


1. Hiểu Actor-Critic với Softmax policies.

2. Hiểu Gaussian policies cho continuous actions. 

Việc nắm rõ hai phần này rất quan trọng khi triển khai các thuật toán policy-based trong RL, đặc biệt với môi trường discrete action và continuous action.

## 2. Actor-Critic với Softmax Policies

### 2.1. Khái niệm Softmax Policy Parameterization

- **Action preferences:** Với discrete actions, thường xây dựng một hàm "preference"  $h(s, a; \theta)$  tùy thuộc state và action, parameterized bởi  $\theta$ . Không nhất thiết là  $Q(s, a)$ ; đây chỉ là giá trị trung gian để so sánh các action.
- **Softmax chuyển thành xác suất:**

$$\pi_{\theta}(a | s) = \frac{\exp(h(s, a; \theta))}{\sum_b \exp(h(s, b; \theta))}.$$


Điều này đảm bảo  $\pi \geq 0$  và  $\sum_a \pi(a|s) = 1$ .

- **Tính gradient  $\nabla_{\theta} \log \pi_{\theta}(a|s)$ :**

$$\nabla_{\theta} \log \pi_{\theta}(a|s) = \nabla_{\theta} h(s, a; \theta) - \sum_b \pi_{\theta}(b|s) \nabla_{\theta} h(s, b; \theta).$$

Với linear preferences ( $h(s, a; \theta) = \theta^T \phi(s, a)$ ), ta có  $\nabla_{\theta} h = \phi(s, a)$ , và  $\nabla \log \pi = \phi(s, a) - \mathbb{E}_{b \sim \pi}[\phi(s, b)]$ . Đây là cơ sở để cập nhật actor trong policy gradient hoặc Actor-Critic [profs.scienze.univr.it](https://profs.scienze.univr.it).

### 2.2. Features của action preference function

- Slide nhấn "The actor's action preferences depend on the state and action" và "Using stacked state features" 
- **Stacked features:** Giả sử ta đã có feature  $\phi_{\text{state}}(s)$  cho state (ví dụ tile coding, coarse coding, embedding, v.v.). Với  $|A|$  hành động, có thể tạo vector  $\phi(s, a)$  kích thước  $|A| \times d$  bằng cách "stack"  $|A|$  khối, mỗi khối dạng  $\phi_{\text{state}}(s)$  ở vị trí tương ứng action  $a$ , các khối khác bằng 0. Khi đó:

$$h(s, a; \theta) = \theta^T \phi(s, a) = \theta_a^T \phi_{\text{state}}(s),$$

tức mỗi action  $a$  có bộ trọng số riêng  $\theta_a$  để tính preference. Cách này đơn giản, sparse, và khi cập nhật semi-gradient TD, chỉ ảnh hưởng phần  $\theta_a$  tương ứng.

- **Ví dụ minh họa:**

- Giả sử  $\phi_{\text{state}}(s)$  chiều 10, có 4 action  $\rightarrow \phi(s, a)$  chiều 40. Nếu agent đang ở state  $s$  và chọn action  $a$ ,  $\phi(s, a)$  có 10 giá trị từ  $\phi_{\text{state}}(s)$  tại segment index tương ứng  $a$ . Khi cập nhật  $w(\theta)$ , update chỉ tác động segment  $a$ . Điều này giúp generalization giữa các state nhưng phân biệt giữa các action.

- **Actor-Critic algorithm:**

- **Critic:** dùng semi-gradient TD để xấp xỉ  $V(s; w)$  (hoặc  $Q$  nếu muốn), cập nhật  $w$  bằng TD error:

$$\delta_t = r_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t).$$

- **Actor:** dùng  $\delta_t$  làm estimate advantage, cập nhật  $\theta$  theo:

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

- Đây là Actor-Critic one-step: critic “đánh giá” bằng TD, actor “cải thiện” chính sách dựa trên gradient log-prob weighted với  $\delta_t$ . [davidstarsilver.wordpress.com](https://davidstarsilver.wordpress.com) [profs.scienze.univr.it](https://profs.scienze.univr.it)

- **Triển khai online:**

1. Khởi  $\theta$  (có thể zeros để policy ban đầu uniform) và  $w$ .
2. Ở mỗi bước: quan sát  $s$ , sample action  $a \sim \pi_{\theta}(\cdot | s)$ , nhận  $r, s'$ .
3. Tính  $\delta_t$ , cập nhật critic và actor.
4. Tiếp tục vô hạn hoặc đến khi policy hội tụ.

- **Lưu ý:**

- Learning rates: critic thường dùng  $\alpha_c$  lớn hơn actor  $\alpha_a$  để critic theo kịp actor.
- Exploration: softmax policy inherent stochastic; có thể thêm entropy bonus:

$$\theta \leftarrow \theta + \alpha [\delta_t \nabla_{\theta} \log \pi_{\theta}(A_t | S_t) + \lambda \nabla_{\theta} H(\pi_{\theta}(\cdot | S_t))].$$

- Với continuing tasks, cần policy ergodic để critic ước lượng giá trị dưới phân phối đủ rộng.

- **Nguồn tham khảo:** Sutton & Barto Chương về Policy Gradient và Actor-Critic [profs.scienze.univr.it](https://profs.scienze.univr.it)  
[opencourse.inf.ed.ac.uk](https://opencourse.inf.ed.ac.uk) ; David Silver Lecture về Policy Gradient Methods [davidstarsilver.wordpress.com](https://davidstarsilver.wordpress.com) .

## 3. Gaussian Policies cho Continuous Actions

### 3.1. Khái niệm và parameterization

- Khi action space continuous, không thể liệt kê hết actions để softmax. Thay vào đó dùng phân phối continuous, thường Gaussian:

$$\pi_{\theta}(a | s) = \mathcal{N}(a; \mu_{\theta}(s), \Sigma_{\theta}(s)).$$

- **Mean  $\mu_{\theta}(s)$ :** một hàm parameterized (linear hoặc neural network) đầu vào state, output mean vector.
- **Covariance  $\Sigma_{\theta}(s)$ :** phải là ma trận dương định; đơn giản nhất dùng diagonal covariance với parameter log-variance  $\phi(s)$  hoặc một tham số cố định  $\sigma^2$ .
- Slide nhấn: “Mu can be any parameterized function. Sigma must be positive. The policy parameters” □
- **Khởi tạo variance lớn:**
  - “We typically initialize the variance to be large so that a wide range of actions are tried.” Khi khởi training, việc sample từ Gaussian với  $\sigma$  lớn giúp exploration rộng khắp action space. Khi policy học dần, có thể giảm  $\sigma$  (hoặc học  $\sigma$  qua gradient) để policy tập trung exploitation. □
- **Gradient of Log Gaussian policy:**
  - Với 1D action  $a \sim \mathcal{N}(\mu(s), \sigma^2)$ ,

$$\log \pi(a|s) = -\frac{(a - \mu(s))^2}{2\sigma^2} - \frac{1}{2} \log(2\pi\sigma^2).$$

$$\nabla_{\theta} \log \pi(a|s) = \frac{(a - \mu(s))}{\sigma^2} \nabla_{\theta} \mu(s) + \left( \frac{(a - \mu(s))^2}{\sigma^3} - \frac{1}{\sigma} \right) \nabla_{\theta} \sigma \quad (\text{nếu } \sigma \text{ cũng parameterized}).$$

- Với diagonal Gaussian multi-dim, tương tự apply per-dimension. Đây là score-function gradient để cập nhật actor:

$$\theta \leftarrow \theta + \alpha \delta_t \nabla_{\theta} \log \pi_{\theta}(a|s).$$

- Nếu học cả  $\mu$  và  $\sigma$ , cần đảm bảo  $\sigma$  luôn  $>0$ , thường parameterize log  $\sigma$  hoặc dùng softplus.
- **Ví dụ minh họa:**
  - Môi trường continuous control (ví dụ CartPole continuous version, MuJoCo tasks): mạng actor nhận input state, output  $\mu(s)$  và log  $\sigma(s)$ . Sample action  $a \sim N(\mu, \sigma^2)$ .
  - Critic học  $V(s)$  bằng TD (discounted) hoặc differential value (average). Actor update theo  $\delta_t \nabla \log \pi$ .
  - Khi training, khởi  $\sigma$  lớn (ví dụ 1.0), agent thử nhiều action; dần dần  $\sigma$  có thể giảm nếu parameterized và gradient hướng giảm exploration khi policy cải thiện.
- **Ưu điểm continuous policies** (theo slide):
  - Không phải discretize action space.
  - Generalization over actions: nearby actions có xác suất cao tương tự, policy mượt.
  - Expressiveness và fine-grained control: agent có thể chọn giá trị liên tục, phù hợp robotics, control tasks.
  - Smoothness: chuyển đổi hành vi mượt mà, tránh giật cục.
  - Efficiency: gradient-based optimization trực tiếp trên parameters của Gaussian dễ thực thi.
  - Optimization: policy gradient methods (REINFORCE, Actor-Critic, PPO, SAC, DDPG) hoạt động tự nhiên với Gaussian policies. [opencourse.inf.ed.ac.uk](https://opencourse.inf.ed.ac.uk)

## 3.2. Implementation details và lưu ý

- **Parameterizing  $\Sigma$ :**
  - Thường dùng diagonal covariance:  $\sigma_i(s) = \exp(h_i(s; \theta))$  hoặc softplus( $h_i$ ). Khởi log  $\sigma$  lớn để exploration ban đầu.
  - Có thể giữ  $\sigma$  constant và chỉ học  $\mu$ ; đơn giản nhưng thiếu flexibility để giảm exploration tự động.
- **Sampling và reparameterization:**
  - Score-function gradient dùng  $\delta \nabla_{\theta} \log \pi$ ; variance có thể cao.
  - Với Gaussian, có thể dùng reparameterization trick ( $a = \mu(s) + \sigma(s) * \epsilon$ ,  $\epsilon \sim N(0,1)$ ) để ước lượng gradient có thể dùng pathwise derivative, giúp giảm variance trong một số phương pháp (thường dùng trong off-policy hoặc variational methods). Nhưng trong RL on-policy thường dùng score-function do  $\delta$  depends on reward.
- **Entropy regularization:** Thêm  $-\lambda \nabla_{\theta} H(\pi(\cdot|s))$  khuyến khích  $\sigma$  không giảm quá sớm, duy trì exploration.
- **Clipping action:** Nếu action cần giới hạn (ví dụ  $\pm 1$ ), sau khi sample Gaussian có thể cần squash (tanh) hoặc rescale. Cần tính gradient tương ứng qua tanh; thường dùng kỹ thuật reparameterization cho tanh-squashed Gaussian.
- **Stability:**

- Learning rate actor thường nhỏ.
  - Critic (value) cần học ổn định (target network, normalization) trong deep RL.
  - Trong continuing tasks hoặc long episodes, discount factor  $\gamma$  chọn gần 1 nếu quan tâm dài hạn. Hoặc dùng average reward setting nếu muốn tối ưu rate. Actor-Critic average reward cũng có thể áp dụng Gaussian policy tương tự.
  - **Ví dụ code:**
    - Trong PyTorch/TensorFlow, network đầu ra two heads: one head output  $\mu(s)$ , one head output  $\log \sigma(s)$ . Sample:  $\epsilon = \text{torch.randn\_like}(\mu)$ ;  $a = \mu + \exp(\log \sigma) * \epsilon$ . Tính  $\log\_prob$  từ distribution object để gradient.
    - Cập nhật actor:  $\text{loss\_actor} = -\log\_prob(a|s) * \delta_t$  (minimize negative); critic loss =  $\delta_t^2$ ; update cả hai mạng.
  - **Nguồn tham khảo:** Sutton & Barto; Spinning Up in Deep RL (OpenAI); David Silver Lecture; bài blog "Policy Gradient Algorithms" [opencourse.inf.ed.ac.uk](https://opencourse.inf.ed.ac.uk).
- 

## 4. Actor-Critic Algorithm tổng quát với Softmax và Gaussian Policies

### 4.1. Actor-Critic chung (one-step TD)

- **Critic:** học value  $V(s; w)$  (discounted) hoặc differential value (average reward) qua TD error.
- **Actor:** cập nhật  $\theta$  qua  $\delta_t \nabla_{\theta} \log \pi_{\theta}(a|s)$ .
- **Khung online:**
  1. Khởi  $\theta, w$  (và  $\rho$  nếu average reward).
  2. Quan sát  $s$ , sample  $a \sim \pi_{\theta}(\cdot|s)$ .
  3. Nhận  $r, s'$ .
  4. Tính  $\delta$ :
    - Discounted:  $\delta = r + \gamma V_w(s') - V_w(s)$ .
    - Average:  $\delta = r - \rho + V_w(s') - V_w(s)$ .
  5. Cập nhật critic:  $w \leftarrow w + \alpha_c \delta \nabla_w V(s)$ . Nếu average: cập nhật  $\rho \leftarrow \rho + \beta \delta$ .
  6. Cập nhật actor:  $\theta \leftarrow \theta + \alpha_a \delta \nabla_{\theta} \log \pi_{\theta}(a|s)$ .
  7.  $s \leftarrow s'$ ; lặp.
- **Softmax vs Gaussian:** công thức actor update giống nhau, chỉ khác cách tính  $\log \pi$  và  $\nabla_{\theta} \log \pi$  tùy policy.
- **Feature/Network:**
  - Softmax:  $\varphi(s, a)$  stacked hoặc network đầu ra preferences.
  - Gaussian: network đầu ra  $\mu(s)$  và  $\log \sigma(s)$ .
- **Exploration:** softmax inherently stochastic; Gaussian sampling inherently exploration. Có thể thêm entropy bonus: actor update thêm term  $\nabla_{\theta} H(\pi_{\theta})$ .
- **Timescales:** critic nhanh, actor trung bình,  $\rho$  chậm (nếu average).
- **Theo dõi:** với discrete, theo dõi average return per episode hoặc running reward; với continuing, theo dõi running average reward per step; với continuous, tùy environment metric.

## 4.2. Ví dụ minh họa Softmax Actor-Critic

- Môi trường Grid World discrete:
  - Feature  $\phi_{\text{state}}(s)$  (one-hot, tile coding, embedding).
  - Softmax preferences:  $h(s,a) = \theta_a^T \phi_{\text{state}}(s)$ .
  - Actor-Critic cập nhật như trên.
  - Quan sát policy dần tập trung vào hành động tốt.
- Ví dụ code pseudocode đã trình bày ở các phần trước.

## 4.3. Ví dụ minh họa Gaussian Actor-Critic

- Môi trường continuous control (ví dụ MountainCarContinuous, Pendulum, MuJoCo tasks):
    - Actor network nhận state, output  $\mu(s)$  và  $\log \sigma(s)$ .
    - Sample  $a = \mu + \sigma * \epsilon$ .
    - Critic network học  $V(s)$ .
    - Update actor qua  $\delta \nabla \log \pi$ ; update critic qua  $\delta \nabla V$ .
    - Khởi  $\sigma$  lớn, exploration tốt; sau dần  $\sigma$  có thể giảm nếu parameterized hoặc dùng entropy bonus để duy trì.
  - Thực nghiệm: các thuật toán như DDPG/SAC dùng khác biệt (deterministic policy gradient hoặc off-policy); nhưng on-policy Actor-Critic (PPO) dùng Gaussian policy tương tự.
- 

# 5. Ưu và nhược điểm của các Policy Parameterizations

## 5.1. Softmax Policies (Discrete)

- **Ưu điểm:**
  - Dễ triển khai, gradient log-prob đơn giản.
  - Stochastic policy đảm bảo exploration liên tục.
  - Kết hợp với Actor-Critic cho discrete action hiệu quả.
- **Nhược điểm:**
  - Nếu số action rất lớn, computing softmax over toàn bộ action có thể tốn kém.
  - Không dùng cho continuous action.
  - Khả năng exploration ngẫu nhiên có thể kém nếu nhiều action không tốt; có thể cần bổ sung entropy bonus hoặc other exploration methods.
- **Feature sharing:** stacked features giúp generalization giữa states nhưng phải cẩn thận feature design.

## 5.2. Gaussian Policies (Continuous)


- **Ưu điểm:**
  - Xử lý trực tiếp action continuous, không cần discretize.

- Expressiveness: fine-grained control, smooth transition.
  - Gradient-based optimization tự nhiên; exploration thông qua sampling noise.
  - **Nhược điểm:**
    - Cần đảm bảo  $\sigma$  positive; khởi và điều chỉnh  $\sigma$  đúng mức.
    - Score-function gradient có variance; có thể dùng reparameterization trong một số frameworks.
    - Nếu action bounds (ví dụ  $[-1, 1]$ ), cần squash/squash-correction gradient.
    - Policy gradient có thể converged chậm hoặc local optima; cần entropy bonus, trust region (PPO).
  - **Exploration:** Gaussian inherently exploration, nhưng có thể quá tập trung nếu  $\sigma$  giảm nhanh; entropy hoặc explicit  $\sigma$  learning cần careful tuning.
- 

## 6. Các lưu ý khi triển khai thực tế

1. **Normalization input:** Với network, normalize state input để ổn định training.
  2. **Gradient clipping:** Tránh gradient actor hoặc critic quá lớn destabilize.
  3. **Entropy regularization:** Giữ stochasticity đủ lâu, tránh policy trở nên quá deterministic sớm.
  4. **Learning rate schedules:** Thường giảm dần hoặc adaptive (Adam) nhưng cần giám sát performance.
  5. **Critic stability:** Với deep critic, có thể dùng target network hoặc n-step returns, TD( $\lambda$ ) để cải thiện độ ổn định và ước lượng chính xác.
  6. **Monitor metrics:**
    - Discrete: average return per episode.
    - Continuous: average cumulative reward, task-specific metrics.
    - Theo dõi distribution of actions,  $\sigma$  in Gaussian to đảm bảo exploration hợp lý.
  7. **Environment specifics:** Với continuous tasks, chú ý action bounds, dynamics noise, reward scaling. Có thể cần reward normalization.
  8. **Policy parameterization lựa chọn:**
    - Discrete action nhỏ: Softmax.
    - Discrete action large/vocabulary: có thể dùng hierarchical policies hoặc parameter sharing.
    - Continuous: Gaussian, Beta, or mixture models nếu phân phối multimodal.
  9. **Advanced methods:**
    - **Natural Policy Gradient / Fisher information:** tăng stability, nhưng tốn tính toán; có thể xấp xỉ (K-FAC).
    - **Trust Region / PPO:** giới hạn update mỗi lần, tránh policy thay đổi quá lớn.
    - **Actor ensembles / distributional critics:** nâng cao performance, stability.
    - **Off-policy variants:** DDPG, SAC, TD3: deterministic policies hoặc stochastic off-policy, khác với on-policy Actor-Critic nhưng vẫn liên quan đến policy parameterization.
-

## 7. Tổng kết

- Slide "Policy Parameterizations" đề cập hai phần chính: Softmax policies trong Actor-Critic và Gaussian policies cho continuous action. Cả hai đều là cách biểu diễn policy có tham số  $\theta$ , khả vi, để tối ưu trực tiếp qua policy gradient. 
- **Softmax**: dùng cho discrete, action preferences phụ thuộc state và action, thường kết hợp stacked features, actor cập nhật theo  $\delta \nabla \log \pi$ , critic semi-gradient TD.
- **Gaussian**: dùng cho continuous, parameterize  $\mu(s)$  và  $\sigma(s)$ , sample action, gradient log Gaussian để cập nhật actor, khởi  $\sigma$  lớn để exploration.
- Cả hai đều cần critic học value function (discounted hoặc differential value), exploration cơ bản qua stochastic policy, entropy bonus để giữ tính ergodic, và careful tuning learning rates, normalization, stability tricks.