

- **Derive a sample-based estimate for the gradient:** Hiểu cách chuyển biểu thức gradient lý thuyết của average reward hoặc discounted return thành cập nhật thực thi dựa trên trải nghiệm (sample-based). 📄
- **Understand the Actor-Critic algorithm:** Hiểu cấu trúc và cơ chế hoạt động của Actor-Critic trong bối cảnh continuing tasks, bao gồm cách ước lượng critic và cập nhật actor. 📄

Việc hiểu rõ hai mục tiêu này giúp triển khai chính xác Actor-Critic cho các tác vụ không có điểm kết thúc rõ ràng, hoặc khi muốn tối ưu reward rate dài hạn.

2. Estimating the Policy Gradient trong Continuing Tasks

2.1. Mục tiêu gradient trong continuing tasks

- Với continuing tasks, chúng ta thường quan tâm hai loại objective:
 1. **Discounted return:** $J(\theta) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}]$ với $0 \leq \gamma < 1$.
 2. **Average reward:** $\rho(\pi) = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}[\sum_{t=1}^T R_t] = \sum_s \mu_{\pi}(s) \sum_a \pi(a|s) r(s, a)$, trong đó $\mu_{\pi}(s)$ là phân phối trạng thái ổn định dưới policy π .
- Policy Gradient Theorem cho cả hai setting cho phép viết gradient $\nabla \theta J$ (hoặc $\nabla \theta \rho$) dưới dạng expectation "on-policy":
 - **Discounted:**

$$\nabla_{\theta} J(\theta) = \sum_s d_{\pi}(s) \sum_a \nabla_{\theta} \pi_{\theta}(a|s) Q_{\pi}(s, a),$$

với $d_{\pi}(s)$ là discounted state distribution.

- **Average reward:**

$$\nabla_{\theta} \rho(\pi_{\theta}) = \sum_s \mu_{\pi}(s) \sum_a \nabla_{\theta} \pi_{\theta}(a|s) q_{\pi}^{\text{diff}}(s, a),$$

trong đó $q_{\pi}^{\text{diff}}(s, a)$ là differential action-value: $\mathbb{E}[\sum_{k=0}^{\infty} (R_{t+k+1} - \rho(\pi)) \mid S_t = s, A_t = a]$.
[arxiv.org](https://arxiv.org/abs/1604.05718)

- **Ý nghĩa:** Mặc dù $\mu_{\pi}(s)$ phụ thuộc θ phức tạp, Policy Gradient Theorem cho thấy ta không cần tính trực tiếp $\partial \mu_{\pi}(s) / \partial \theta$; thay vào đó, gradient có thể ước lượng qua trải nghiệm thực thi policy hiện tại. [doutoury2.tistory.com](https://doutoury2.tistory.com/11) [arxiv.org](https://arxiv.org/abs/1604.05718)

2.2. Sample-based estimate

- Theo stochastic gradient descent/ascent, để cập nhật θ , ta cần một sample estimate cho $\nabla \theta J$ hoặc $\nabla \theta \rho$:

$$\widehat{g}_t \leftarrow (\text{estimate of } Q \text{ or } q^{\text{diff}}) \times \nabla_{\theta} \ln \pi_{\theta}(A_t | S_t).$$

- **Discounted episodic:** ước lượng $Q(s,a)$ bằng return G_t ; nhưng continuing thì không có episode rõ rệt, có thể dùng truncated discounted return hoặc TD return: $R_{t+1} + \gamma V(S_{t+1})$ để ước lượng Q .
- **Average reward:** ước lượng differential action-value $q^{\text{diff}}(S_t, A_t)$ bằng differential TD error (critic estimate), cụ thể $\delta_t = R_{t+1} - \hat{p} + V_w(S_{t+1}) - V_w(S_t)$. Khi đó δ_t là ước lượng một bước cho $q^{\text{diff}} - V$. doutoury2.tistory.com mohammadghavamzadeh.github.io
- **Cập nhật actor:**

$$\theta \leftarrow \theta + \alpha \delta_t \nabla_{\theta} \ln \pi_{\theta}(A_t | S_t),$$

với δ_t phù hợp discounted TD error (nếu tối ưu discounted return) hoặc differential TD error (nếu tối ưu average reward). arxiv.org

2.3. Các phương pháp ước lượng gradient đề cập trong slide

Slide liệt kê một số cách estimate gradient:

1. **Finite Difference Methods:** perturb trực tiếp θ và đo thay đổi expected return \rightarrow đắt chi phí, không phổ biến trong RL lớn.
2. **Score Function Methods (Likelihood Ratio Methods):** sử dụng $\nabla_{\theta} \ln \pi_{\theta}$ để tính gradient từ samples. Đây là căn bản cho REINFORCE và Actor-Critic:

$$\nabla_{\theta} J \approx \mathbb{E}[(\text{return or advantage}) \nabla_{\theta} \ln \pi_{\theta}(A|S)].$$

Subtract baseline (value) để giảm variance nhưng không gây bias. ai.stackexchange.com

3. **Actor-Critic Methods:** kết hợp policy gradient (score function) với value-function estimation (critic) dùng TD để ước lượng advantage, giảm variance và cập nhật ngay mỗi bước.
4. **Natural Policy Gradient:** Cân nhắc hình học θ -space, dùng Fisher information matrix để precondition gradient, cải thiện convergence và stability. Thường áp dụng trong continuing tasks phức tạp để tránh update quá mạnh/kém. mohammadghavamzadeh.github.io

Những phương pháp này đều dùng sampling on-policy và tùy mục tiêu (discounted hay average reward) điều chỉnh critic và δ_t phù hợp.


3. Actor-Critic Algorithm cho Continuing Tasks

Actor-Critic kết hợp hai thành phần:

- **Actor:** policy parameterized $\pi_{\theta}(a|s)$, chịu trách nhiệm chọn action.
- **Critic:** xấp xỉ value function (V hoặc differential V), cung cấp ước lượng TD error δ để cập nhật actor.

3.1. Initialization

- **Actor network/parameters θ** khởi random (có thể zero cho softmax preferences).

- **Critic parameters w :** khởi random hoặc zero. Nếu average reward: khởi ước lượng average reward \hat{p} (scalar) = 0 hoặc small. 

3.2. Loop tương tác (online, continuing)

Trong mỗi bước hoặc mỗi nhỏ batch:

1. **Sample action:** tại state S_t , sample $A_t \sim \pi_\theta(\cdot|S_t)$.
2. **Quan sát:** nhận R_{t+1} , next state S_{t+1} .
3. **Critic update:**
 - **Discounted setting:**
 - Tính TD error: $\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$.
 - Cập nhật critic: $w \leftarrow w + \alpha_c \delta_t \nabla_w V_w(S_t)$.
 - **Average reward setting:**
 - Tính differential TD error: $\delta_t = R_{t+1} - \hat{p} + V_w(S_{t+1}) - V_w(S_t)$.
 - Cập nhật critic weights: $w \leftarrow w + \alpha_c \delta_t \nabla_w V_w(S_t)$.
 - Cập nhật average reward estimate: $\hat{p} \leftarrow \hat{p} + \beta \delta_t$.
Trong cả hai, critic dùng semi-gradient TD(0) để học V (discounted) hoặc differential value (average). arxiv.org mohammadghavamzadeh.github.io
4. **Actor update:**
 - Dùng cùng δ_t :
$$\theta \leftarrow \theta + \alpha_a \delta_t \nabla_\theta \ln \pi_\theta(A_t|S_t).$$
 - Ở discounted case, δ_t = TD error discounted; average case dùng δ_t differential. Việc này dựa trên policy gradient theorem, thay Q bằng TD error * gradient log policy. arxiv.org
5. **Tiếp tục:** Lặp vô hạn hoặc đến khi policy ổn định (đánh giá performance periodic). Exploration được duy trì nhờ policy stochastic (softmax hoặc Gaussian) và có thể thêm entropy bonus nếu cần.

3.3. Implementation details

- **Policy parameterization:**
 - **Discrete actions:** softmax preferences $h(s, a; \theta) \rightarrow \pi_\theta(a|s) = \text{softmax}(h)$. Gradient log-prob để tính: $\nabla_\theta \ln \pi_\theta(a|s) = \nabla_\theta h(s, a) - \sum_b \pi_\theta(b|s) \nabla_\theta h(s, b)$. doutoury2.tistory.com
 - **Continuous actions:** Gaussian policy parameterized mean $\mu_\theta(s)$ và có thể học log-variance; gradient log-prob dùng score-function hoặc reparameterization trick.
- **Critic architecture:**
 - Với discounted: network/linear approximator cho $V(s)$.
 - Với average: network/linear approximator cho differential value $h(s)$; scalar \hat{p} cập nhật riêng.
- **Step-sizes:** α_c (critic) thường lớn hơn hoặc cùng order với α_a (actor) để critic theo kịp actor. Với average reward cần thêm β cho \hat{p} ; β thường nhỏ hơn α_c để \hat{p} cập nhật chậm trên

timescale phiem.

- **Exploration & stability:**

- Đảm bảo policy không quá deterministic quá sớm → có thể thêm entropy bonus:

$$\theta \leftarrow \theta + \alpha_a [\delta_t \nabla \theta \ln \pi \theta(A_t | S_t) + \lambda \nabla \theta H(\pi \theta(\cdot | S_t))].$$

- Với continuing tasks, cần policy ergodic (thăm đủ states) để ước lượng μ_π và critic hội tụ.
- Với discounted, weighting d_π khác steady-state, nhưng Actor-Critic online vẫn hiệu quả nếu discount < 1 ; truncated trajectories hoặc bootstrapping liên tục.
- **Batch vs online:** Có thể accumulate gradients qua nhiều bước trước khi cập nhật; hoặc cập nhật ngay mỗi bước (on-policy). Trong continuing tasks, thường dùng online update để phản hồi nhanh thay đổi policy. [geeksforgeeks.org](https://www.geeksforgeeks.org/) neuralpai.medium.com

3.4. Actor-Critic Example trong Grid World continuing

- Giả sử grid world không có terminal: agent đi mãi, mỗi bước nhận reward có thể dương/âm.
- **Policy (actor):** softmax trên feature state-action: $h(s,a;\theta) = \theta^T \phi(s,a)$.
- **Value (critic):** ước lượng differential $V(s)$ và ρ .
- **Cập nhật:**
 - Tại mỗi bước: sample A_t theo softmax, nhận R_{t+1} , S_{t+1} .
 - Tính $\delta_t = R_{t+1} - \rho + V_w(S_{t+1}) - V_w(S_t)$.
 - Cập nhật w và ρ , rồi θ như trên.
- **Theo dõi performance:** đo running average reward per step theo window dài, để đánh giá policy cải thiện.
- **Chú ý:** nếu policy quá ưu tiên một hướng, không đủ exploration, critic ước lượng sai, có thể policy converged đến local suboptimal. Cần giữ entropy cao ban đầu và giảm từ từ.

doutoury2.tistory.com

3.5. Actor-Critic for discounted continuing

- Tương tự nhưng $\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$. Không cần ρ .
- Ví dụ robot liên tục điều khiển với discount γ gần 1 để ưu tiên reward dài hạn. Actor-Critic discounted phổ biến rộng (A2C, PPO, v.v.).

4. Các phương pháp Estimate Gradient để cập

4.1. Finite Difference Methods

- **Nguyên lý:** perturbe $\theta \rightarrow \theta + \epsilon e_i \rightarrow$ chạy policy dài hạn đo $J(\theta + \epsilon e_i) - J(\theta)$ để ước lượng $\partial J / \partial \theta_i \approx [J(\theta + \epsilon) - J(\theta)] / \epsilon$.

- **Nhược:** tốn kém do phải chạy nhiều lần cho mỗi chiều tham số hoặc batch, không khả thi khi θ dimension lớn. Thường dùng làm tham khảo nhỏ hoặc debug, ít dùng thực tế. [geeksforgeeks.org](https://www.geeksforgeeks.org/)

4.2. Score Function / Likelihood Ratio Methods

- **Cơ sở:** $\nabla_{\theta} E[f(X)] = E[f(X) \nabla_{\theta} \ln p_{\theta}(X)]$. Với RL: f là return/advantage, $p_{\theta}(X)$ là xác suất trajectory do π_{θ} tạo ra.
- **Ưu:** Không cần biết model dynamics; dùng trực tiếp samples trên policy hiện tại.
- **Baseline:** subtract baseline $b(s)$ (thường là $V(s)$) để ước lượng advantage, giảm variance: $\widehat{A}_t = \mathbb{E}(S_t, A_t) - b(S_t)$.
- **Actor-Critic:** critic ước lượng $b(S)=V(s)$.
- **Công thức cập nhật:** $\theta \leftarrow \theta + \alpha \widehat{A}_t \nabla_{\theta} \ln \pi_{\theta}(A_t|S_t)$ ai.stackexchange.com

4.3. Actor-Critic Methods

- **Kết hợp:** policy gradient + TD critic. Critic estimate V hoặc Q , actor dùng δ_t (TD error) làm estimate advantage.
- **Ưu:** cập nhật mỗi bước, variance thấp hơn REINFORCE, sample efficient hơn.
- **Continuing:** dùng discounted TD(0) hoặc differential TD cho average reward.
- **Cần đảm bảo:** critic theo kịp actor; step-size phù hợp.
- **Ví dụ phổ biến:** A2C/A3C (discounted), Average Reward Actor-Critic (continuing tasks lâu dài). neuralpai.medium.com

4.4. Natural Policy Gradient

- **Ý tưởng:** Gradient thông thường (Euclidean) có thể không phản ánh khoảng cách thật trong space của distribution π_{θ} . Natural gradient dùng Fisher Information Matrix F để precondition: update direction = $F^{-1} \nabla_{\theta} J$.
- **Ưu:** cải thiện convergence stability, tránh update quá mạnh sai hướng.
- **Áp dụng:** có thể dùng trong continuing tasks, cả discounted và average reward. Tuy nhiên tốn tính toán do F^{-1} . Có thể xấp xỉ (Kronecker-Factored, etc.). mohammadghavamzadeh.github.io

5. Derive Sample-based Estimate cho Gradient of Average Reward

Slide đề cập: "The gradient of the average reward: We simply make updates from states we observe while following policy π . This gradient from state S_t provides an approximation to the gradient of the average reward. The stochastic gradient descent update looks like for the policy parameters."

- **Gradient lý thuyết:**

$$\nabla_{\theta} \rho(\pi_{\theta}) = \sum_s \mu_{\pi}(s) \sum_a \nabla_{\theta} \pi_{\theta}(a|s) q_{\pi}^{\text{diff}}(s, a).$$

- Khi sample state S_t under on-policy long-run, $S_t \sim \mu_\pi$ (giả định policy ergodic), ta có ước lượng:

$$E[\nabla_{\theta} \ln \pi_{\theta}(A_t | S_t) q_{\pi}^{\text{diff}}(S_t, A_t)] = \sum_s \mu_{\pi}(s) \sum_a \pi_{\theta}(a | s) \nabla_{\theta} \ln \pi_{\theta}(a | s) q_{\pi}^{\text{diff}}(s, a) = \nabla_{\theta} \rho.$$

- **Không có q^{diff}_{π} thực:** dùng critic estimate differential value để ước lượng q^{diff}_{π} .
- **One-step bootstrap:** differential TD error δ_t ước lượng $R_{t+1} - \rho + V(S_{t+1}) - V(S_t)$, gần bằng advantage.
- **Update actor:** $\theta \leftarrow \theta + \alpha \delta_t \nabla_{\theta} \ln \pi_{\theta}(A_t | S_t)$. Mỗi bước dùng sample $(S_t, A_t, R_{t+1}, S_{t+1})$. Khi chạy lâu, phối hợp với critic hội tụ, ước lượng gần đúng gradient average reward.

doutoury2.tistory.com

mohammadghavamzadeh.github.io

6. Actor-Critic Algorithm Steps (Continuing, Average Reward)

Tóm tắt cụ thể:

1. **Khởi:** θ , w (critic weights), ρ estimate.
2. **Lặp mỗi bước:**
 - Quan sát S_t .
 - Chọn $A_t \sim \pi_{\theta}(\cdot | S_t)$.
 - Thực hiện, nhận R_{t+1} , S_{t+1} .
 - Tính $\delta_t = R_{t+1} - \rho + V_w(S_{t+1}) - V_w(S_t)$.
 - **Cập nhật critic:** $w \leftarrow w + \alpha_c \delta_t \nabla_w V_w(S_t)$.
 - **Cập nhật average reward scalar:** $\rho \leftarrow \rho + \beta \delta_t$.
 - **Cập nhật actor:** $\theta \leftarrow \theta + \alpha_a \delta_t \nabla_{\theta} \ln \pi_{\theta}(A_t | S_t)$.
3. **Repeat** vô hạn hoặc đến khi policy ổn định.
4. **Theo dõi:** running average reward thực thi policy sau mỗi N bước/episode dài. Nếu average reward tăng, policy cải thiện.
5. **Hyperparameter:**
 - $\alpha_c > 0$, $\alpha_a > 0$, $\beta > 0$; thường β nhỏ hơn α_c , α_a , để ρ thay đổi chậm hơn critic; critic thay đổi nhanh hơn actor để cung cấp δ đúng hướng.
 - Exploration: policy stochastic, có thể thêm entropy bonus.
 - Kiểm tra ergodicity: đảm bảo policy ban đầu đủ khám phá, tránh stuck local.

doutoury2.tistory.com

7. Actor-Critic cho Discounted Continuing

- Tương tự như average nhưng:
 - Không có ρ .
 - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$.
 - Actor update: $\theta \leftarrow \theta + \alpha \delta_t \nabla_{\theta} \ln \pi(A_t|S_t)$.
 - Critic update: $w \leftarrow w + \alpha_c \delta_t \nabla_w V_w(S_t)$.
- Discounted Actor-Critic rất phổ biến (A2C, PPO, v.v.), stable với replay buffer hạn chế (on-policy) hoặc off-policy variants (DDPG, SAC). neuralpai.medium.com

8. Ví dụ Code và Total Reward của Learned Policy

Slide yêu cầu "Run that code and show a total reward obtained by learned policy". Dưới đây là minh hoạ pseudocode (Python-like) cho average reward Actor-Critic trong một môi trường continuing đơn giản (ví dụ Grid World infinite hoặc custom env):

python

 Sao chép  Chính sửa

```
# Pseudocode cho Average-Reward Actor-Critic # Giả sử ta có: # - function approx for
V(s): V_w(s), param w # - policy softmax: pi_theta(a|s), param theta # - environment
cung cấp reset() trả state ban đầu, step(a) trả (next_state, reward, done) # với
continuing, done=False luôn hoặc theo điều kiện custom, ta ignore done. theta =
initialize_actor_params() w = initialize_critic_params() rho = 0.0 # estimate average
reward alpha_c = 1e-3 alpha_a = 1e-4 beta = 1e-5 # step-size for rho, nhỏ hơn critic #
Feature/Network definitions omitted; assume functions: # get_policy_probs(s, theta),
sample_action(probs), grad_log_policy(s,a,theta) # V_w(s), grad_V(s,w)
running_avg_reward = [] # dùng để theo dõi performance state = env.reset() for t in
range(1, max_steps+1): probs = get_policy_probs(state, theta) # softmax outputs action
= sample_action(probs) next_state, reward, done, info = env.step(action) # Compute TD
error differential v_s = V_w(state) v_next = V_w(next_state) delta = reward - rho +
v_next - v_s # Critic update w += alpha_c * delta * grad_V(state, w) # Average reward
update rho += beta * delta # Actor update theta += alpha_a * delta *
grad_log_policy(state, action, theta) # Optionally entropy bonus: # theta += alpha_a *
entropy_coeff * grad_entropy(probs) # For monitoring: accumulate reward and
periodically compute average reward per step running_avg_reward.append(reward) if t %
eval_interval == 0: avg_r = sum(running_avg_reward[-eval_interval:]) / eval_interval
print(f"Step {t}, estimated avg reward: {rho:.3f}, recent avg: {avg_r:.3f}") state =
next_state # If environment has natural terminal (e.g. episodic), can env.reset() when
done; # for continuing, may ignore done or treat resets accordingly. # Sau khi
training, có thể test policy: test_rewards = [] for ep in range(num_test_episodes): s =
env.reset() total = 0 for step in range(max_test_steps): a =
sample_action(get_policy_probs(s, theta)) # or choose greedy s, r, done, _ =
env.step(a) total += r if done: break test_rewards.append(total) print("Average test
reward per episode:", sum(test_rewards)/len(test_rewards))
```

- Khi chạy, “total reward obtained by learned policy” có thể tính theo average per step (continuing) hoặc tổng reward per episode (nếu giả episodic cho đánh giá).
 - Kết quả cụ thể phụ thuộc task và hyperparameters; bạn sẽ thấy ρ estimate tăng dần và average test reward cải thiện theo thời gian. doutoury2.tistory.com
-

9. Thách thức và Lưu ý

1. **Exploration trong continuing:** policy stochastic, entropy bonus giúp duy trì thăm đủ trạng thái. Nếu policy quá nhanh deterministic, critic ước lượng sai, gradient không đúng hướng.
 2. **Step-size scheduling:**
 - Critic cần theo kịp actor: α_c thường lớn hơn α_a .
 - ρ cần cập nhật chậm: β nhỏ hơn α_c .
 - Nếu nonstationary, giảm step-sizes theo thời gian giúp hội tụ gần điểm tối ưu cục bộ.
 3. **Ergodicity:** MDP phải ergodic dưới policy, tức policy phải có xác suất nonzero thăm mọi trạng thái quan trọng.
 4. **Variance reduction:** even with Actor-Critic, gradient noise còn; có thể dùng n-step returns hoặc TD(λ) cho critic để cải thiện ước lượng value.
 5. **Natural Policy Gradient:** cao hơn stability nhưng tốn tính toán; có thể thử cho môi trường phức tạp hoặc sensitivity cao.
 6. **Neural network vs linear:**
 - Với linear features đơn giản, lý thuyết hội tụ on-policy average reward Actor-Critic tương đối rõ ràng.
 - Với deep nets, cần kỹ thuật ổn định (target networks, normalization, gradient clipping) nếu môi trường phức tạp.
 7. **Monitoring & Debug:** Theo dõi ρ estimate, running average reward, TD error distribution; nếu TD error dao động lớn, giảm α_c ; nếu ρ drift bất thường, điều chỉnh β .
 8. **So sánh discounted vs average:** Trong nhiều ứng dụng, discounted Actor-Critic phổ biến hơn; average reward Actor-Critic ít dùng do complexity. Dùng average khi thực sự cần tối ưu reward rate dài hạn, không quan tâm ưu tiên reward gần.
-

10. Summary

- Slide “Actor-Critic for Continuing Tasks” tập trung vào cách ước lượng sample-based policy gradient cho average reward và cách triển khai Actor-Critic tương ứng.
- **Estimating Policy Gradient:** Dùng score-function ($\nabla \log \pi$) kết hợp TD error (differential hoặc discounted) làm estimate advantage.
- **Actor-Critic:** Critic học value/differential value bằng TD; actor cập nhật policy parameter theo gradient thu được.

- **Continuing vs Episodic:** Continuing yêu cầu average reward hoặc discounted infinite-horizon; average reward Actor-Critic cần thêm ước lượng p .
- **Implementation:** softmax/Gaussian policy, critic approximator, step-size, exploration stochastic, cập nhật online.
- **Thách thức:** exploration ergodic, variance reduction, stability khi policy thay đổi ảnh hưởng distribution, tuning hyperparameters, đặc biệt với deep nets.
- **Ứng dụng:** Khi cần tối ưu rate reward dài hạn (scheduling, điều khiển liên tục vô hạn), dùng average reward Actor-Critic; nếu episodic hoặc ưu tiên reward gần, dùng discounted Actor-Critic phổ biến (A2C, PPO...).