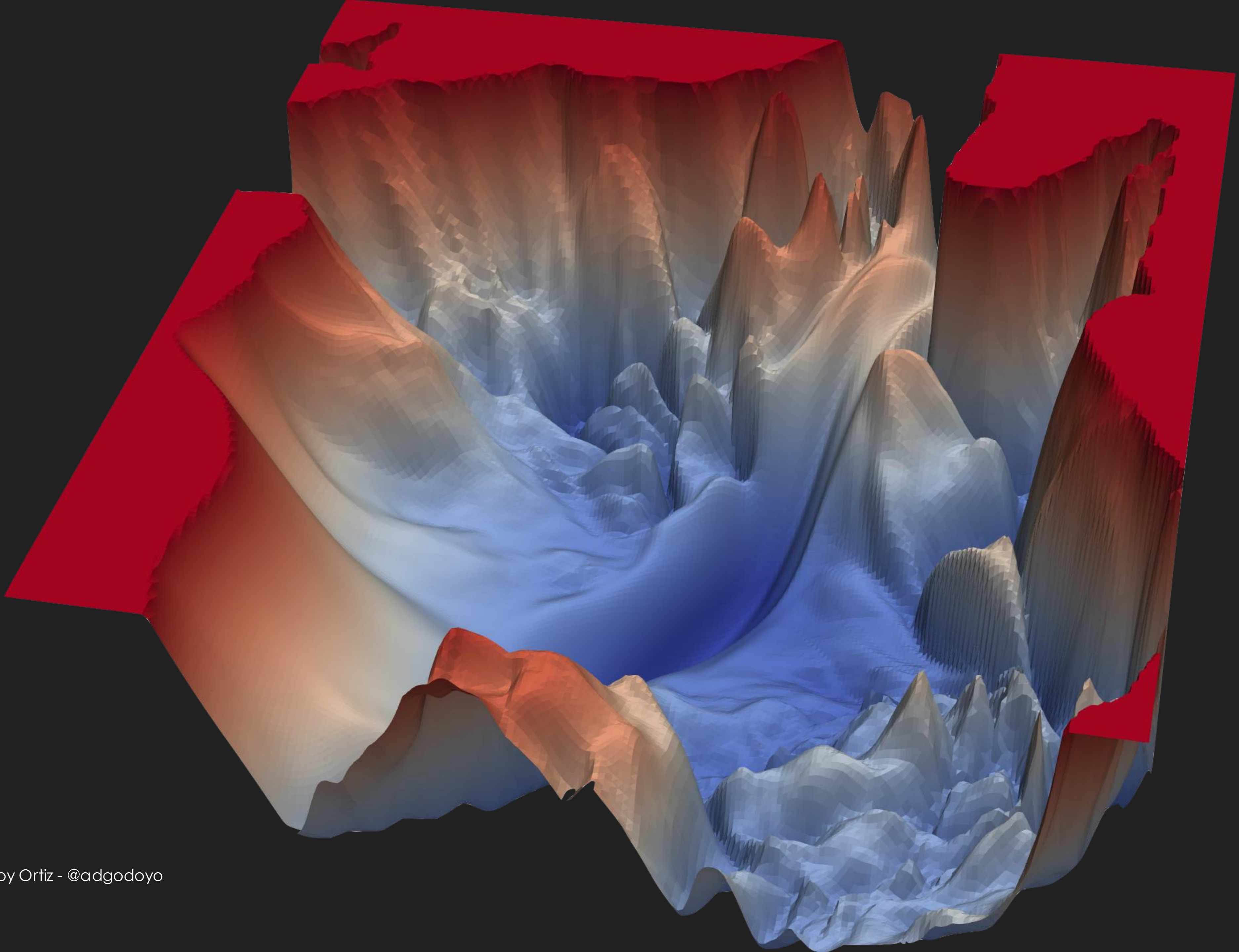
Gradiente Descendente

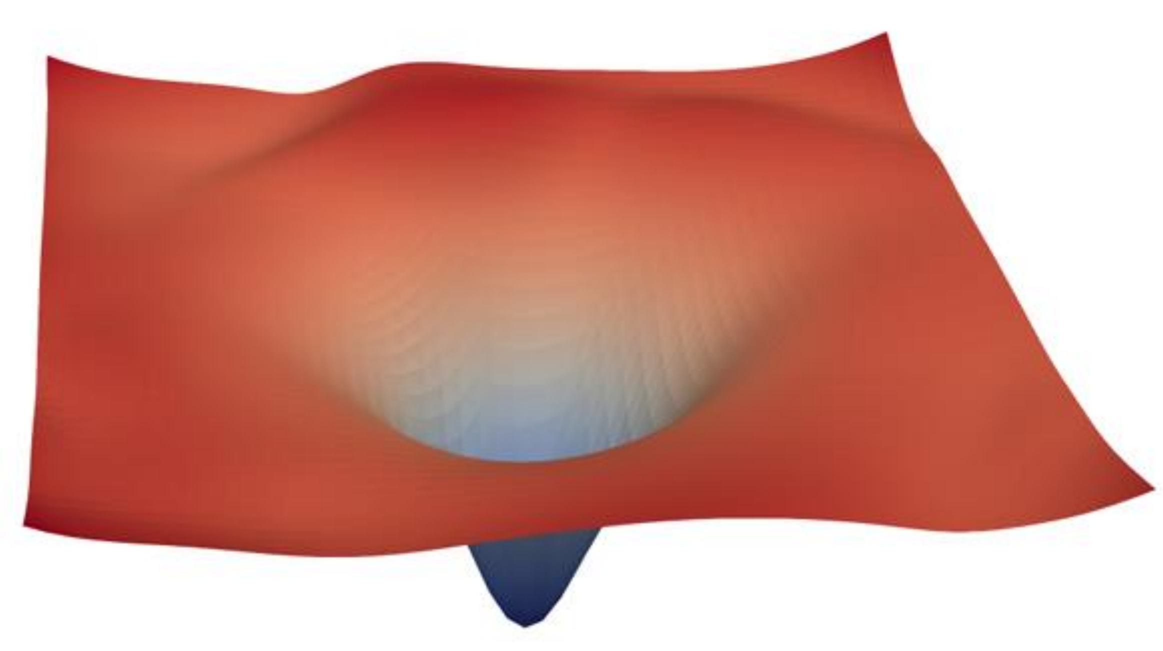
Andrés Daniel Godoy Ortiz

Muchos recursos gráficos tomados de 3blue1Brown y Emergent Garden

¿cómo aprenden las redes neuronales?



Andrés Daniel Godoy Ortiz - @adgodoyo



El objetivo es encontrar el mínimo de la función de costo, del error que sistemáticamente comete el modelo.

Error cuadrático medio (MSE):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Error absoluto medio (MAE):

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

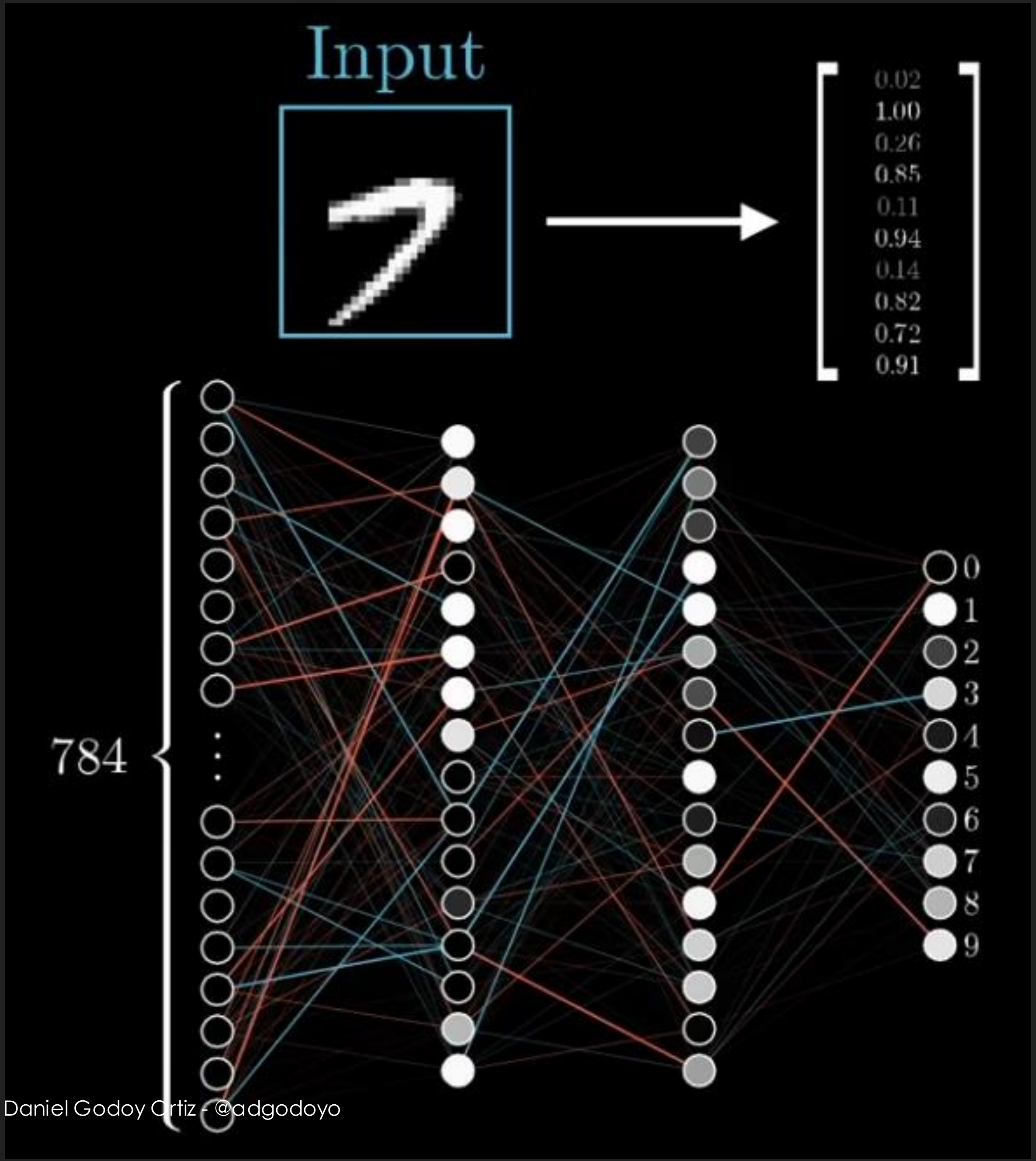
Clasificación binaria (activación final: SIGMOID):

$$\mathcal{L}_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

Clasificación multiclase (activación final: SOFTMAX): :

$$\mathcal{L}_{CCE} = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(\hat{y}_{ij})$$

¿Cómo logro reducir la función de costo?
Ajustando mis parámetros



Función Neuronal:

Input: 784 pixeles

Output: 10 dígitos

Parámetros: 13,002

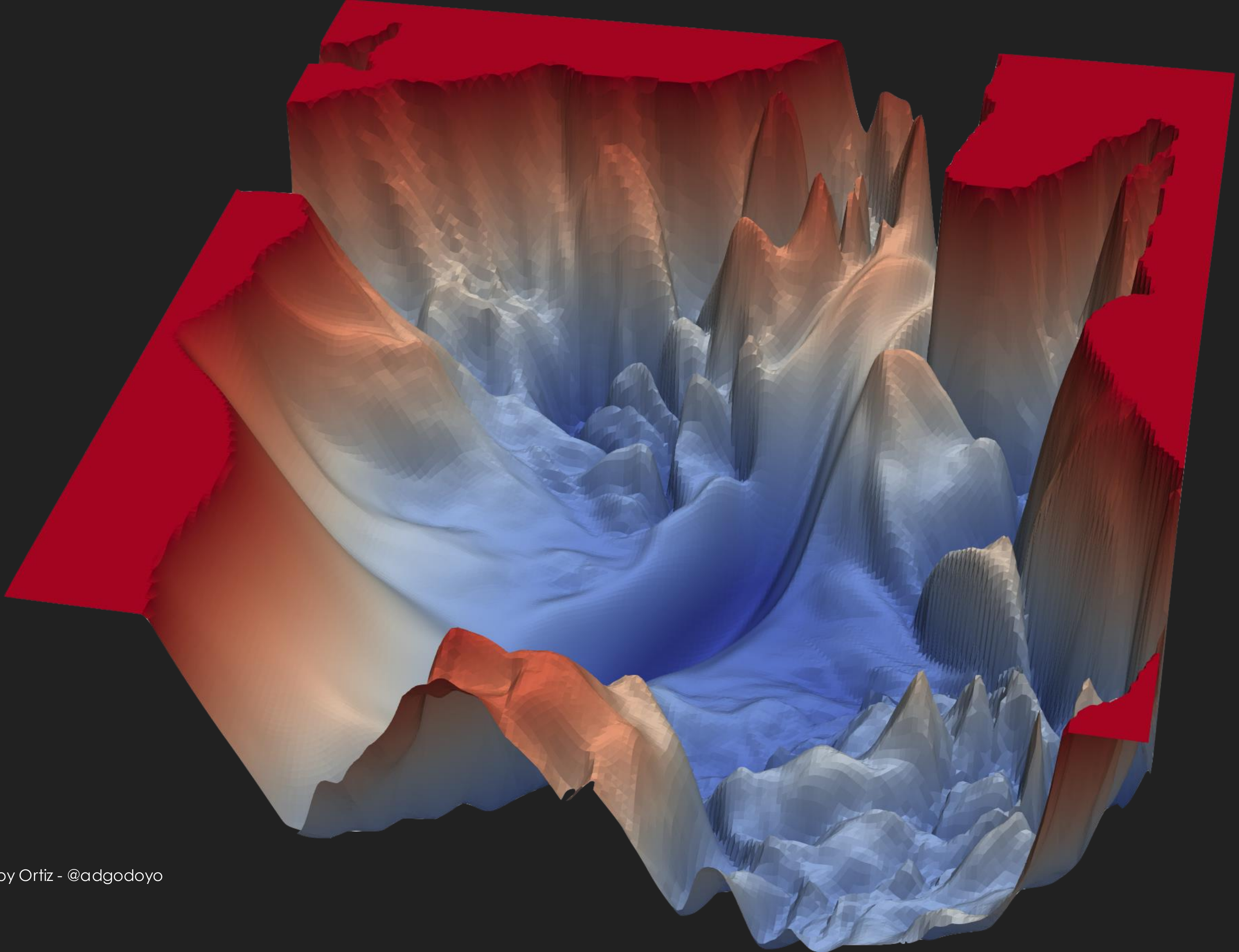
Función de Costo: $C(w_1, w_2, \ldots, w_{13,002})$

Input: 13,002 valores de pesos (w) y sesgos(b)

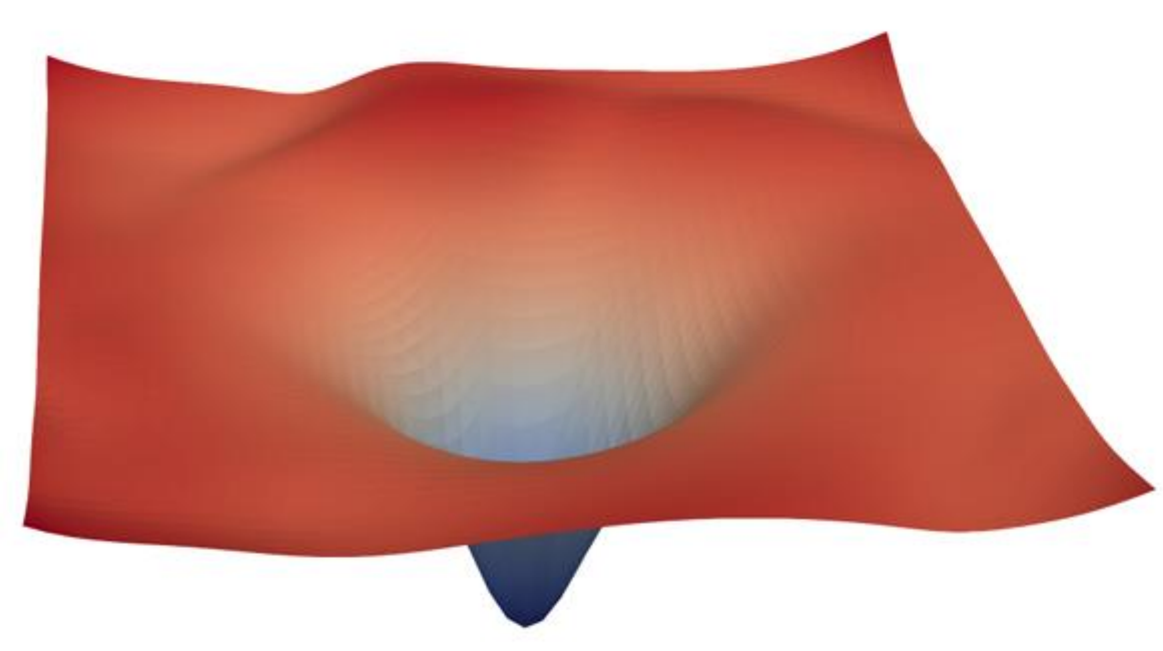
Output: 1 valor el costo

Parámetros: La data/ las imagenes

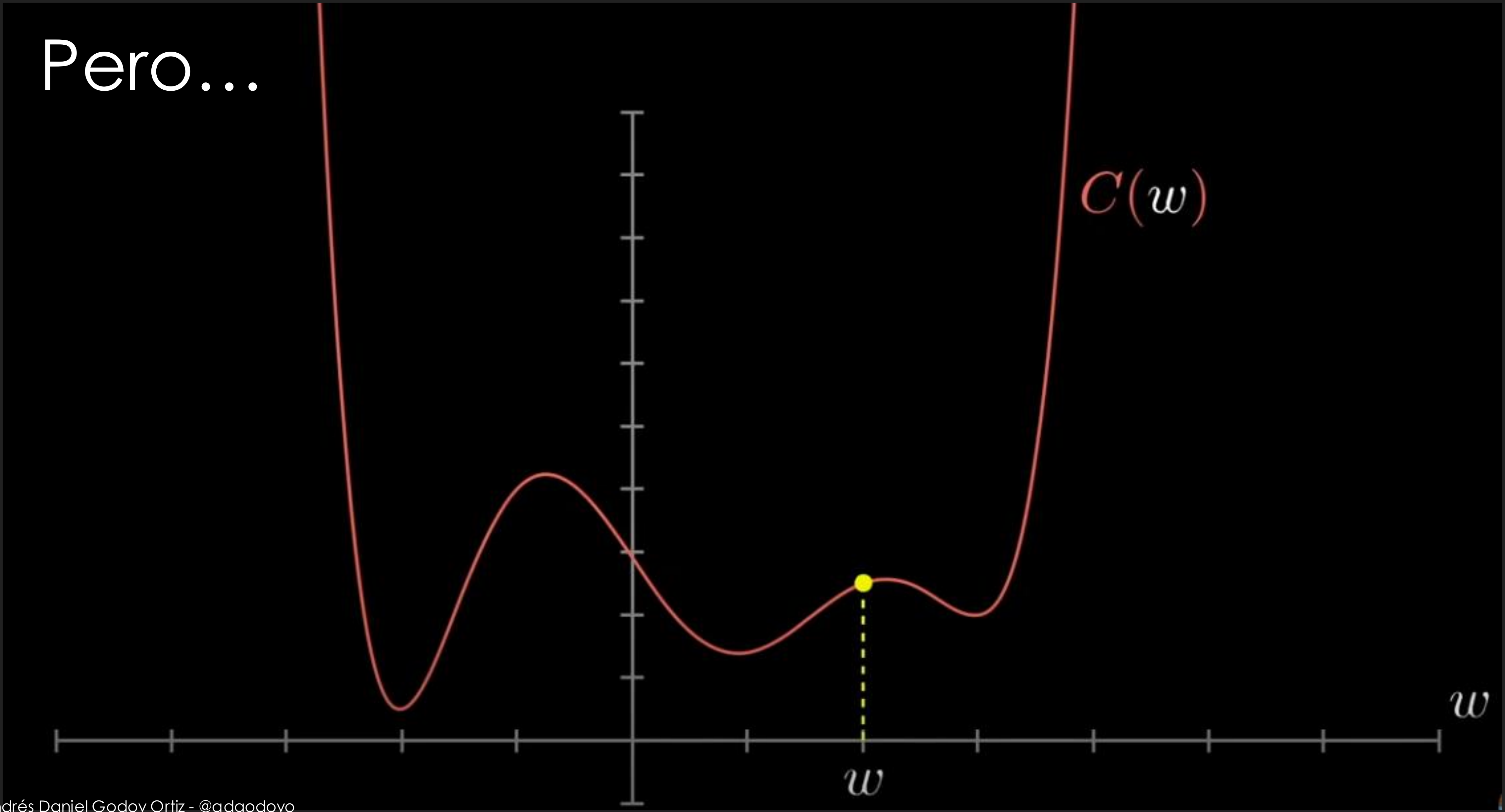
$$
\begin{aligned}
f(\theta) = \ & \mathrm{cost}(\theta, \ ) \\
& + \mathrm{cost}(\theta, \ ) \\
& + \mathrm{cost}(\theta, \ ) \\
& + \mathrm{cost}(\theta, \ )
\end{aligned}
$$



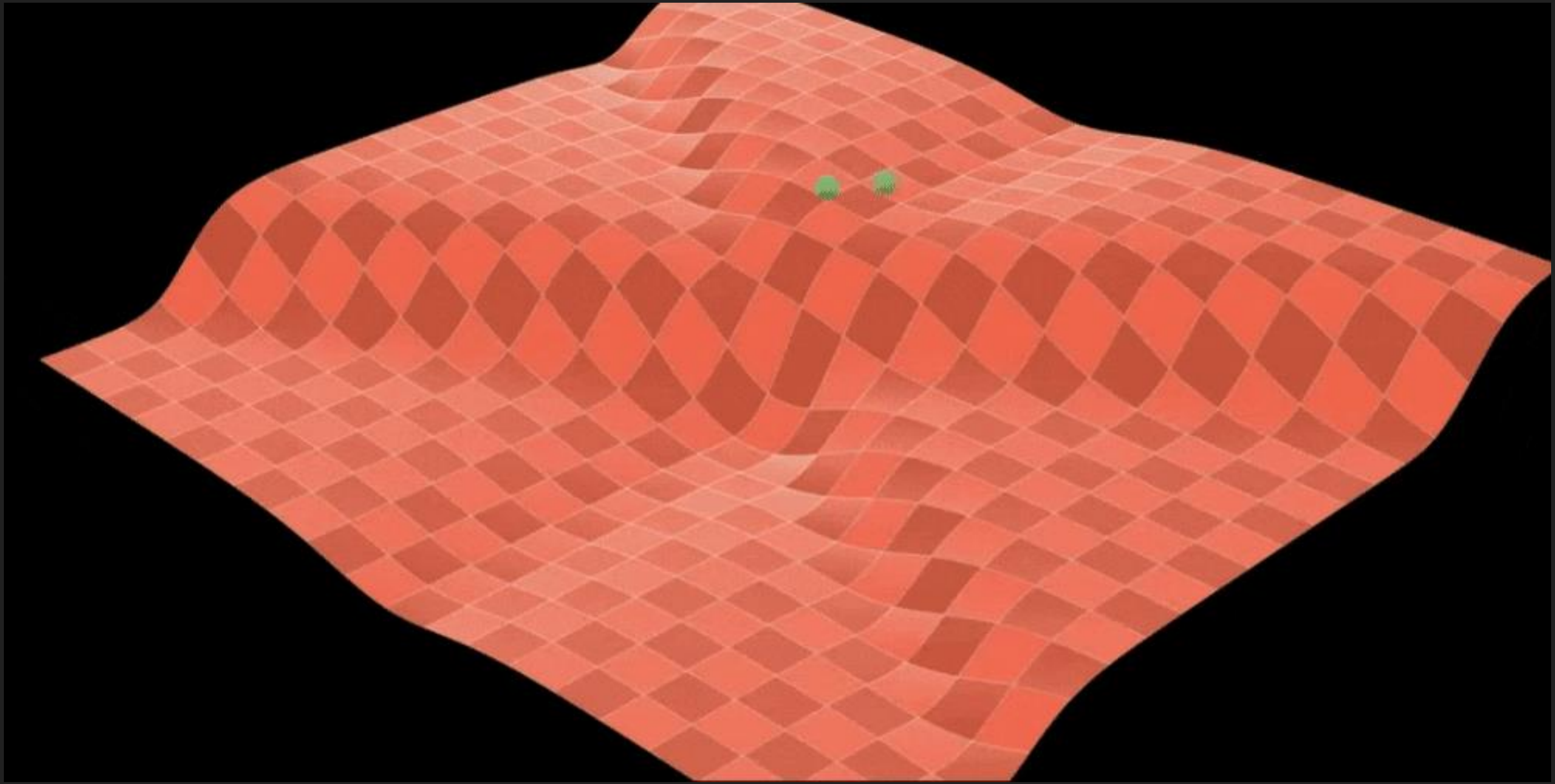
Andrés Daniel Godoy Ortiz - @adgodoyo

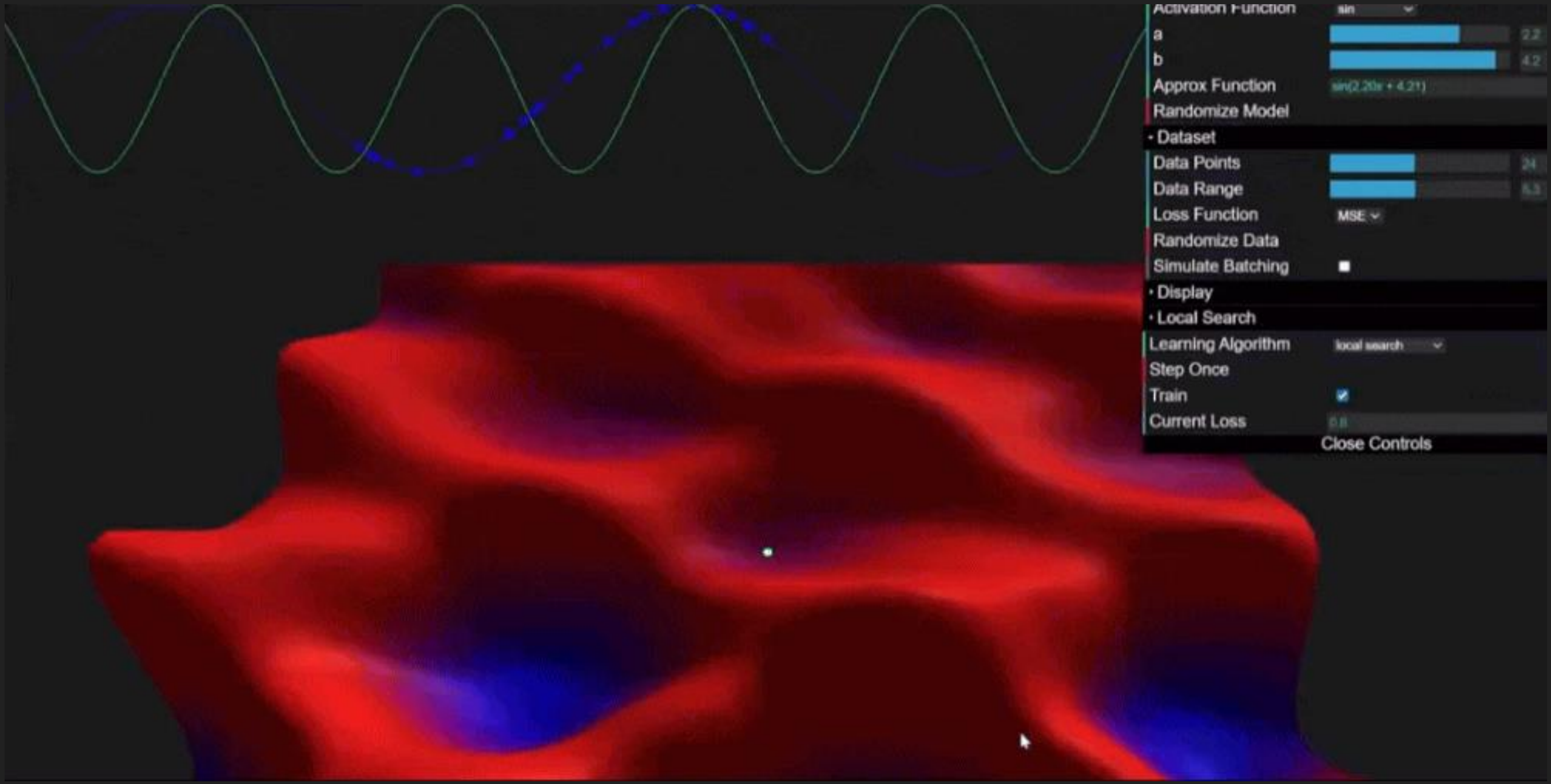


Pero…

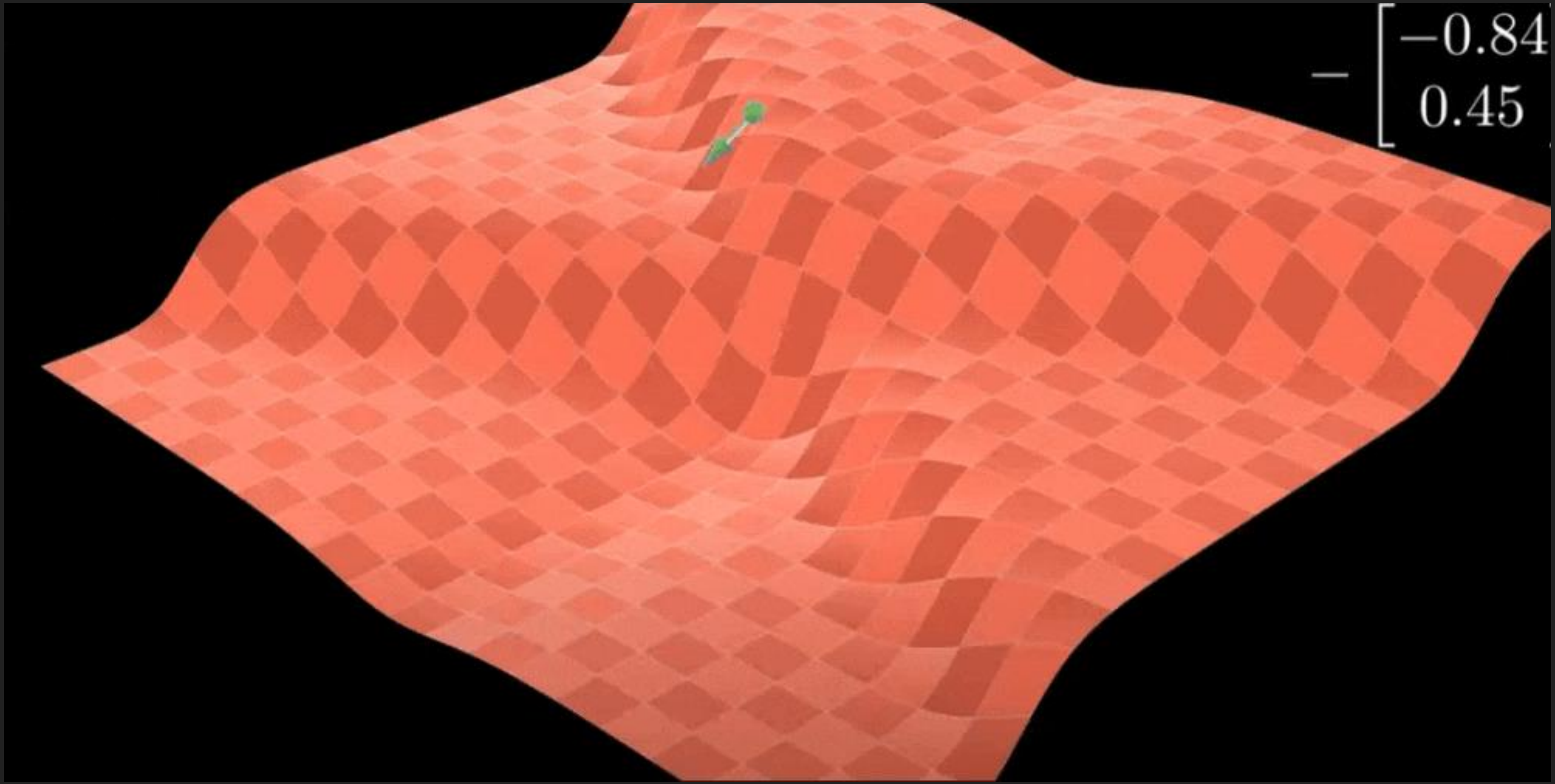


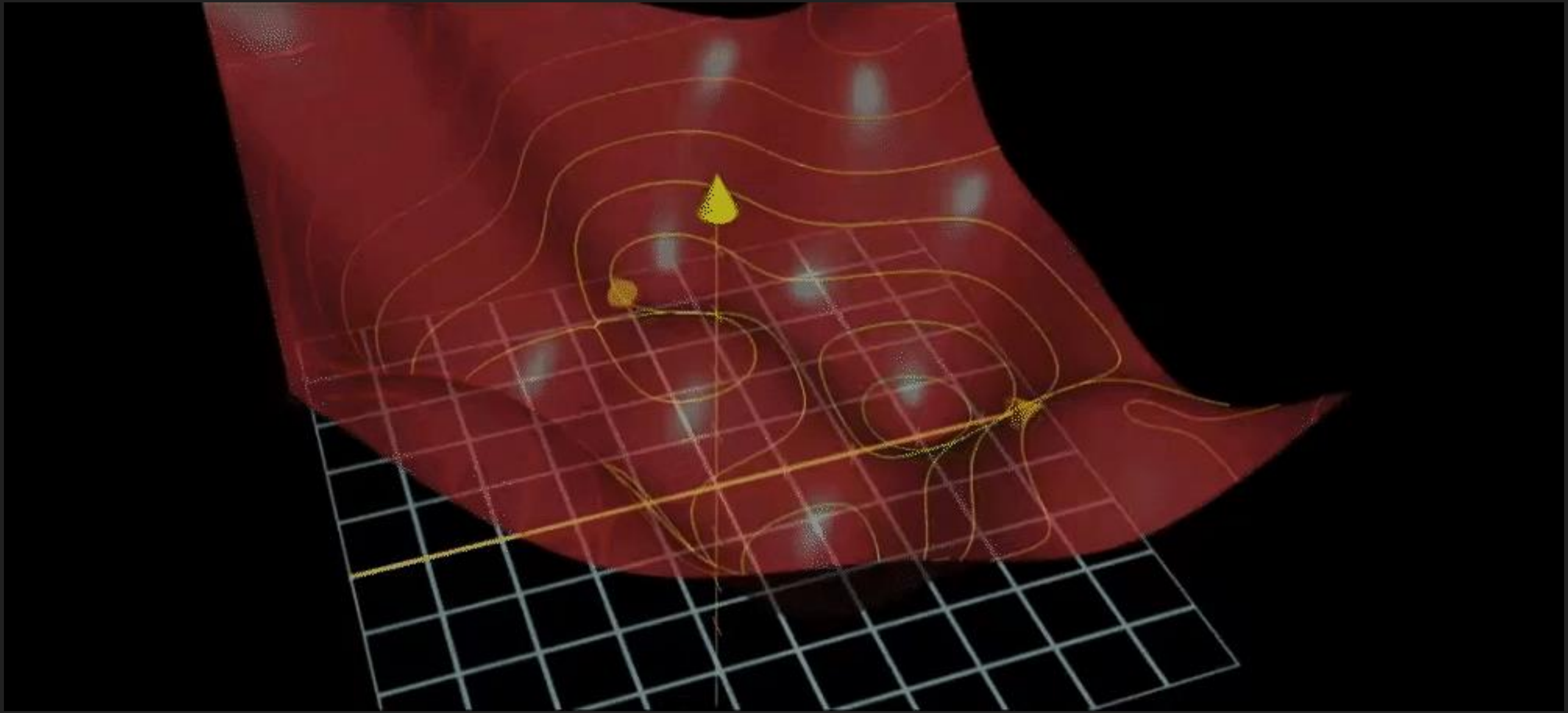
Local Search…




Activation Function
sin
a
2.2
b
4.2
Approx Function
sin(2.20x + 4.21)
Randomize Model
Dataset
Data Points
24
Data Range
Loss Function
MSE
Randomize Data
Simulate Batching
Display
Local Search
Learning Algorithm
local search
Step Once
Train
Current Loss
Close Controls

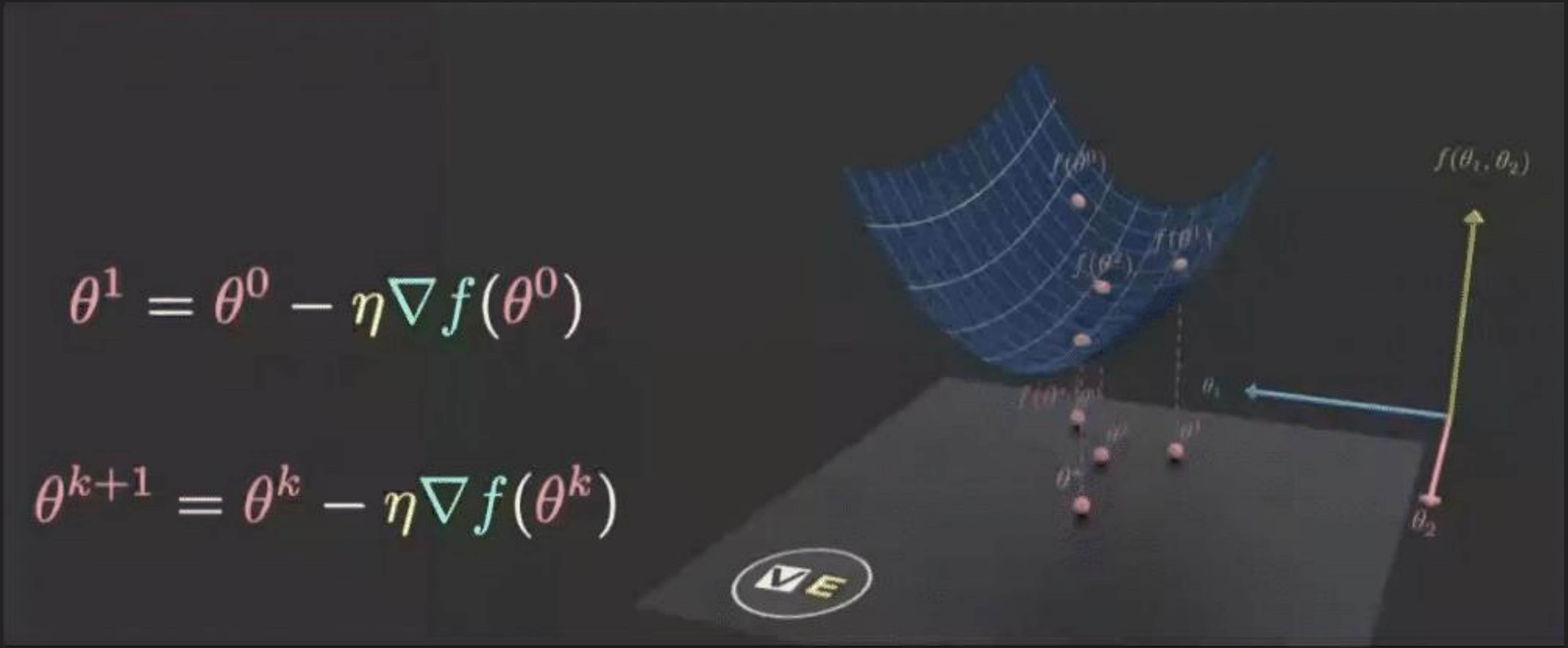
Gradient Descent…



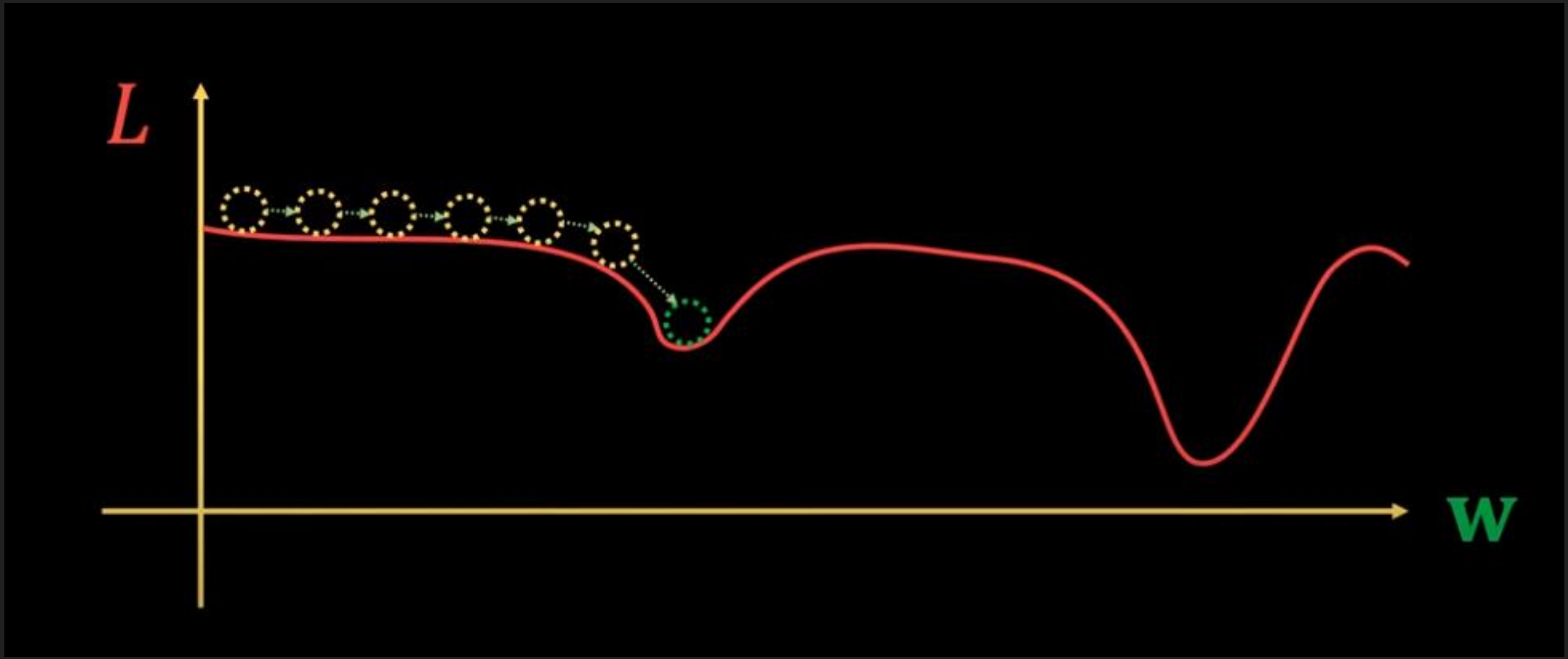


¿qué nos dice el gradiente sobre cómo deben cambiar los pesos?

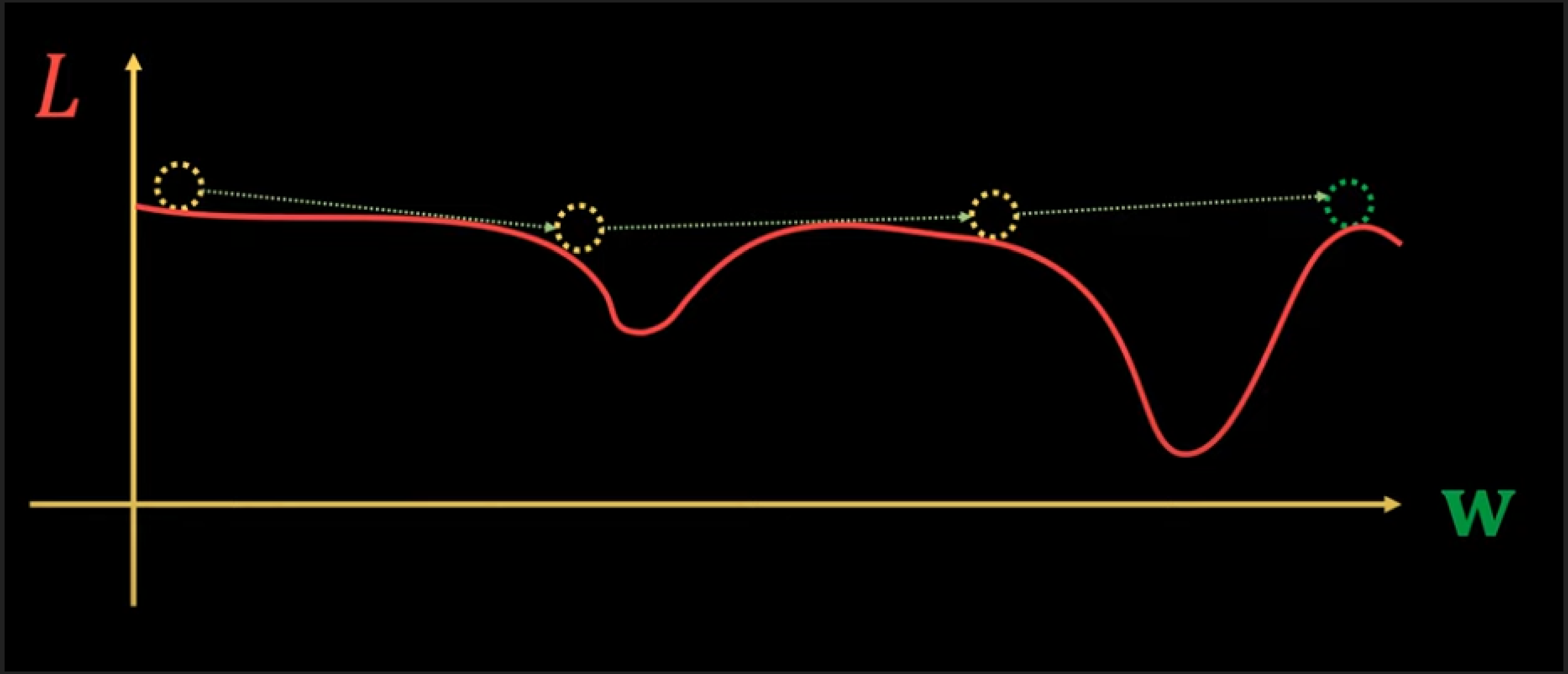
$$\vec{\mathbf{W}} = \begin{bmatrix} 2.25 \\ -1.57 \\ 1.98 \\ \vdots \\ -1.16 \\ 3.82 \\ 1.21 \end{bmatrix} \qquad -\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.18 \\ 0.45 \\ -0.51 \\ \vdots \\ 0.40 \\ -0.32 \\ 0.82 \end{bmatrix}$$

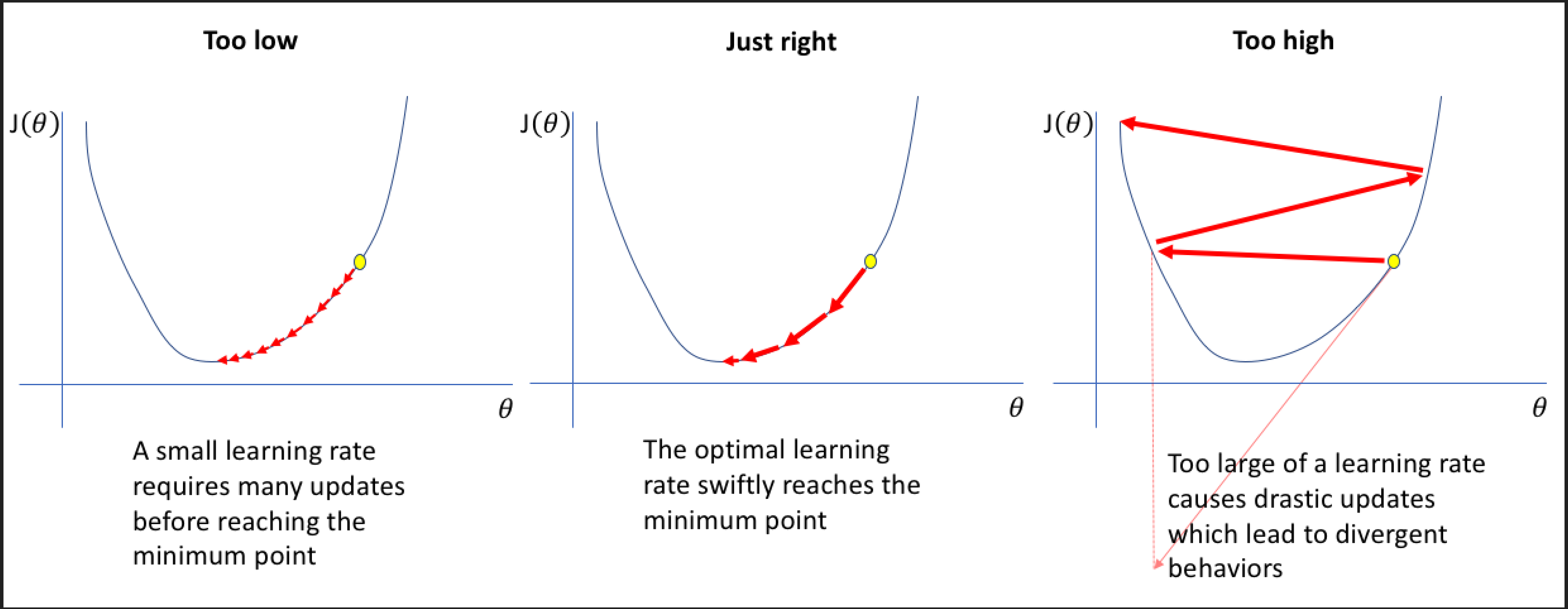

$\theta^1 = \theta^0 - \eta \nabla f(\theta^0)$
$\theta^{k+1} = \theta^k - \eta \nabla f(\theta^k)$
$f(\theta_1, \theta_2)$
θ_1
θ_2

Tasa de aprendizaje baja:



Tasa de aprendizaje alta:




Too low
$J(\theta)$
θ
A small learning rate requires many updates before reaching the minimum point
Just right
$J(\theta)$
θ
The optimal learning rate swiftly reaches the minimum point
Too high
$J(\theta)$
θ
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Variaciones de Gradiente descendente:

1. Batch GD: Se calcula sobre todo el dataset
2. Stochastic GD: 1 sola observación aleatoria (noise)
3. Mini-batch GD: Un pequeño subconjunto aleatorio

4. Momentum GD:

Se acumula un promedio exponencial de los gradientes pasados.

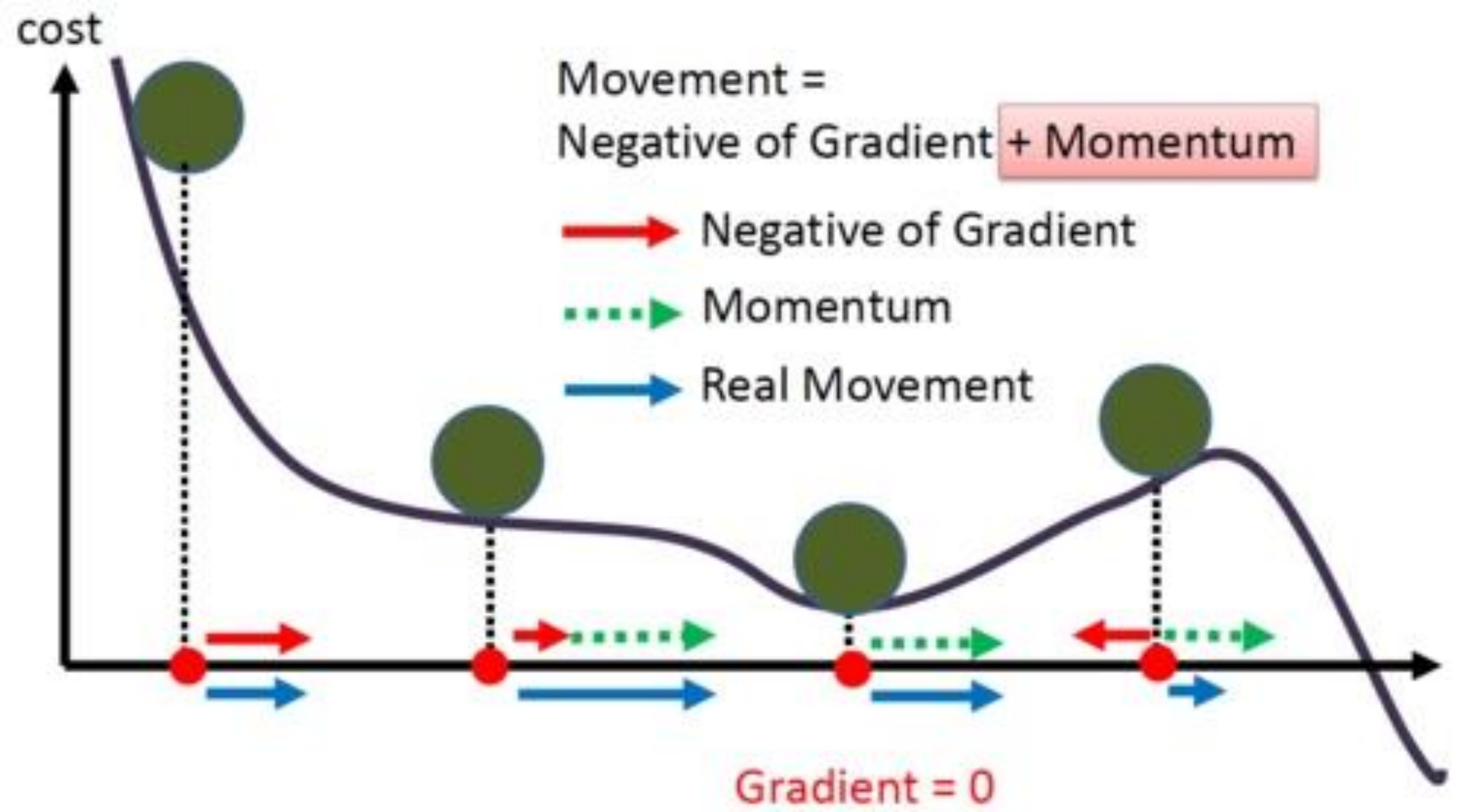
En lugares donde el gradiente cambia de dirección bruscamente, el gradiente descendente simple puede “rebotar” de un lado a otro.

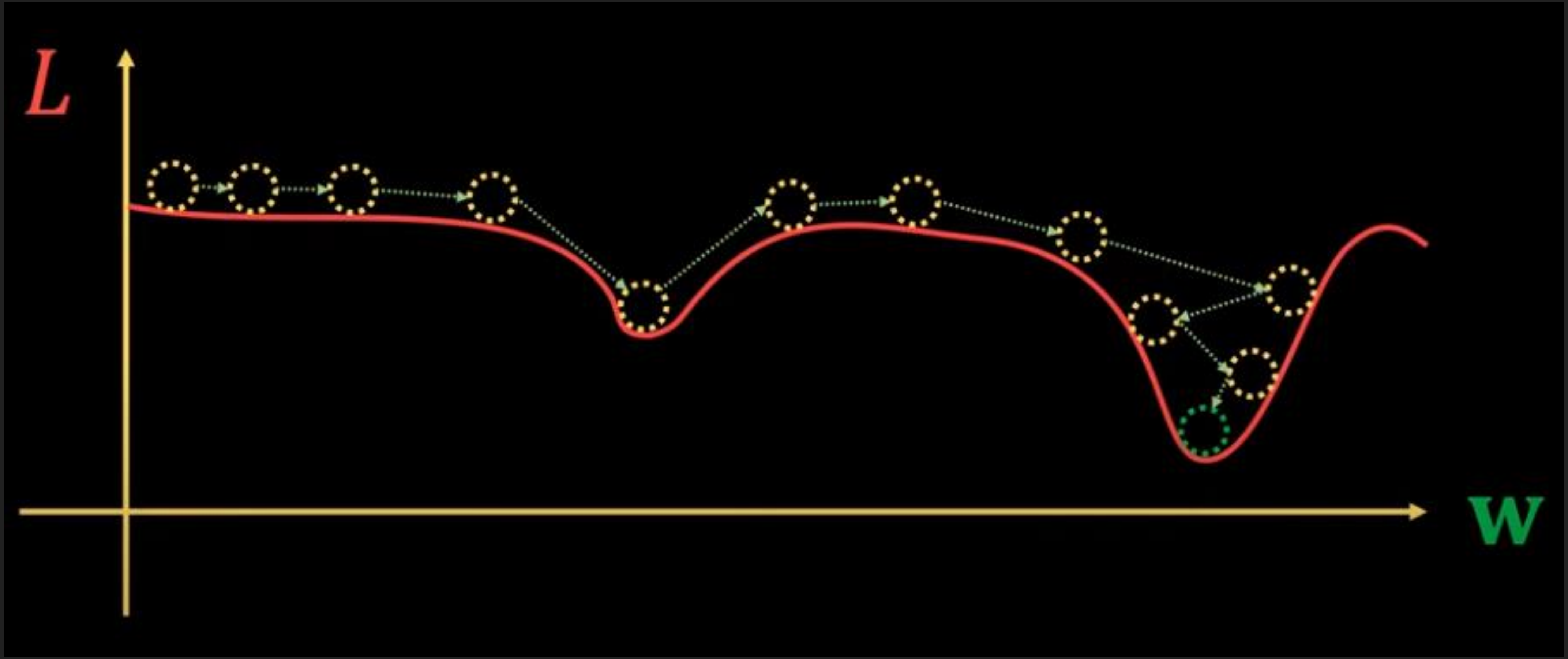
Momentum suaviza el gradiente usando algo parecido a la “inercia”.

1. $$v_t = \beta\, v_{t-1} + (1-\beta)\,\nabla L(w_t).$$

2. $$w_{t+1} = w_t - \eta\, v_t.$$

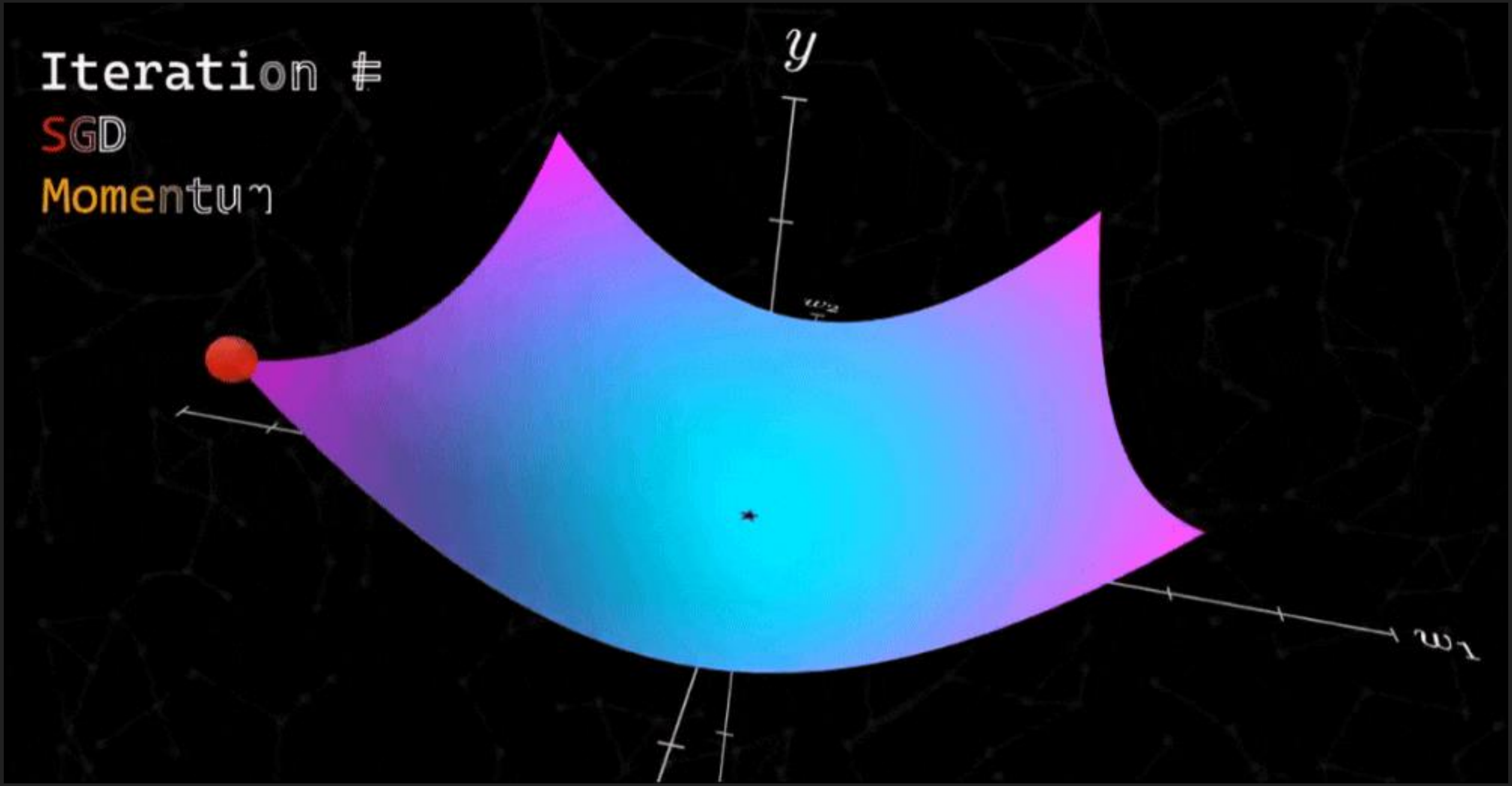
Aquí, β (entre 0 y 1) es el factor de inercia y η la tasa de aprendizaje.


cost
Movement =
Negative of Gradient + Momentum
Negative of Gradient
Momentum
Real Movement
Gradient = 0


L
w

Interpretación:

- **Si el gradiente sigue apuntando en la misma dirección** durante varios pasos, el término $\beta\, v_{t-1}$ ayuda a "acumular" esa dirección, "empujando" con mayor fuerza en la dirección adecuada.
- **Si el gradiente cambia de dirección** bruscamente, la parte "vieja" $\beta\, v_{t-1}$ se va reduciendo con el tiempo, y la parte nueva $(1-\beta)\,\nabla L(w_t)$ hace que la dirección se ajuste gradualmente.


Iteration #
SGD
Momentum
y
w1

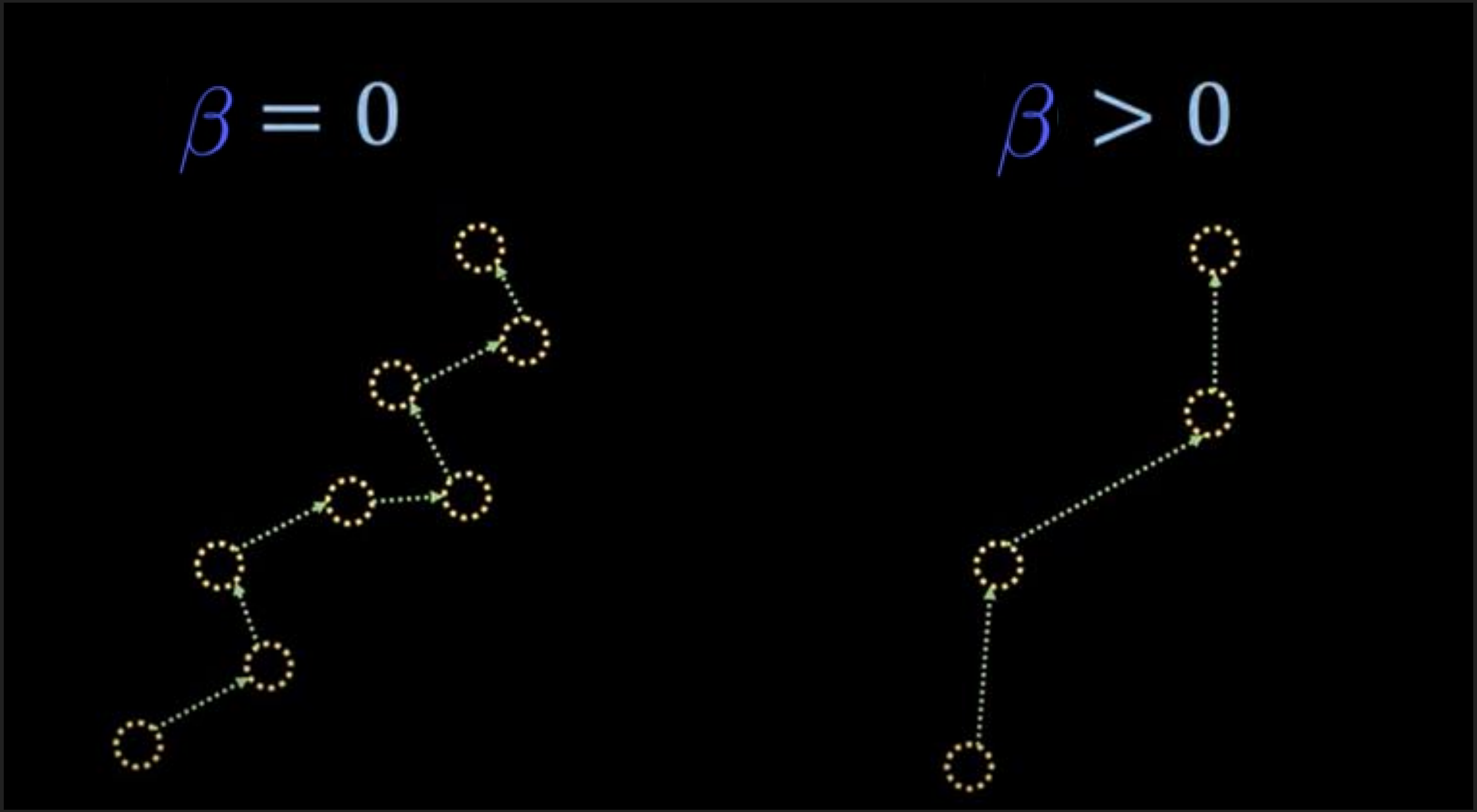
Unrolling the Exponentially Weighted Average

$$V_{t+1} = \beta V_t + (1 - \beta)\nabla W_t$$

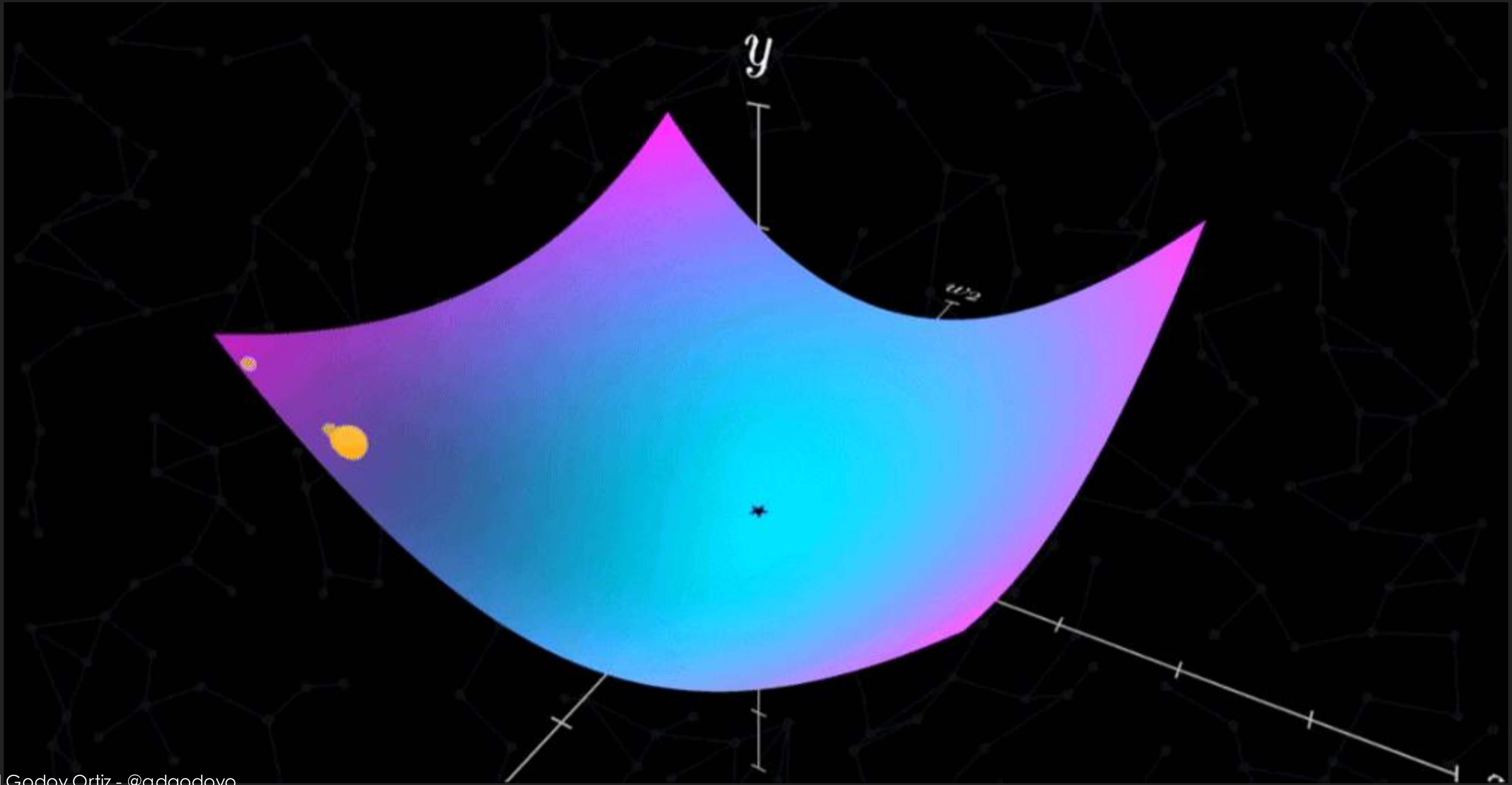
$$V_{t+1} = \beta^2 V_{t-1} + \beta(1 - \beta)\nabla W_{t-1} + (1 - \beta)\nabla W_t$$

$$V_{t+1} = \beta^2(\beta V_{t-2} + (1 - \beta)\nabla W_{t-2}) + \beta(1 - \beta)\nabla W_{t-1} + (1 - \beta)\nabla W_t$$

Andrés Daniel Godoy Ortiz - @adgodoyo

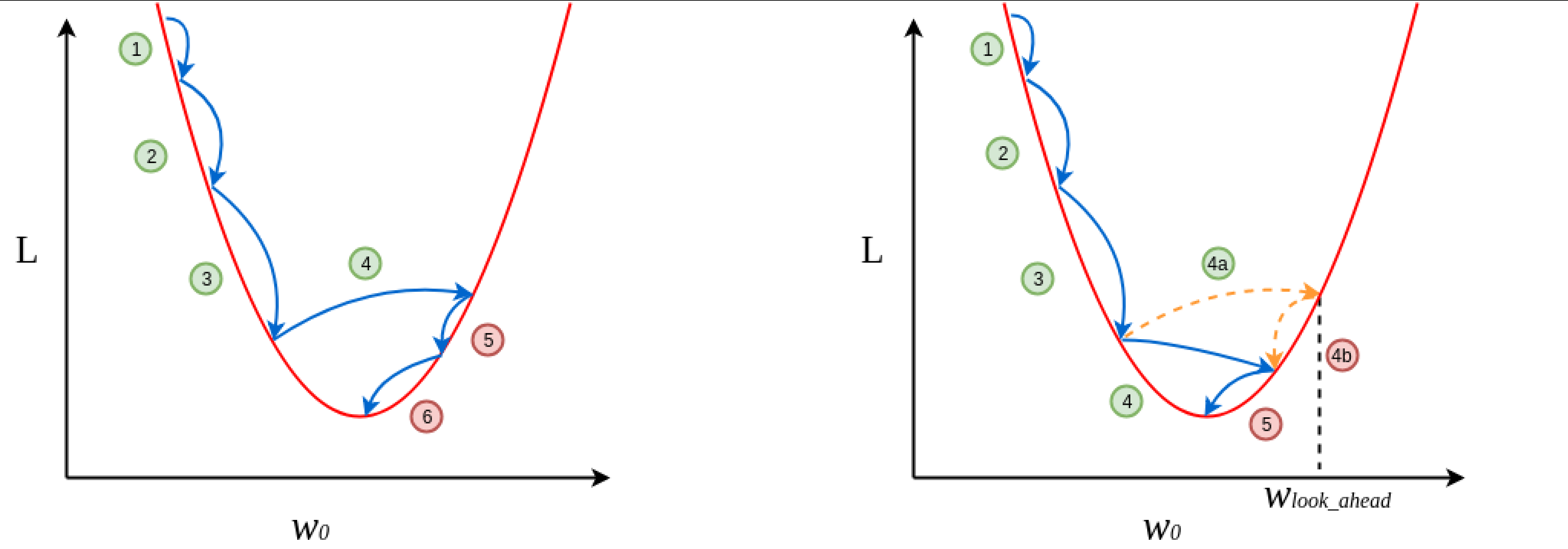

$\beta = 0$
$\beta > 0$

Pero….



5. Nesterov Accelerated Gradient (NAG): calcular el gradiente **“un paso antes”** de aplicar el momento, es decir, evaluar el gradiente en una posición “adelantada”.
Esto ayuda a “corregir” la dirección antes de dar el salto completo, reduciendo la posibilidad de pasarte de largo.

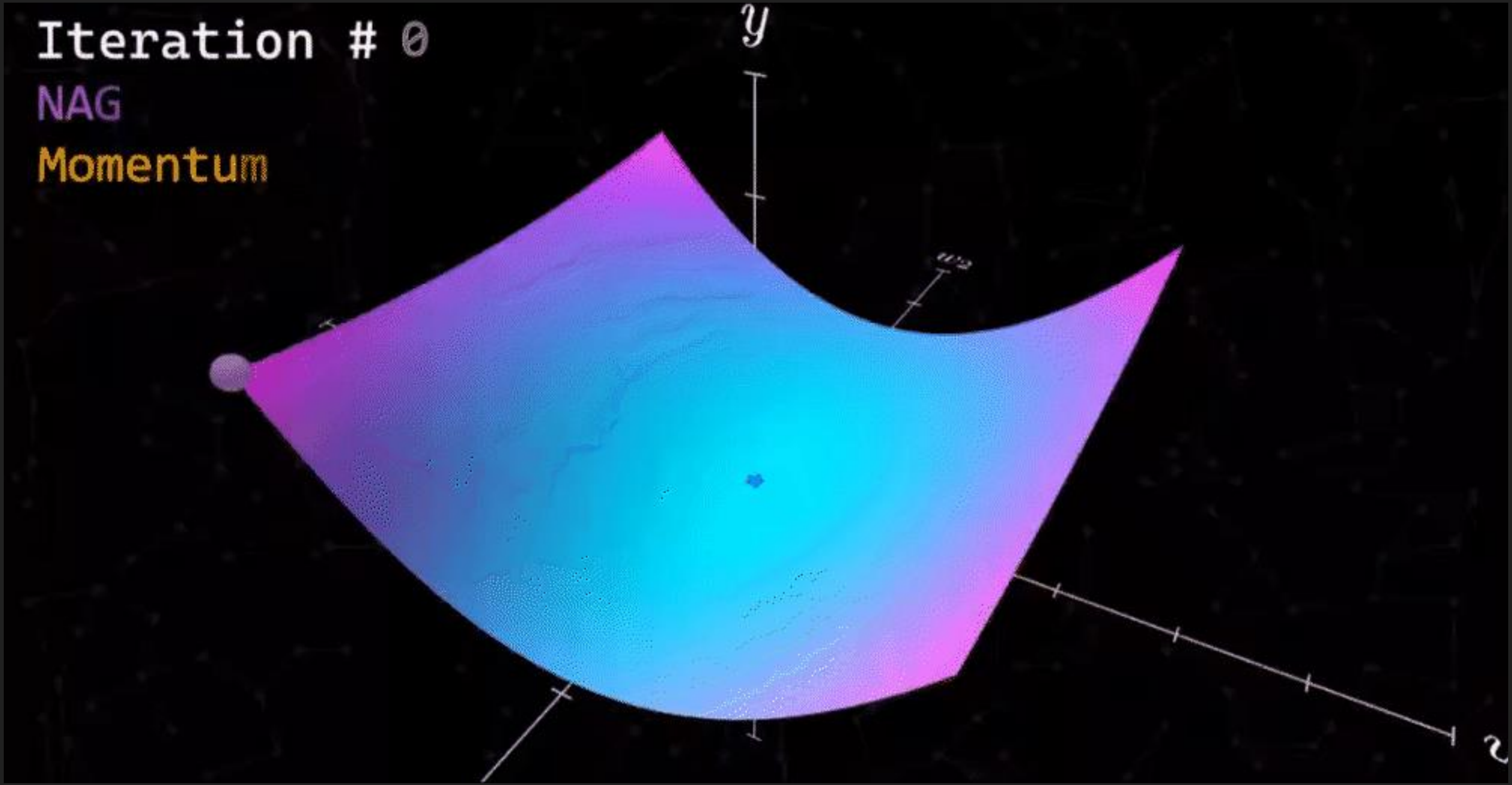
“Classical” momentum	Nesterov momentum
$\mathbf{v}_{t+1} = \rho \mathbf{v}_t - \eta \mathbf{g}(\mathbf{w}_t)$	$\mathbf{v}_{t+1} = \rho \mathbf{v}_t - \eta \mathbf{g}(\mathbf{w}_t + \rho \mathbf{v}_t)$
$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1}$	$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1}$



(a) Momentum-Based Gradient Descent

(b) Nesterov Accelerated Gradient Descent

$$\bigcirc \Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Positive(+)} \qquad \bigcirc \Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Negative(-)}$$


Iteration # 0
NAG
Momentum
y

5. RMSprop (Root Mean Square Propagation) :

Es un algoritmo de optimización que ajusta la tasa de aprendizaje para cada parámetro de manera adaptativa.

¿Por qué necesitamos algo como RMSprop?

Una tasa de aprendizaje fija para todos los parámetros puede causar problemas cuando:

- Algunas direcciones tienen gradientes grandes → los saltos son muy largos.
- Otras tienen gradientes pequeños → el avance es lento.

Además, en problemas con funciones de pérdida muy onduladas (no suaves), el optimizador puede oscilar mucho y no converger bien.

Cuidarse de la inestabilidad…..

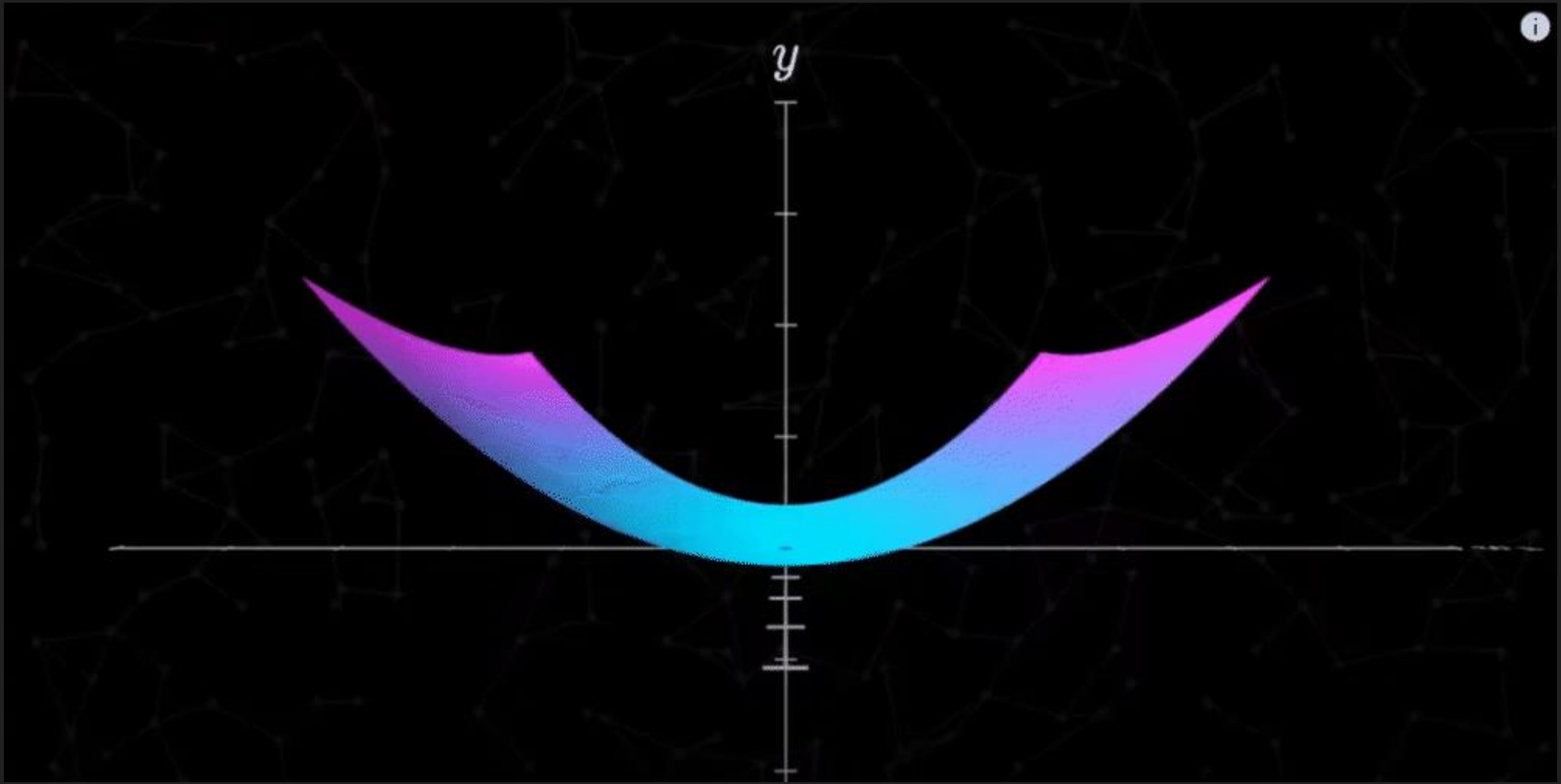
$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t + (1-\beta)\mathbf{g}(\mathbf{w}_t)^2$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\boldsymbol{\varepsilon} + \sqrt{\mathbf{v}_{t+1}}}\mathbf{g}(\mathbf{w}_t)$$

🔍 Intuición

- Si el gradiente es **consistentemente grande (oscila, mucha varianza)** → el paso será pequeño.
- Si el gradiente es **pequeño** → el paso será más grande.

Esto permite moverse más eficientemente en direcciones donde hay poco cambio, y con más cuidado donde el gradiente fluctúa mucho.


y

6. Adam (Hannah Montana) :

Combinar ideas de “promedio móvil de gradientes” (Momentum) y “promedio móvil de los cuadrados del gradiente” (RMSprop), para conseguir que el método sea estable y eficiente.

Promedio móvil de los gradientes (como Momentum):

$$m_t = \beta_1 \, m_{t-1} \; + \; (1 - \beta_1) \, \nabla L(w_t)$$

Promedio móvil de los cuadrados de los gradientes

$$v_t = \beta_2 \, v_{t-1} \; + \; (1 - \beta_2) \, (\nabla L(w_t))^2$$

Promedio móvil de los gradientes (como Momentum):

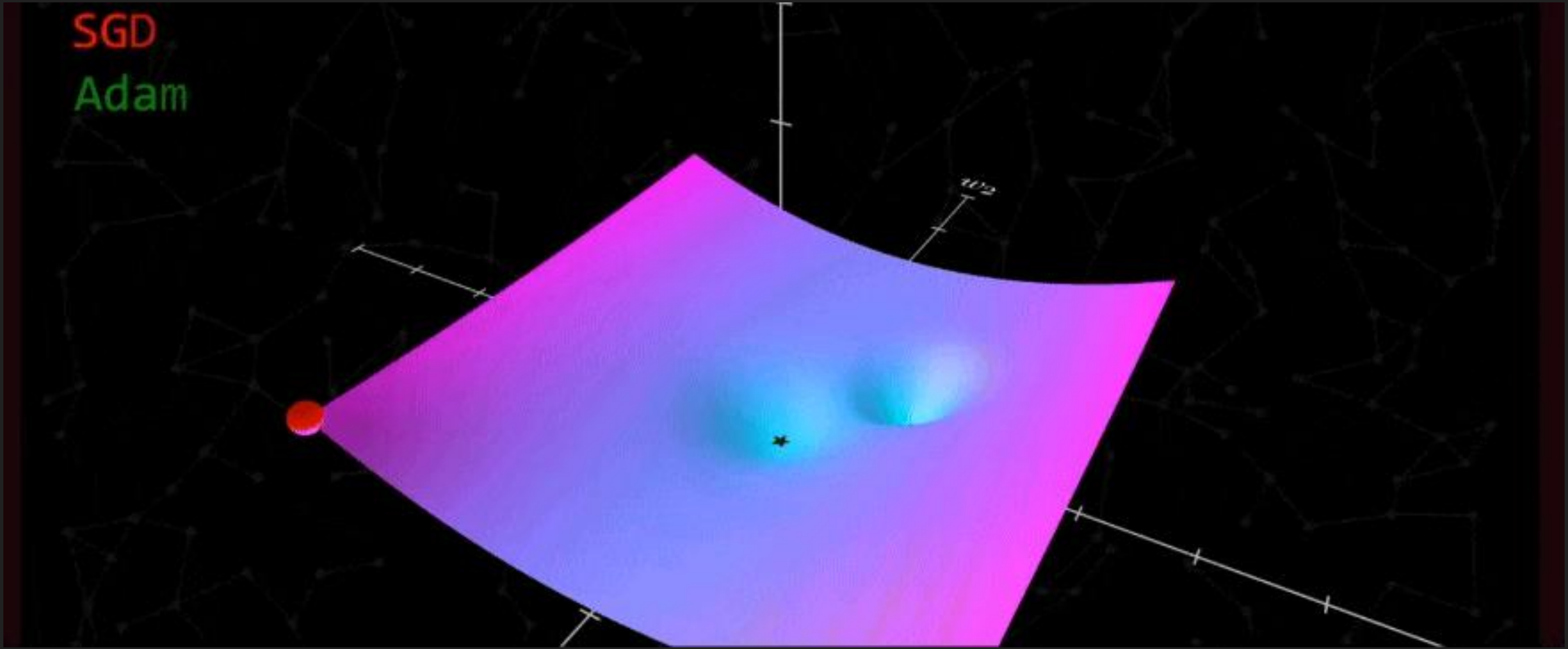
$$m_t = \beta_1 \, m_{t-1} \; + \; (1 - \beta_1) \, \nabla L(w_t)$$

Promedio móvil de los cuadrados de los gradientes

$$v_t = \beta_2 \, v_{t-1} \; + \; (1 - \beta_2) \, (\nabla L(w_t))^2$$

$$w_{t+1} = w_t \; - \; \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

$$w_{t+1} = w_t - \underbrace{\left(\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\right)}_{\text{“tasa de aprendizaje efectiva”}} \hat{m}_t$$

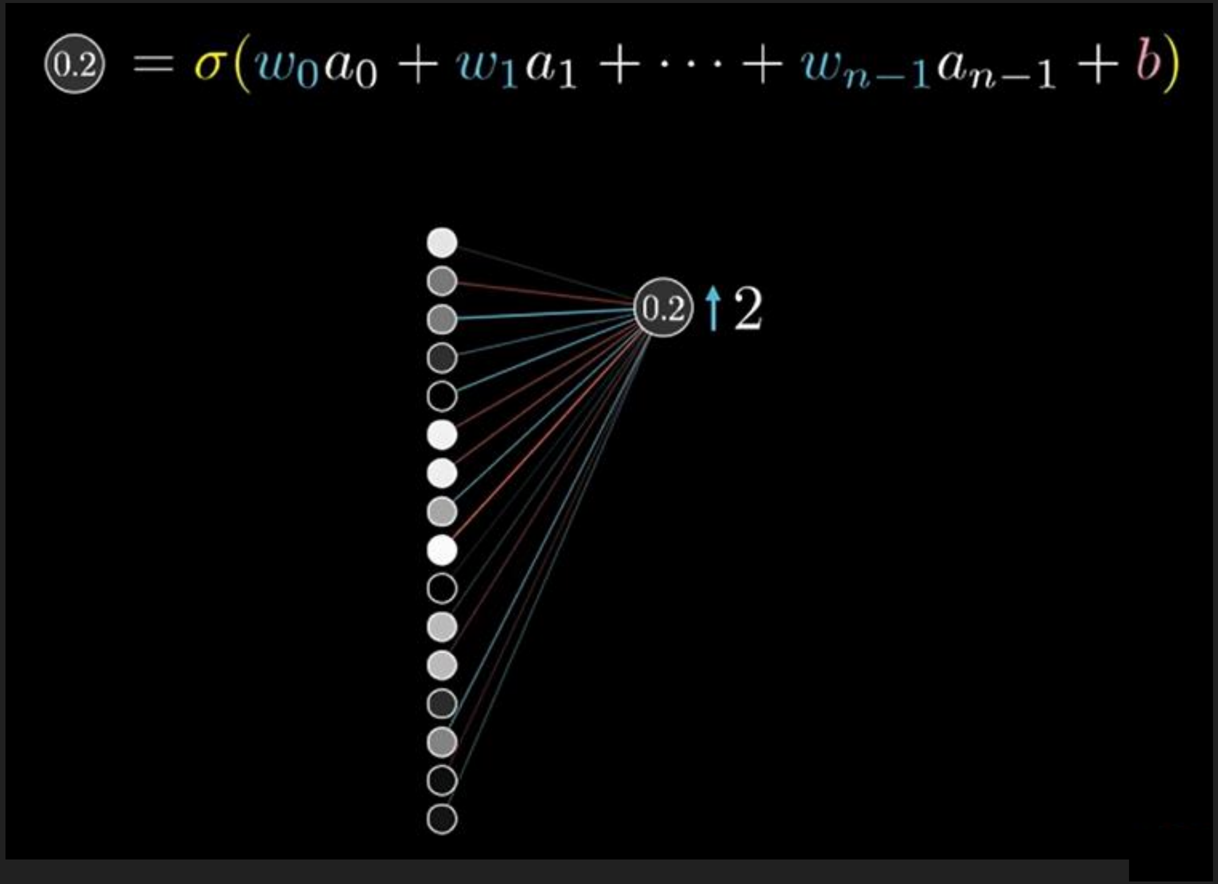

SGD
Adam
w2

¿Cómo calcular el gradiente?

Backpropagation

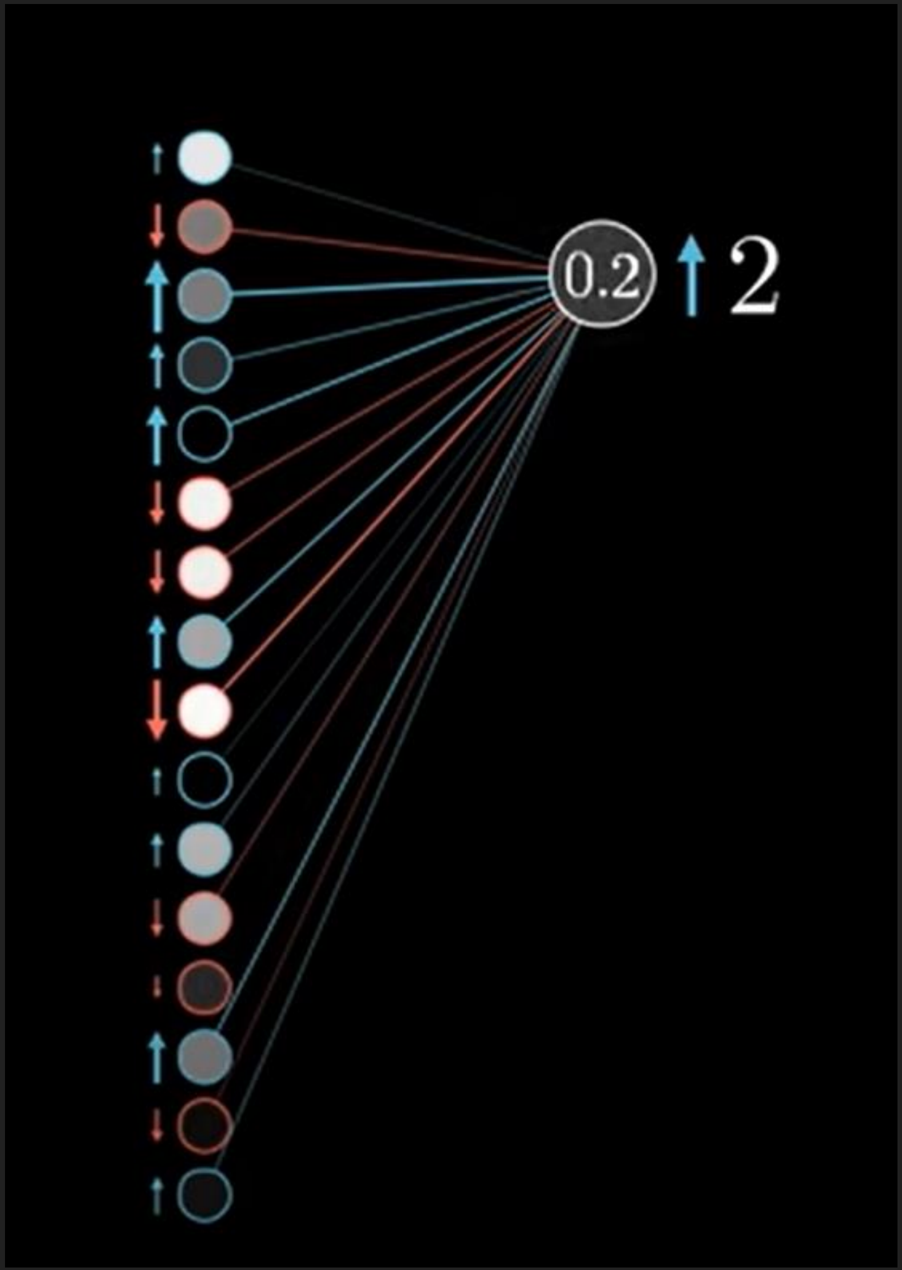
Andrés Daniel Godoy Ortiz

¿Cómo logro que esta neurona incremente su valor?



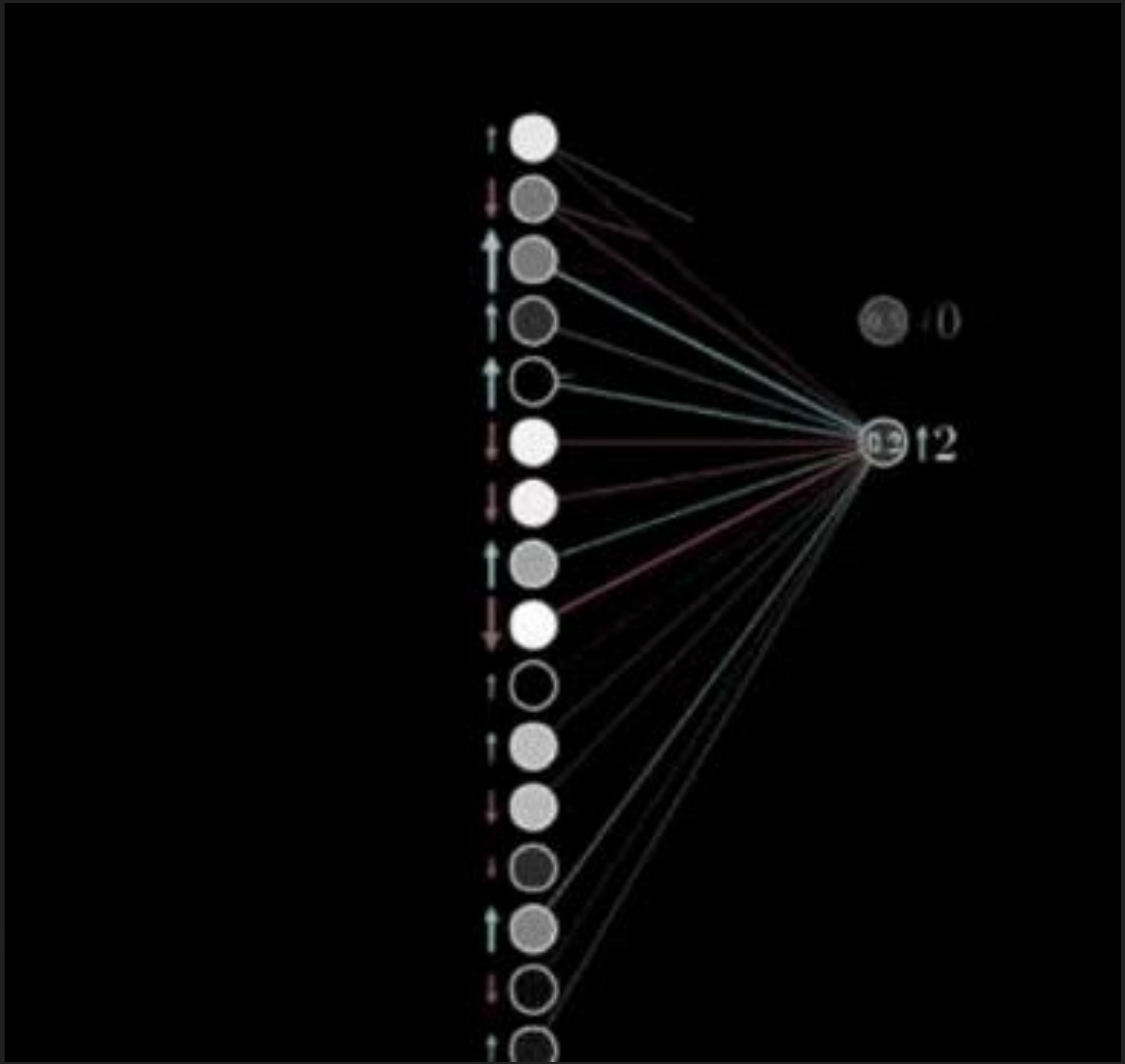
Incrementando w, b, o los valores de las neuronas anteriores

Incrementa los **pesos** en la proporción de los valores de las neuronas

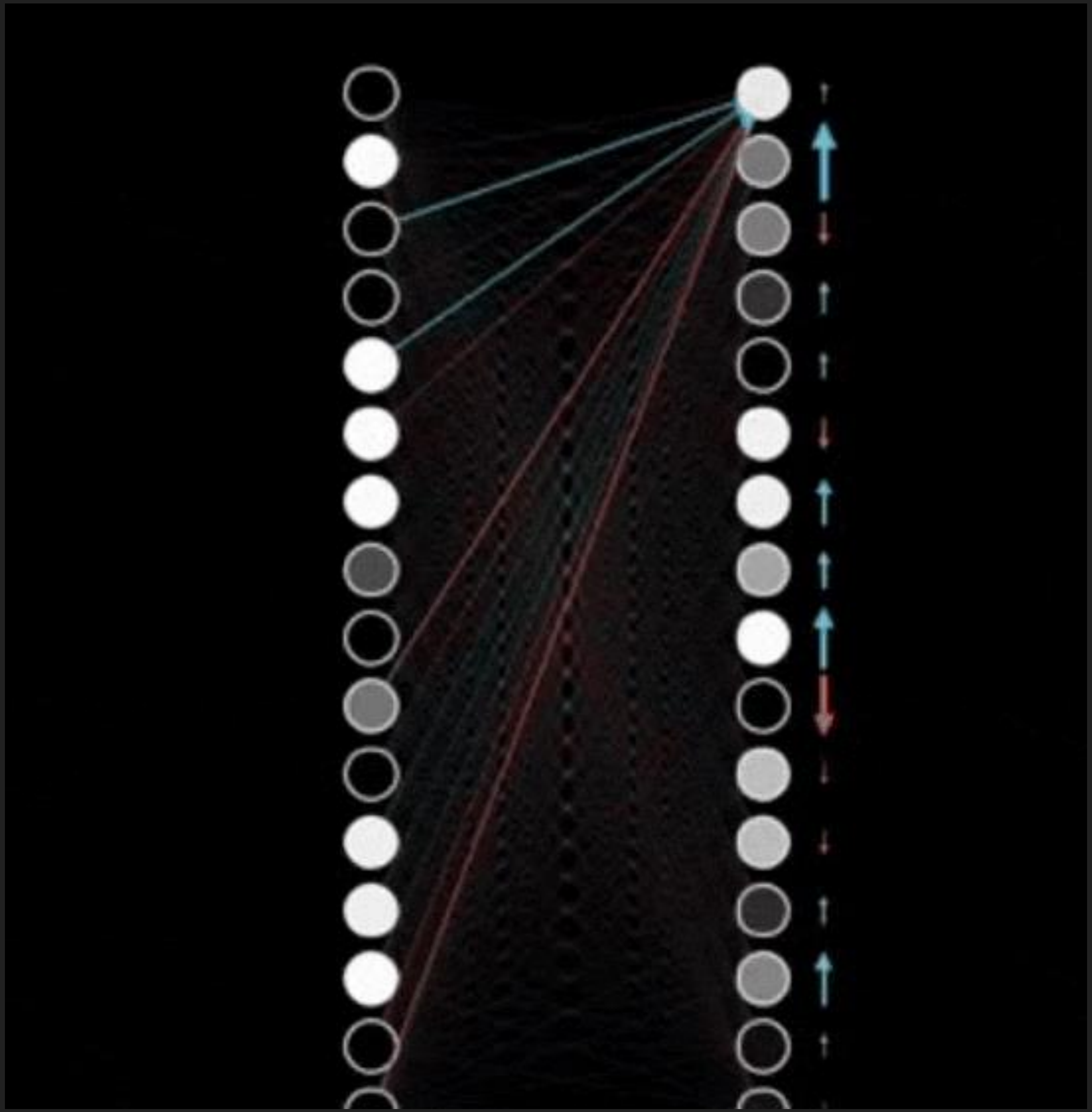


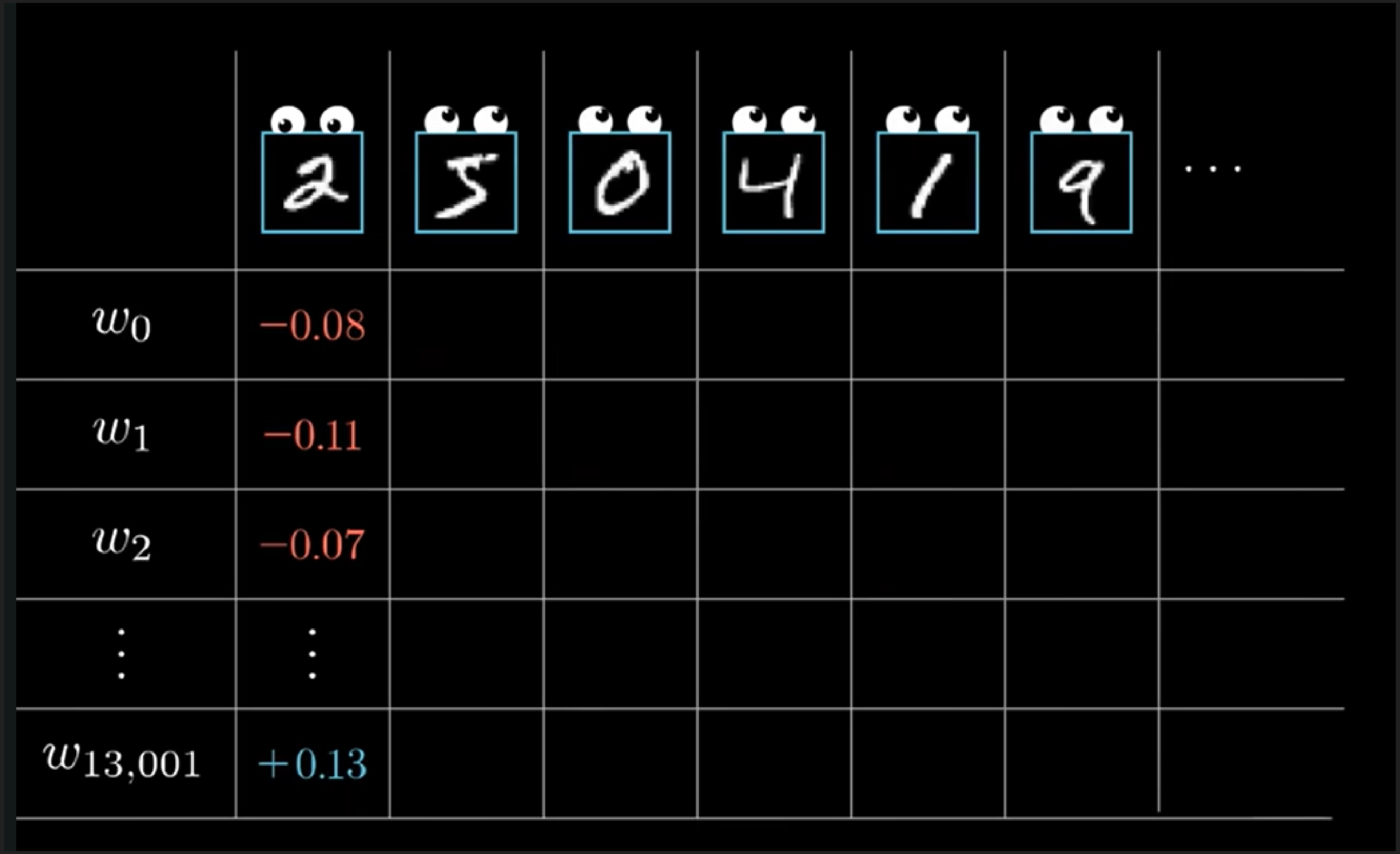
Aquí es donde entra la idea de backpropagation.

Al sumar todos los efectos deseados, obtienes una lista de ajustes que quieres aplicar a la penúltima capa.



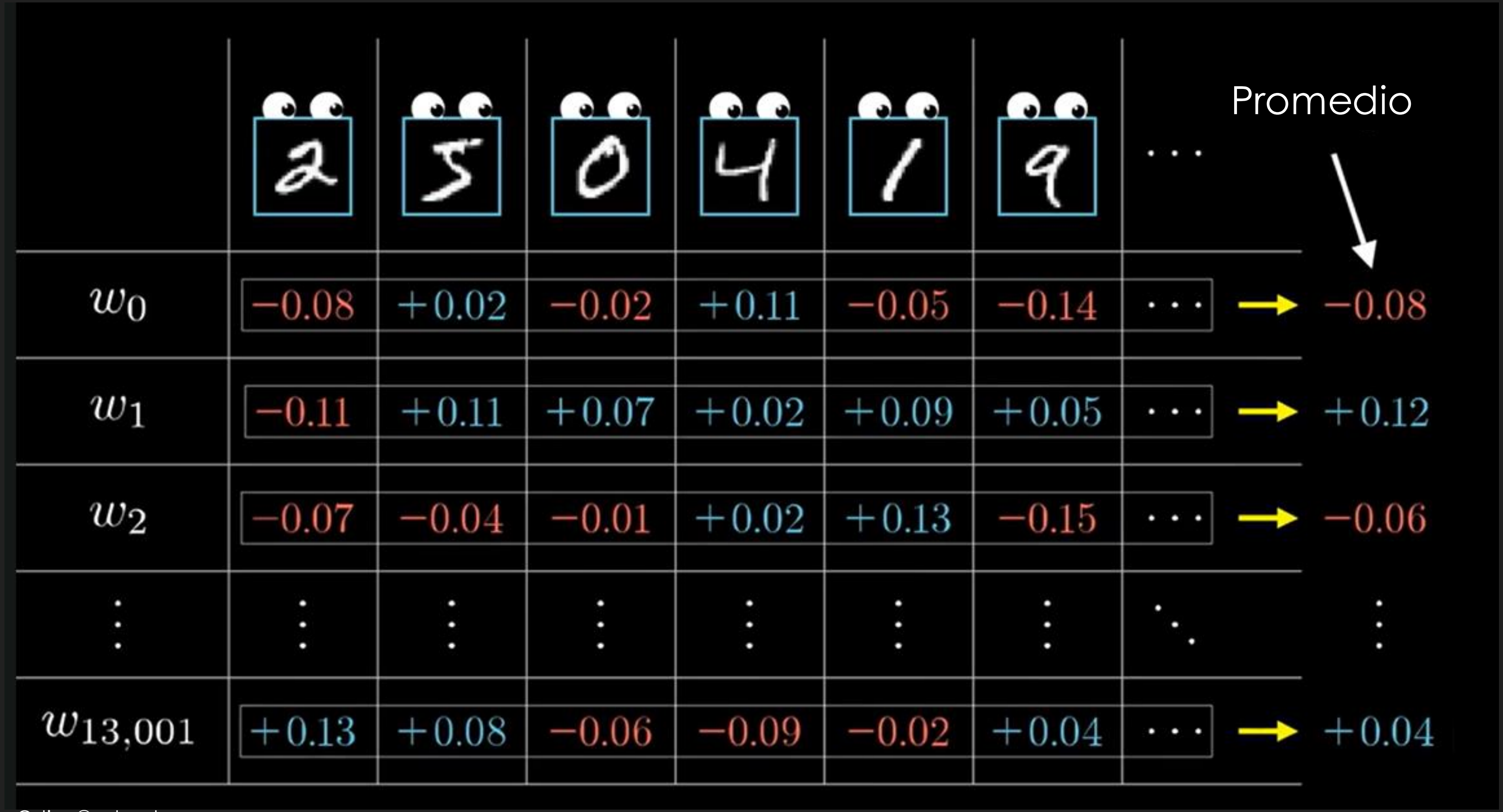
Luego, puedes repetir el mismo proceso con los pesos y sesgos que influyen en esa capa, retrocediendo por toda la red.





w_0
-0.08
w_1
-0.11
w_2
-0.07
$w_{13,001}$
$+0.13$

Es en este punto, donde resulta importante preguntarse con cuántas observaciones voy a calcular el gradiente



	2	5	0	4	1	9	...	Promedio
w_0	−0.08	+0.02	−0.02	+0.11	−0.05	−0.14	...	−0.08
w_1	−0.11	+0.11	+0.07	+0.02	+0.09	+0.05	...	+0.12
w_2	−0.07	−0.04	−0.01	+0.02	+0.13	−0.15	...	−0.06
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$w_{13,001}$	+0.13	+0.08	−0.06	−0.09	−0.02	+0.04	...	+0.04

Gracias

adgodoyo@gmail.com