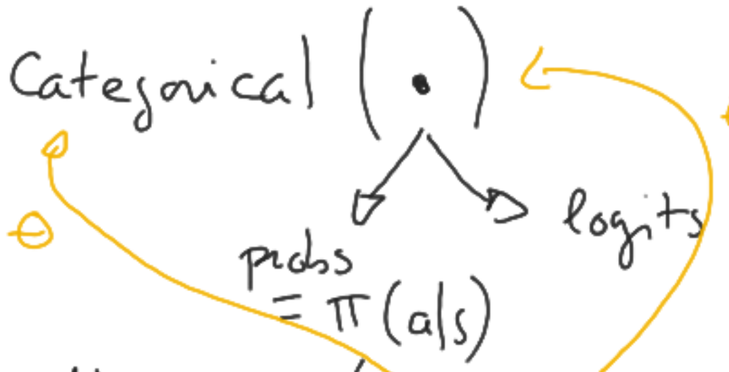
TME6 (RLD)

→ Finir TME 5 (Reinforce - acra - critic) !

→ PPO < KL / Clip]

$$\alpha_{\theta_k}(\theta) = \mathbb{E}_{old}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t\right)$$

(KL) torch.distributions.Categorical(·)



dθ

torch.distributions.kl_divergence(dθ, dθ_old)

(PPO) tensor.clamp(minimum, maximum) → $\begin{cases} x \text{ si } x \in [min, max] \\ min \text{ si } x < min \\ max \text{ si } x > max \end{cases}$

① Commencer par utiliser une baseline pour estimer l'avantage
(→ marche pour CartPole) → cf "réduction de la variance : baseline"

$$b(s_t) = \frac{1}{N} \sum_{\tau} R_t(\tau)$$

② 

⚠ done = True lorsque l'épisode se termine ou est arrêté arbitrairement (≥ 500 pas de long si pas de baseline)

ob, reward, done, info = env.step()

info.get("TimeLimit.truncated", False)

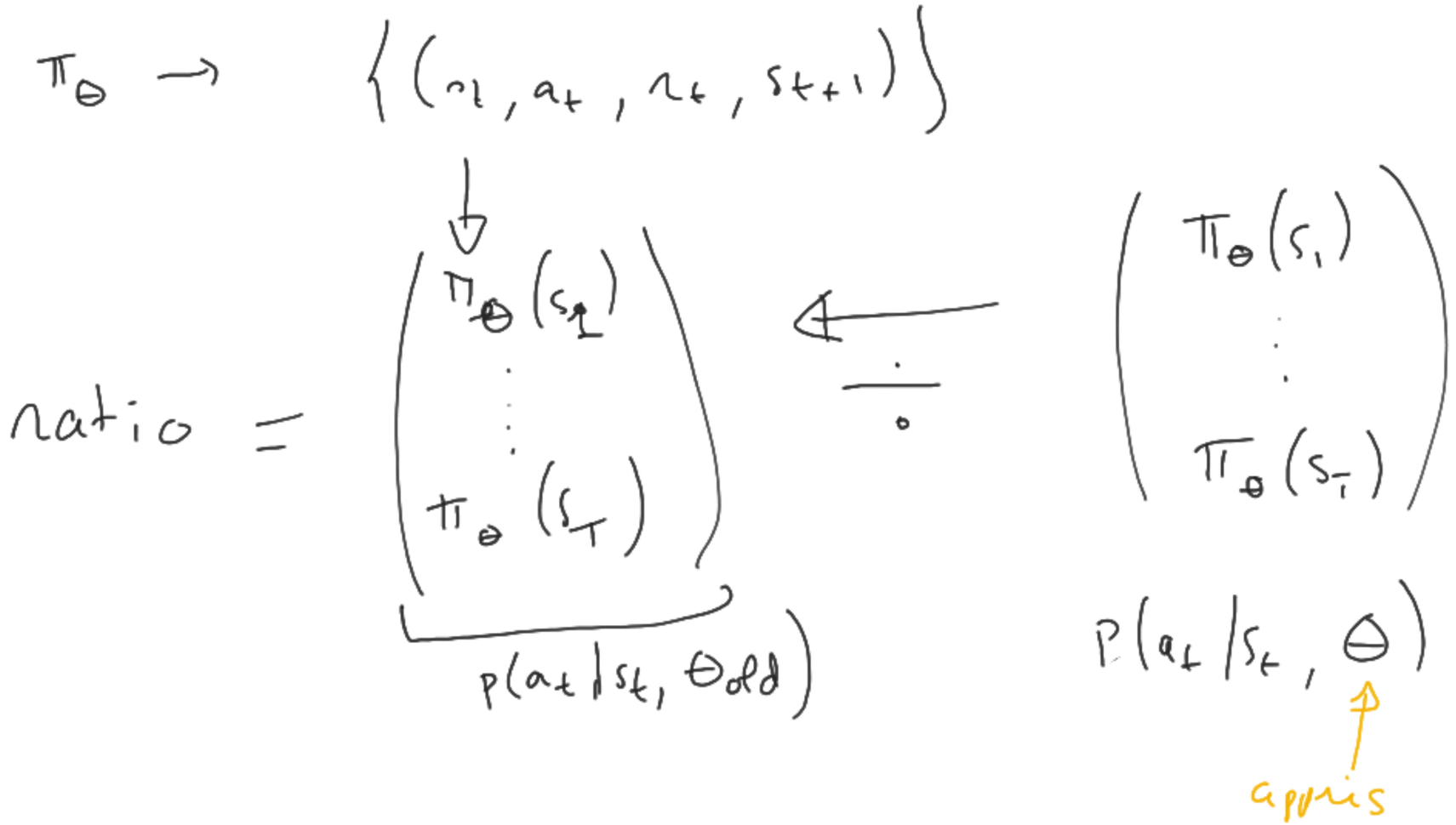
truncated =
- True → si arrêt
- False → sinon

pas correct { r_t si done || pas bon si arrêt
$r_t + \gamma V^{\pi}_{target}(s_{t+1})$ sinon

correct { r_t si done ∧ not(truncated)
$r_t + \gamma V^{\pi}_{target}(s_{t+1})$ sinon

!!!

cible pour apprentissage V^{π}


$\pi_\theta \rightarrow \{(s_1, a_t, r_t, s_{t+1})\}$
ratio =
$\pi_\theta(s_1)$
$\pi_\theta(s_T)$
$p(a_t|s_t, \theta_{old})$
$\pi_\theta(s_1)$
$\pi_\theta(s_T)$
$p(a_t|s_t, \theta)$
appris

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ

for $k = 0, 1, 2, \ldots$ **do**

Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$$

by taking K steps of minibatch SGD (via Adam)

if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**

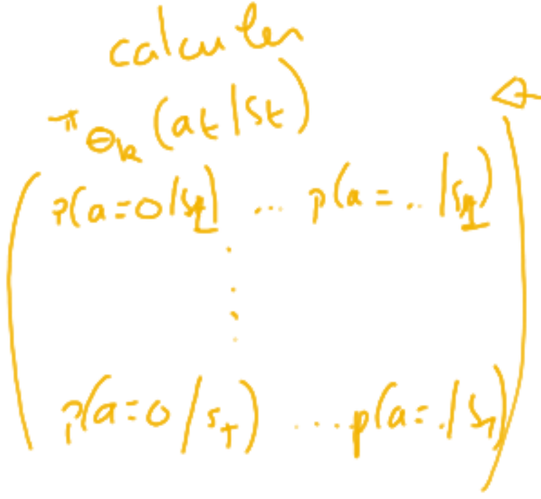
$\beta_{k+1} = 2\beta_k$

else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**

$\beta_{k+1} = \beta_k/2$

end if

end for



à utiliser pour la KL

actions = [...]

prob_a_s = probas[range(len(actions)), actions]

0	–	a_0
1	–	a_1
2	–	a_2
⋮	⋮	
T-1	.	a_{T-1}

] au utiliser gather