# polish

# Uczenie maszynowe:
# *wykład 14*

Paweł Cichosz

polish

# Strategie zachłanne

Względem $V^\pi$:

$$\pi'(x) = \arg\max_a \left[ R(x,a) + \gamma \sum_y P_{xy}(a) V^\pi(y) \right]$$

(lepsza niż $\pi$ albo obie optymalne)

Względem $Q^\pi$:

$$\pi'(x) = \arg\max_a Q^\pi(x,a)$$

(lepsza niż $\pi$ albo obie optymalne)

Strategie optymalne: zachłanne względem własnych funkcji wartości i wartości akcji ($V^*$, $Q^*$).

# Iteracja strategii

1. Strategia $\pi_0$ dowolna.
2. Dla $k = 0, 1, \ldots$:
   - funkcja wartości $V^{\pi_k}$ (lub $Q^{\pi_k}$) wyznaczana przez iteracyjne zastosowanie równania Bellmana dla każdego stanu, np.:

$$V_{i+1}^{\pi_k}(x) := R(x, \pi_k(x)) + \gamma \sum_y P_{xy}(\pi_k(x)) V_i^{\pi_k}(y)$$

   - strategia $\pi_{k+1}$ wyznaczana jako zachłanna względem $V^{\pi_k}$ (lub $Q^{\pi_k}$).
3. Stop jeśli $\pi_{k+1}$ nie jest lepsza niż $\pi_k$.

# Iteracja wartości

1. Optymalna funkcja wartości $V^*$ (lub $Q^*$) wyznaczana przez iteracyjne zastosowanie równania Bellmana dla każdego stanu, np.:

$$V^*_{k+1} := \max_a \left[ R(x,a) + \gamma \sum_y P_{xy}(a) V^*_k(y) \right]$$

lub

$$Q^*_{k+1}(x,a) := R(x,a) + \gamma \sum_y P_{xy}(a) \max_{a'} Q^*_k(y,a')$$

2. Strategia $\pi^*$ wyznaczana jako zachłanna względem $V^*$ (lub $Q^*$).

# Uczenie się ze wzmocnieniem a programowanie dynamiczne

- Uczenie się ze wzmocnieniem nie wymaga znajomości środowiska (prawdopodobieństw przejść i wartości oczekiwanych nagród).
- Uczenie się ze wzmocnieniem jest realizowane w trakcie wykonywania zadania i umożliwia ciągłą poprawę.
- Uczenie się ze wzmocnieniem umożliwia uzyskanie optymalnego zachowania bez konieczności wyznaczania pełnej optymalnej strategii.
- Uczenie się ze wzmocnieniem można wyposażyć w mechanizmy generalizacji dla podobnych stanów.

polish

# Algorytm Q-Learning

Funkcja $Q$: przyrostowo modyfikowane wartości akcji względem strategii optymalnej.

Aktualizacja $Q$ w kroku $t$:

$$Q_{t+1}(x_t, a_t) := (1 - \beta)Q_t(x_t, a_t) + \beta\big(r_t + \gamma \max_a Q_t(x_{t+1}, a)\big)$$

gdzie $0 < \beta < 1$ – rozmiar kroku.

Interpretacja: symulacja Monte-Carlo równania Bellmana dla $Q^*$:

- zamiast nieznanej wartości oczekiwanej $R(x_t, a_t)$ faktyczna wartość $r_t$,
- zamiast uwzględniania wszystkich możliwych stanów następnych o nieznanych prawdopodobieństwach przejść $P_{x_t y}(a_t)$ uwzględniany jeden faktyczny stan następny $x_{t+1}$.

# Algorytm Q-Learning

Tablicowa reprezentacja $Q$: przechowywana i aktualizowana tablica wartości $Q(x, a)$ dla wszystkich $x \in X$ i $a \in A$.

Zbieżność: $Q$ zbiega do $Q^*$ jeśli:

- do kolejnych aktualizacji dla każdej pary $(x, a)$ stosowany malejący ciąg rozmiarów kroku $\beta_{x,a}(i)$:

$$\sum_{i=1}^{\infty} \frac{1}{\beta_{x,a}(i)} = \infty$$

$$\sum_{i=1}^{\infty} \frac{1}{\beta_{x,a}^2(i)} < \infty$$

- w każdym stanie prawdopodobieństwo wyboru każdej akcji zawsze pozostaje niezerowe.

# Wybór akcji

Eksploracja kontra eksploatacja: do poprawy jakości działania wymagany wybór akcji zbliżony do zachłannego, do poprawnego wyznaczania $Q$ wymagane wykonywanie również akcji, które dotychczas wydają się słabe.

Strategia Boltzmanna (soft-max): akcja $a$ wybierana w stanie $x$ z prawdopodobieństwem:

$$\pi(x,a) = \frac{\exp(Q(x,a)/T)}{\sum_{a'} \exp(Q(x,a')/T)}$$

gdzie $T > 0$ reguluje stopień losowości.

Strategia $\epsilon$-zachłanna: z prawdopodobieństwem $1-\epsilon$ wybierana akcja zachłanna, z prawdopodobieństwem $\epsilon$ akcja wybierana jednostajnie losowo.

# Reprezentacja $Q$

Ograniczenia reprezentacji tablicowej: brak możliwości użycia dla ciągłych przestrzeni stanów, wolne uczenie się dla dużych przestrzeni stanów, brak generalizacji dla podobnych stanów.

Aproksymacja funkcji: $Q$ reprezentowana przez generalizujący aproksymator – może pokonać ograniczenia reprezentacji tablicowej, chociaż zazwyczaj bez formalnej gwarancji zbieżności, np.:

dyskretyzacja: dyskretyzacja każdego elementu wektora stanu i reprezentacja tablicowa dla zdyskretyzowanych stanów,

aproksymacja liniowa:

$$Q(x,a) = \sum_{i=1}^{n+1} w_{a,i}\phi_i(x)$$

$$w_{a_t,i} := w_{a_t,i} + \beta\big(r_t + \gamma \max_a Q_t(x_{t+1},a) - Q_t(x_t,a_t)\big)\phi_i(x_t)$$

gdzie $w_{a,1}, w_{a,2}, \ldots, w_{a,n+1}$ – parametry aproksymatora dla $Q(,a)$, $\phi_1(x), \ldots, \phi_n(x)$ – elementy wektora reprezentującego stan $x$, $\phi_{n+1}(x) \equiv 1$,

sieć neuronowa: uniwersalny aproksymator nieliniowy (poza zakresem wykładu).

# Algorytm Sarsa

*State action reward state action*

Funkcja $Q$: przyrostowo modyfikowane wartości akcji względem aktualnej strategii (zachłannej względem $Q$).

Aktualizacja $Q$ w kroku $t$:

$$Q_{t+1}(x_t, a_t) := (1 - \beta)Q_t(x_t, a_t) + \beta\big(r_t + \gamma Q_t(x_{t+1}, a_{t+1})\big)$$

(wymaga opóźnienia aktualizacji do czasu wyboru kolejnej akcji)

Wybór akcji: w przybliżeniu zachłanny względem $Q$.

Algorytm zależny od strategii: wymaga posługiwania się strategią reprezentowaną przez $Q$.

# Podsumowanie uczenia się ze wzmocnieniem

- Obszar aktywnych badań, lecz także udanych zastosowań.
- Słabe wymagania na informację trenującą.
- Potencjalnie łatwe do użycia w różnorodnych zadaniach z zakresu sterowania, podejmowania decyzji i optymalizacji.
- Szczególnie popularne w połączeniu z sieciami neuronowymi do reprezentacji funkcji.
- Główne ograniczenia: nie zawsze zadowalająca szybkość uczenia się, nie zawsze spełnione założenie o własności Markowa, problemu ze zbieżnością i stabilnością w połączeniu z aproksymatorami funkcji.