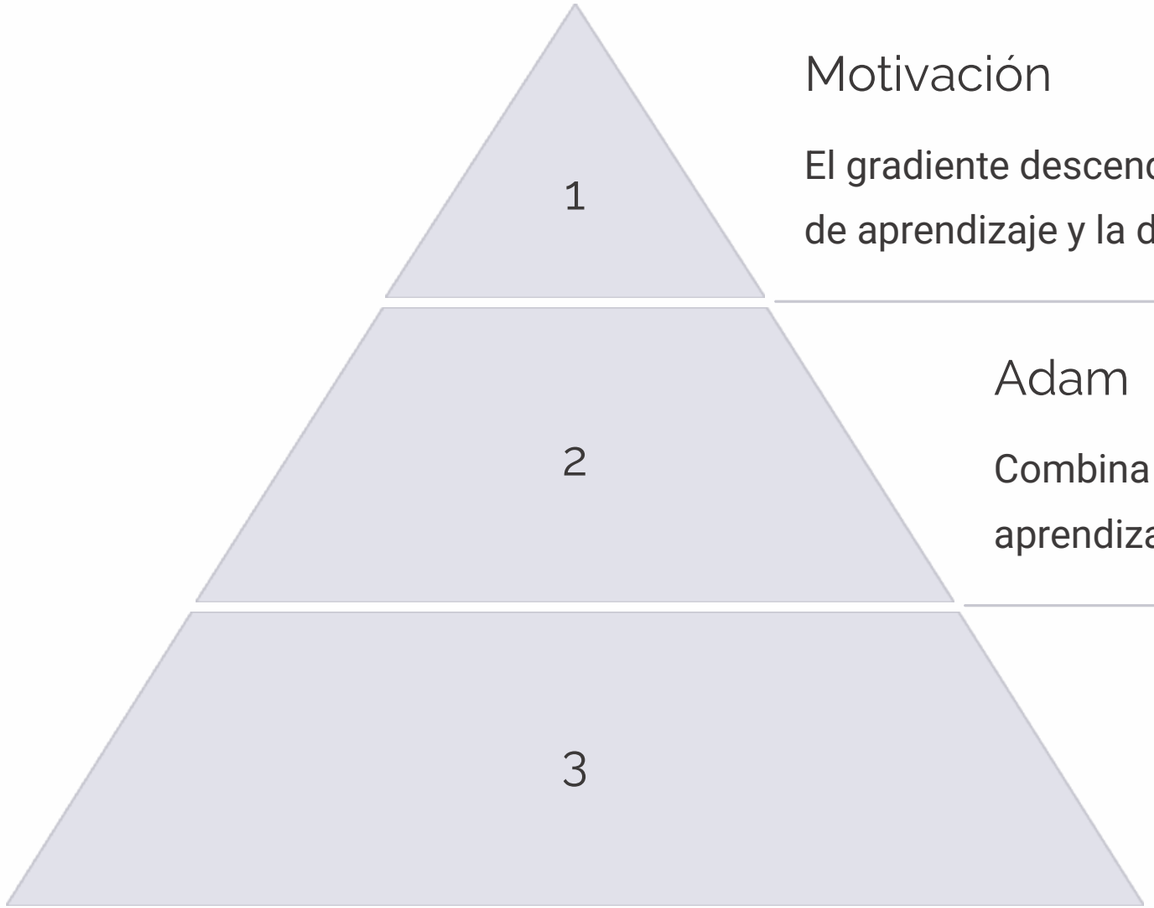
- Entrada x = 2
- Peso inicial w = 1
- Valor real y = 4
- Bias $b = 0$
- *Tasa de aprendizaje α=0.1*

Iteración	w	$\hat{y} = w \cdot x$	Error $C(w)$	$\frac{dC}{dw}$	w_{nuevo}
1	1.00	$1 \times 2 = 2$	$\frac{1}{2}(2-4)^2 = 2$	$2(2-4) = -4$	$1 - 0.1(-4) = 1.4$
2	1.40	$1.4 \times 2 = 2.8$	$\frac{1}{2}(2.8-4)^2 = 0.72$	$2(2.8-4) = -2.4$	$1.4 - 0.1(-2.4) = 1.64$
3	1.64	$1.64 \times 2 = 3.28$	$\frac{1}{2}(3.28-4)^2 = 0.26$	$2(3.28-4) = -1.44$	$1.64 - 0.1(-1.44) = 1.78$

Gradiente Descendente:



Optimizador Adam: Una Mejora Adaptativa



Motivación

El gradiente descendente simple tiene limitaciones, como la sensibilidad a la tasa de aprendizaje y la dificultad para manejar funciones con gradientes ruidosos.

Adam

Combina RMSProp y Momentum, adaptando dinámicamente la tasa de aprendizaje de cada parámetro.

Actualizaciones

Utiliza momentos de primer y segundo orden para ajustar la tasa de aprendizaje de cada parámetro.

Optimizador Adam:

Primer Momento:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

Donde:

- m_t: Vector de primeros momentos.
- g_t: Gradiente en la actualización t.

Segundo Momento:

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

Donde:

- v_t: Vector de segundos momentos.

Corrección de Bias:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

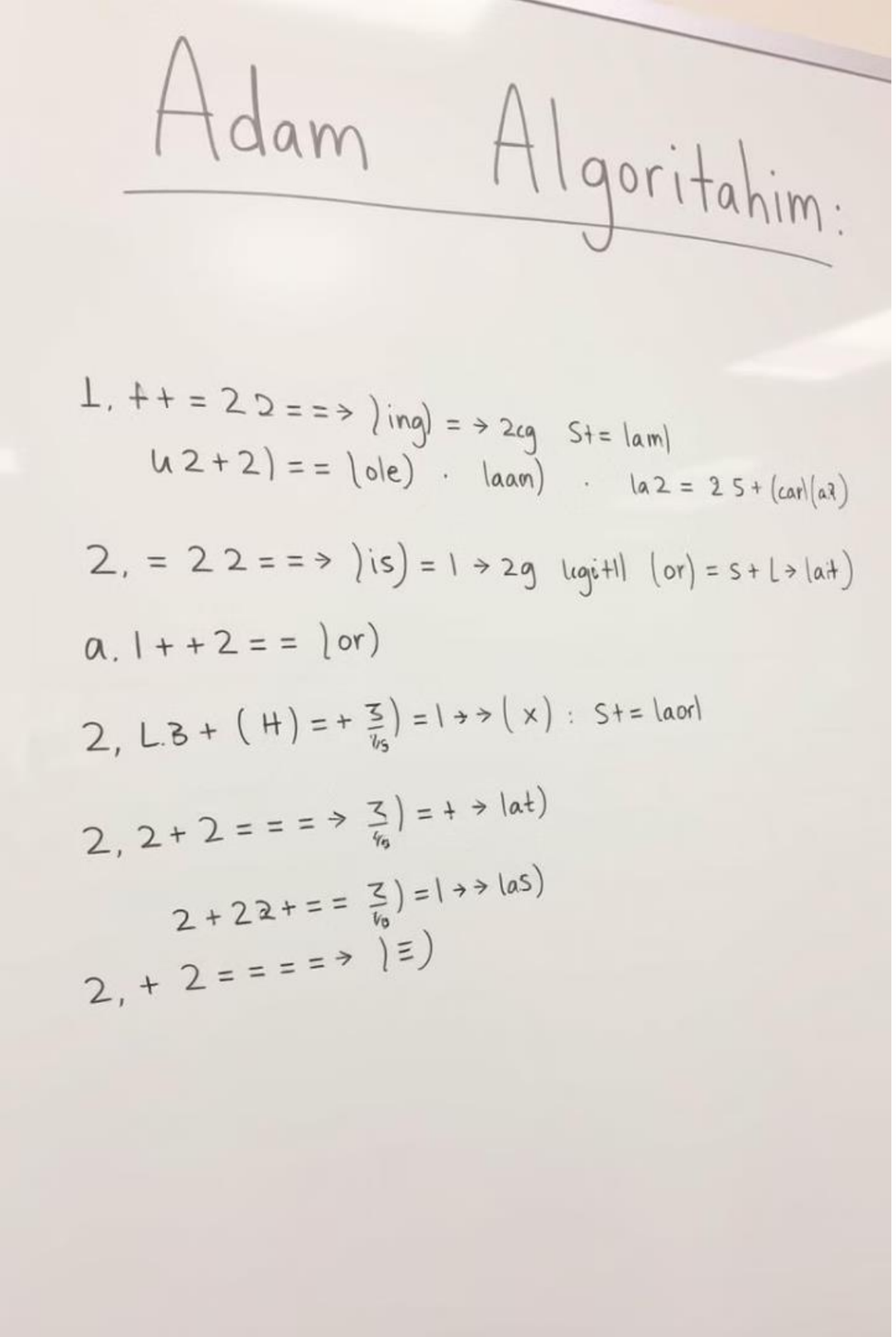
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Actualización de Parámetros:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Donde:

- ϵ: pequeña constante para evitar divisiones por cero.



Ejercicio Práctico: Adam

1. Calcular los momentos

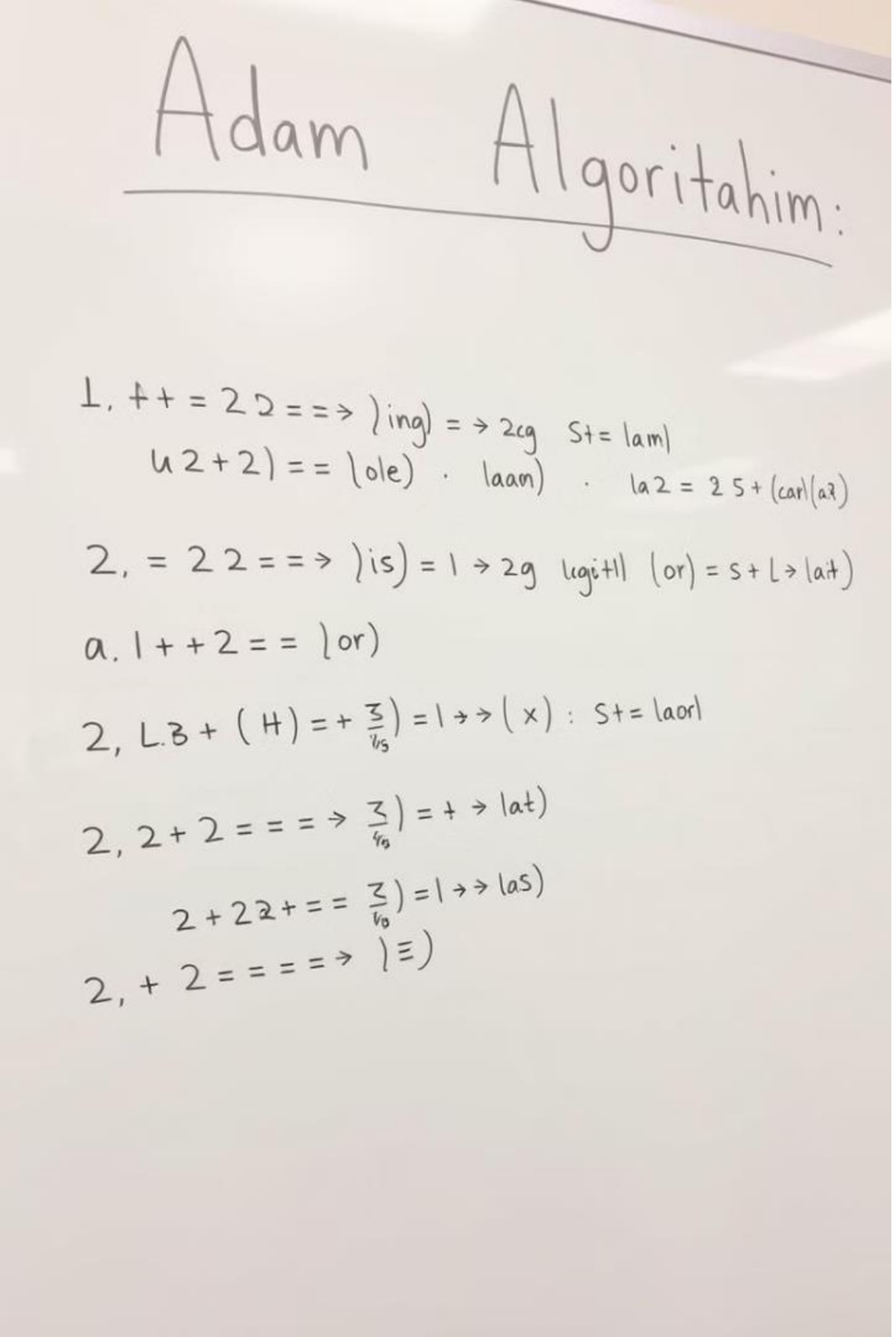
   Calcular el primer momento (media del gradiente) y el segundo momento (varianza del gradiente) para cada parámetro.

2. Adaptar la tasa de aprendizaje

   Ajustar la tasa de aprendizaje de cada parámetro usando los momentos calculados.

3. Actualizar los parámetros

   Utilizar la tasa de aprendizaje adaptada para actualizar los parámetros del modelo.



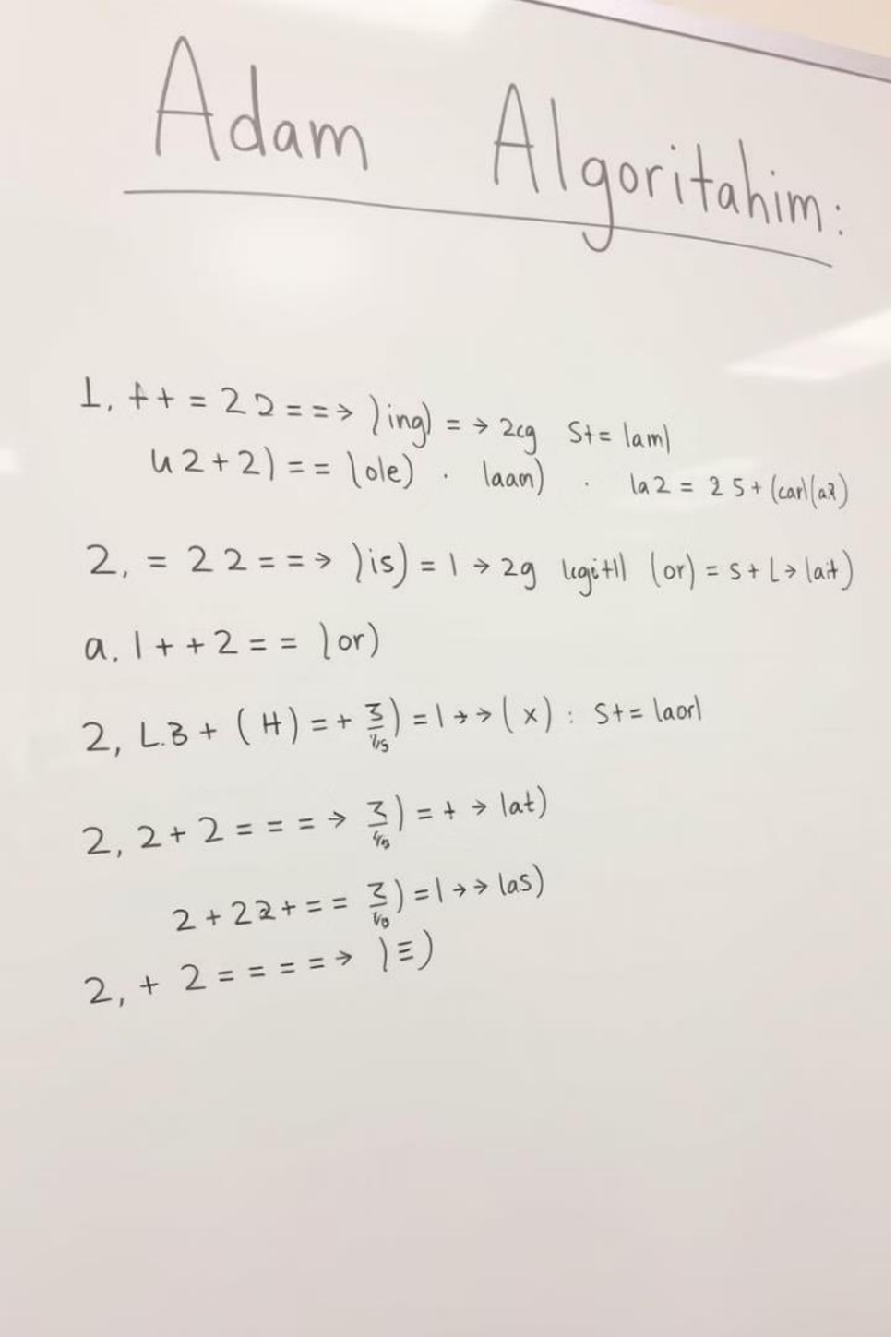
Ejercicio Práctico: Adam

Minimizar la función $f(x) = x^2$

Inicialización: $x_0 = 5$, $\alpha = 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$

- Gradiente en x_0: $g_0 = 2 \times 5 = 10$.
- Actualización de momentos:
  - $m_1 = 0.9 \times 0 + 0.1 \times 10 = 1$
  - $v_1 = 0.999 \times 0 + 0.001 \times 100 = 0.1$
- Corrección de sesgo:
  - $\hat{m}_1 = \frac{1}{1-0.9^1} = 10$
  - $\hat{v}_1 = \frac{0.1}{1-0.999^1} = 100$
- Actualización de x:

$$x_1 = 5 - 0.1 \times \frac{10}{\sqrt{100} + 10^{-8}} = 4$$



Ejercicio Práctico: Adam

- Iteración 1:
  - Gradiente: $g_1 = 2 \times 5 = 10$
  - Momentos: $m_1 = 1,\ v_1 = 0.1$
  - Corrección de sesgo: $\hat{m}_1 = 10,\ \hat{v}_1 = 100$
  - Actualización: $x_1 = 4$
- Iteración 2:
  - Gradiente: $g_2 = 8$
  - Momentos: $m_2 = 1.7,\ v_2 = 0.164$
  - Corrección de sesgo: $\hat{m}_2 = 8.947,\ \hat{v}_2 = 82.041$
  - Actualización: $x_2 \approx 3.015$
- Iteración 3:
  - Gradiente: $g_3 \approx 6.03$
  - Momentos: $m_3 \approx 2.133,\ v_3 \approx 0.200$
  - Corrección de sesgo: $\hat{m}_3 \approx 7.655,\ \hat{v}_3 \approx 66.875$
  - Actualización: $x_3 \approx 2.081$

Otros Optimizadores Populares

Adagrad

Ajusta la tasa de aprendizaje en función de la frecuencia de actualización de cada parámetro, útil para datos esparsos.

Adadelta

Variante de Adagrad que limita la acumulación de tasas de aprendizaje para mejorar la convergencia en el largo plazo.

Nesterov Accelerated Gradient

Una variante de Momentum que utiliza una "mirada hacia adelante" para obtener una mejor dirección de búsqueda.

Cómo Elegir el Optimizador Adecuado

1

Tipo de problema

La elección depende del tipo de problema, la naturaleza de los datos y las características del modelo.

2

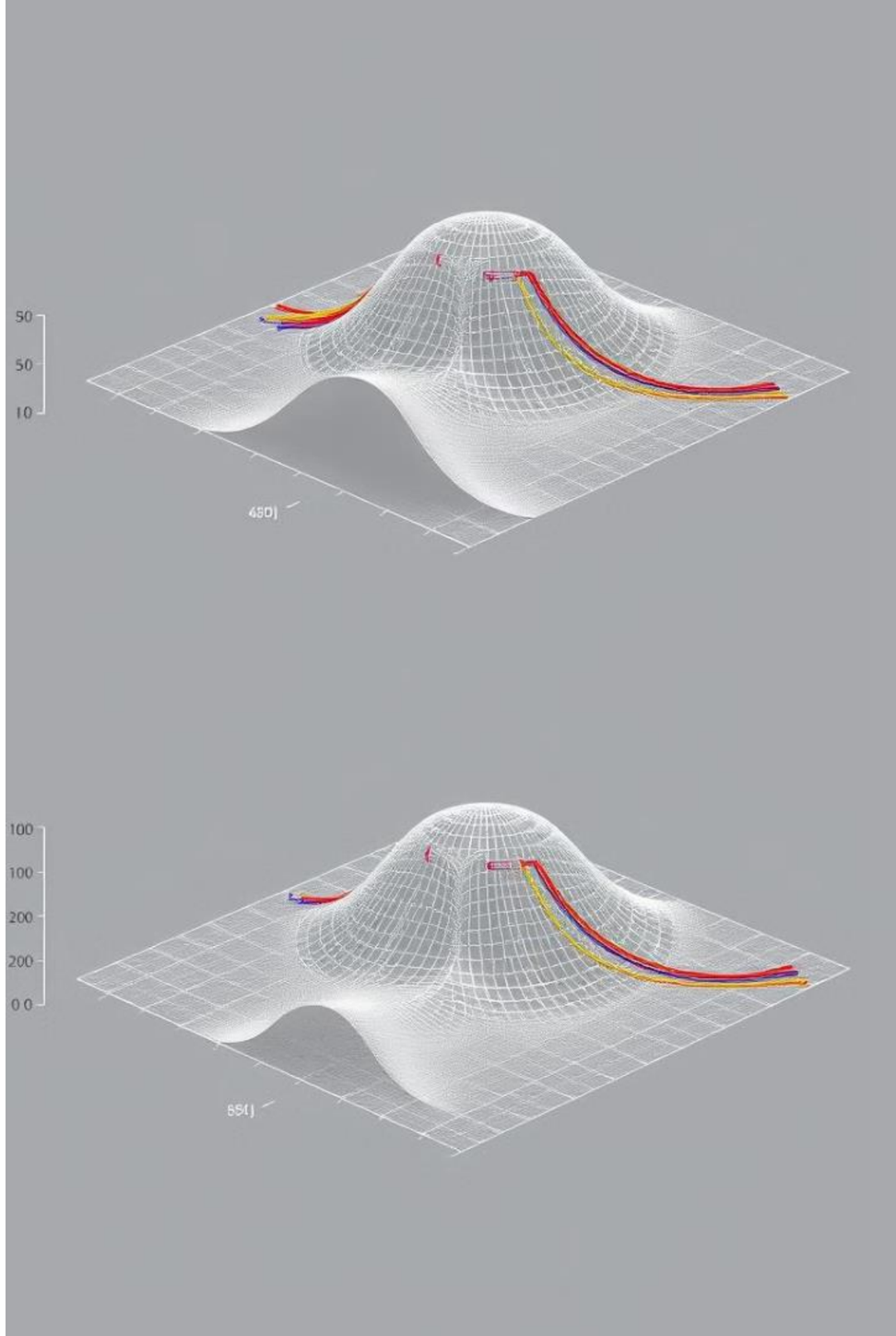
Experimentación

Probar diferentes optimizadores y ajustar sus hiperparámetros para encontrar la mejor configuración para tu modelo.

3

Recursos

Existen tutoriales, artículos y bibliotecas que proporcionan información detallada sobre los optimizadores.



Conclusión

La optimización es esencial para el entrenamiento eficaz de modelos de aprendizaje automático. Elegir el optimizador adecuado y ajustar sus hiperparámetros pueden mejorar significativamente el rendimiento y la capacidad predictiva de tu modelo.