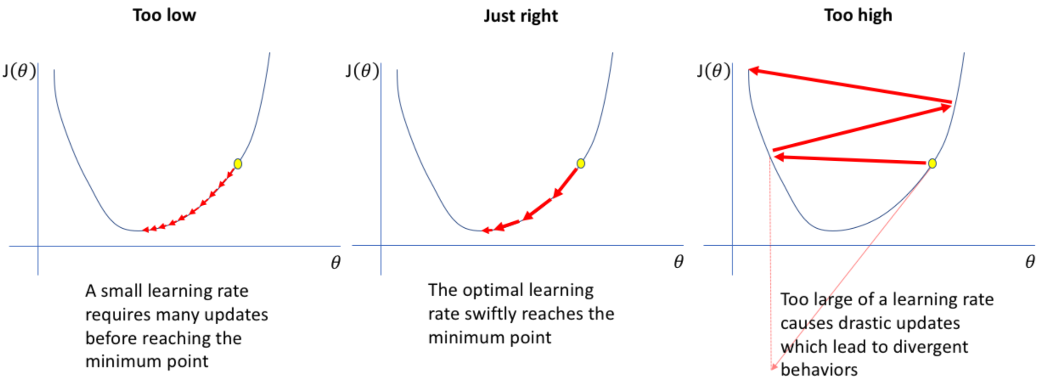
Modelos de Deep Learning

Variantes de Gradient Descent: Momento, Fricción y ADAM

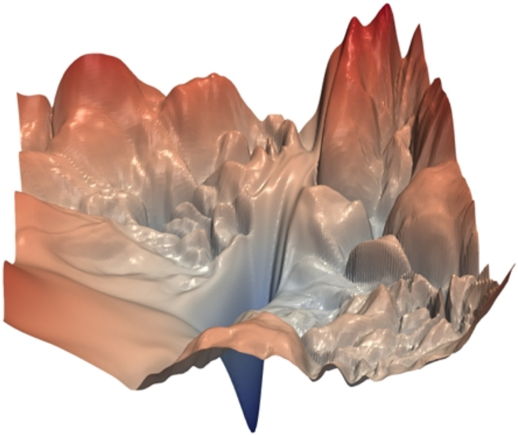
Universidad ORT Uruguay

15 de Setiembre, 2025

Gradient Descent - Learning Rate



¿Cómo encontrar el mínimo de esta función?



Costo empírico de ResNet-56

Visualizing the Loss Landscape of Neural Nets (2017)

Regularización

- Modificación a un algoritmo de aprendizaje que pretende reducir el error en validación (V) pero no su error empírico (en T).

- En su versión más simple consiste en agregar términos de penalización a la función de costo para desalentar modelos complejos:

$$\begin{pmatrix} \text{Costo} \\ \text{nuevo} \end{pmatrix} = \begin{pmatrix} \text{Costo} \\ \text{original} \end{pmatrix} + \lambda \times \begin{pmatrix} \text{Penalización} \\ \text{a la complejidad} \end{pmatrix}$$

- λ es un parámetro que controla la importancia del término de regularización.

- **Penalización L2 o Ridge** Para parámetros $\boldsymbol{\theta}$ (tensor) y costo $J(\boldsymbol{\theta})$

$$J_\lambda(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \tfrac{\lambda}{2}\,\|\boldsymbol{\theta}\|_2^2, \qquad \lambda \geq 0.$$

- **Gradiente regularizado**

$$\nabla J_\lambda(\boldsymbol{\theta}) = \nabla J(\boldsymbol{\theta}) + \lambda\,\boldsymbol{\theta}.$$

- **Actualización SGD con WD (L2 acoplado)**
  Sea $\mathbf{g}_t = \nabla J(\boldsymbol{\theta}_t)$ el gradiente en el paso t y $\eta > 0$ la tasa de aprendizaje:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\big(\mathbf{g}_t + \lambda\,\boldsymbol{\theta}_t\big) = (1 - \eta\lambda)\,\boldsymbol{\theta}_t - \eta\,\mathbf{g}_t.$$

- En `torch.optim.SGD`, el parámetro `weight_decay` implementa exactamente el término L2 acoplado (añade $\lambda\,\boldsymbol{\theta}_t$ al gradiente).

SGD completo:

Entrada: γ (lr), θ_0 (parms), λ (weight decay), μ (momentum), τ (dampening), `nesterov`.

1. **Para** $t = 1, 2, \ldots$ **hacer**:

   1.1 **Gradiente del paso**

   $$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$

   1.2 **L2 acoplado (weight decay)** si $\lambda \neq 0$:

   $$\mathbf{g}_t \leftarrow \mathbf{g}_t + \lambda\, \boldsymbol{\theta}_{t-1}$$

   1.3 **Momentum** si $\mu \neq 0$:

   $$\text{si } t = 1\text{: } \mathbf{v}_t \leftarrow \mathbf{g}_t \qquad \text{si } t > 1\text{: } \mathbf{v}_t \leftarrow \mu\, \mathbf{v}_{t-1} + (1 - \tau)\, \mathbf{g}_t$$

   Nesterov:

   $$\mathbf{g}_t \leftarrow \mathbf{g}_t + \mu\, \mathbf{v}_t, \text{ si } \texttt{nesterov} \qquad \mathbf{g}_t \leftarrow \mathbf{v}_t, \text{ si no}$$

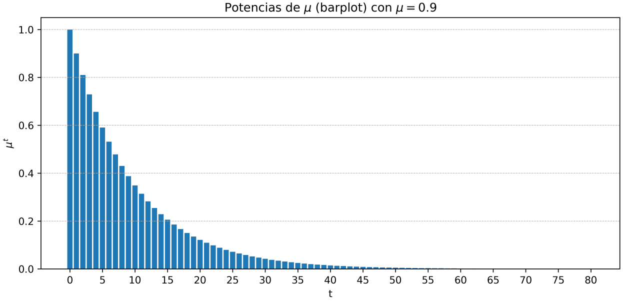
   1.4 **Actualización** $\quad \boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \gamma\, \mathbf{g}_t$

Devolver $\boldsymbol{\theta}_t$.

Interpretación del momento

Si $\lambda = 0$ (Weight Decay) y $\tau = 0$ (Dampening) y $\texttt{Nesterov}$=False:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \sum_{i=1}^{t} \mu^{t-i} \mathbf{g}_i$$



Momento y amortiguamiento: modelo físico (heavy–ball)

- Modelo continuo: Una partícula se mueve sobre el paisaje de $f(\boldsymbol{\theta})$:

$$\dot{\boldsymbol{\theta}} = \mathbf{v}, \qquad m\,\dot{\mathbf{v}} = -\nabla f(\boldsymbol{\theta}) - c\,\mathbf{v}$$

  - $-\nabla f(\boldsymbol{\theta})$: "fuerza" que empuja cuesta abajo.
  - $c\,\mathbf{v}$: fricción (amortiguamiento) que disipa energía.

- Discretización (esquema heavy–ball / momentum). Con paso Δt:

$$\mathbf{v}_{t+1} \approx \mu\,\mathbf{v}_t - \eta\,\nabla f(\boldsymbol{\theta}_t), \qquad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{v}_{t+1},$$

- $\mu = 1 - \frac{c}{m}\Delta t$ (cuánta *inercia* sobrevive)
- $\eta = \frac{\Delta t}{m}$ (escala del empuje por el gradiente).

Intuición

- Momentum (μ): conserva velocidad (*inercia*) $\Rightarrow$ suaviza zig–zag y acelera a lo largo del los valles.

- Amortiguamiento (fricción): reduce oscilaciones y evita "rebotes".

Interpretación geométrica en valles elípticos

- En una cuenca "alargada" (una elipse con ejes muy distintos en longitud):
  - En la dirección "empinada" el gradiente cambia de signo $\Rightarrow$ zig–zag.
  - En la dirección del valle el gradiente es coherente $\Rightarrow$ se puede acelerar.
- Con $\mathbf{g}_t = \nabla f(\boldsymbol{\theta}_t)$ y *dampening* τ:

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t + (1-\tau)\mathbf{g}_t \quad \Rightarrow \quad \mathbf{v}_t = \mu^t \mathbf{g}_1 + (1-\tau) \sum_{k=1}^{t} \mu^{t-i}\, \mathbf{g}_i.$$

  - μ actúa como suavizador de alta frecuencia (menos zig–zag transversal).
  - La "memoria" efectiva es $\approx \frac{1}{1-\mu}$ pasos (p.ej. $\mu = 0.9 \Rightarrow 10$).
  - τ recorta la inyección de gradiente nuevo $\Rightarrow$ pasos más contenidos.

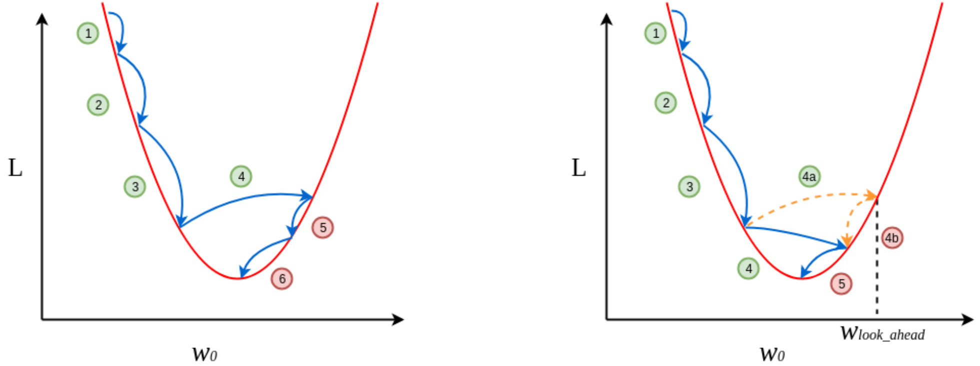
Nesterov: forma conceptual (*look–ahead*)

Idea: mirar el gradiente en el punto adelantado por el momentum.

$$\begin{aligned}
\mathbf{y}_t &:= \boldsymbol{\theta}_t - \eta\,\mu\,\mathbf{v}_t && \text{(punto adelantado)}\\
\tilde{\mathbf{g}}_t &:= \nabla f(\mathbf{y}_t) && \text{(gradiente en } \mathbf{y}_t\text{)}\\
\mathbf{v}_{t+1} &:= \mu\,\mathbf{v}_t - \eta\,\tilde{\mathbf{g}}_t && \text{(actualizar velocidad)}\\
\boldsymbol{\theta}_{t+1} &:= \boldsymbol{\theta}_t + \mathbf{v}_{t+1} && \text{(actualizar parámetros).}
\end{aligned}$$

En la expresión $\mathbf{y}_t = \boldsymbol{\theta}_t - \eta\,\mu\,\mathbf{v}_t$ no aparece un término $-\eta\mathbf{g}_t$ porque $\mathbf{y}_t$ no es el update de los parámetros, sino solo el punto "adelantado" donde se "mira" el gradiente. Ese desplazamiento previo usa solo la componente de inercia (el momentum $\mu\mathbf{v}_t$, no el empuje nuevo del gradiente $-\eta\mathbf{g}_t$ El término con $\mathbf{g}_t$ se aplica después, al hacer el paso real.

Nesterov Accelerated Gradient



(a) Momentum-Based Gradient Descent (b) Nesterov Accelerated Gradient Descent

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

Nesterov en PyTorch: forma implementada

Actualización usada por `torch.optim.SGD` (con L2 y dampening):

$$\boxed{\begin{aligned}
\mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_t) \; + \; \lambda\,\boldsymbol{\theta}_t \\
\mathbf{v}_{t+1} &\leftarrow \mu\,\mathbf{v}_t \; + \; (1-\tau)\,\mathbf{g}_t \\
\mathbf{d}_t &\leftarrow \mathbf{g}_t \; + \; \mu\,\mathbf{v}_{t+1} \qquad (\texttt{nesterov=True}) \\
\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t - \eta\,\mathbf{d}_t
\end{aligned}}$$

Si `nesterov=False`, se usa $\mathbf{d}_t = \mathbf{v}_{t+1}$ (momentum clásico).

Adam: idea y regla de actualización

- Adaptive Moment Estimation. Mantiene medias móviles exponenciales
- Con $\mathbf{g}_t = \nabla J(\boldsymbol{\theta}_t)$:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\mathbf{g}_t \qquad \text{(1er momento)}$$
$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\,\mathbf{g}_t^2 \qquad \text{(2do momento)}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}, \qquad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t} \quad \text{(corrección de sesgo)}$$

$$\boxed{\boldsymbol{\theta}_{t+1} \;=\; \boldsymbol{\theta}_t \;-\; \alpha\,\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\varepsilon}}$$

- Parámetros típicos: $\beta_1 = 0.9,\ \beta_2 = 0.999,\ \varepsilon = 10^{-8}$.

Adam en la práctica

- Paso *adaptativo por coordenada*:

  Cada coordenada tiene un learning rate efectivo distinto

- Costo de memoria: guarda $\mathbf{m}_t$ y $\mathbf{v}_t$.

- En Adam "clásico" el L2 es *acoplado* (se suma $\lambda\boldsymbol{\theta}$ al gradiente).

- Suele converger rápido con poco *tuning*; a veces se pasa luego a SGD+momentum para mejor generalización.