1. **Understanding Parameterized Policies**: Khái niệm về chính sách được biểu diễn qua tham số θ, khác với lưu bảng (tabular) đặc tả xác suất hành động.
2. **Understanding Softmax Policy Parameterization**: Cách dùng softmax để chuyển các "action preferences" thành xác suất hợp lệ.
3. **Understanding the Advantages of Policy Parameterization**: Tại sao biểu diễn chính sách qua tham số lại quan trọng và hữu ích trong RL, đặc biệt khi không gian trạng thái/hành động lớn hoặc liên tục. lcalem.github.io

Dưới đây là phân tích chi tiết từng phần, kèm các dẫn chứng từ tài liệu học thuật và ví dụ minh họa.

---

1. Khái niệm Policy Parameterization

1.1. Tại sao cần Parameterized Policies?

- Trong RL truyền thống bảng (tabular), ta lưu Q(s,a) hoặc π(s,a) riêng cho từng cặp state-action. Khi số trạng thái hoặc hành động nhỏ, hoặc có thể liệt kê, cách này đơn giản. Tuy nhiên:
  - **Không gian trạng thái/hành động lớn**: Nếu trạng thái rất nhiều (hoặc continuous), không thể lưu bảng đầy đủ.
  - **Generalization**: Muốn dùng chung thông tin giữa các trạng thái tương tự, hoặc học chính sách cho continuous action space.
  - **End-to-end learning**: Trong deep RL, thường trực tiếp tối ưu tham số mạng neural biểu diễn policy.
- Do đó, **Parameterized policy** là hàm πθ(a|s) có tham số θ (vector hoặc weights mạng), đầu vào state, đầu ra là xác suất chọn action a. Khi thay đổi θ, policy thay đổi. Việc học policy trở thành tối ưu θ để cải thiện hiệu năng. lilianweng.github.io

1.2. Đặc tả tổng quát

- Ký hiệu θ ∈ ℝ^d là vector tham số. Policy πθ(a|s) là một hàm xác suất:

$$\pi_\theta(a \mid s) \geq 0, \quad \sum_a \pi_\theta(a \mid s) = 1 \quad \forall s.$$

- **Yêu cầu**: πθ(a|s) phải khả vi theo θ để có thể lấy gradient. Tức ∇θ πθ(a|s) tồn tại và hữu hạn. lcalem.github.io
- Với discrete action space, phổ biến là định nghĩa một hàm "action preference" h(s,a;θ) ∈ ℝ, rồi dùng softmax để đảm bảo điều kiện xác suất:

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

- Với continuous action space, thường dùng các phân phối tham số (ví dụ Gaussian) với tham số θ điều chỉnh mean/covariance:

$$\pi_\theta(a \mid s) = \mathrm{N}(a \mid \mu_\theta(s), \Sigma_\theta(s)),$$

  hoặc các phân phối khác phù hợp domain. milvus.io

1.3. Hạn chế của Parameterization

- Policy phải tuân ràng buộc xác suất: softmax cho discrete, phân phối xác suất hợp lệ cho continuous. Khi thiết kế hàm h(s,a;θ) hoặc network cấu trúc đầu ra, cần đảm bảo non-negative và sum to one (softmax) hoặc positive-definite covariance (Gaussian). elearning.di.unipi.it

- Với network, có thể thiết kế đầu ra raw preferences rồi áp softmax, hoặc đầu ra mean/variance rồi áp exp để đảm bảo dương, ...

---

2. Softmax Policy Parameterization

2.1. Định nghĩa và công thức

- **Action preference** h(s,a;θ): một hàm đầu vào state và action, tham số θ, cho giá trị bất kỳ thực. Ví dụ với linear: h(s,a;θ) = θ^T φ(s,a). Với neural: h(s,a;θ) = NN(s,a;θ) (thường NN đầu vào state, đầu ra vector preferences, hoặc đầu vào cả state và action).
- **Softmax** chính sách:

$$\pi_\theta(a \mid s) = \frac{\exp(h(s,a;\theta))}{\sum_{b \in A} \exp(h(s,b;\theta))}.$$

  Do đó π_θ(a|s) ≥ 0 và ∑_a π_θ(a|s) = 1. incompleteideas.net
- **Action preferences không phải action values**:
  - h(s,a;θ) chỉ đóng vai trò intermediate để tính softmax. Không phải Q(s,a). Thường Q-value dùng để học rồi derive policy (value-based), nhưng ở policy parameterization, h(s,a) là tham số riêng, không yêu cầu phải xấp xỉ Q. Tuy nhiên trong Actor-Critic, critic xấp xỉ Q hoặc V để tính gradient, còn h(s,a) là actor.
  - Quan trọng: h(s,a) chỉ dùng để so sánh tương đối giữa các action trong cùng state; softmax ấn định xác suất. Nếu thêm h(s,a) thêm một hằng c cho mọi a, softmax không thay đổi: softmax invariant to constant shift elearning.di.unipi.it .

2.2. Gradient của Softmax Policy

- Trong policy gradient, cần tính ∇θ log πθ(a|s). Với softmax và h linear hoặc mạng:

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

  →

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

  Với linear preferences h(s,a;θ) = θ^T φ(s,a):

  ∇_θ h(s,a;θ) = φ(s,a), \quad ∇_θ \log π_θ(a|s) = φ(s,a) - \sum_{b} π_θ(b|s)\,φ(s,b). \]
  :contentReference[oaicite:7]{index=7}
- Phương pháp tính này là nền tảng cho policy gradient: Eπ[∇_θ log π_θ(a|s) * advantage].

2.3. Ví dụ Softmax trong Grid World

- Giả sử Grid World với finite discrete actions (ví dụ lên/xuống/trái/phải). Xây dựng h(s,a;θ) linearly: features φ(s,a) có thể là tile-coded state kết hợp one-hot action.
- Tại mỗi state s, compute h(s,a;θ) = θ^T φ(s,a) cho mọi a. Sau đó tính softmax probabilities: π(a|s) = exp(h)/sum.
- Khi triển khai policy gradient (ví dụ REINFORCE):
  1. Khởi θ ngẫu nhiên (có thể zero để policy ban đầu là uniform).

2. Tương tác theo π_θ: sample actions theo softmax, collect rewards.
3. Tính returns G_t, rồi update θ theo:

$$\theta \leftarrow \theta + \alpha\, G_t\, \nabla_\theta \log \pi_\theta(A_t|S_t).$$

4. Với Actor-Critic, thay G_t bằng advantage ≈ Q(s,a) - V(s).

- **Giải thích**: Softmax cho phép policy mềm dẻo: tại state s, xác suất từng a phụ thuộc preferences. Khi update, nếu action A_t mang reward cao, ∇ log π tăng θ theo hướng tăng h(s,A_t) và giảm tương đối h(s,b) cho b≠A_t. Qua nhiều trải nghiệm, policy dần ưu tiên action tốt. elearning.di.unipi.it ai.stackexchange.com

---

3. Ưu điểm của Policy Parameterization

Slide liệt kê nhiều ưu điểm, dưới đây phân tích chi tiết:

3.1. Compact Representation

- **Ý nghĩa**: Với tham số θ, có thể biểu diễn chính sách phức tạp mà không cần lưu trữ từng state-action. Tham số có thể nhỏ hơn rất nhiều so với số trạng thái.
- Ví dụ: state space lên tới hàng triệu điểm (hoặc continuous), nhưng ta dùng mạng neural nhỏ hoặc feature tuyến tính kết hợp basis, tổng độ lớn tham số θ << bảng Q size. lilianweng.github.io
- **Tác động**: Tiết kiệm bộ nhớ, khả năng áp dụng RL vào các domain lớn.

3.2. Generalization Across States

- Khi parametrization chia sẻ tham số giữa nhiều state (ví dụ network có layers chung), học trên một state cũng ảnh hưởng đến state tương tự. Điều này dẫn đến generalization: policy có thể phản hồi hợp lý ở states chưa gặp.
- Ví dụ: network đầu vào là cảm biến môi trường, các weights học patterns chung; khi gặp trạng thái mới có đặc trưng tương tự, policy vẫn chọn action hợp lý. lilianweng.github.io
- Trái ngược với tabular, không generalize giữa states. Parameterized policy tận dụng đặc trưng (features hoặc representation learning) để generalize.

3.3. Continuous Action Spaces

- Tabular hoặc softmax trên discrete action không áp dụng cho continuous actions (infinite choices). Với parameterized policy, có thể dùng phân phối liên tục, ví dụ Gaussian policy:

$$\pi_\theta(a|s) = \mathrm{N}(a; \mu_\theta(s), \sigma^2),$$

  hoặc Beta, v.v. Do đó agent có thể học trực tiếp chọn action continuous bằng sampling. milvus.io
- Hơn nữa, policy gradient methods (REINFORCE, PPO, DDPG, SAC, v.v.) dựa hoàn toàn trên parameterized policies để xử lý continuous control.

3.4. Flexibility

- Parameterization linh hoạt: có thể chọn cấu trúc linear, neural network, Gaussian Mixture Models, spline basis, v.v.
- Dễ mở rộng để biểu diễn chính sách phức tạp: multi-modal distributions, hierarchical policies, conditional networks.
- Khi domain thay đổi, có thể điều chỉnh cấu trúc network/feature mà không thay đổi algorithm core.

3.5. Learning Complex Policies

- Trong môi trường high-dimensional (hình ảnh, ngôn ngữ, robotics), policy phải xử lý input phức tạp. Mạng neural sâu (deep networks) làm nhiệm vụ trích xuất feature và mapping đến phân phối action trực tiếp.
- Actor networks trong Actor-Critic hoặc policy networks trong PPO, TRPO, DDPG... minh họa learning complex stochastic/deterministic policy end-to-end. lilianweng.github.io

3.6. End-to-End Learning

- Khi biểu diễn policy bằng neural network, toàn bộ pipeline từ raw input (hình ảnh, sensor data) đến action được học trực tiếp qua tối ưu θ nhằm tối đa expected return. Không cần tách rõ feature engineering và policy design.
- Kết hợp với perception modules (CNN, RNN) để xử lý raw observations, rồi output phân phối action. Toàn bộ gradient lan truyền từ reward signal qua critic network (actor-critic) hoặc REINFORCE update. Cho phép học end-to-end. spinningup.openai.com

3.7. Khả năng mở rộng và áp dụng vào các môi trường phức tạp

- Từ game AI (Atari, Go) đến robotics, tài chính, recommendation: parameterized policies (deep RL) là nền tảng.
- Khi cần policy stochatic để thăm dò, softmax cho discrete, Gaussian cho continuous.
- Khung policy gradient, actor-critic, PPO, SAC... đều dựa trên policy parameterization.

3.8. Một số lưu ý/nhược điểm cần cân nhắc

- **Local Optima**: Policy gradient tối ưu trực tiếp θ, thường hội tụ đến cực trị cục bộ; cần careful initialization, entropy bonus để tránh premature convergence.
- **Variance của gradient estimate**: REINFORCE ước lượng gradient có variance cao; dùng baseline (value function) để giảm variance.
- **Khó điều chỉnh hyperparameters**: Learning rate, network architecture, entropy coefficient… cần tuning.
- **Requirement differentiable**: Mọi thành phần trong policy phải khả vi để backprop; với discrete action softmax thỏa mãn, nhưng sampling discrete action phi tuyến cần tính gradient qua log-prob trick.
- **Cần exploration cơ bản**: Stochastic policy tự khám phá; nhưng cần thêm entropy bonus hoặc noise scheduling để đủ exploration.

---

4. Ví dụ Cụ thể: Softmax Policy trong Grid World

Dưới đây là pseudocode minh họa cách xây dựng và học softmax policy (policy gradient) trong Grid World đơn giản:

1. **Feature representation**:
   - Xác định φ(s): vector feature state (ví dụ one-hot encoding, tile coding, embedding).
   - Nếu dùng linear preferences: h(s,a;θ) = θ_a^T φ(s), với θ là ma trận kích thước |A|×d. Mỗi action có weight riêng.
   - Với neural: NN đầu vào φ(s) hoặc raw state, output vector kích thước |A| là preferences; sau đó softmax.
2. **Chọn hành động**:
   - Tại state s: compute preferences h_i = h(s,a_i;θ) cho mọi a_i, sau đó π(a_i|s) = exp(h_i)/∑ exp(h_j).
   - Sample action A ~ π(·|s).

3. **Thu thập dữ liệu episode** (REINFORCE) hoặc trải nghiệm từng bước (Actor-Critic).
4. **Tính return hoặc advantage**:
   - Với REINFORCE: sau mỗi episode, tính G_t cho từng t.
   - Với Actor-Critic: estimate V(s;w_c) (critic) để tính advantage A_t = R_{t+1} + γ V(s_{t+1}) − V(s_t).
5. **Cập nhật θ**:
   - Với REINFORCE:

$$\theta \leftarrow \theta + \alpha \sum_t G_t \nabla_\theta \log \pi_\theta(A_t|S_t).$$

   - Với Actor-Critic:
     θ \leftarrow θ + α \sum_t A_t ∇_θ \log π_θ(A_t|S_t), \quad \text{và critic update để học V(s;w_c)}.
6. **Constraints & implementation**:
   - Softmax computation: chú ý numerical stability (subtract max preference trước khi exp).
   - Entropy bonus (optional): thêm −β ∇θ ∑_a πθ(a|s) log πθ(a|s) để khuyến khích stochasticity ban đầu.
   - Learning rate decay, batch updates/truncated trajectories, v.v.

Qua nhiều episode, θ học sao cho policy πθ tập trung vào các action có expected return cao. Softmax phù hợp vì discrete action small/medium. elearning.di.unipi.it ai.stackexchange.com

---

5. Kết hợp với các thuật toán RL hiện đại

- **Actor-Critic**: Actor là policy parameterized (softmax discrete hoặc Gaussian continuous), Critic xấp xỉ value function (V hoặc Q) để cung cấp advantage cho actor update.
- **Policy Optimization Algorithms**: PPO, TRPO, A2C/A3C, DDPG/SAC (cho continuous) đều dựa trên parameterized policies. Softmax hoặc Gaussian hay các phân phối phức tạp hơn.
- **Entropy Regularization**: Trong policy parameterization, thường thêm entropy của πθ để duy trì exploration: loss_actor = −E[A_t log πθ(a|s) + β H(πθ(·|s))].
- **Trust Region Methods**: Giới hạn thay đổi policy mỗi update để tránh diverge.
- **Multimodal Policies**: Với Gaussian Mixture hoặc Normalizing Flows để mô hình hóa phân phối complex trong continuous action.

---

6. Tổng kết

- **Policy Parameterization** là khái niệm then chốt trong RL hiện đại, cho phép học trực tiếp chính sách phức tạp với tham số θ.
- **Softmax Parameterization**: Cách phổ biến cho discrete action space; dùng action preferences h(s,a;θ) rồi softmax để ra xác suất hợp lệ. Gradient log-prob có công thức rõ ràng, trọng tâm cho policy gradient.
- **Action Preferences ≠ Action Values**: Preferences là intermediate để tính xác suất, không phải Q-value. Actor-Critic có critic xử lý Q/V song song, nhưng không nhầm lẫn h và Q.
- **Ưu điểm**: Compact representation, generalization across states, xử lý continuous actions, flexibility, end-to-end learning, learning complex policies trong high-dimensional tasks.
- **Nhược điểm/chú ý**: Local optima, variance high gradient, cần baseline, exploration strategy, tuning hyperparameters, requirement differentiable.

- **Ví dụ**: Grid World softmax policy gradient hoặc Actor-Critic.
- **Mở rộng**: Policy optimization (PPO, TRPO...), entropy regularization, trust region, multimodal distributions, neural architectures phức tạp, hierarchical policies.