FMDPs $p(s', r \mid s, a)$ – opis środowiska

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Funkcja wartościująca stany $v_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$

Funkcja wartościująca akcje $q_\pi(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$

Równania Bellmana i Bellman Optimality Principle

$$v_\pi(s) = \sum_a q_\pi(s,a)\,\pi(a \mid s)$$

$$q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v_\pi(s')]$$

NOTE: zamień je to w. równań innych dla $v_\pi(s)$ lub $q_\pi(s,a)$.

$$v^*(s) = \max_\pi v_\pi(s) = \max_a q^*(s,a)$$

$$q^*(s,a) = \max_\pi q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v^*(s')]$$

Iterative Policy Evaluation

1. Initialize $v_\pi^{(0)}(s)$ s.t. $v_\pi^{(0)}(\text{terminal}) = 0$
2. $v_\pi^{(k+1)}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v_\pi^{(k)}(s')]$

Policy Iteration

1. Initialize $\pi_0(s)$ (deterministic policy)
2. Oblicz $v_{\pi_k}(s)$ korzystając z IPE
3. Popraw $\pi_{k+1}(s) = \arg\max_a q_{\pi_k}(s,a)$

Value Iteration

1. Initialize $v_0(s)$ s.t. $v_0(\text{terminal}) = 0$
2. $v_{k+1}(s) = \max_{a \in A} \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v_k(s')]$
3. zwróć politykę zachłanną.

Bandyci • zbiór akcji $a \in A$, $p(r \mid a)$

Funkcja wartościująca akcje

$$N_t(a) = \sum_{i=1}^{t-1} [a_i = a]$$

$$Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{t-1} r_i\,[a_i = a]$$

$\Rightarrow$ $N_{t+1}(a) = N_t(a) + [a_t = a]$, $Q_{t+1}(a) = Q_t(a) + \frac{[a_t = a]}{N_t(a)} \ldots$

Exponential recency-weighted average

$$Q_{t+1}(a) = Q_t(a) + \alpha\,[a_t = a]\,(r_t - Q_t(a))$$

- ε-Greedy a_t = rand(a) if rand() < ε else $\arg\max_a Q_t$
- UCB $a_t = \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right\}$
- Gradient approach $\pi(a) = \frac{\exp w_a}{\sum_{b \in A} \exp w_b}$

$$w_{t+1}(a) = w_t(a) + \alpha\,(r_t - \bar{r}_t)\,\big[[a_t = a] - \pi_t(a)\big]$$

- Thompson sampling

$$p_t(\theta_{a_t} \mid r_t) \propto p(r_t \mid \theta_{a_t})\,p_t(\theta_{a_t})$$

$$p_{t+1}(\theta_{a_t}) = p_t(\theta_{a_t} \mid r_t)$$

$$a_t = \arg\max_a \mathbb{E}[r \mid \underline{\theta_a}], \quad \underline{\theta_a} \sim p_t(\theta_a)$$

Bernoulli bandits

$$p(r \mid \theta_a) \equiv \mathrm{Ber}(\theta_a)(r) = \theta_a^r (1-\theta_a)^{1-r}$$

$$p_1(\theta_a) \equiv \mathrm{Beta}(\alpha_1, \beta_1) \propto \theta_a^{\alpha_1 - 1} \cdot (1-\theta_a)^{\beta_1 - 1}$$

$$p_2(\theta_a) \equiv \theta_a^{r_1} \cdot (1-\theta_a)^{1-r_1} \cdot \theta_a^{\alpha_1 - 1} \cdot (1-\theta_a)^{\beta_1 - 1} =$$

$$= \mathrm{Beta}(\alpha_1 + r_1, \beta_1 + 1 - r_1) = \mathrm{Beta}(\alpha_2, \beta_2)$$

MONTE CARLO on-policy – ewaluujemy i ulepszamy tę samą politykę wg. której wybieramy akcje

Importance sampling

Jeśli używamy off-policy to mamy 2 polityki: π – optymalna, b – behawioralna / eksplorująca. Ponieważ prawdopodobieństwa wystąpienia danego epizodu są różne dla π i b więc musimy to naprawić

$$\alpha \leftarrow \left(\prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)} \right) \cdot \alpha$$

TD $\delta_t = R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$

SARSA (on-policy TD(0) control)

for each episode:
- Take action A using $Q(S,\cdot)$ (e.g. ε-greedy)
- Take action A' using $Q(S',\cdot)$ (e.g. ε-greedy)
- $Q(S,A) \leftarrow Q(S,A) + \alpha\,[R + \gamma Q(S',A') - Q(S,A)]$

Q-Learning (off-policy TD(0) control)

for each episode:
- Take action A using $Q(S,\cdot)$ (e.g. ε-greedy)
- $Q(S,A) \leftarrow Q(S,A) + \alpha\,[R + \gamma \max_a Q(S',a) - Q(S,A)]$

SARSA znajduje optymalną politykę wśród polityk eksplorujących. Jest więc lepszym wyborem, gdy mamy ciągłą eksplorację (przykład z klifem: SARSA znajduje ścieżkę nieco gorszą, ale znacznie bezpieczniejszą).

Q-Learning znajduje optymalną politykę, ale w czasie treningu korzysta z polityki behawioralnej (która może spadać z klifu). Jest więc lepszym wyborem, gdy mamy 2 fazy: eksperymentów i ewaluacji i eksperymenty nie są kosztowne.

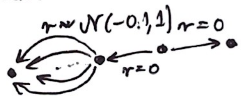
Expected SARSA

$$Q(S,A) \leftarrow Q(S,A) + \alpha\Big[R + \gamma \sum_a \pi(a \mid S')\,Q(S',a) - Q(S,A)\Big]$$

(Jest droższa kosztowna obliczeniowo)

Maximisation Bias

Q-Learning (i trochę Exp SARSA) wymienia przypadek możliwości nadmiernej optymistyczne bo bierze max.



Dla tego przykładu Q-Learning będzie długo chodził w lewo bo maksymalna wartość wartościowana w następnym kroku może być duża (mamy wzorce o dużej wariancji). Algorytm będzie powielał dodatnie wartości. Źródłem max. bias jest to, że oceniamy nasz wybór na ocenie to co było do tego wyboru użyte (wybieramy akcję na podstawie Q, a potem oceniamy ją Q)

Double-Q-Learning

Mamy dwie funkcje Q_1, Q_2, które aktualizujemy losowo w każdym kroku

$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big[R + \gamma Q_2\big(S', \arg\max_a Q_1(S',a)\big) \ldots$$

W przypadku problemów nie-epizodycznych (czy. continual learning) discount factor γ nam przeszkadza. Wtedy mamy wprowadzić zwrot wyrównany $\tilde{G}_t = R_{t+1} - \bar{R}_\pi + R_{t+2} - \bar{R}_\pi + \dots$

$\delta_t = R - \bar{R} + q(S', A') - q(S, A)$

Deadly Triad Metody off-policy w połączeniu z metodami semi-gradientowymi są niestabilne.

Złożenie 3 czynników, które osobno są porządne, ale razem powodują niestabilność treningu.

I. Chcemy aproksymować (np. neuronowo), aby generalizować i radzić sobie z ogromnymi przestrzeniami stanów / akcji.

II. Chcemy bootstrap, bo w przypadku czystego MC wariancja jest duża, aktualizacje rzadkie i nie ma generalizacji (przecieku wiedzy) między epizodami.

III. Chcemy się uczyć off-policy, bo chcemy mieć z jednej strony polityką optymalną, a z drugiej polityką, która umie eksplorować.

Przykład ze współdzielonymi stanami. $\omega \longrightarrow 2\omega$

Podbicie wartościowania jednego stanu podbija wartościowanie innego stanu przez (1) generalizację NN, (2) bootstrap, co jeszcze bardziej zwiększa TD error i wagi nam rosną do ∞. On-policy nas ratuje bo z drugiego stanu trzeba wyjść co obniży jego wartościowanie.

METODY N-KROKOWE

W przypadku n-krokowej SARSY (on-policy)

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left[R_1 + \gamma R_2 + \dots + \gamma^{n-1} R_n + \gamma^n Q(S_{n+1}, A_{n+1}) - Q(S,A) \right]$$

W przypadku off-policy pamiętamy o importance sampling.

Metody n-krokowe umożliwiają szybsze przeciekanie informacji między stanami, ale zwiększenie zbytnio wartości n obniża performance (zbliżamy się wtedy do MC ze wszystkimi problemami: rzadkie update'y, duża wariancja).

METODY APROKSYMUJĄCE

Do tej pory używaliśmy algorytmów tablicowych które wymagały przechowywania wartości Q dla wszystkich spotkanych par s, a. Chcemy teraz aproksymacji

$$\hat{q}(s, a; \theta) \approx q_\pi(s,a)$$

Jak coś takiego uczyć i to online?

Podejście semi-gradientowe

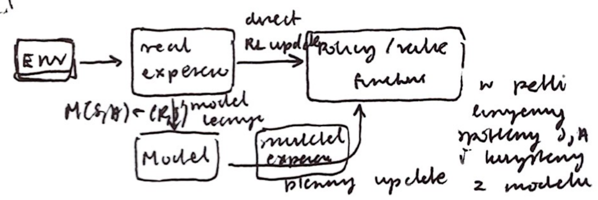
$$\theta_{t+1} = \theta_t + \alpha \left[G_t - \hat{q}(S_t, A_t; \theta_t) \right] \nabla_\theta \hat{q}(S_t, A_t; \theta_t)$$

Jeśli G_t to zwrot z MC to wszystko jest ok (brak obciążenia, być może spora wariancja). Wielkim problemem jest dla TD, ale zaskakująco działa, np. semi-gradient SARSA

$$\theta_{t+1} = \theta_t + \alpha \left[R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}; \theta_t) - \hat{q}(S_t, A_t; \theta_t) \right] \nabla_\theta \hat{q}$$

PLANNING

Dyna-Q



Dyna-Q+ nagroda zwiększona na etapie planowania wynosi

$r_t = r + \kappa\sqrt{\tau}$ gdzie τ to czas od ostatniego odwiedzenia przejścia między stanami.

POLICY GRADIENT

Zamiast korzystać z funkcji wartościujących v / q możemy próbować bezpośrednio poprawiać optymalnej polityki $\pi(s; \theta)$ korzystając z SGD

$$\theta_{t+1} = \theta_t + \alpha \frac{\partial J}{\partial \theta}, \quad J(\theta) = v_{\pi_\theta}(s_0)$$

dla pewnego meta-stanu początkowego s_0.

Tw. o gradiencie polityki

$$\frac{\partial J}{\partial \theta} \propto \mathbb{E}\left[G_t \nabla_\theta \ln \pi(A_t | S_t; \theta) \right]$$

REINFORCE (Monte Carlo Policy Gradient)

1. Choose actions according to $\pi(a | S_t; \theta_t)$
2. $\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \ln \pi(A_t | S_t; \theta_t)$

REINFORCE w/ baseline

Dodatkowo możemy wykorzystać trik z baselinem

$$\theta_{t+1} = \theta_t + \alpha (G_t - b(S_t)) \nabla_\theta \ln \pi(A_t | S_t; \theta_t)$$

Naturalnym wyborem jest $b(S_t) = \hat{v}(S_t; \phi)$

$$\phi_{t+1} = \phi_t + \alpha \left[G_t - \hat{v}(S_t; \phi_t) \right] \nabla_\phi \hat{v}(S_t; \phi_t)$$
$$\theta_{t+1} = \theta_t + \alpha \left[G_t - \hat{v}(S_t; \phi_t) \right] \nabla_\theta \ln \pi(A_t | S_t; \theta_t)$$

Zwrot G_t jest tutaj liczony MC, ale skoro mamy już estymację $\hat{v}(S_t; \phi_t)$ to szkoda z niej nie skorzystać do bootstrapingu

$$\phi_{t+1} = \phi_t + \alpha \left[R_{t+1} + \gamma \hat{v}(S_{t+1}; \phi_t) - \hat{v}(S_t; \phi_t) \right] \nabla_\phi \hat{v}(S_t; \phi_t)$$
$$\theta_{t+1} = \theta_t + \alpha \left[R_{t+1} + \gamma \hat{v}(S_{t+1}; \phi_t) - \hat{v}(S_t; \phi_t) \right] \nabla_\theta \ln \pi(A_t | S_t; \theta_t)$$

Dyna-Q / TD(λ) dla λ≫1 działa w szczególności dobrze w problemach, w których tylko wyróżnione stany dają duże nagrody, a pozostałe tylko do nich prowadzą (np. maze search). Między istotne jest „wlekanie się wiedzy".