Uczenie przez Wzmacnianie

Bartosz Hanc

Skondensowany przegląd podstawowych metod uczenia przez wzmacnianie: problem wielorękiego bandyty i strategie eksploracji, skończone procesy decyzyjne Markowa, programowanie dynamiczne, algorytmy Monte Carlo i TD (SARSA, Q-Learning, Double Q-Learning, metody n-krokowe), planowanie w architekturze Dyna-Q, aproksymacja funkcyjna i deadly triad oraz metody gradientu polityki (REINFORCE, Actor–Critic).

Spis treści

§ 1. Bandyci 2
1.1. Strategie eksploracji 2
1.2. Thompson Sampling 3
§ 2. Skończone procesy decyzyjne Markowa (FMDP) 4
2.1. Równania Bellmana i zasada optymalności 5
2.2. Programowanie dynamiczne 5
§ 3. Metody Monte Carlo i TD 6
3.1. Monte Carlo 6
3.2. Temporal Difference (TD) 7
3.3. Expected SARSA 8
3.4. Błąd maksymalizacji i Double Q-Learning 8
3.5. Metody n-krokowe 8
§ 4. Planowanie (Dyna-Q) 9
§ 5. Metody aproksymujące i Deadly Triad 10
5.1. Deadly Triad 10
§ 6. Metody gradientu polityki (Policy Gradient) 11
6.1. REINFORCE z baseline i Actor–Critic 11

§ 1. Bandyci

Problem wielorękiego bandyty to najprostszy model uczenia przez wzmacnianie, w którym agent podejmuje decyzje w jednym stanie. Rozważamy zbiór akcji $a \in A$ o nieznanych rozkładach nagród $p(r \mid a)$. Liczbę wyborów akcji a oraz estymowaną funkcję wartościującą akcje do chwili t definiujemy jako

$$\begin{aligned} N_t(a) &= \sum_{i=1}^{t-1} [a_i = a] \quad , \\ Q_t(a) &= \frac{1}{N_t(a)} \sum_{i=1}^{t-1} r_i [a_i = a] \quad , \end{aligned} \tag{1.1}$$

co prowadzi do inkrementalnych reguł aktualizacji bez konieczności przechowywania całej historii nagród:

$$\begin{aligned} N_{t+1}(a) &= N_t(a) + [a_t = a] \quad , \\ Q_{t+1}(a) &= Q_t(a) + \frac{[a_t = a]}{N_{t+1}(a)} (r_t - Q_t(a)) \quad . \end{aligned} \tag{1.2}$$

W środowiskach niestacjonarnych, gdzie rozkłady nagród zmieniają się w czasie, zamiast malejącego kroku $\frac{1}{N_{t+1}}(a)$ stosuje się stały krok $\alpha \in (0, 1]$, co daje wykładniczą średnią ruchomą (exponential recency-weighted average):

$$Q_{t+1}(a) = Q_t(a) + \alpha [a_t = a] (r_t - Q_t(a)) \quad . \tag{1.3}$$

1.1. Strategie eksploracji

Kluczowym wyzwaniem w problemie bandytów jest balans między **eksploracją** (zbieraniem informacji o niepewnych akcjach) a **eksploatacją** (wyborem akcji o najwyższej dotychczasowej estymacie):

- **ε-greedy**: z prawdopodobieństwem ε wybieramy losową akcję $a \sim \mathrm{Unif}(A)$, a z prawdopodobieństwem $1-\varepsilon$ akcję zachłanną $a_t = \arg\max_a Q_t(a)$.
- **UCB (Upper Confidence Bound)**: faworyzujemy akcje o wysokiej estymacie lub dużej niepewności (małym $N_t(a)$),

$$a_t = \arg\max_a \left\{ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right\} \quad . \tag{1.4}$$

- **Podejście gradientowe**: zamiast estymować wartości $Q_t(a)$, uczymy numeryczne preferencje $w_t(a)$ i definiujemy politykę przez funkcję softmax,

$$\begin{aligned} \pi_t(a) &= \frac{\exp w_t(a)}{\sum_{b \in A} \exp w_t(b)} \quad , \\ w_{t+1}(a) &= w_t(a) + \alpha(r_t - \overline{r}_t)([a_t = a] - \pi_t(a)) \quad , \end{aligned} \tag{1.5}$$

  gdzie $\overline{r}_t$ jest średnią dotychczasowych nagród pełniącą rolę punktu odniesienia (baseline).

1.2. Thompson Sampling

W podejściu Bayesowskim utrzymujemy rozkład a posteriori nad parametrami nagród θ_{a_t} i aktualizujemy go po obserwacji r_t:

$$\begin{aligned} p_t\big(\theta_{a_t} \mid r_t\big) &\propto p\big(r_t \mid \theta_{a_t}\big) p_t\big(\theta_{a_t}\big) \quad , \\ p_{t+1}\big(\theta_{a_t}\big) &= p_t\big(\theta_{a_t} \mid r_t\big) \quad . \end{aligned} \tag{1.6}$$

W każdym kroku próbkujemy parametry $\underline{\theta}_a \sim p_t(\theta_a)$ dla wszystkich akcji i wybieramy akcję maksymalizującą oczekiwaną nagrodę,

$$a_t = \arg\max_a \mathbb{E}\big[r \mid \underline{\theta}_a\big] \quad . \tag{1.7}$$

Dla **bandytów Bernoulliego**, gdzie nagrody są binarne i $p(r \mid \theta_a) = \mathrm{Ber}(\theta_a)(r) = \theta_a^r(1-\theta_a)^{1-r}$, prior sprzężony stanowi rozkład Beta:

$$p_1(\theta_a) = \mathrm{Beta}(\alpha_1, \beta_1) \propto \theta_a^{\alpha_1 - 1}(1-\theta_a)^{\beta_1 - 1} \quad . \tag{1.8}$$

Dzięki sprzężeniu po obserwacji $r_1 \in \{0, 1\}$ rozkład a posteriori zachowuje tę samą postać:

$$\begin{aligned} p_2(\theta_a) &\propto \theta_a^{r_1}(1-\theta_a)^{1-r_1}\theta_a^{\alpha_1-1}(1-\theta_a)^{\beta_1-1} \\ &= \mathrm{Beta}(\alpha_1 + r_1, \beta_1 + 1 - r_1) \quad . \end{aligned} \tag{1.9}$$

§ 2. Skończone procesy decyzyjne Markowa (FMDP)

W ogólnych zadaniach sekwencyjnych decyzje agenta wpływają nie tylko na natychmiastową nagrodę, lecz także na kolejne stany środowiska. Dynamika środowiska jest opisana rozkładem przejść i nagród $p(s', r \mid s, a)$ (własność Markowa: przyszłość zależy wyłącznie od obecnego stanu i akcji). Zdyskontowany zwrot od chwili t definiujemy jako

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad , \tag{2.1}$$

gdzie $\gamma \in [0, 1)$ jest współczynnikiem dyskontującym, który określa wagę przyszłych nagród i zapewnia zbieżność sumy. Funkcje wartościujące stany oraz pary stan–akcja dla ustalonej polityki $\pi(a \mid s)$ mają postać

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \quad , \\ q_\pi(s, a) &= \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \quad . \end{aligned} \tag{2.2}$$

2.1. Równania Bellmana i zasada optymalności

Dla ustalonej polityki π zachodzą równania Bellmana, które rekurencyjnie wiążą wartość stanu z wartościami jego następników (tworząc zawsze układ równań liniowych dla $v_\pi(s)$ lub $q_\pi(s,a)$):

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a \mid s) q_\pi(s,a) \quad , \\ q_\pi(s,a) &= \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \quad . \end{aligned} \tag{2.3}$$

Optymalne funkcje wartościujące v^* i q^* spełniają nieliniowe równania optymalności Bellmana:

$$\begin{aligned} v^*(s) &= \max_\pi v_\pi(s) = \max_a q^*(s,a) \quad , \\ q^*(s,a) &= \max_\pi q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)[r + \gamma v^*(s')] \quad . \end{aligned} \tag{2.4}$$

2.2. Programowanie dynamiczne

Gdy znamy pełny model środowiska $p(s', r \mid s, a)$, możemy rozwiązać równania Bellmana iteracyjnie za pomocą metod programowania dynamicznego.

Algorytm: Iterative Policy Evaluation (IPE)

Zainicjalizuj $v_\pi^{(0)}(s)$ tak, aby $v_\pi^{(0)}(\text{terminal}) = 0$.
Powtarzaj aż do zbieżności:

$$v_\pi^{(k+1)}(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi^{(k)}(s')\Big] \quad . \tag{2.5}$$

Algorytm: Policy Iteration

Zainicjalizuj deterministyczną politykę $\pi_0(s)$.
Powtarzaj aż do stabilizacji polityki:

1. Oblicz $v_{\pi_k}(s)$ korzystając z IPE.
2. Popraw politykę zachłannie:

$$\pi_{k+1}(s) = \arg\max_a q_{\pi_k}(s,a) \quad . \tag{2.6}$$

Algorytm: Value Iteration

Zainicjalizuj $v_0(s)$ tak, aby $v_0(\text{terminal}) = 0$.
Powtarzaj aż do zbieżności:

$$v_{k+1}(s) = \max_{a \in A} \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')] \quad . \tag{2.7}$$

zwróć politykę zachłanną względem v_k

§ 3. Metody Monte Carlo i TD

W praktyce zazwyczaj nie znamy modelu $p(s', r \mid s, a)$ i musimy uczyć się bezpośrednio z próbek doświadczenia.

3.1. Monte Carlo

Metody Monte Carlo estymują wartości na podstawie pełnych epizodów (bez bootstrappingu). W metodach **on-policy** ewaluujemy i ulepszamy tę samą politykę, według której wybieramy akcje. Jeśli używamy podejścia **off-policy**, mamy dwie polityki: π — politykę docelową (optymalną) oraz b — politykę behawioralną (eksplorującą). Ponieważ prawdopodobieństwa wystąpienia danego epizodu

są różne dla π i b, stosujemy **importance sampling**, ważąc krok aktualizacji iloczynem ilorazów wiarygodności:

$$\alpha \leftarrow \left(\prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)} \right) \cdot \alpha \quad . \tag{3.1}$$

3.2. Temporal Difference (TD)

Metody TD łączą uczenie z próbek (jak MC) z bootstrappingiem (jak DP), aktualizując estymaty po każdym kroku bez czekania na koniec epizodu. Błąd TD definiujemy jako

$$\delta_t = R_{t+1} + \gamma v(S_{t+1}) - v(S_t) \quad . \tag{3.2}$$

Algorytm: SARSA (on-policy TD(0) control)

Dla każdego epizodu wybieraj akcję A według $Q(S, \cdot)$ oraz akcję A' według $Q(S', \cdot)$ (np. ε-greedy) i aktualizuj:

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)] \quad . \tag{3.3}$$

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

Dla każdego epizodu wybieraj akcję A według $Q(S, \cdot)$ (np. ε--greedy) i aktualizuj:

$$\begin{aligned} Q(S, A) \leftarrow & \, Q(S, A) \\ & + \alpha \Big[R + \gamma \max_a Q(S', a) - Q(S, A) \Big] \quad . \end{aligned} \tag{3.4}$$

SARSA znajduje optymalną politykę wśród polityk eksplorujących. Jest więc lepszym wyborem, gdy mamy ciągłą eksplorację (np. problem z klifem: SARSA znajduje ścieżkę nieco dłuższą, ale znacznie bezpieczniejszą). Q-Learning znajduje politykę optymalną, ale w czasie treningu korzysta z polityki behawioralnej (która może

spaść z klifu) — jest więc lepszym wyborem, gdy mamy dwie fazy (eksperymentu i ewaluacji) i eksperymenty nie są kosztowne.

3.3. Expected SARSA

Wariantem zmniejszającym wariancję aktualizacji (kosztem większej złożoności obliczeniowej w pojedynczym kroku) jest **Expected SARSA**, która zamiast jednej wylosowanej akcji A' używa wartości oczekiwanej po polityce:

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \sum_a \pi(a \mid S') Q(S',a) - Q(S,A)\right]. \tag{3.5}$$

3.4. Błąd maksymalizacji i Double Q-Learning

Q-Learning (i częściowo Expected SARSA) wycenia wartości nadmiernie optymistycznie (**maximisation bias**), ponieważ używa operatora max. Jeśli nagrody mają dużą wariancję, algorytm powiela dodatnie wahania — źródłem błędu jest to, że oceniamy ten sam wybór na podstawie tej samej funkcji Q, której użyliśmy do wyboru akcji.

Rozwiązaniem jest **Double Q-Learning**: utrzymujemy dwie niezależne funkcje Q_1, Q_2, które aktualizujemy losowo w każdym kroku, rozdzielając wybór akcji od jej oceny:

$$Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big[R + \gamma Q_2\Big(S', \arg\max_a Q_1(S',a)\Big) - Q_1(S,A)\Big]. \tag{3.6}$$

3.5. Metody n-krokowe

Metody n-krokowe uogólniają TD(0) ($n = 1$) oraz Monte Carlo ($n \to \infty$), korzystając z n rzeczywistych nagród przed wykonaniem

bootstrappingu. Dla n-krokowej SARSY (on-policy) aktualizacja ma postać

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[\sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n Q(S_{t+n}, A_{t+n}) - Q(S_t, A_t) \right]. \tag{3.7}$$

W przypadku off-policy należy uwzględnić ważenie importance sampling. Metody n-krokowe umożliwiają szybsze przekazywanie informacji między stanami, ale zbliżanie $n \to \infty$ powoduje rzadsze aktualizacje i dużą wariancję.

§ 4. Planowanie (Dyna-Q)

W architekturze **Dyna-Q** łączymy uczenie z rzeczywistego doświadczenia (**direct RL update** funkcji wartości/polityki) z uczeniem modelu środowiska $M(S, A) \to (R, S')$. W pętli planowania (**planning updates**) losujemy przeszłe pary (S, A) i symulujemy przejścia korzystając z modelu $M(S, A)$, co pozwala znacznie efektywniej wykorzystać zebrane próbki.

Wariant **Dyna-Q+** nagradza powrót do dawno nieodwiedzanych przejść na etapie planowania (co pomaga w środowiskach niestacjonarnych):

$$r_+ = r + \kappa\sqrt{\tau} \quad , \tag{4.1}$$

gdzie τ oznacza czas od ostatniego odwiedzenia danego przejścia między stanami. Ponadto Dyna-Q oraz TD(λ) dla $\lambda \gg 1$ sprawdzają się szczególnie dobrze w problemach, w których tylko wyróżnione stany dają duże nagrody, a pozostałe do nich prowadzą (np. maze search), ponieważ istotne jest wówczas „cofanie się wstecz".

§ 5. Metody aproksymujące i Deadly Triad

Dotychczasowe algorytmy tabelaryczne wymagały przechowywania wartości dla wszystkich par (s, a), co jest niemożliwe w dużych lub ciągłych przestrzeniach. Stosujemy wówczas aproksymację funkcyjną parametryzowaną wektorem $\boldsymbol{\theta}$:

$$\hat{q}(s, a; \boldsymbol{\theta}) \approx q_\pi(s, a) \quad . \tag{5.1}$$

W podejściu **semi-gradientowym** parametry aktualizujemy w kierunku celu G_t (ignorując zależność samego celu od $\boldsymbol{\theta}$ przy liczeniu gradientu):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[G_t - \hat{q}(S_t, A_t; \boldsymbol{\theta}_t)]\nabla_{\boldsymbol{\theta}} \hat{q}(S_t, A_t; \boldsymbol{\theta}_t) \quad . \tag{5.2}$$

Jeśli G_t jest zwrotem Monte Carlo, estymator jest nieobciążony (choć może mieć dużą wariancję). Dla celu TD otrzymujemy np. **semi-gradient SARSA**:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[&R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}; \boldsymbol{\theta}_t) - \hat{q}(S_t, A_t; \boldsymbol{\theta}_t)] \\ &\cdot \nabla_{\boldsymbol{\theta}} \hat{q}(S_t, A_t; \boldsymbol{\theta}_t) \quad . \end{aligned} \tag{5.3}$$

5.1. Deadly Triad

Metody off-policy w połączeniu z metodami semi-gradientowymi są niestabilne. Istnieją trzy czynniki, które osobno są pożądane, ale razem powodują niestabilność treningu:

1. **Aproksymacja funkcyjna** (np. sieci neuronowe) — aby generalizować i radzić sobie z ogromnymi przestrzeniami stanów/akcji.
2. **Bootstrapping** — w Monte Carlo wariancja jest duża, aktualizacje rzadkie i nie ma transferu wiedzy w trakcie epizodów.
3. **Uczenie off-policy** — aby osobno utrzymywać politykę optymalną i politykę eksplorującą.

Podbicie wartości jednego stanu powoduje (przez generalizację oraz bootstrapping z błędem TD), że wartości innych stanów mogą rozbiegać do nieskończoności. W uczeniu on-policy do tego nie dochodzi, ponieważ trajektorie opuszczają przeszacowane stany i wyrównują ich wartości.

§ 6. Metody gradientu polityki (Policy Gradient)

Zamiast wyprowadzać politykę pośrednio z funkcji wartościujących v, q, możemy bezpośrednio parametryzować i optymalizować politykę $\pi(a \mid s; \boldsymbol{\theta})$ metodą wzrostu gradientowego:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \frac{\partial J}{\partial \boldsymbol{\theta}} \quad , \quad J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(s_0) \quad , \tag{6.1}$$

dla pewnego meta-stanu początkowego s_0. Z **twierdzenia o gradiencie polityki** mamy analityczną postać gradientu niewymagającą znajomości dynamiki środowiska:

$$\frac{\partial J}{\partial \boldsymbol{\theta}} \propto \mathbb{E}[G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t)] \quad . \tag{6.2}$$

Algorytm: REINFORCE (Monte Carlo Policy Gradient)

---

1. Wygeneruj epizod zgodnie z polityką $\pi(a \mid s; \boldsymbol{\theta}_t)$.
2. Dla każdego kroku t zaktualizuj parametry polityki:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t) \quad . \tag{6.3}$$

6.1. REINFORCE z baseline i Actor–Critic

Wariancję estymatora gradientu redukujemy przez odjęcie funkcji bazowej (baseline) $b(S_t)$ niezależnej od wybranej akcji (co nie wprowadza obciążenia):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha (G_t - b(S_t)) \nabla_{\boldsymbol{\theta}} \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t) \quad . \tag{6.4}$$

Naturalnym wyborem jest $b(S_t) = \hat{v}(S_t; \boldsymbol{\varphi})$, gdzie krytyk $\boldsymbol{\varphi}$ uczy się równolegle z aktorem $\boldsymbol{\theta}$:

$$\begin{aligned} \boldsymbol{\varphi}_{t+1} &= \boldsymbol{\varphi}_t + \alpha[G_t - \hat{v}(S_t; \boldsymbol{\varphi}_t)]\nabla_{\boldsymbol{\varphi}} \hat{v}(S_t; \boldsymbol{\varphi}_t) \quad , \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha[G_t - \hat{v}(S_t; \boldsymbol{\varphi}_t)]\nabla_{\boldsymbol{\theta}} \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t) \quad . \end{aligned} \tag{6.5}$$

Ponieważ mamy już estymację $\hat{v}(S_t; \boldsymbol{\varphi}_t)$, możemy zastąpić zwrot Monte Carlo G_t jednokrokowym celem TD (bootstrapping), łącząc zalety metod wartościujących i politykowych w architekturze **Actor–Critic**:

$$\begin{aligned} \boldsymbol{\varphi}_{t+1} &= \boldsymbol{\varphi}_t + \alpha \delta_t \nabla_{\boldsymbol{\varphi}} \hat{v}(S_t; \boldsymbol{\varphi}_t) \quad , \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha \delta_t \nabla_{\boldsymbol{\theta}} \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t) \quad , \end{aligned} \tag{6.6}$$

gdzie $\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}; \boldsymbol{\varphi}_t) - \hat{v}(S_t; \boldsymbol{\varphi}_t)$.

W zadaniach nieepizodycznych (**continual learning**) klasyczny zwrot G_t traci sens z powodu dyskontowania; wprowadzamy wówczas zwrot różnicowy względem średniej nagrody $\overline{R}$:

$$\begin{aligned} G_t &= R_{t+1} - \overline{R} + R_{t+2} - \overline{R} + \cdots \quad , \\ \delta_t &= R - \overline{R} + q(S', A') - q(S, A) \quad . \end{aligned} \tag{6.7}$$