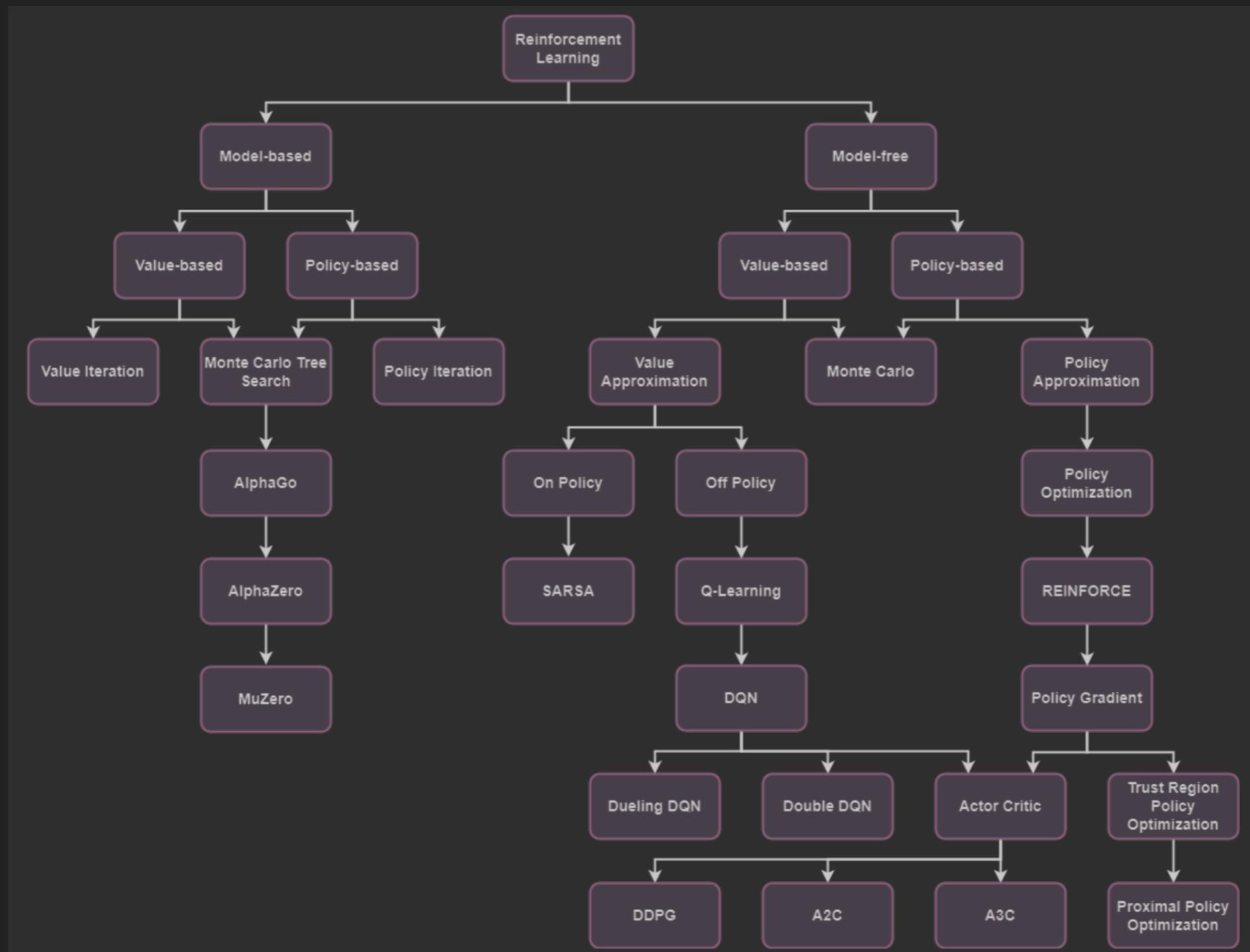


4

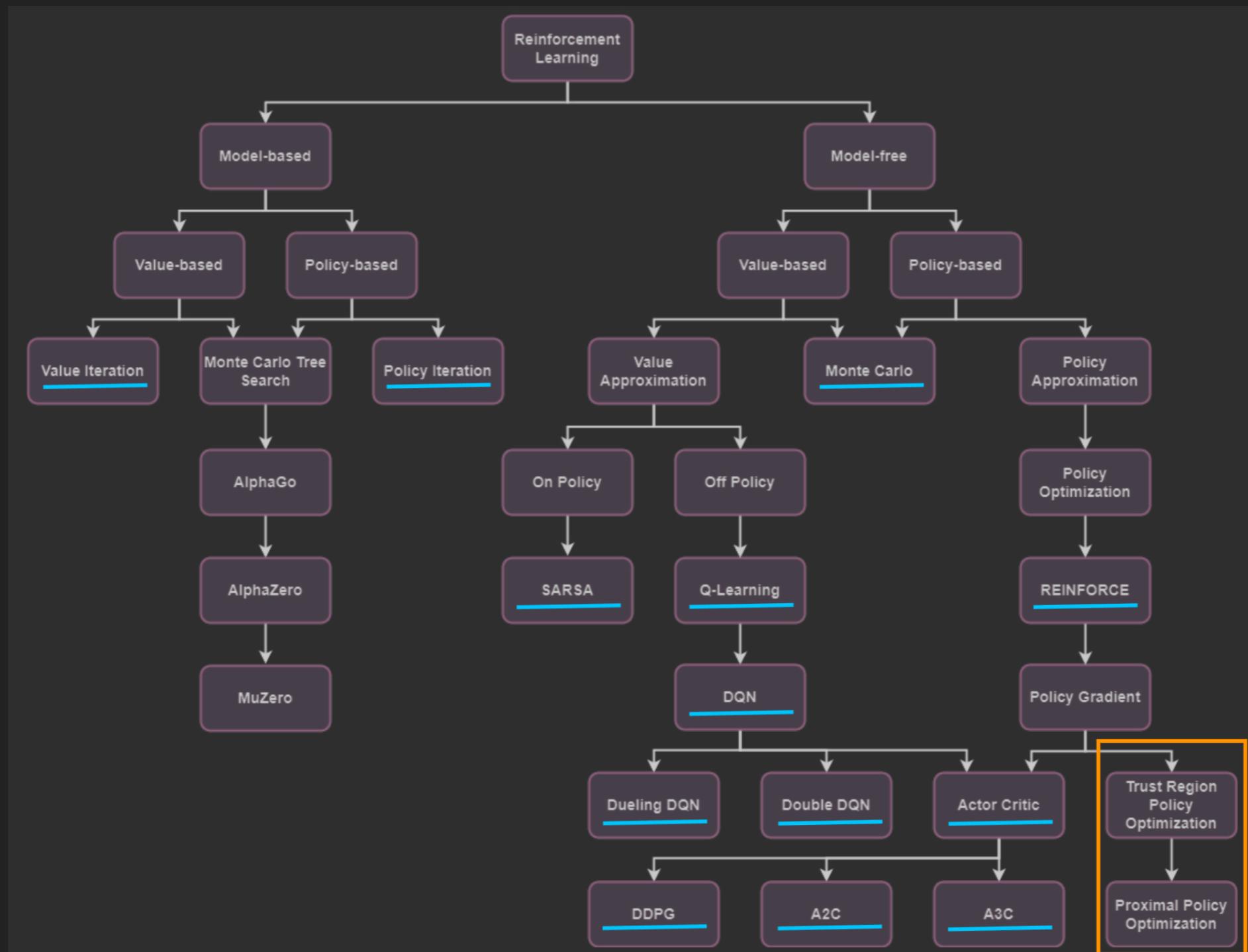
---

# TRPO - PPO

# CLASIFICACIÓN DE MÉTODOS RL



# CLASIFICACIÓN DE MÉTODOS RL

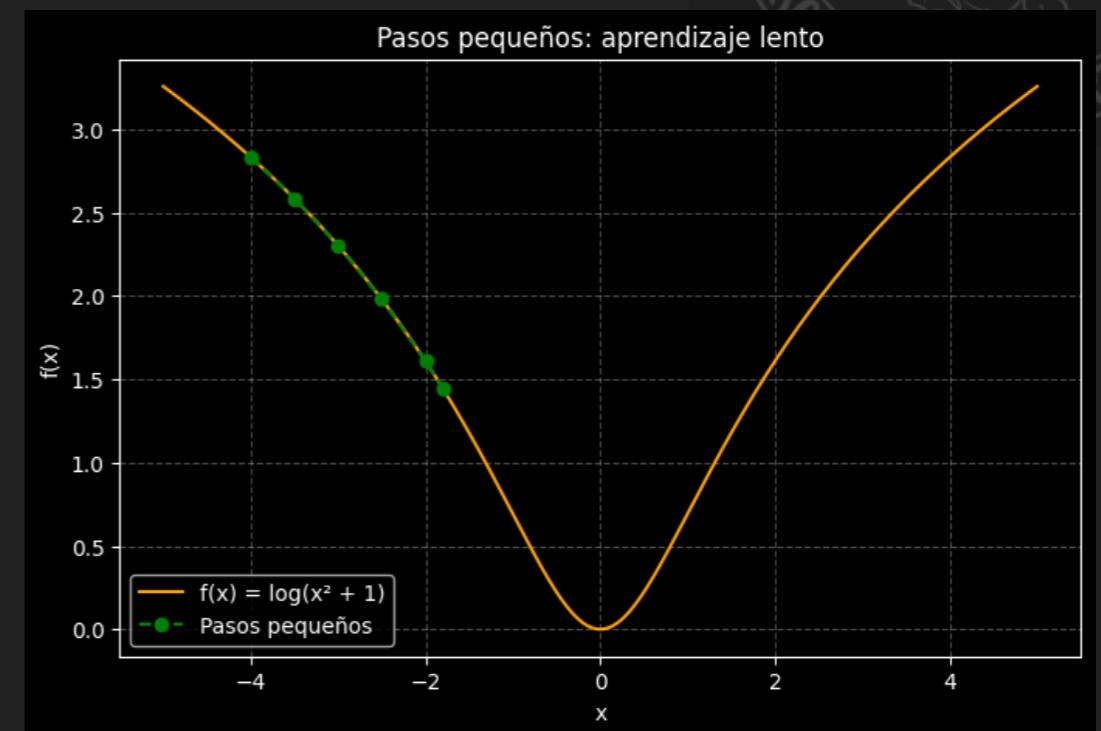
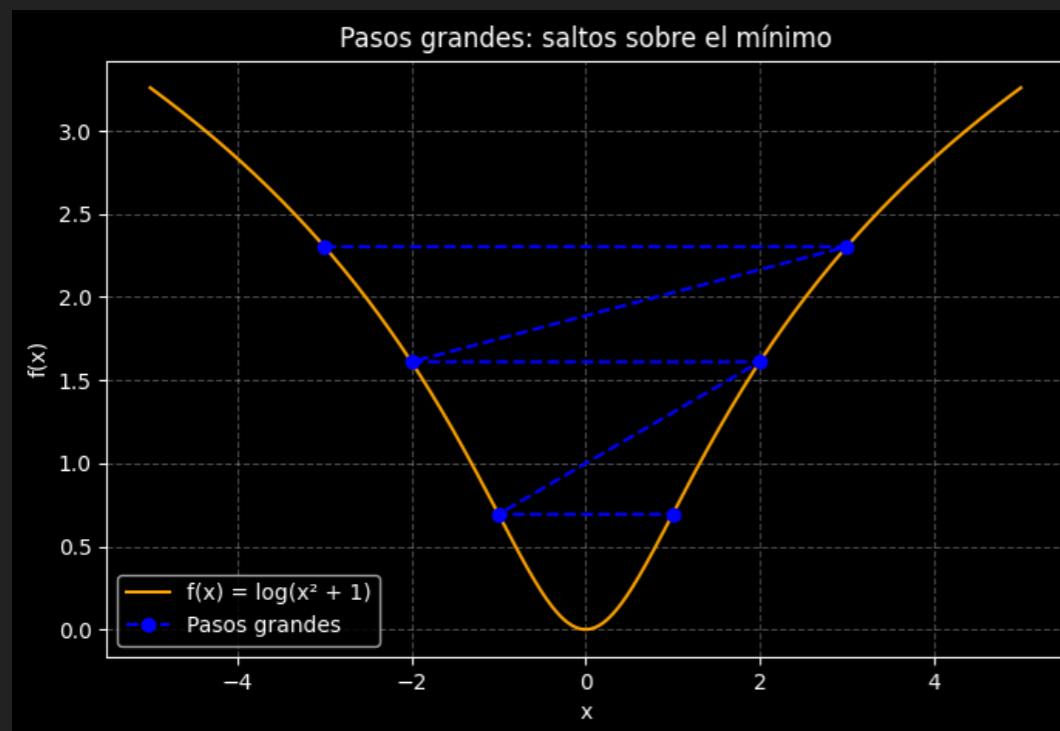


# EL PROBLEMA DE LOS ALGORITMOS DE GRADIENTE DE POLÍTICA

- ▶ Los algoritmos PG (REINFORCE, A2C) → actualizan los pesos de la política usando el gradiente del retorno esperado.

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

- ▶ Un optimizador de primer orden solo usa la pendiente local (el gradiente) para actualizar los parámetros.
- ▶ En regiones del espacio de pesos donde existe una alta curvatura (cercanas a un mínimo), los pasos pueden:
  - ✓ Ser demasiado grandes y saltar el mínimo (exceso de confianza).
  - ✓ Ser demasiado pequeños, ralentizando el aprendizaje.



## TRPO - ANALOGÍA

- ▶ El alpinista aprendiz y el guía experimentado.



# EL ALPINISTA APRENDIZ Y EL GUÍA EXPERIMENTADO

- ▶ El alpinista aprendiz (Actor)
- ▶ La montaña y los precipicios (Entorno)
- ▶ El guía experimentado (Crítico)



# EL ALPINISTA APRENDIZ Y EL GUÍA EXPERIMENTADO

► Exceso de confianza en PG:

✓ Si el paso es demasiado grande el alpinista cae al precipicio.

✓ Si el paso es demasiado pequeño, el alpinista aprende demasiado lento.



# EL ALPINISTA APRENDIZ Y EL GUÍA EXPERIMENTADO

## ► Región de confianza:

- ✓ Es una restricción que asegura que la nueva política (estilo de escalar del alpinista basada en lo que dice el guía) no se desvíe demasiado de su política antigua.
- ✓ Asegura de que el "paso" sea pequeño y esté dentro de una zona válida.



# TRPO

- ▶ TRPO fue presentado en 2015 con el título de "Optimización de política por región de confianza" (Schulman et al., 2015).
- ▶ TRPO se basa en gradientes de política de REINFORCE y la función de valor para estimar la Ventaja y reducir la varianza de Actor-Critic.



**Trust Region Policy Optimization**

---

John Schulman  
Sergey Levine  
Philipp Moritz  
Michael Jordan  
Pieter Abbeel

JOSCHU@EECS.BERKELEY.EDU  
SLEVINE@EECS.BERKELEY.EDU  
PCMORITZ@EECS.BERKELEY.EDU  
JORDAN@CS.BERKELEY.EDU  
PABBEEL@CS.BERKELEY.EDU

University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

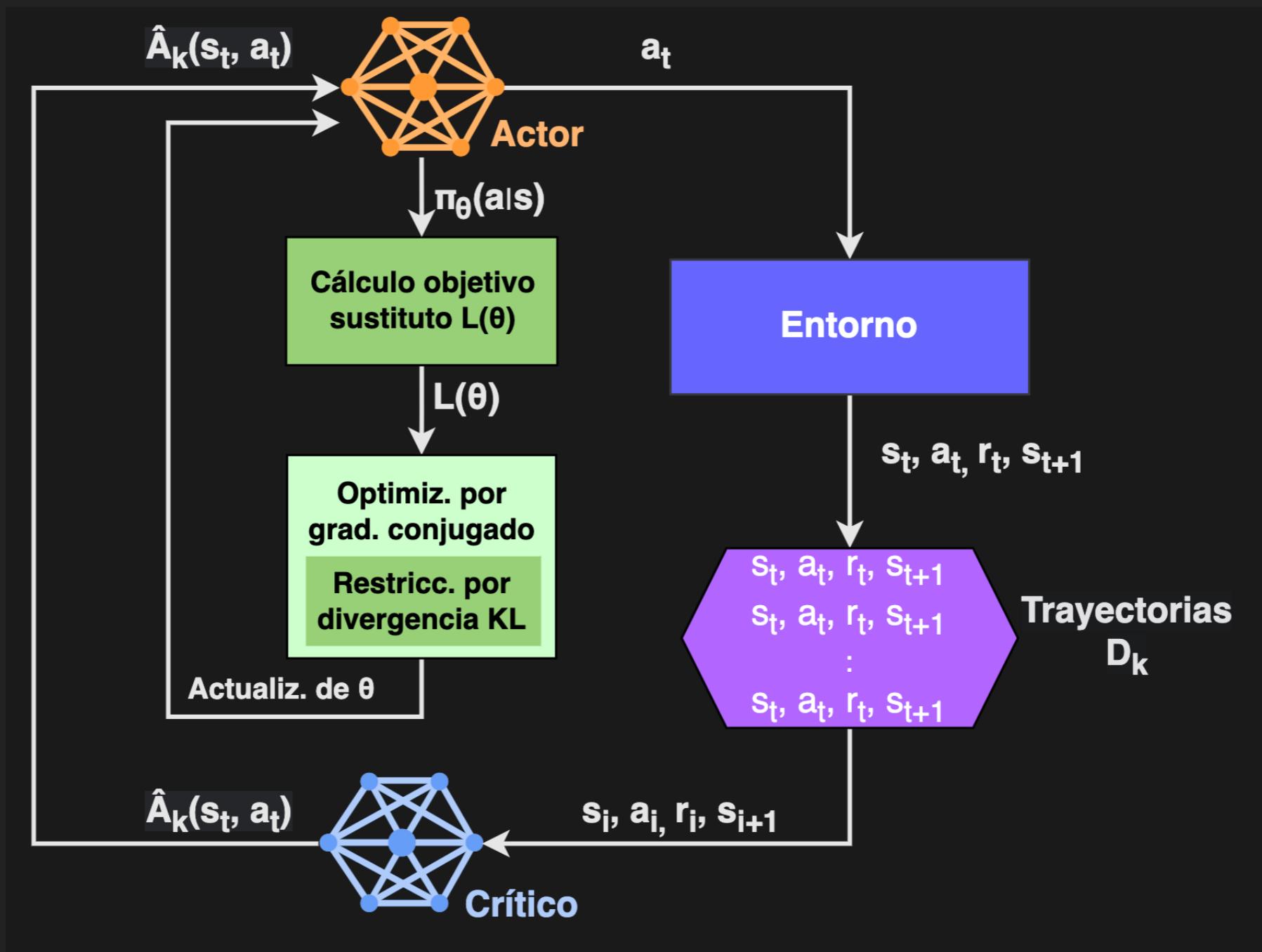
---

**Abstract**

In this article, we describe a method for optimizing control policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified scheme, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

namic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabilon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

# ARQUITECTURA DE TRPO



# PSEUDOCÓDIGO DE TRPO

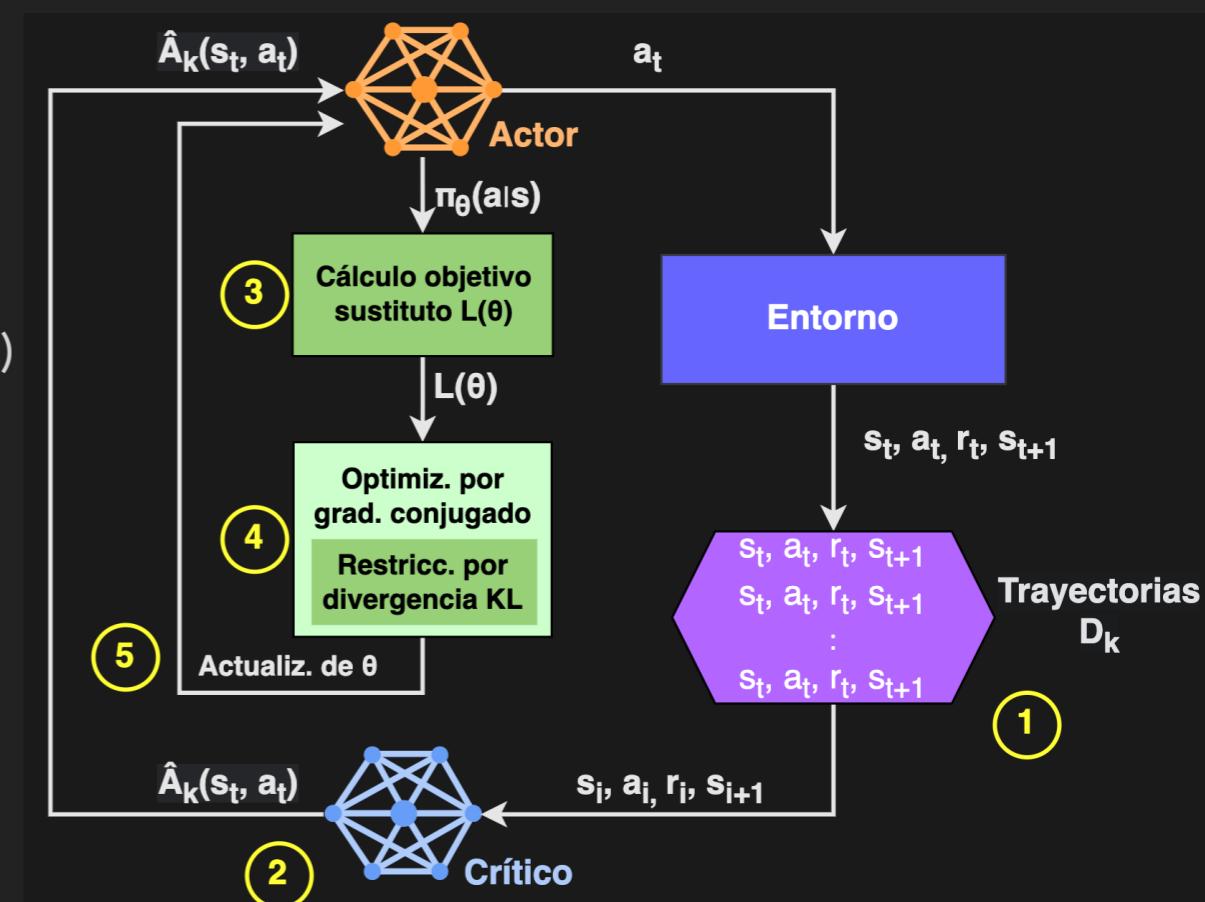
Iniciar los parámetros de la política  $\theta_0$

Iniciar los parámetros de la función de valor  $\phi_0$

Iniciar tamaño de la región de confianza  $\delta$

Loop para  $k = 0, 1, 2, \dots$ :

- 1 Recopilar un conj. de trayect.  $D_k = \{\tau_i\}$  dada la polít.  $\pi_k = \pi(\cdot | \cdot, \theta_k)$
- Para cada trayectoria en  $D_k$  y para cada paso de tiempo  $t$ :
  - Calcular los retornos  $R_t$
  - Calcular las ventajas estimadas  $\hat{A}_k(s_t, a_t)$
  - Calcular el objetivo sustituto (surrogate objective)
  - Optimizar ese objetivo respetando una trust region
  - Actualizar los pesos



# FUNCIÓN DE VENTAJA

Iniciar los parámetros de la política  $\theta_0$

Iniciar los parámetros de la función de valor  $\phi_0$

Iniciar tamaño de la región de confianza  $\delta$

Loop para  $k = 0, 1, 2, \dots$ :

① Recopilar un conj. de trayect.  $D_k = \{\tau_i\}$  dada la polít.  $\pi_k = \pi(\cdot|\cdot, \theta_k)$

Para cada trayectoria en  $D_k$  y para cada paso de tiempo  $t$ :

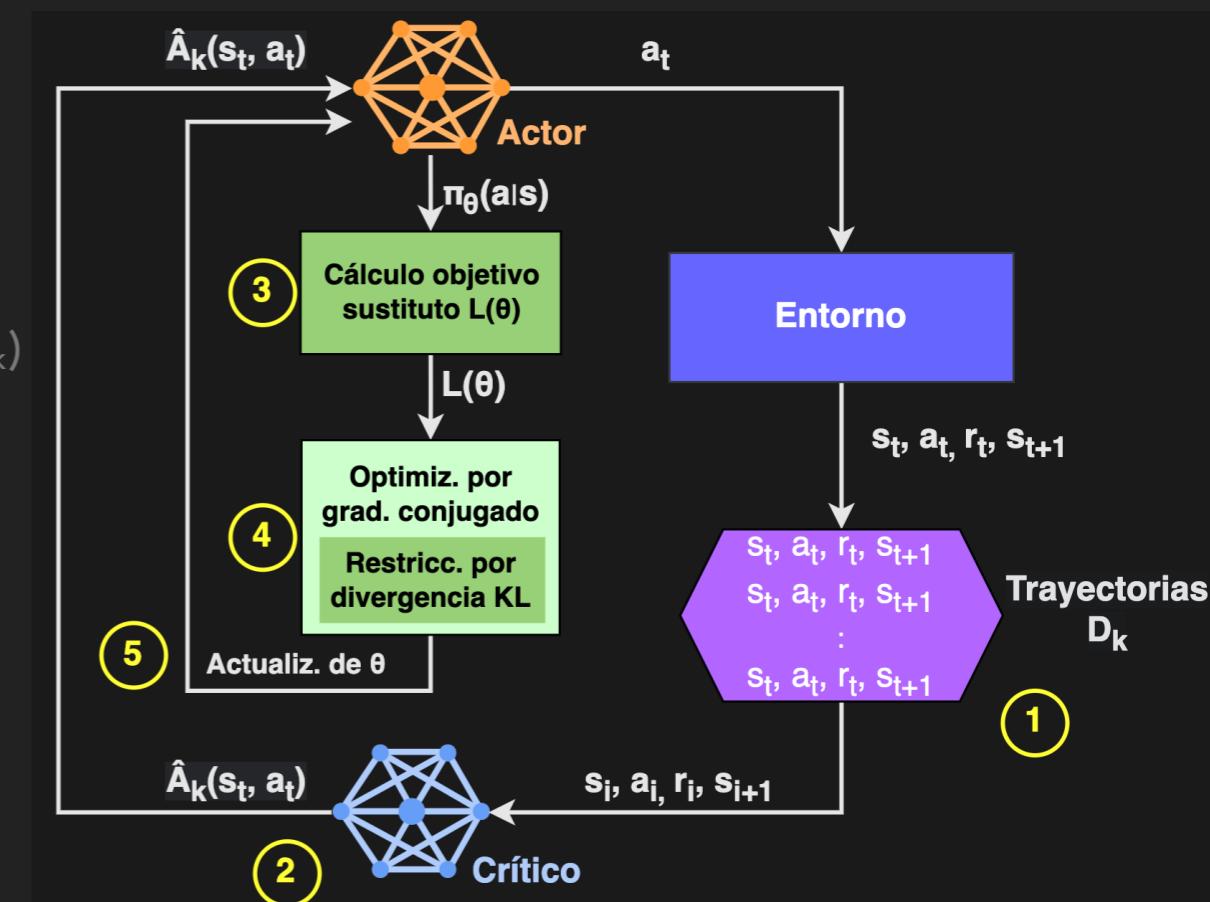
Calcular los retornos  $R_t$

② Calcular las ventajas estimadas  $\hat{A}_k(s_t, a_t)$

③ Calcular el objetivo sustituto (surrogate objective)

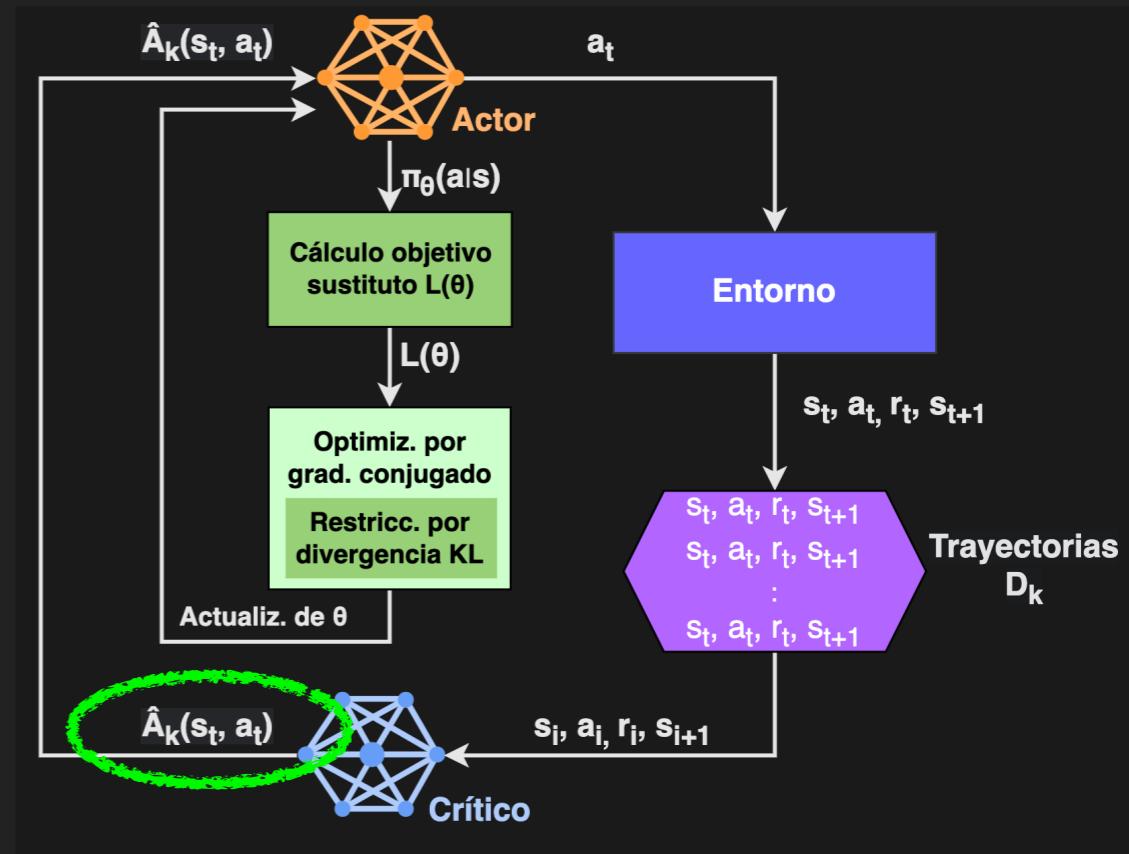
④ Optimizar ese objetivo respetando una trust region

⑤ Actualizar los pesos



## FUNCIÓN DE VENTAJA

- ▶  $A(s,a)$ : Después de que el **Actor** da un paso (**acción  $a_t$** ), el guía evalúa ese paso específico:
- ▶ **Guía**: "Ese paso condujo a un lugar mucho mejor de lo que esperaba desde la posición anterior." (**Ventaja positiva**).
- ▶ **Guía**: "Ese paso no fue tan bueno. Esperaba un avance mas o que conduzca a una mejor posición." (**Ventaja negativa**).



# CÁLCULO DE LA FUNCIÓN DE VENTAJA

- ▶ Dueling DQN. Se obtiene calculando:

$$Q(s, a) = V(s) + A(s, a)$$

- ▶ Diferencia Temporal (TD). se calcula con:

$$A^t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

- ▶ GAE Generalized Advantage Estimation (Schulman et al., 2016):

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}$$

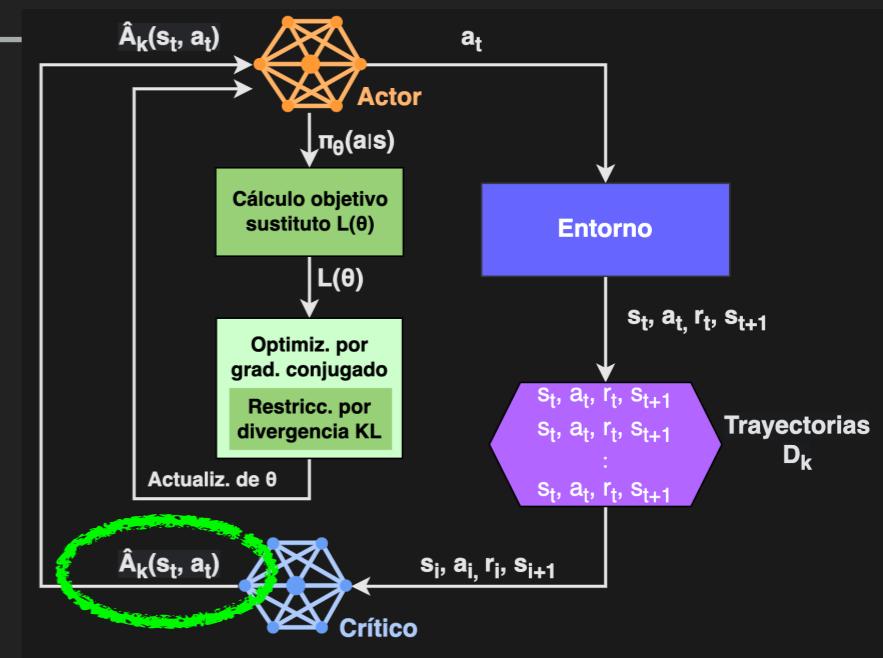
- ▶ donde:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

✓ Si  $\lambda=1$ : la estimación es de tipo Monte Carlo (alta varianza, bajo sesgo).

✓ Si  $\lambda=0$ : la estimación no está optimizada ni ajustada al entorno real (bajo sesgo, alta varianza).

✓ Típicamente se usa  $\lambda=0.95$ .



# CÁLCULO DEL OBJETIVO SUSTITUTO

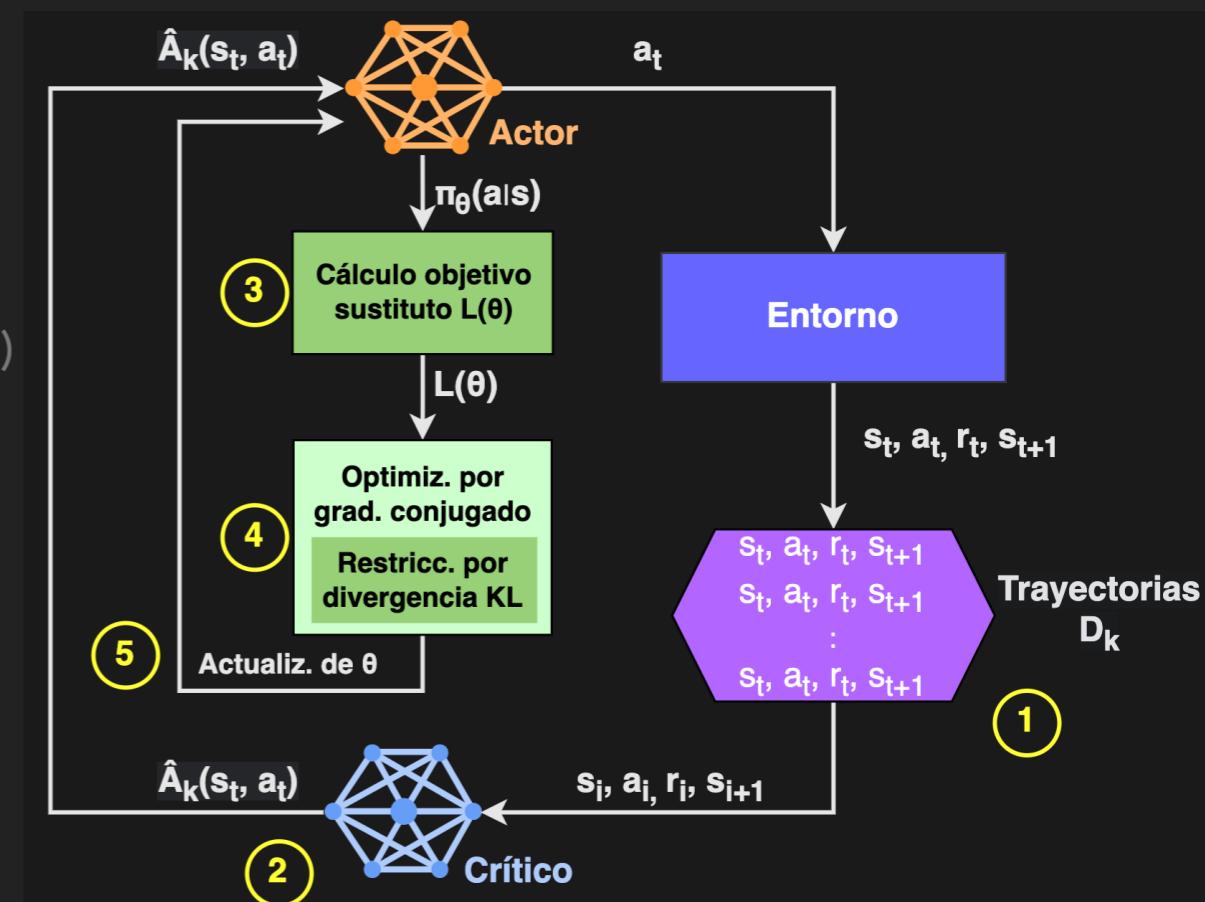
Iniciar los parámetros de la política  $\theta_0$

Iniciar los parámetros de la función de valor  $\phi_0$

Iniciar tamaño de la región de confianza  $\delta$

Loop para  $k = 0, 1, 2, \dots$ :

- 1 Recopilar un conj. de trayect.  $D_k = \{\tau_i\}$  dada la polít.  $\pi_k = \pi(\cdot | \cdot, \theta_k)$
- Para cada trayectoria en  $D_k$  y para cada paso de tiempo  $t$ :
  - Calcular los retornos  $R_t$
  - Calcular las ventajas estimadas  $\hat{A}_k(s_t, a_t)$
  - Calcular el objetivo sustituto (surrogate objective)**
  - Optimizar ese objetivo respetando una trust region
  - Actualizar los pesos



## CÁLCULO DEL OBJETIVO SUSTITUTO

- ▶ El objetivo en política estocástica es maximizar el rendimiento esperado:

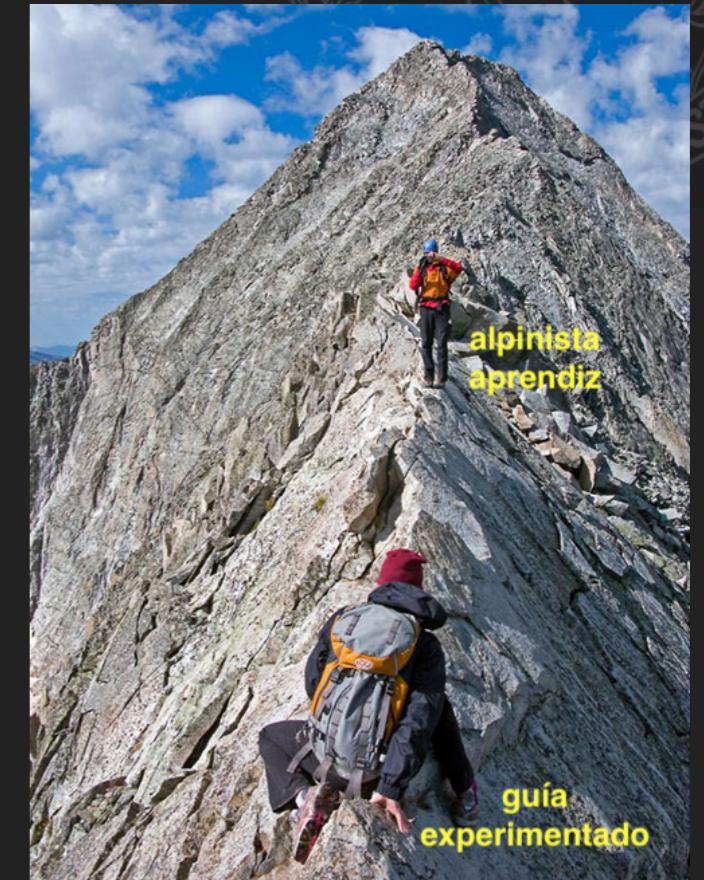
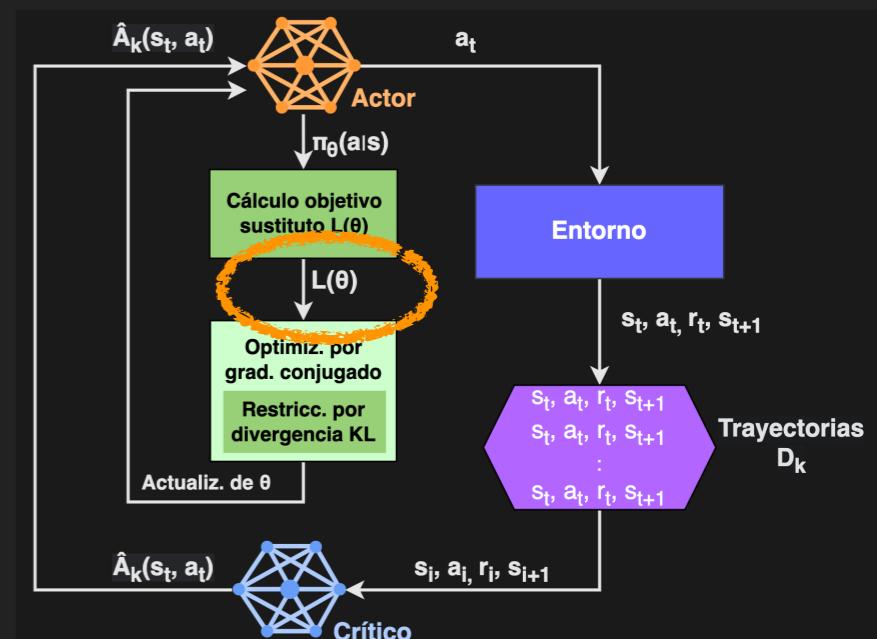
$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^T r_t \right]$$

- ▶ TRPO no lo utiliza porque su gradiente puede ser inestable o llevar a pasos críticos.

- ▶ En su lugar, TRPO define una aproximación más segura y estable del objetivo basada en una política anterior  $\pi_{\theta_{\text{old}}}$ :

$$L^{\text{TRPO}}(\theta) = \mathbb{E}_{(s,a) \sim \pi_{\theta_{\text{old}}}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}(s,a) \right]$$

- ▶ Este **sustituye** al verdadero objetivo  $J$ .

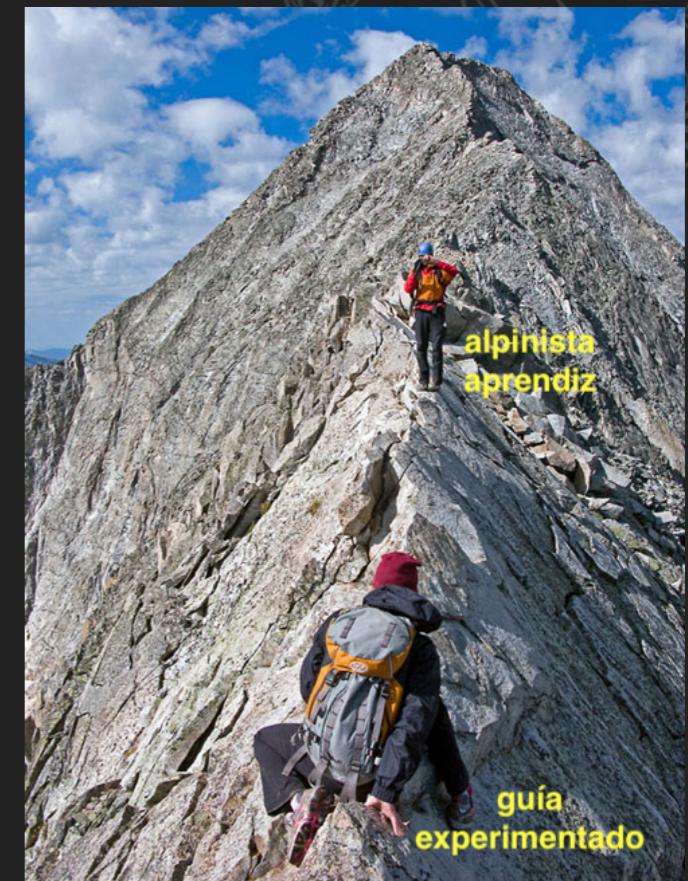
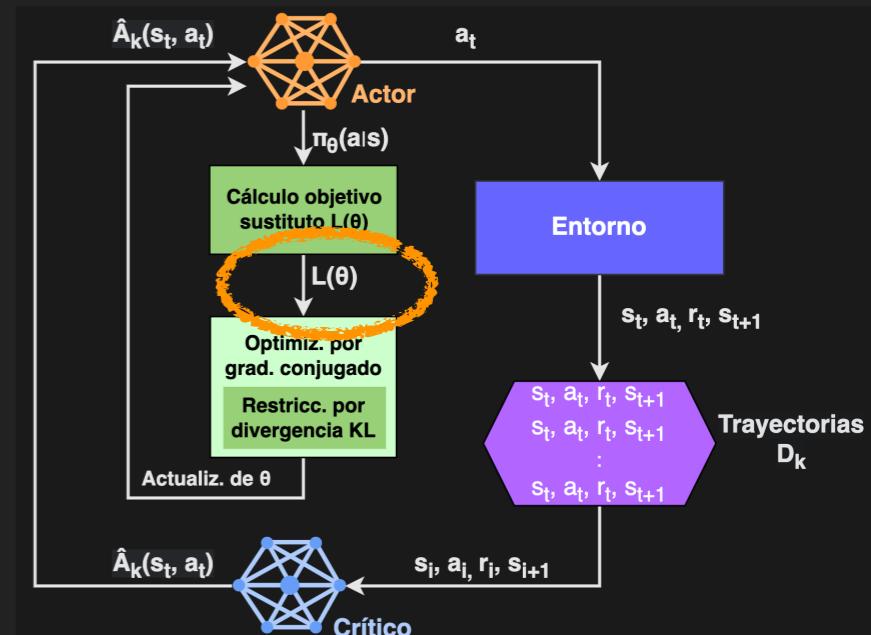


## INTERPRETACIÓN DEL OBJETIVO SUSTITUTO

- ▶ **Alpinista:** "Si cambio mi estilo un poquito de  $\pi_{\theta_{\text{old}}}$  a  $\pi$ , ¿cómo se espera que cambie la ventaja de mis acciones, según la evaluación de mi guía basada en mi experiencia reciente con  $\pi_{\theta_{\text{old}}}$ ?"

$$L^{\text{TRPO}}(\theta) = \mathbb{E}_{(s,a) \sim \pi_{\theta_{\text{old}}}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}(s,a) \right]$$

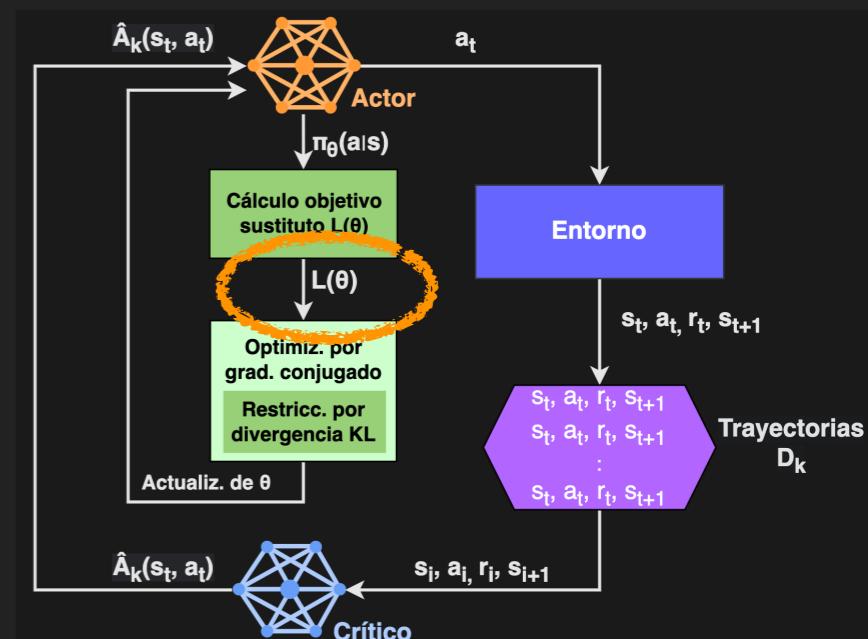
- ▶ **Surrogate objective** utiliza la razón de probabilidad de importancia (**importance sampling ratio**):  $r_t(\theta) = \pi_\theta(a_t|s_t) / \pi_{\theta_{\text{old}}}(a_t|s_t)$
- ▶ Este ratio mide cuánto más (o menos) probable es tomar una acción  $a_t$  en el estado  $s_t$  con la nueva política  $\pi_\theta$  en comparación con la política antigua  $\pi_{\theta_{\text{old}}}$ .



# INTERPRETACIÓN DEL OBJETIVO SUSTITUTO

- ▶ El objetivo sustituto básico es:

$$L^{\text{TRPO}}(\theta) = \mathbb{E}_{(s,a) \sim \pi_{\theta_{\text{old}}}} \left[ \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}(s, a) \right]$$



- ▶  $\pi_{\theta}(a_t|s_t) / \pi_{\theta_{\text{old}}}(a_t|s_t)$ : "Si hubiera usado mi nuevo estilo potencial en esta situación que experimenté con mi estilo antiguo, ¿qué tan diferente habría sido mi elección?"
- ▶  $A_{\theta_{\text{old}}}(s, a)$ : "¿cuál fue la ventaja de la acción que realmente tomé con mi estilo antiguo, según mi guía?"



# OPTIMIZACIÓN DE L MEDIANTE REGIÓN DE CONFIANZA

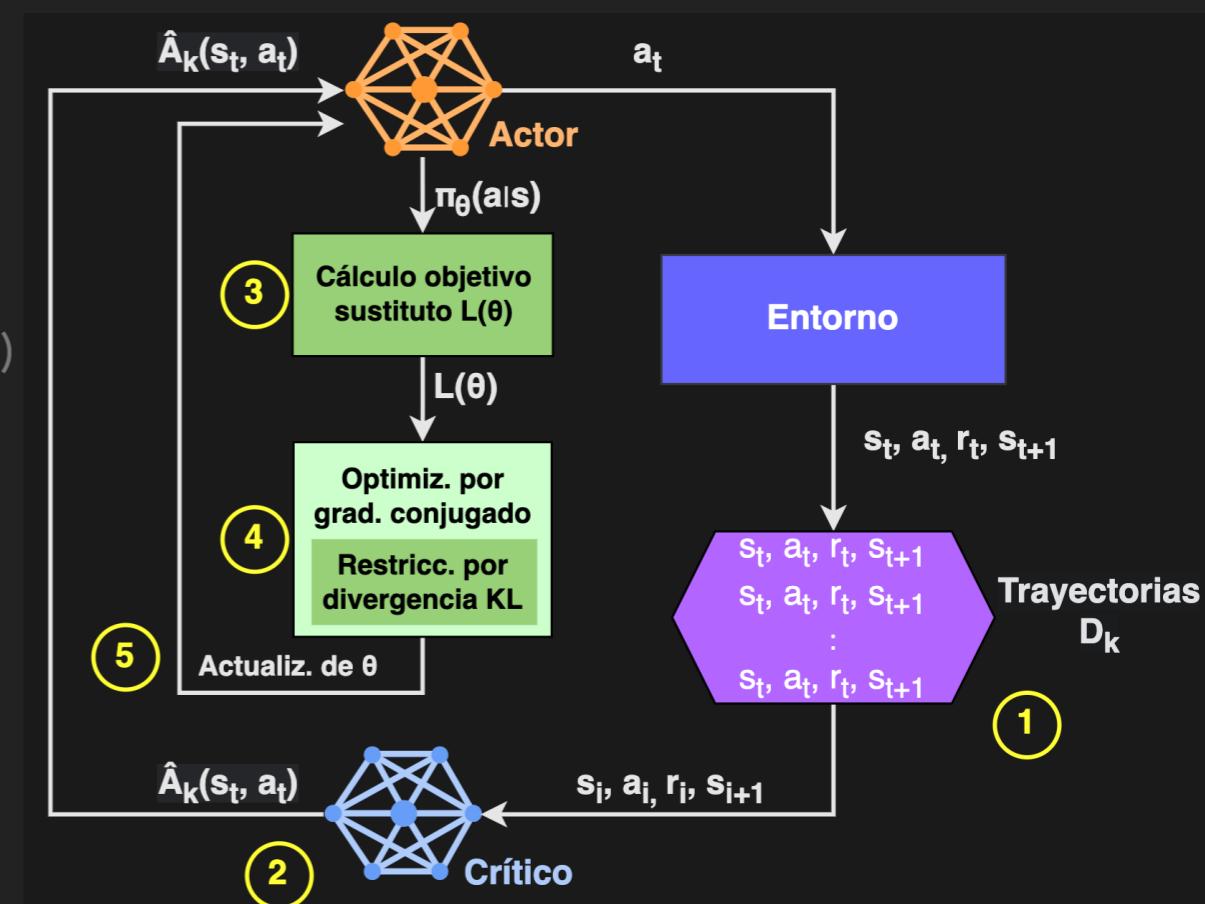
Iniciar los parámetros de la política  $\theta_0$

Iniciar los parámetros de la función de valor  $\phi_0$

Iniciar tamaño de la región de confianza  $\delta$

Loop para  $k = 0, 1, 2, \dots$ :

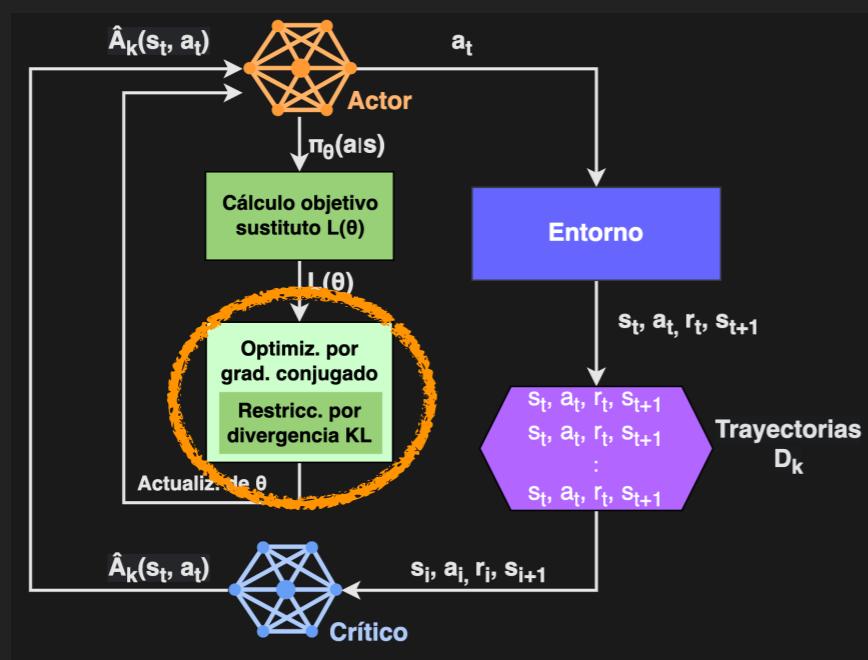
- 1 Recopilar un conj. de trayect.  $D_k = \{\tau_i\}$  dada la polít.  $\pi_k = \pi(\cdot | \cdot, \theta_k)$
- Para cada trayectoria en  $D_k$  y para cada paso de tiempo  $t$ :
  - Calcular los retornos  $R_t$
  - Calcular las ventajas estimadas  $\hat{A}_k(s_t, a_t)$
  - Calcular el objetivo sustituto (surrogate objective)
  - Optimizar ese objetivo respetando una trust region**
  - Actualizar los pesos



# OPTIMIZACIÓN DE L MEDIANTE REGIÓN DE CONFIANZA

- Se calcula el gradiente de L para saber en qué dirección habría que modificar los parámetros  $\theta$  para luego maximizar  $L(\theta)$ :

$$g = \nabla_{\theta} L(\theta)$$

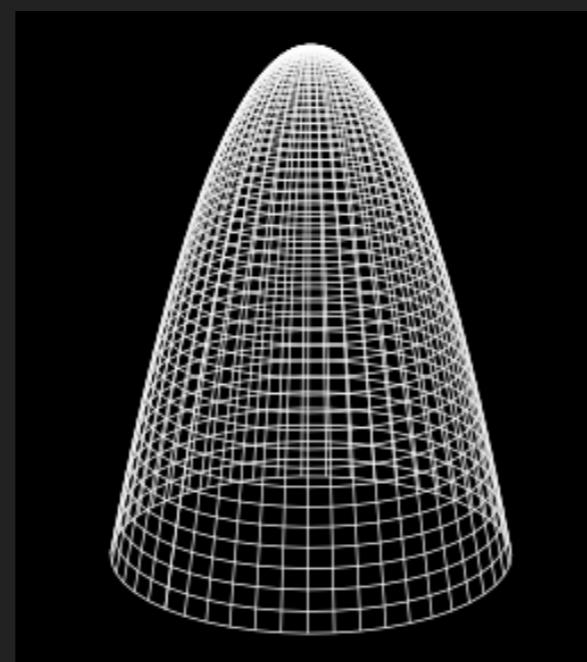
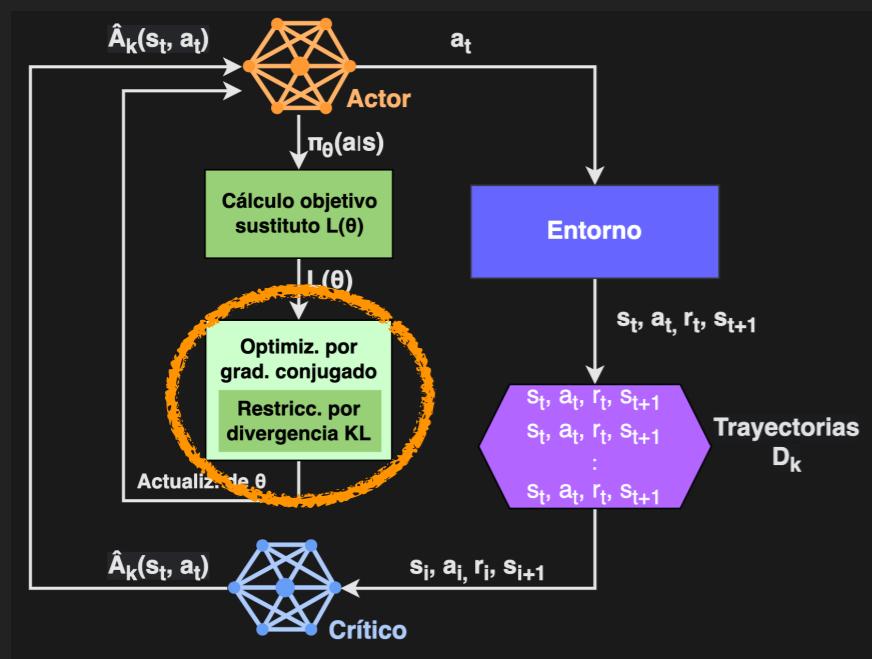


# OPTIMIZACIÓN DE L MEDIANTE REGIÓN DE CONFIANZA

- ▶ Pero la variación de la política  $\pi$  no debería excederse de cierto límite.
- ▶ Para limitarlo TRPO usa la divergencia KL.

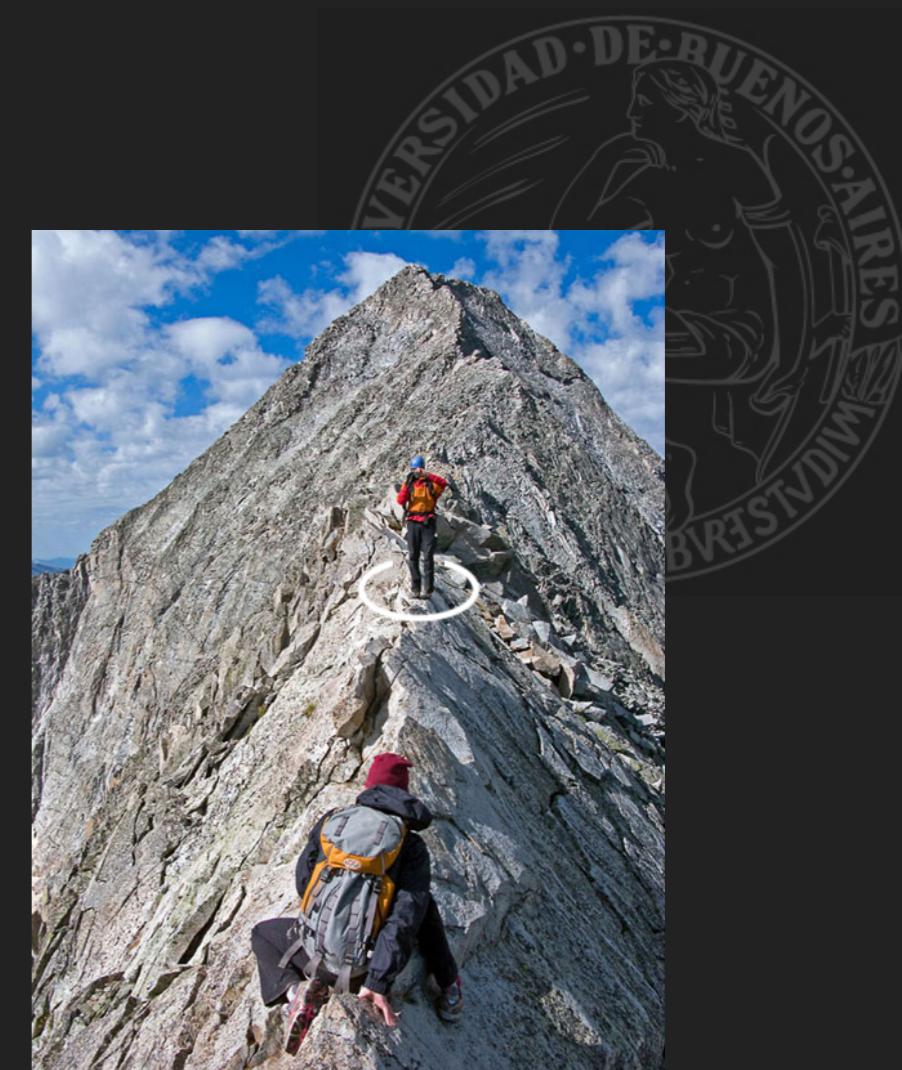
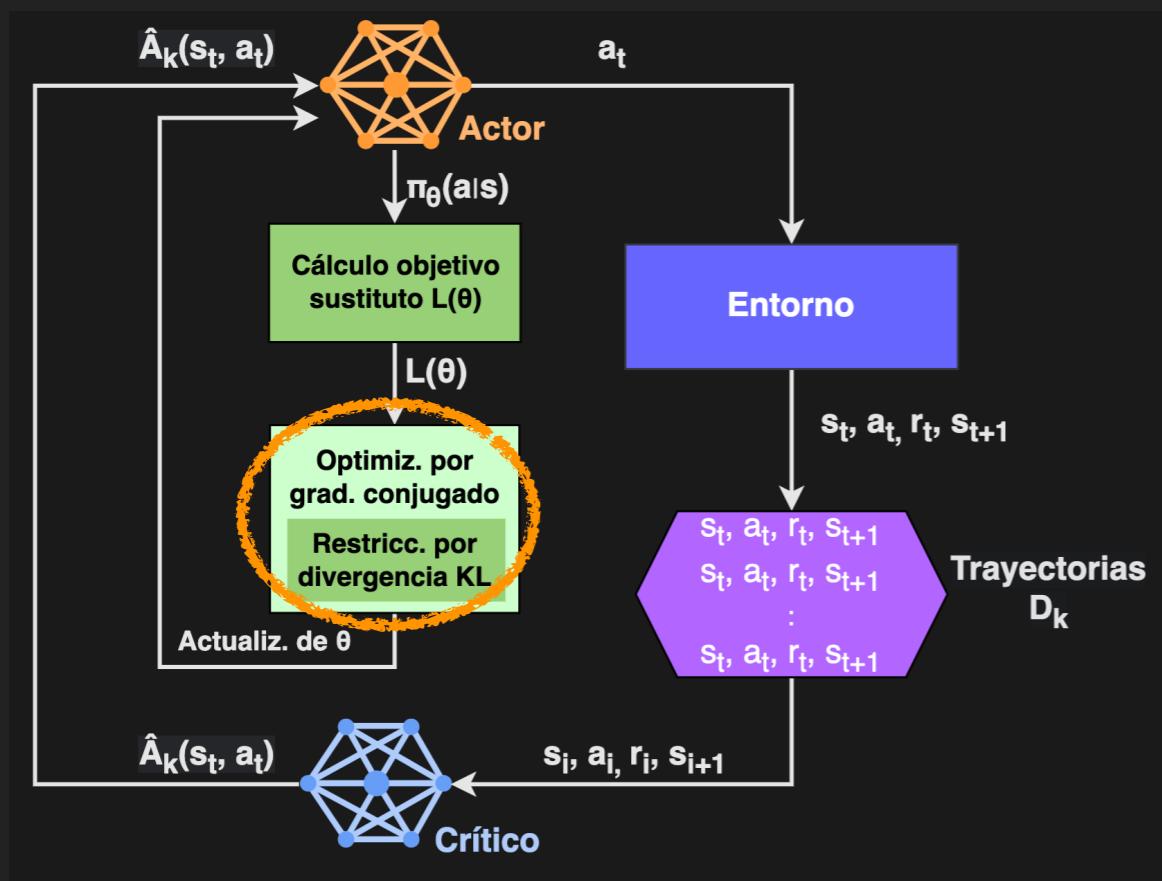
$$D_{\text{KL}}(\pi_{\theta+\Delta\theta} \parallel \pi_\theta) \approx \frac{1}{2} \Delta\theta^\top F \Delta\theta$$

- ▶ donde  $F$  es la matriz de Fisher, que es una aproximación al Hessiano de la KL.



# OPTIMIZACIÓN DE L MEDIANTE REGIÓN DE CONFIANZA

- Para optimizar se requiere un algoritmo de optimización de segundo orden (como **conjugate gradient**), en lugar de usar Adam, SGD o RMSProp.



# ¿CÓMO SE OBTIENE $\Delta\theta$ USANDO GRADIENTE CONJUGADO?

- ▶ Se busca encontrar una dirección de paso  $\Delta\theta$ , que sea una solución de:

$$\max_{\Delta\theta} \quad g^T \Delta\theta \quad \text{sujeto a} \quad \frac{1}{2} \Delta\theta^T H \Delta\theta \leq \delta \quad H \Delta\theta = g$$

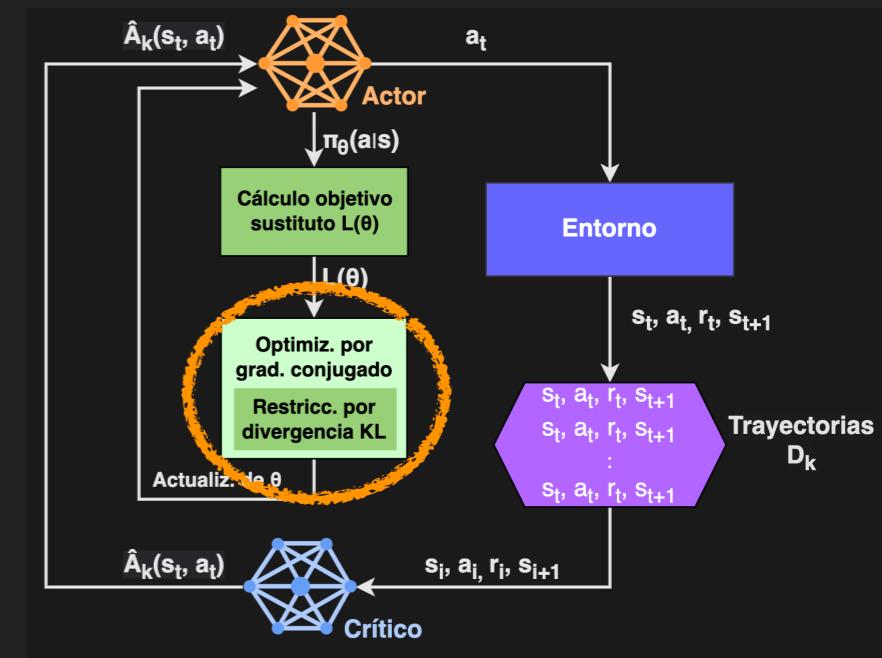
- ▶ En lugar de invertir  $H$  (carísimo computacionalmente), se aplica el **método de gradiente conjugado (CG)** para aproximar la solución a ese sistema lineal. Es decir, el CG devuelve una aproximación de:

$$\Delta\theta \approx H^{-1}g$$

- ▶ Una vez obtenida esa dirección, hay que escalar el tamaño del paso para cumplir la restricción de  $KL \leq \delta$ :

$$\text{step} = \sqrt{\frac{2\delta}{\Delta\theta^T H \Delta\theta}}$$

$$\theta_{\text{new}} = \theta_{\text{old}} + \text{step} \cdot \Delta\theta$$



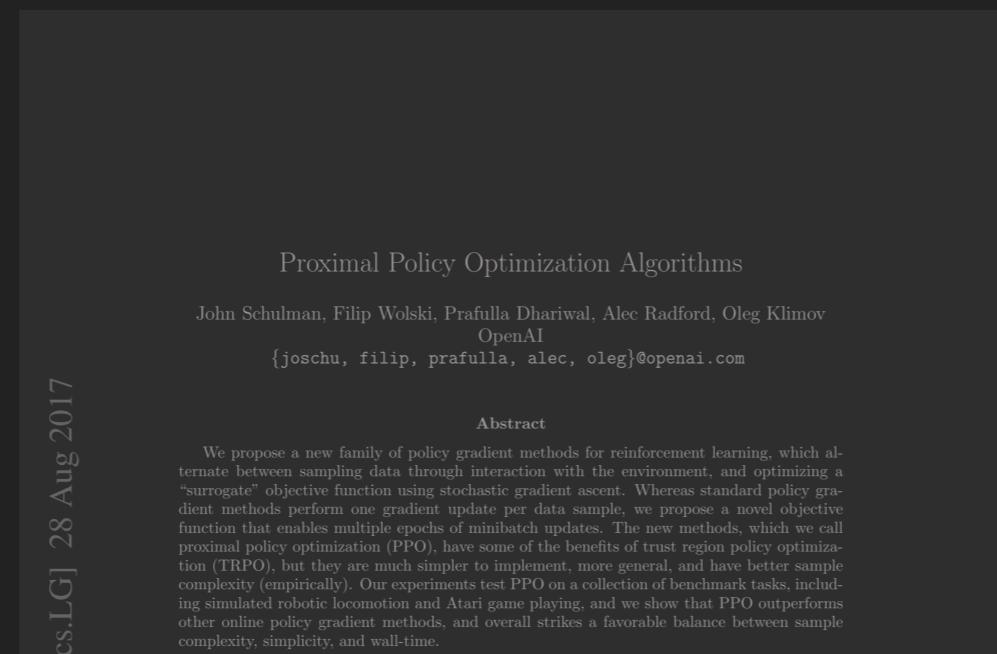
## LIMITACIONES DE TRPO

- ✓ Es difícil de implementar.
- ✓ Es computacionalmente caro.
- ✓ Requiere resolver problemas de optimización más complejos (conjugate gradient, producto del Hessian-vector)

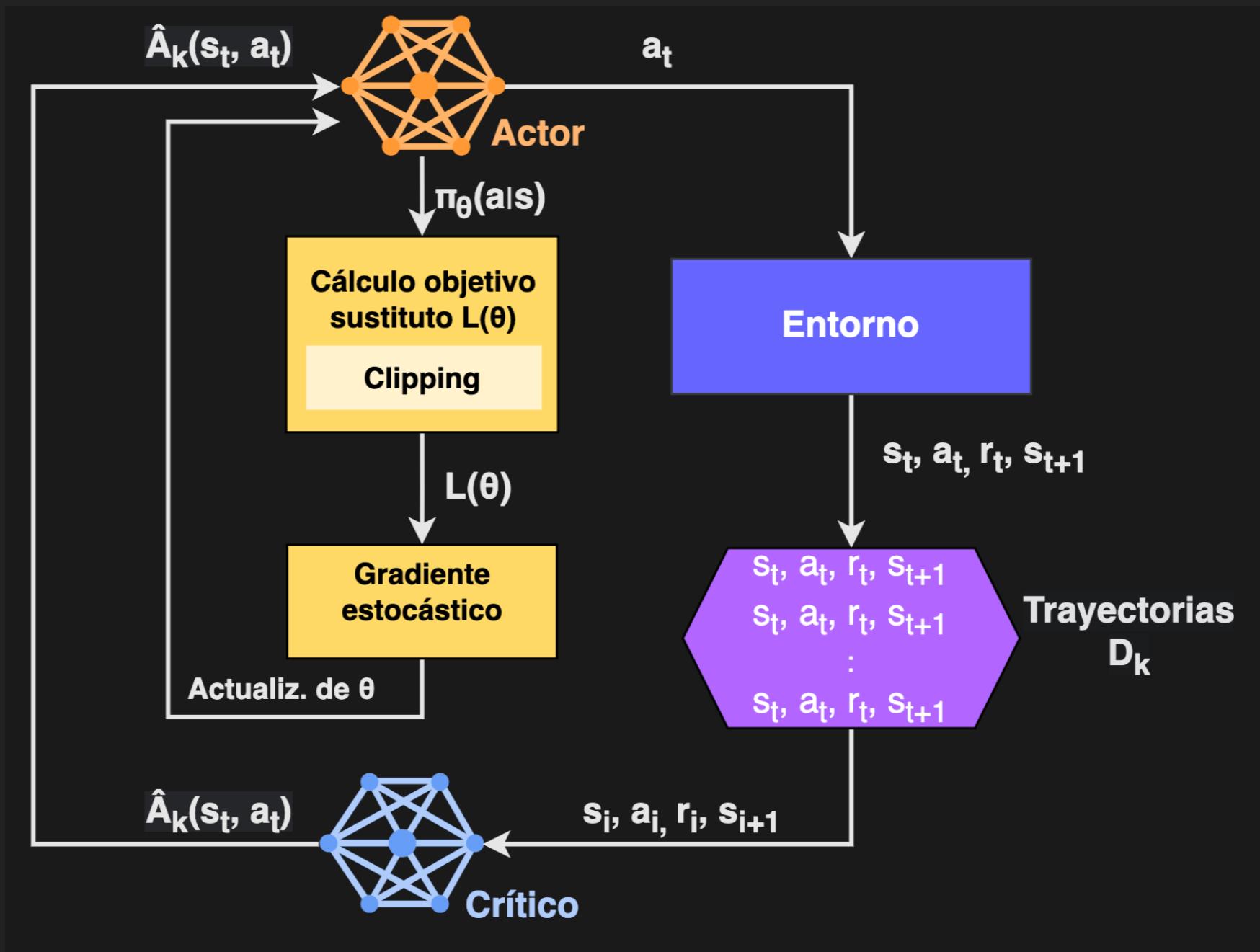


# ALGORITMO PPO

- ▶ PPO surge en 2017 con el título de "Algoritmos de optimización de políticas próximas" (Schulman et al., 2017).
- ▶ Es una mejora de su predecesor TRPO al evaluar variantes de PPO:
  - ✓ PPO-Penalty (o PPO con penalización KL)
  - ✓ PPO-Clip (o PPO con objetivo recortado)
  - ✓ Otras variantes
- ▶ Los autores encontraron que la variante PPO-Clip (con el objetivo recortado) generalmente funciona mejor o igual de bien que la variante con penalización KL adaptativa, y es más simple de implementar.



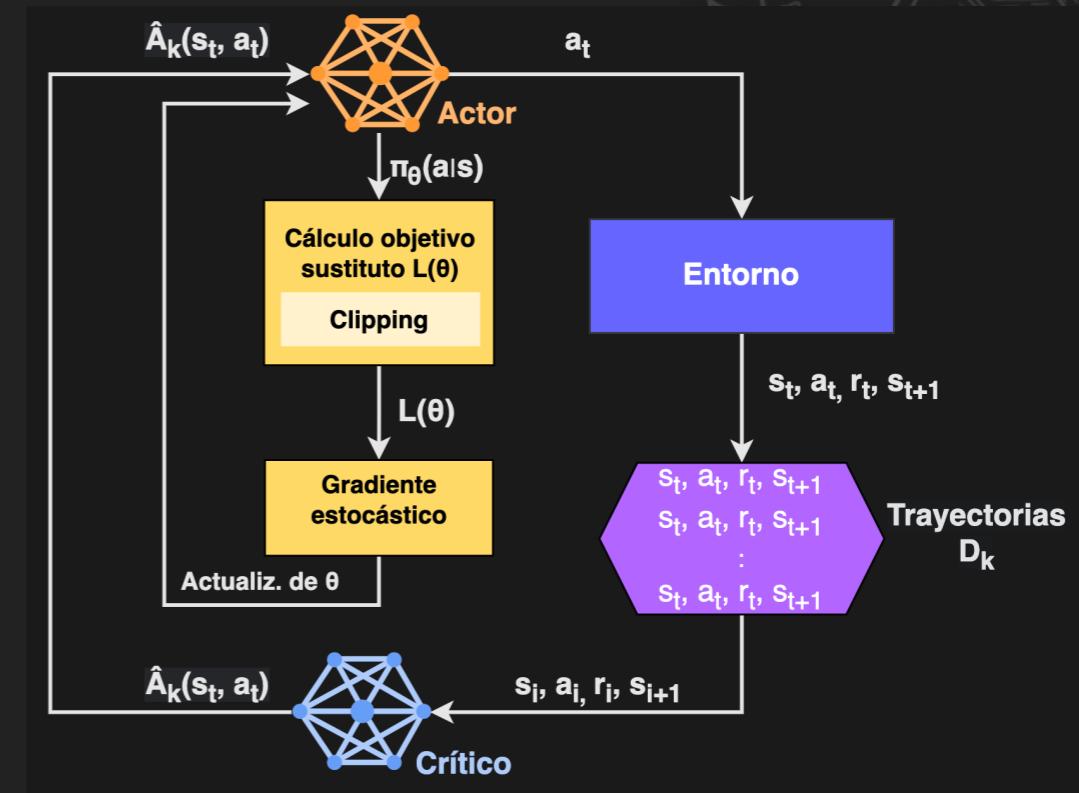
# ARQUITECTURA DE PPO



## COMO FUNCIONA EL RECORTE? (CLIPPING)

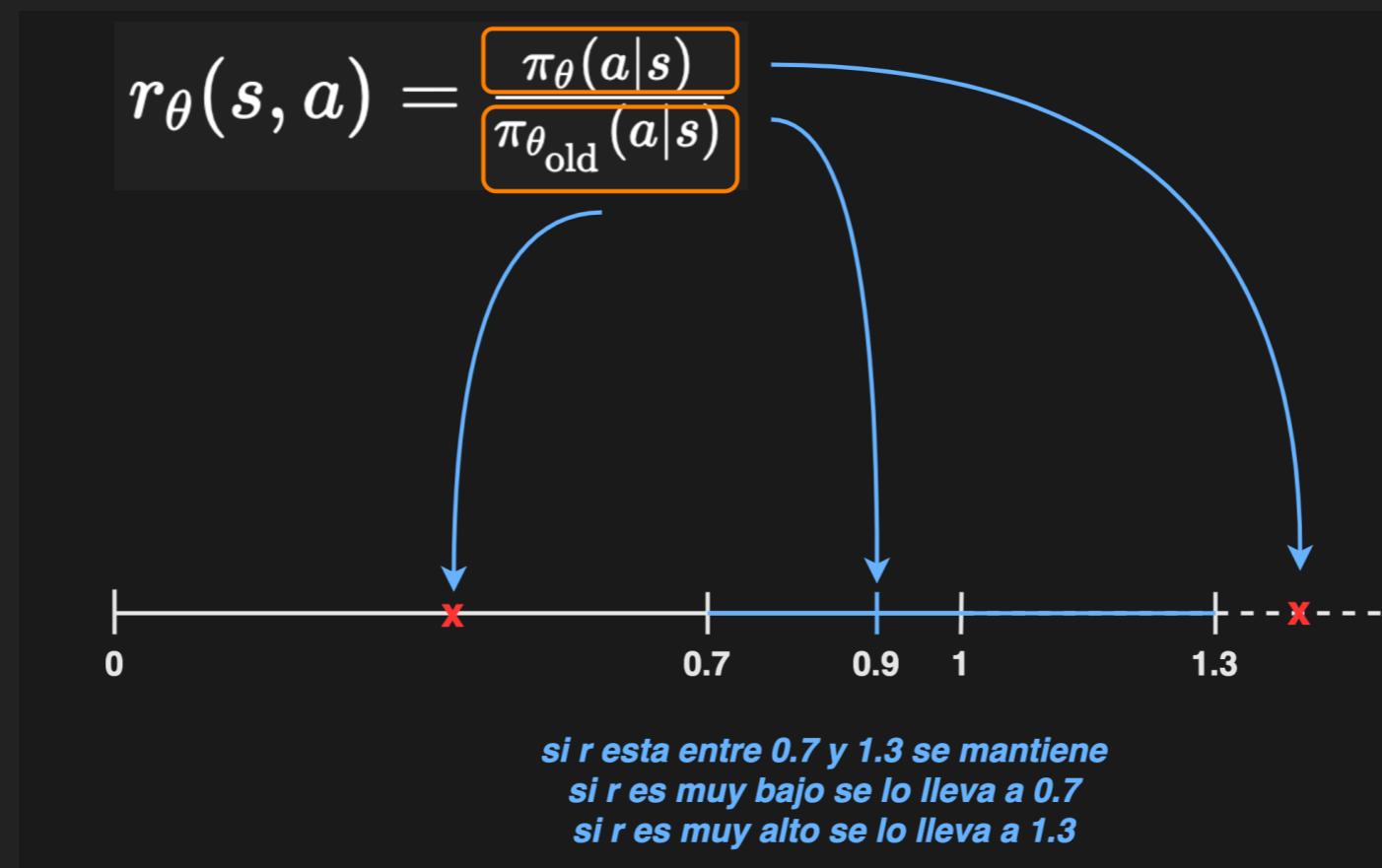
- ▶ Supongamos que dada una política la probabilidad de una determinada acción actual es 0.4 pero en la iteración anterior era 0.0001.
- ▶ El resultado 4000 es alto pero es mas alto aun si se multiplica por la ventaja.
- ▶ Una red neuronal no puede procesar adecuadamente ese resultado muy alto.

$$r_\theta(s, a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} = \frac{0.4}{0.0001} = 4000$$



## COMO FUNCIONA EL RECORTE? (CLIPPING)

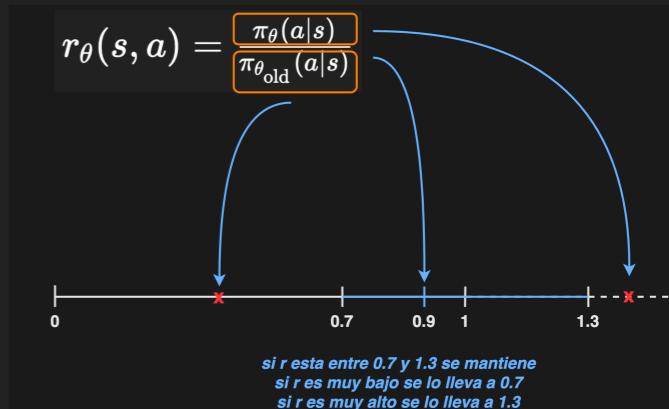
- ▶ Es deseable que  $L$  no sea muy grande ni muy pequeño.
- ▶ El numerador y el denominador están en  $[0, 1]$ , pero la relación entre ambos debería estar cercana a 1 porque se supone que ambas no varían demasiado.
- ▶ Un radio de 0.3 alrededor de 1 es un valor lógico.



## COMO FUNCIONA EL RECORTE? (CLIPPING)

- ▶ El objetivo sustituto  $L$  en TRPO era:

$$L^{\text{TRPO}}(\theta) = \mathbb{E}_{(s,a) \sim \pi_{\theta_{\text{old}}}} \left[ \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}(s,a) \right]$$



- ▶ En PPO es:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) \hat{A}_t, \underline{\text{clip}}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right]$$

- ▶ Donde  $\text{clip}()$  es el recorte y  $\text{Epsilon}$  es el 0.3 del ejemplo.

- ✓ si  $\text{clip}$  esta entre  $1-\epsilon$  y  $1+\epsilon$  se mantiene
- ✓ si  $\text{clip}$  es menor que  $1-\epsilon$  se mantiene  $1-\epsilon$
- ✓ si  $\text{clip}$  es mayor que  $1+\epsilon$  se mantiene  $1+\epsilon$



## PÉRDIDA TOTAL EN PPO

- ▶ La función de pérdida total es:

$$L_{total} = L_{policy}^{CLIP} + c_1 \cdot L_{value} - c_2 \cdot H(\pi)$$

- ▶ Donde:
  - ✓ LCLIP: pérdida de política (con clipping)
  - ✓ Lvalue=(V(s)–R<sub>t</sub>)<sup>2</sup>: pérdida del crítico
  - ✓ H(π): entropía de la política (para exploración)
  - ✓ C1 es el peso de la perdida del crítico
  - ✓ C2 es el peso de la entropía del actor



# COMPARATIVA TRPO VS PPO

## ▶ TRPO

- ✓ Usa un **surrogate objective** + **constrained optimization** ( $KL \leq \delta$ )
- ✓ Requiere algoritmo de optimización de segundo orden (como **conjugate gradient**)
- ✓ Muy estable, pero más **pesado**
- ✓ Se usa más en **papers teóricos** o cuando se prioriza garantías fuertes

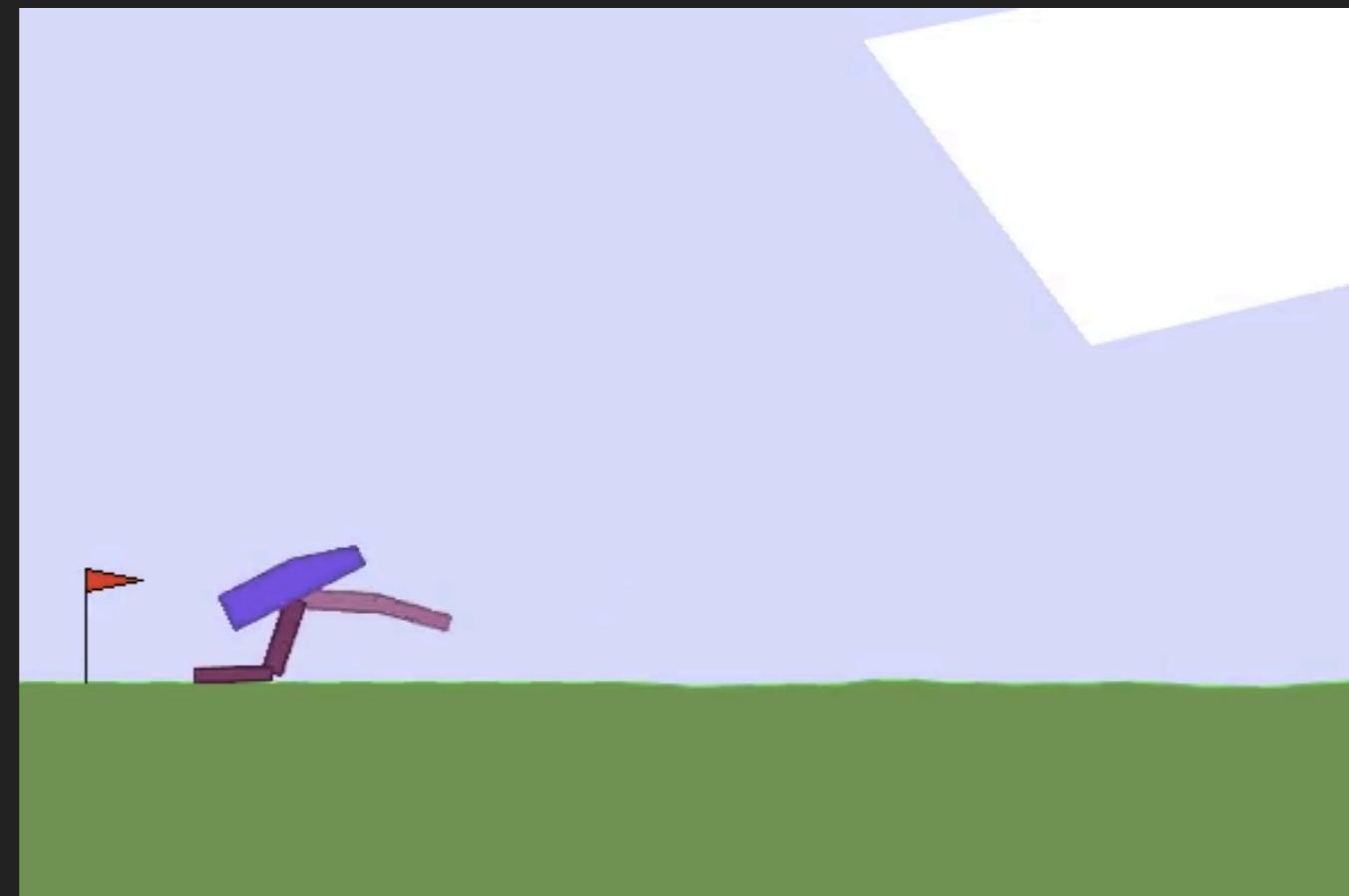
## ▶ PPO

- ✓ Usa un **surrogate objective** con **clip** en el ratio de políticas
- ✓ Usa **SGD** o **Adam** directamente
- ✓ También estable, pero **más rápido**
- ✓ Se usa ampliamente en la **práctica**



## EJEMPLO

- ▶ Aprendizaje del robot con BipedalWalker-v3 de Gymnasium y con la biblioteca de agentes Stable Baseline 3.
- ▶ <https://github.com/aear-uba/ar2/tree/main/src/ppo>



# PARÁMETROS EN SB3

```
model = PPO(  
    policy='MlpPolicy',  
    env=train_env,  
    learning_rate=3e-4,  
    n_steps=2048, # Número de pasos a correr por cada actor por actualización  
    batch_size=64, # Tamaño del minibatch  
    n_epochs=10, # Núm. de épocas p/ optim. el obj. surrogate x c/ actualización  
    gamma=0.99,  
    gae_lambda=0.95, # Factor para GAE (Generalized Advantage Estimation)  
    clip_range=0.2, # Parámetro de clipping para PPO  
    ent_coef=0.0, # Coeficiente de entropía (puede ayudar a la exploración)  
    vf_coef=0.5, # Coeficiente de la función de valor en la pérdida  
    max_grad_norm=0.5, # Clipping del gradiente  
    verbose=1,  
    seed=42,  
)
```



## APLICACIÓN DE PPO

- ▶ Robótica
- ▶ Ajuste fino de LLMs
- ▶ Sistemas de recomendación adaptativos
- ▶ Drones, autos autónomos
- ▶ Diseño de circuitos y layouts



# FASES DEL PIPELINE RLHF

- ▶ Preentrenamiento (Pretraining)
  - ✓ Texto de internet
- ▶ Entrenamiento supervisado (Supervised Fine-Tuning – SFT)
  - ✓ Anotadores humanos escriben respuestas correctas a prompts reales.
- ▶ Entrenamiento del modelo de recompensa (Reward Model – RM)
  - ✓ Evaluadores humanos comparan múltiples respuestas a un mismo prompt y eligen la mejor.
- ▶ Entrenamiento con PPO
  - ✓ Objetivo: Afinar el modelo para que produzca respuestas que maximicen la recompensa del reward model.
  - ✓ Toma un prompt.
  - ✓ El modelo genera una respuesta.
  - ✓ El reward model evalúa esa respuesta y da una recompensa.
  - ✓ PPO ajusta los pesos del modelo principal para maximizar esa recompensa, cuidando que los cambios no sean muy bruscos



## REFERENCIAS BIBLIOGRÁFICAS Y WEB (I)

- ▶ Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In International conference on machine learning (pp. 1889-1897). PMLR.
- ▶ Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438.
- ▶ Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

