1. Mục tiêu của Policy Gradient Algorithms

- **Mục tiêu học chính sách (Learning Policy Objective)**: Trong Reinforcement Learning, agent cần tìm một chính sách π (có thể parameterized bằng θ) sao cho tối đa hóa phần thưởng dài hạn. Với episodic tasks, objective thường là giá trị khởi đầu hoặc tổng phần thưởng trong một episode. Với continuing tasks (không phân chia rõ episode), có thể dùng discounted return vô hạn hoặc average reward (reward trung bình mỗi bước). Việc lựa chọn objective ảnh hưởng cách tính gradient và thuật toán cập nhật. lilianweng.github.io datacamp.com .
- **Formalizing the goal**:
  - **Episodic**: $J(\theta) = \mathrm{E}_\pi[\sum_{t=0}^{T-1} R_{t+1}]$, với T độ dài episode ngẫu nhiên. Gradient-based methods (REINFORCE, Actor-Critic episodic) dùng công thức policy gradient tương ứng.
  - **Continuing**: Thường có hai cách:
    1. **Discounted return**: $J(\theta) = \mathrm{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_{t+1}]$ với $0 \le \gamma < 1$. Dùng khi muốn ưu tiên reward gần hơn hoặc cần đảm bảo hội tụ khi tổng reward vô hạn.
    2. **Average reward**: $J(\theta) = \rho(\pi_\theta) = \lim_{T\to\infty} \frac{1}{T}\mathrm{E}[\sum_{t=1}^{T} R_t]$, hay dưới phân phối ổn định $\mu_\pi(s)$: $\rho(\pi) = \sum_s \mu_\pi(s) \sum_a \pi(a|s) r(s,a)$. Phù hợp với continuing tasks không rõ ranh giới episode và khi muốn tối ưu reward rate dài hạn. opencourse.inf.ed.ac.uk en.wikipedia.org .
- **Thách thức**: Khi policy πθ thay đổi, phân phối trạng thái ổn định μ_π cũng thay đổi, khiến objective phụ thuộc phức tạp vào θ không chỉ qua πθ(a|s) mà còn qua μ_π(s). Khi triển khai gradient ascent, phải xét đến ảnh hưởng của θ lên phân phối trạng thái lâu dài; tuy nhiên policy gradient theorem cho phép tránh trực tiếp tính đạo hàm của μ_π bằng các giả thiết và chuyển sang dạng expectation trên các trải nghiệm on-policy opencourse.inf.ed.ac.uk .

---

2. Average Reward Objective trong Continuing Tasks

- **Định nghĩa**:

$$\rho(\pi) = \lim_{T\to\infty} \frac{1}{T}\mathrm{E}[\sum_{t=1}^{T} R_t] = \sum_s \mu_\pi(s) \sum_a \pi(a|s)\, r(s,a),$$

  với $\mu_\pi(s)$ là phân phối trạng thái ổn định (steady-state distribution) khi chạy policy π∞. Mục tiêu tìm θ sao cho $\rho(\pi_\theta)$ lớn nhất. opencourse.inf.ed.ac.uk en.wikipedia.org .
- **Ưu/Nhược so với Discounted**:
  - **Ưu**: Không cần chọn γ gần 1 để tập trung dài hạn; trực tiếp tối ưu reward rate, tránh bias do discount quá nhỏ (myopic) hoặc variance lớn do discount quá gần 1. Phù hợp với các tác vụ liên tục (continuous control, scheduling) không có điểm kết thúc tự nhiên.
  - **Nhược**: Cần ước lượng μ_π hoặc ước lượng average reward ρ song song với policy learning; thuật toán phức tạp hơn. Khi reward ngắn hạn quan trọng hơn hoặc muốn ưu tiên nhanh, average reward không biểu diễn được ưu tiên đó. datascience.stackexchange.com reddit.com .

- **Ví dụ minh họa**: Giả sử môi trường hình vòng (ring) với hai lựa chọn: vòng trái trả reward 1 ngay, vòng phải trả reward 2 nhưng sau một chu kỳ dài hơn. Với discounted, agent cần γ đủ lớn để thấy giá trị vòng phải vượt vòng trái; với average reward, agent so sánh rate trung bình: vòng phải có rate 2/m vs vòng trái 1/m (m độ dài chu kỳ), nên chọn vòng phải trực tiếp mà không cần điều chỉnh γ. linkedin.com .

---

3. Policy Gradient Theorem cho Continuing Tasks

- **Mục đích**: Tính gradient $\nabla_\theta J(\theta)$ (với J có thể là discounted return hoặc average reward) mà không cần biết explicit mô hình chuyển động môi trường P(s'|s,a).
- **Công thức tổng quát**:
  - Với **discounted**:

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

    với $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr(S_t = s|\pi)$ (discounted weighting).
  - Với **average reward**:

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

    hoặc tương đương dùng differential action-value function $q_\pi(s,a)$ định nghĩa differential return (R − ρ + next differential value). Steady-state distribution μ_π đóng vai trò weighting tương tự d_π. opencourse.inf.ed.ac.uk en.wikipedia.org .
- **Ý tưởng chính**:
  - Thay vì phải tính đạo hàm của μ_π(s) theo θ (phức tạp), policy gradient theorem cho thấy gradient có thể viết dưới dạng expectation trên phân phối on-policy (discounted hoặc steady-state), nhân với Q-function. Khi ước lượng Q-function (critic), có thể thực hiện gradient ascent cho θ.
  - Với average reward, cần estimate differential Q-function $q_\pi(s,a) = \mathrm{E}[\sum_{k=0}^{\infty}(R_{t+k+1} - \rho(\pi))|S_t = s, A_t = a]$, nhưng gradient formula tương tự.
- **Trích dẫn lý thuyết**: Policy Gradient Theorem lần đầu bởi Sutton et al. (1999) và trong sách Sutton & Barto, có kết quả cho cả discounted và average reward settings; bài giảng RL của Edinburgh (opencourse) cũng trình bày công thức với steady-state distribution. opencourse.inf.ed.ac.uk .

---

4. Ước lượng Policy Gradient qua Sampling

- **Sampling-based estimation**: Vì ∇J(θ) là expectation theo phân phối on-policy, ta có thể lấy mẫu trải nghiệm (s,a,r,s') từ policy hiện tại π_θ và ước lượng gradient qua các ước lượng Monte

Carlo hoặc TD:

$$\nabla_\theta J(\theta) = \mathrm{E}_{s\sim d_\pi, a\sim\pi_\theta(\cdot|s)}[Q_\pi(s,a)\,\nabla_\theta \log \pi_\theta(a|s)]$$

(cho discounted hoặc average reward phù hợp). opencourse.inf.ed.ac.uk .

- **Ước lượng Q_π(s,a)**:
  - **Monte Carlo**: Trong episodic, dùng return G_t để ước lượng Q; trong continuing discounted, dùng truncated return hoặc infinite-horizon with discount.
  - **Actor-Critic**: Estimate value function (V(s) hoặc Q(s,a)) bằng critic (TD, TD(λ), v.v.), sau đó dùng advantage function $A(s,a) = Q(s,a) - V(s)$ để giảm variance gradient. Với average reward, critic cần estimate differential value, thường gọi average reward actor-critic.
- **Cập nhật θ**:

$$\theta \leftarrow \theta + \alpha\, \widehat{\nabla_\theta J(\theta)},$$

  với $\widehat{\nabla_\theta J}$ từ sampling.
- **Giảm variance**: Dùng baseline (V(s) hoặc ρ) để subtract offset trong return, giảm variance trong ước lượng gradient mà không bias: ví dụ advantage $A(s,a) = Q(s,a) - V(s)$ trong discounted; trong average reward, dùng differential TD error $\delta = R - \hat{\rho} + V(s') - V(s)$. opencourse.inf.ed.ac.uk .

---

5. Thuật toán Policy Gradient cho Continuing Tasks

5.1. Discounted Actor-Critic (Continuing)

- **Objective**: maximize $\mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}]$.
- **Actor update**:

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

  với $\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$ là TD error.
- **Critic update**: cập nhật w để xấp xỉ V(s) (e.g., semi-gradient TD).
- **Exploration**: policy π_θ (softmax hoặc Gaussian) inherently stochastic, exploration được điều khiển qua entropy bonus hoặc noise.
- **Cân nhắc**: vì discounted, weighting state theo d_π(s) = ∑ γ^t P(S_t=s), không hoàn toàn steady-state nhưng ưu tiên gần hiện tại. en.wikipedia.org opencourse.inf.ed.ac.uk .

5.2. Average Reward Actor-Critic (Continuing)

- **Objective**: maximize average reward ρ(π).
- **Differential TD error**:

$$\delta_t = R_{t+1} - \hat{\rho} + V_w(S_{t+1}) - V_w(S_t).$$

- **Critic (value estimator)**: cập nhật w để xấp xỉ differential value h(s):

$$w \leftarrow w + \alpha\ \delta_t\ \nabla_w V_w(S_t).$$

- **Average reward estimate**: cập nhật $\hat{\rho} \leftarrow \hat{\rho} + \beta\ \delta_t$ (step-size β nhỏ hơn α hoặc cùng timescale chậm) để ước lượng average reward dưới policy hiện tại.
- **Actor update**:

$$\theta \leftarrow \theta + \eta\ \delta_t\ \nabla_\theta \log \pi_\theta(A_t|S_t).$$

- **Giải thích**: Khi R > ρ, δ>0 → tăng xác suất action dẫn đến reward cao hơn average; khi R < ρ, δ<0 → giảm xác suất. Qua thời gian, policy tối ưu reward rate.
- **Thách thức**: ước lượng ρ và V(s) phải ổn định; policy thay đổi làm phân phối ổn định mới, cần exploration ergodic. Cần đảm bảo step-size scheduling hợp lý theo lý thuyết stochastic approximation. opencourse.inf.ed.ac.uk arxiv.org .

5.3. Triển khai Softmax / Gaussian Policy

- **Discrete action**: softmax preferences h(s,a;θ), dùng ∇ log π dễ tính.
- **Continuous action**: Gaussian policy parameterized mean μ_θ(s) và có thể parameterize variance, dùng reparameterization trick hoặc score function gradient để cập nhật.
- **Numerical stability**: Khi tính softmax, trừ max preference trước exp; khi tính Gaussian, đảm bảo variance dương (exp hoặc softplus).
- **Entropy regularization**: Thêm term $+\lambda H(\pi_\theta(\cdot|s))$ vào objective để khuyến khích exploration bền vững.
- **Batching vs online**: Có thể cập nhật online theo từng bước hoặc dùng minibatch (giữ buffer trải nghiệm hoặc dùng multi-worker).

---

6. Thách thức và Lưu ý Khi Thay Đổi Policy Ảnh Hưởng Phân Phối Trạng Thái

- **Distribution shift**: Khi θ thay đổi, phân phối trạng thái s theo policy (discounted weighting hay steady-state) thay đổi; do đó gradient estimate ước lượng cũ có thể không phù hợp hoàn toàn. Actor-Critic online phải xử lý non-stationarity này qua sampling liên tục on-policy.
- **Ergodicity / sufficient exploration**: Cần policy duy trì stochasticity đủ để đảm bảo khám phá đầy đủ states; nếu policy trở nên quá deterministic quá sớm, không thu đủ dữ liệu để ước lượng critic và ρ chính xác. Thường dùng entropy bonus hoặc ε-greedy decay chậm. opencourse.inf.ed.ac.uk .
- **Learning rate annealing**: Step-sizes α, β, η cần decay hoặc scheduling để đảm bảo hội tụ gần điểm tối ưu cục bộ; nếu quá lớn, policy cập nhật quá nhanh, critic không theo kịp dẫn đến divergence; nếu quá nhỏ, học quá chậm.

- **Variance cao**: Gradient estimator có variance; dùng baseline (value) hoặc Generalized Advantage Estimation (GAE) để giảm variance; đối với average reward, dùng differential advantage tương tự.
- **Stationarity assumptions**: Một số lý thuyết average reward policy gradient giả định MDP cần ergodic; thay đổi policy cần đủ chậm để ổn định gần-steady-state mỗi giai đoạn. Trong practice, phải điều chỉnh tốc độ cập nhật để xấp xỉ điều kiện lý thuyết. arxiv.org .

---

7. Ví dụ minh họa

- **Episodic vs Continuing**: Trong môi trường không có terminal rõ (ví dụ robot điều khiển liên tục), dùng average reward actor-critic thay vì discounted actor-critic. Ví dụ robot di chuyển liên tục thu thập reward mỗi bước; objective là maximize reward rate (ví dụ năng lượng tiêu thụ/travel distance).
- **Grid World Continuing**: Nếu biến thể grid world không có terminal (agent đi mãi), dùng discounted với γ<1 hoặc average reward. Khi muốn policy ổn định vùng nào agent nên lui về recharge station đều đặn, average reward formulation có thể trực tiếp học chu kỳ tối ưu.
- **CartPole vô tận**: Thông thường dùng discounted với γ<1; nếu muốn maximize thời gian cân bằng trung bình vô hạn, có thể thử average reward actor-critic though hiếm dùng.
- **Learning loop**:
  1. Khởi θ, w, ρ.
  2. Tại mỗi bước: lấy S_t, sample A_t ~ π_θ; nhận R_{t+1}, S_{t+1}.
  3. Tính δ_t: nếu discounted: $R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$; nếu average: $R_{t+1} - \hat{\rho} + V_w(S_{t+1}) - V_w(S_t)$.
  4. Cập nhật critic w và ρ (nếu average), rồi cập nhật actor θ theo δ_t ∇ log π.
  5. Lặp vô hạn hoặc đến khi policy ổn định.
- **Theo dõi**: Với continuing, không rõ điểm dừng, thường theo dõi running average reward ước lượng và performance metric thực thi policy (ví dụ average reward measured trên window dài) để đánh giá cải thiện.

---

8. Tổng kết và Khuyến nghị

1. **Hiểu rõ objective**: Xác định task là episodic hay continuing, rồi chọn discounted return hay average reward cho phù hợp mục tiêu thực tế.
2. **Policy Gradient Theorem**: Chính là nền tảng cho cả discounted và average reward settings, cho phép tính gradient mà không cần model dynamics. Cần nắm công thức chung và cách ước lượng qua sampling. opencourse.inf.ed.ac.uk en.wikipedia.org .
3. **Actor-Critic cho Continuing**: Khi dùng discounted continuously, actor-critic tương tự episodic nhưng dùng truncated trajectories hoặc online update; khi dùng average reward, dùng differential TD error và estimate ρ song song.

4. **Thiết kế policy**: Discrete dùng softmax preferences; continuous dùng Gaussian hoặc các phân phối thích hợp. Đảm bảo khả vi để tính gradient log-prob.
5. **Critic / Baseline**: Estimate V(s) hoặc Q(s,a) để giảm variance, hoặc estimate differential value và average reward khi cần.
6. **Điều chỉnh exploration & learning rates**: Giữ stochasticity đủ lâu, tránh policy quá deterministic sớm; scheduling step-size để critic kịp theo actor. Entropy bonus hữu ích.
7. **Theo dõi convergence**: Với continuing, theo dõi estimate ρ(π) hoặc running average reward thực thi policy; chú ý non-stationarity khi policy liên tục thay đổi.