Uczenie ze wzmocnieniem

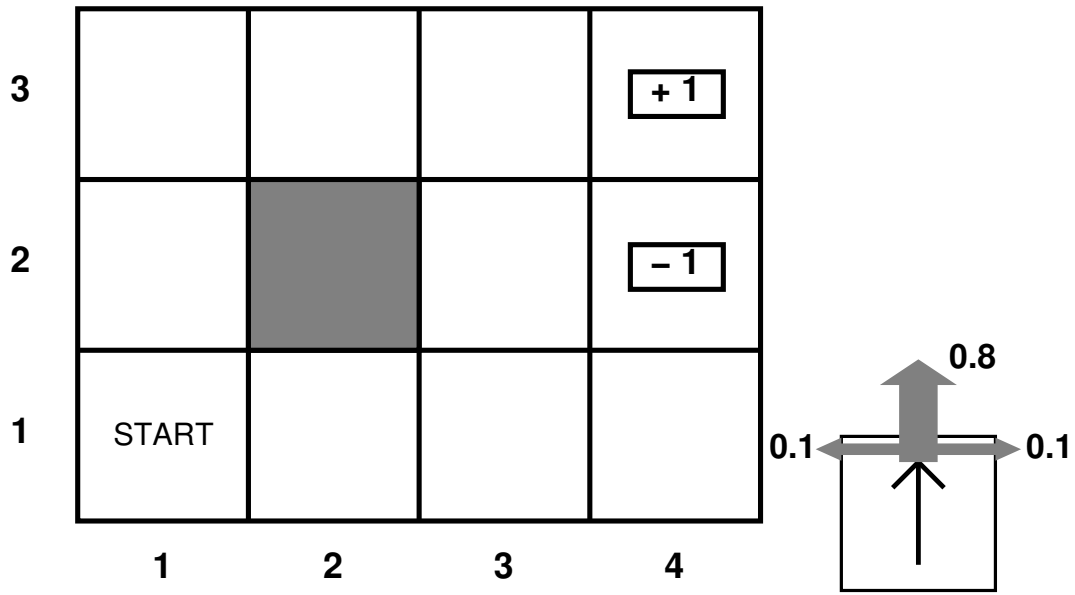
Na podstawie: AIMA ch21

Wojciech Jaśkowski

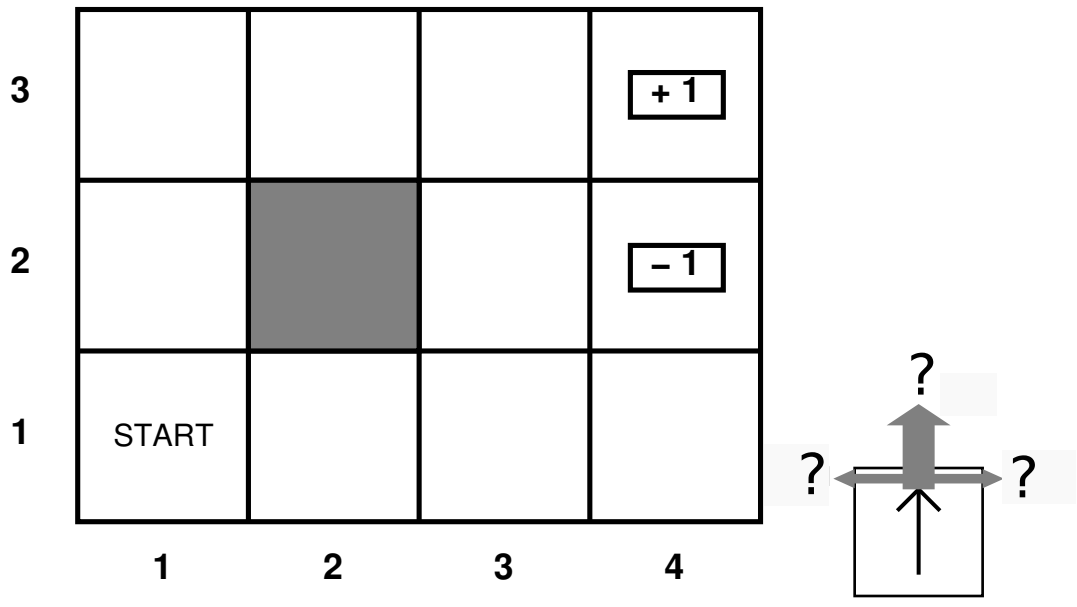
Instytut Informatyki,
Politechnika Poznańska

20 listopada 2013

Problem decyzyjny Markova

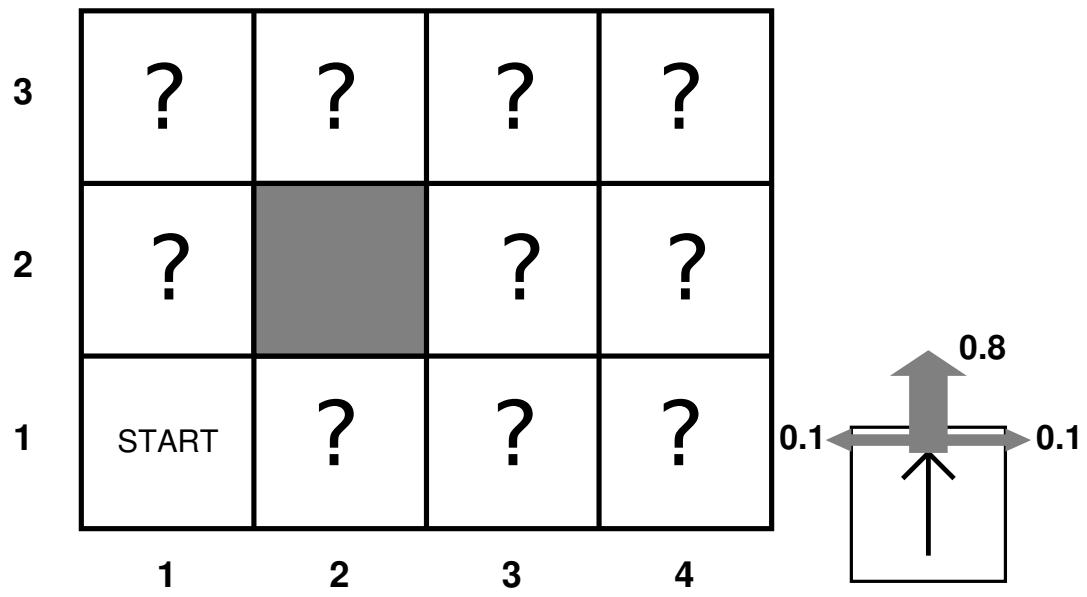


MDP bez modelu przejść $P(s'|s,a)$

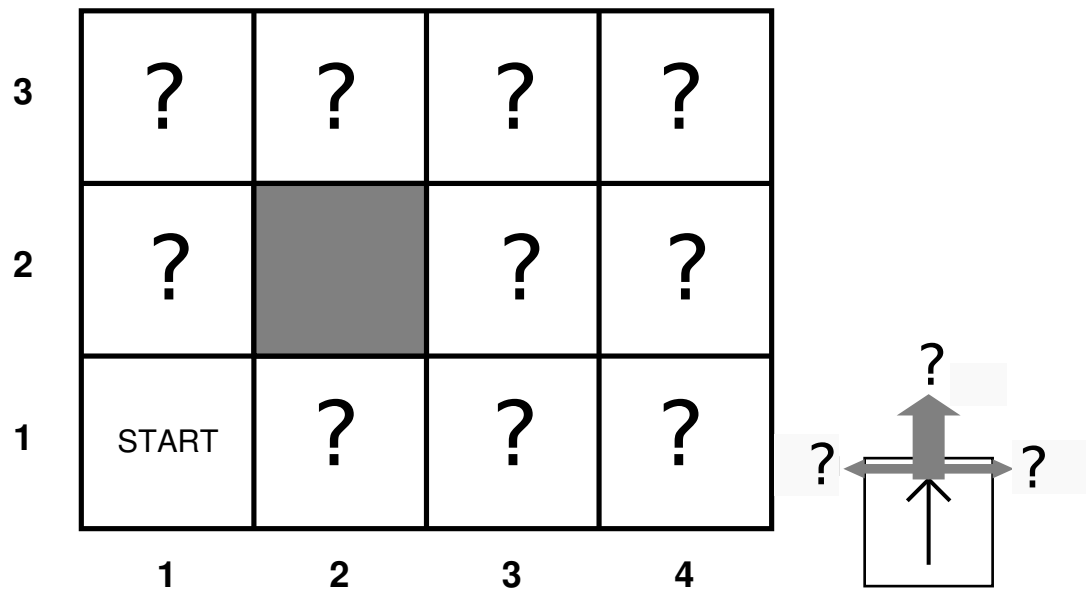


- Jak się nazywa takie środowisko? [zadanie 1]

MDP bez funkcji nagrody $R(s)$



MDP bez f. nagrody i bez modelu przejść



Uczenie ze wzmocnieniem

Problem uczenia ze wzmocnieniem

= MDP bez modelu przejść i bez funkcji nagrody = **nieznany MDP**

- Agent (gracz) musi nauczyć się:
  1. czy **ruch jest dobry czy zły**
  2. **dokąd prowadzą jego akcje** (model przejść)
     - **przewidywać** ruchy przeciwnika

Wzmocnienie

- Bez żadnej informacji ze środowiska agent nie ma podstaw, aby decydować, który jaki ruch wykonać
  - Musi wiedzieć, że coś dobrego się stało, gdy zaszachował przeciwnika
    $\rightarrow$ **nagroda** (reward), **wzmocnienie** (reinforcement)
- **Wzmocnienie**:
  - W szachach tylko na końcu gry.
  - W ping pongu: za każde odbicie
  - W nauce pływania za przesuwanie się do przodu
- Cel: **optymalna polityka**

Aplikacje i przykłady

- W wielu domenach RL jest jedyną drogą, np.
  - gry (kary/nagrody za wygraną/przegraną)
  - kontroler helikoptera (kary/nagrody za rozbicie się/chybotanie się/nietrzymanie kierunku)
- Robot uczący się ruchu:
  - `http://www.youtube.com/watch?v=RZf8fR1SmNY`

Podejścia do RL

Równanie Bellman'a:

$$U(s) = R(s) + \gamma \max_{a \in A} \sum_{s'} U(s')P(s'|s,a)$$

Trzy podejścia do RL:

1. **agent odruchowy** (ang. *direct policy search*)
   - Uczy się polityki $\pi : S \rightarrow A$
2. **agent z funkcją użyteczności**
   - uczy się f. użyteczności $U(s)$ i używa jej, aby wybierać akcje, które maksymalizują wartość oczekiwaną nagród
3. **agent z funkcją Q**
   - Uczy się funkcji $Q(s,a)$, która zwraca oczekiwaną użyteczność podjęcia danej akcji w danym stanie

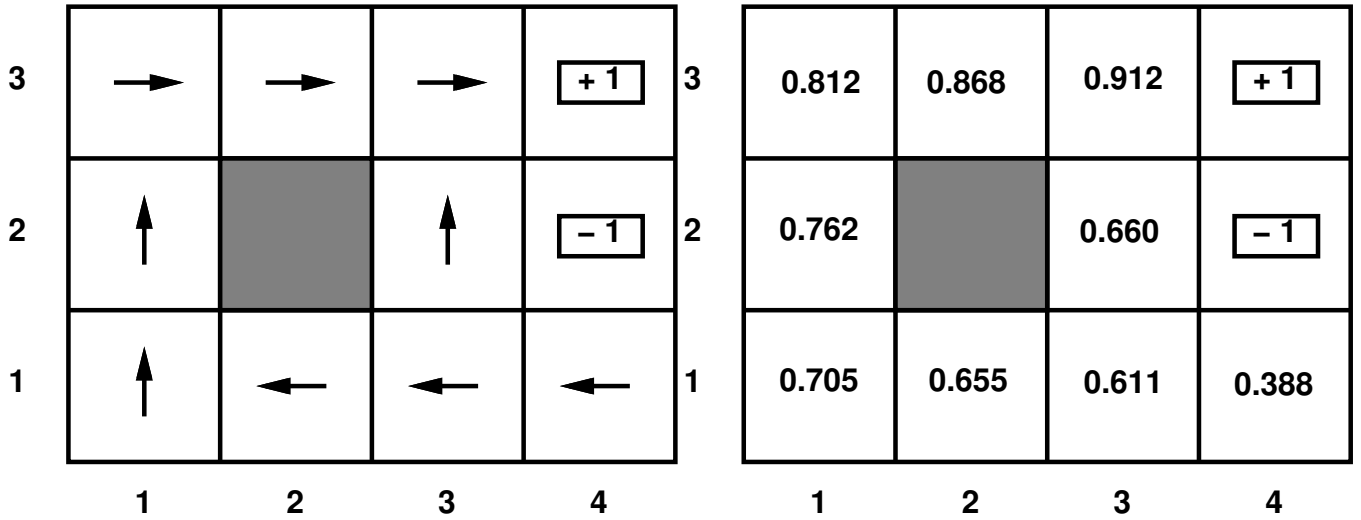
Który agent potrzebuje modelu świata?[zadanie 2]

Typy uczenia ze wzmocnieniem

Typy uczenia ze wzmocnieniem:

- **pasywne.** Polityka π jest dana.
  - Uczymy się tylko użyteczności stanów funkcja U (lub par: (stan, akcja): funkcja Q)
- **aktywne.** Musimy również nauczyć się polityki („co mam robić?")
  - Eksploracja

Pasywne uczenie ze wzmocnieniem



Znane:

- Środowisko całkowicie obserwowalne
- polityka π (agent w stanie s wykonuje akcję $\pi(s)$)

Nieznane:

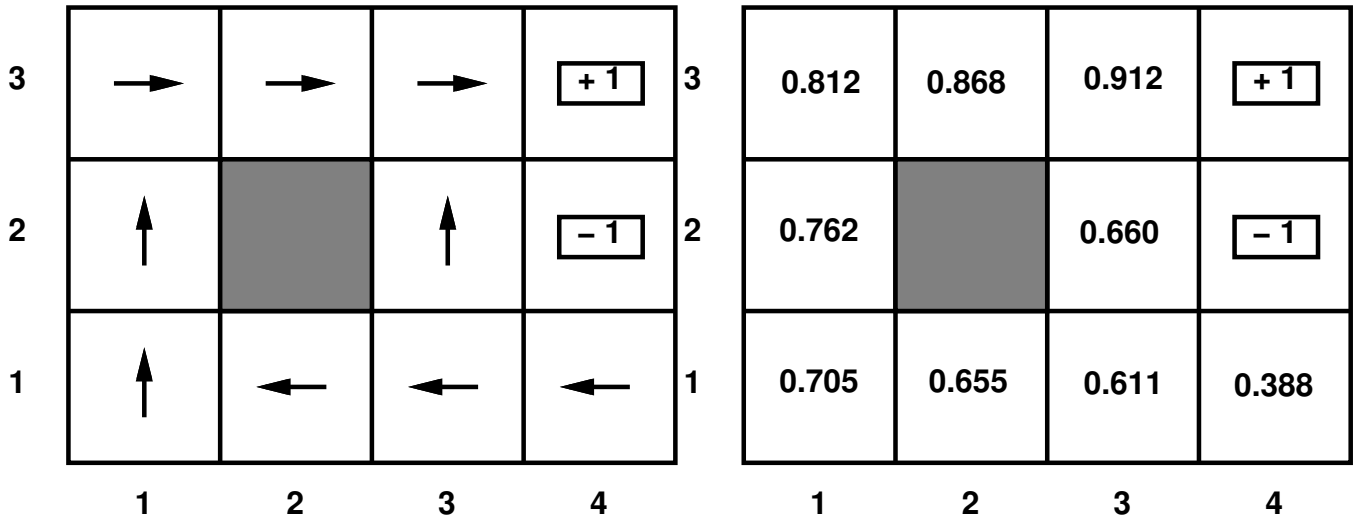
- model przejść $P(s'|s,a)$.
- funkcja nagrody $R(s)$

Cel: Jak „dobra" jest ta polityka?

- znaleźć wartości funkcji użyteczności $U^{\pi}(s)$.

Pasywne uczenie ze wzmocnieniem (c.d)

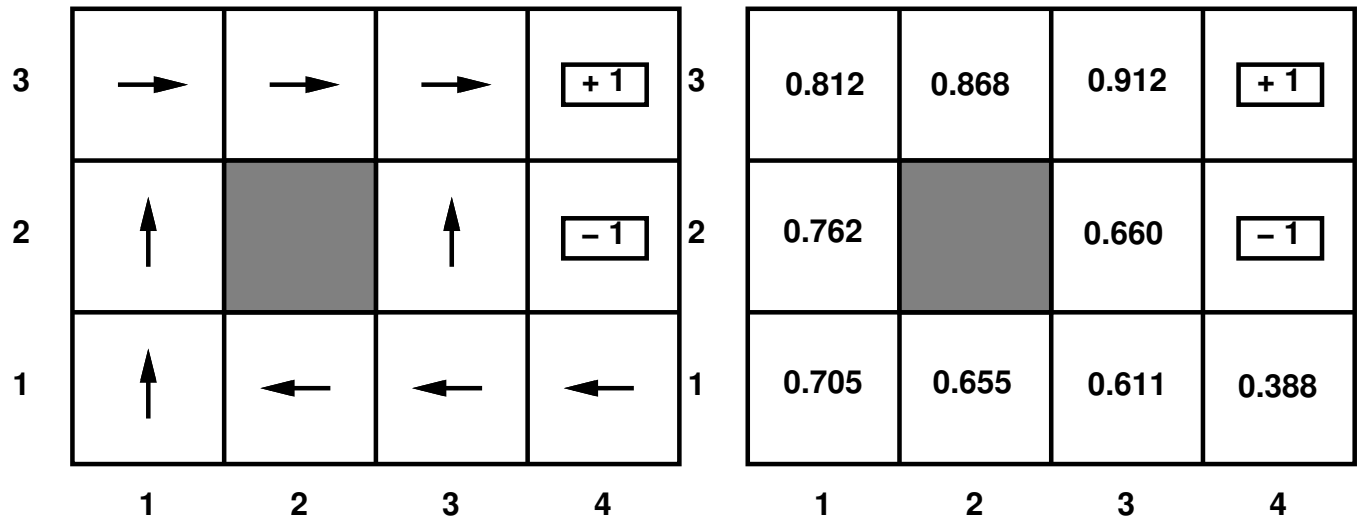
Jak policzyć użyteczność polityki?



Czy wystarczy skorzystać z rekurencyjnego wzoru, który już widzieliśmy (gdzie? [zadanie 3])?

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} U^{\pi}(s') P(s'|s, \pi(s))$$

Pasywne uczenie ze wzmocnieniem (c.d)



- Agent wykonuje serię „eksperymentów" (ang. *trial*) używając polityki π.
- Przykładowe „eksperymenty" (zebrane doświadczenie):
  1. $(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$
  2. $(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (3,2)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$
  3. $(1,1)_{-0.04} \rightarrow (2,1)_{-0.04} \rightarrow (3,1)_{-0.04} \rightarrow (3,2)_{-0.04} \rightarrow (4,2)_{-1}$

Pasywne uczenie ze wzmocnieniem (c.d)

- (Przypomnienie) użyteczność polityki w stanie s jest zdefiniowana jako:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right],$$

gdzie:

- S_t — zmienna losowa „w czasie t jestem w danym stanie"
- γ — współczynnik dyskontowy (przyjmiemy 1)

Czyli:

- Użyteczność stanu = oczekiwana całkowita nagroda z tego stanu dalej (oczekiwana **reward-to-go**).

Algorytm: Bezpośrednia estymacja użyteczności (Widrow and Hoff, 1960)

Zauważmy:

- Próbka daje informację o **reward-to-go** dla danego stanu
- Wiele próbek $\rightarrow$ **estymacja** $U^{\pi}(s)$ dla każdego stanu

Przykład:

1. $(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$
2. $(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (3,2)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$
3. $(1,1)_{-0.04} \rightarrow (2,1)_{-0.04} \rightarrow (3,1)_{-0.04} \rightarrow (3,2)_{-0.04} \rightarrow (4,2)_{-1}$

Dla $(3,3)$ mamy próbki, czyli
$U^{\pi}(3,3) = (0.88 + 0.96 + 0.96)/3 \approx 0.93$
Ile jest próbek i jakie mają wartości dla $(1,3)$? [zadanie 4]

Bezpośrednia estymacja użyteczności (c.d.)

Bezpośrednia estymacja użyteczności:

- sprowadza problem (pasywgnego) RL do problemu uczenia nadzorowanego:
  - stan: nagroda
- Nie uwzględnia informacji o zależnościach pomiędzy stanami.
  - **Użyteczności sąsiednich stanów nie są niezależne**!
  - Użyteczność stanu = nagroda w tym stanie + oczekiwana użyteczność jego **następników**, czyli:

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} U^{\pi}(s') P(s'|s, \pi(s))$$

- Stracona okazja do nauki $\rightarrow$ algorytm wolno zbiega.

Algorytm: Adaptatywne Programowanie Dynamiczne (ADP)

- Bierze pod uwagę zależności pomiędzy użytecznościami stanów.
- Bezpośrednio uczy się:
  - modelu przejść $P(s'|s,a)$
  - funkcji nagrody $R(s)$

Algorytm: Adaptatywne Programowanie Dynamiczne

Agent

- ze stanu s wykonał akcję $\pi[s] = a$.
- dotarł do stanu s' i otrzymał nagrodę r'.

procedure PASSIVE-ADP(s, a, s', r')

if s' jest nowym stanem **then**

$U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$

$N[s, a] \leftarrow N[s, a] + 1$

$M[s', s, a] \leftarrow M[s', s, a] + 1$

for w **in** znane następniki stanu s (tzn. $M[w, s, a] > 0$) **do**

$P(w|s, a) \leftarrow M[w, s, a]/N[s, a]$

$U \leftarrow$ Policy-Evaluation (π, P, U)

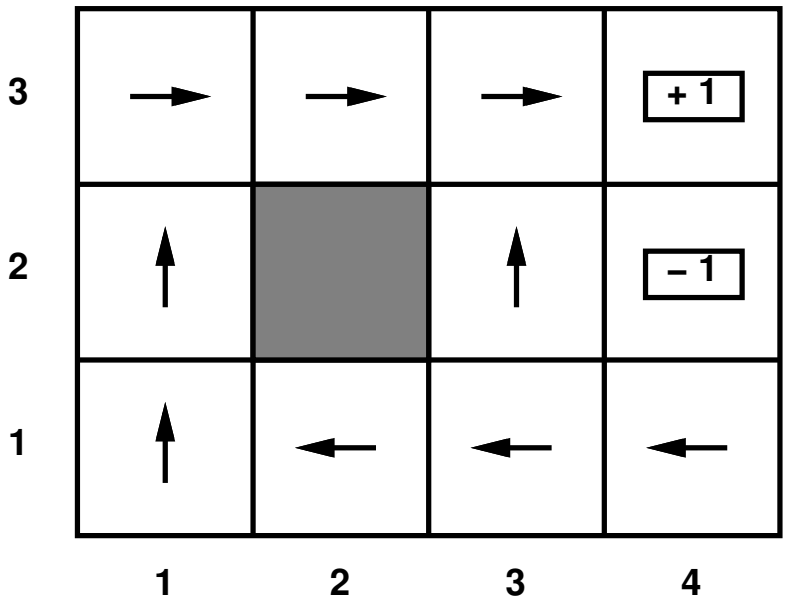
return $\pi[s']$

Policy-Evaluation: układ równań lub iteracyjnie.

ADP — Przykład

Przykład:

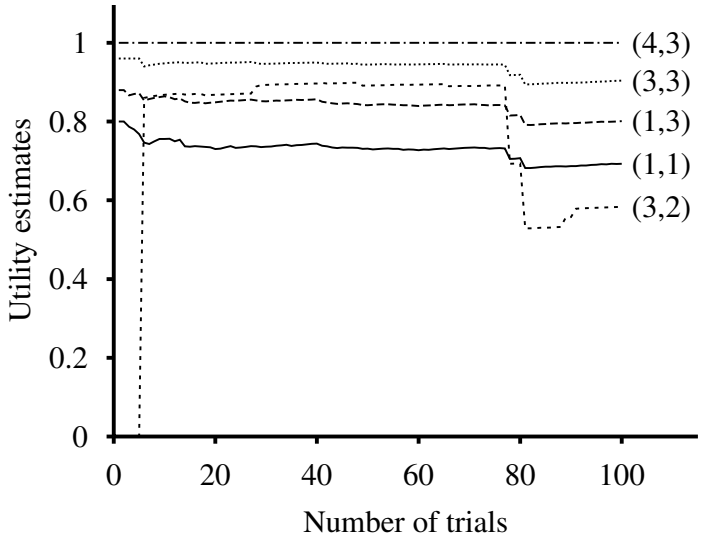
1. $(1,1)_{-0.04} \to (1,2)_{-0.04} \to (1,3)_{-0.04} \to (1,2)_{-0.04} \to (1,3)_{-0.04} \to (2,3)_{-0.04} \to (3,3)_{-0.04} \to (4,3)_{+1}$
2. $(1,1)_{-0.04} \to (1,2)_{-0.04} \to (1,3)_{-0.04} \to (2,3)_{-0.04} \to (3,3)_{-0.04} \to (3,2)_{-0.04} \to (3,3)_{-0.04} \to (4,3)_{+1}$
3. $(1,1)_{-0.04} \to (2,1)_{-0.04} \to (3,1)_{-0.04} \to (3,2)_{-0.04} \to (4,2)_{-1}$

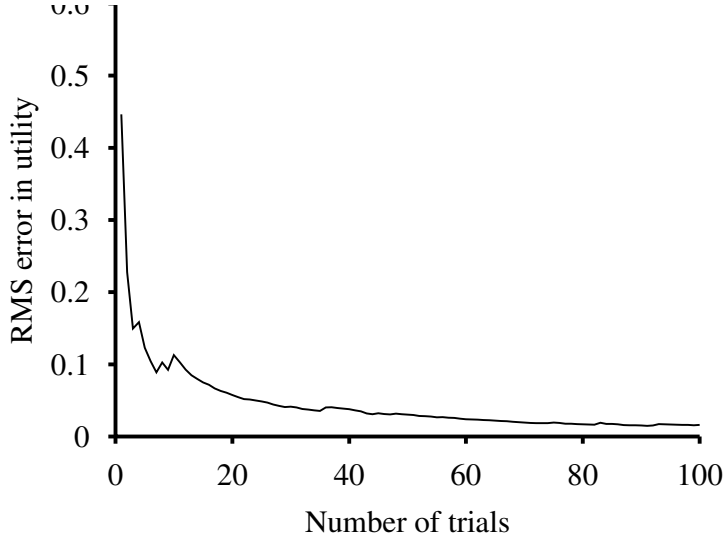


Wykonaj ADP[zadanie 5] :

- $R((1,3)) = ?$
- $R((4,1)) = ?$
- $P((1,3)|(1,2), góra) = ?$
- $P((1,3)|(1,2), dół) = ?$
- $P((2,3)|(1,3), prawo) = ?$

ADP — wykresy





Uwagi:

- ADP implementuje ideę Oszacowania Maksymalnego Prawdopodobieństwa (Maximum Likelihood Estimation)
  - Znajduje najbardziej prawdopodobny model (najlepiej pasujący do danych)
- ADP zbiega całkiem szybko (jest tylko ograniczony tym jak szybko potrafi nauczyć się modelu przejść).
- Policy-Evaluation jest dość wolne.

Uczenie różnicowe (TDL)

ang. *temporal difference* (TD) *learning* (TDL)

- **Pomysł:**
  - Użyj obserwacji $(s \rightarrow_a s')$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby one współgrały z ograniczeniami.
- **Przykład**:
  - Załóżmy, że $U^\pi(1,3) = 0.84$ i $U^\pi(2,3) = 0.94$.
  - Obserwacja: $(1,3) \rightarrow_{góra} (2,3)$.
  - Jeżeli to przejście zawsze ma miejsce, to oczekiwalibyśmy, że

$$U^\pi(1,3) = -0.04 + \gamma U^\pi(2,3) = 0.90$$

  - **Wniosek**: $U^\pi(1,3) = 0.84$ jest za małe o $0.90 - 0.84 = 0.06$
  - Zwiększmy je „trochę” $(\alpha = 0.01)$, tzn.

$$U^\pi(1,3) = U^\pi(1,3) + \alpha 0.06 = 0.846$$

Uczenie różnicowe (TDL) — ogólnie

- Próbka: $s \rightarrow_a s'$ i nagroda w stanie s to $R(s)$.
- Początkowo $U^\pi(s)$
- Oczekujemy, że

$$U'^\pi(s) = R(s) + \gamma U^\pi(s')$$

- Modyfikujemy $U^\pi(s)$ o ważoną (α) różnicę pomiędzy „oczekiwanym" U'^π a starym U^π.
  - Różnica:

$$\Delta = U'^\pi(s) - U^\pi(s)$$

  - „Nowe" $U^\pi(s)$:

$$U^\pi(s) = U^\pi(s) + \alpha\Delta$$

Uczenie różnicowe

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha\left(R(s) + \gamma U^\pi(s') - U^\pi(s)\right)$$

- α — współczynnik uczenia

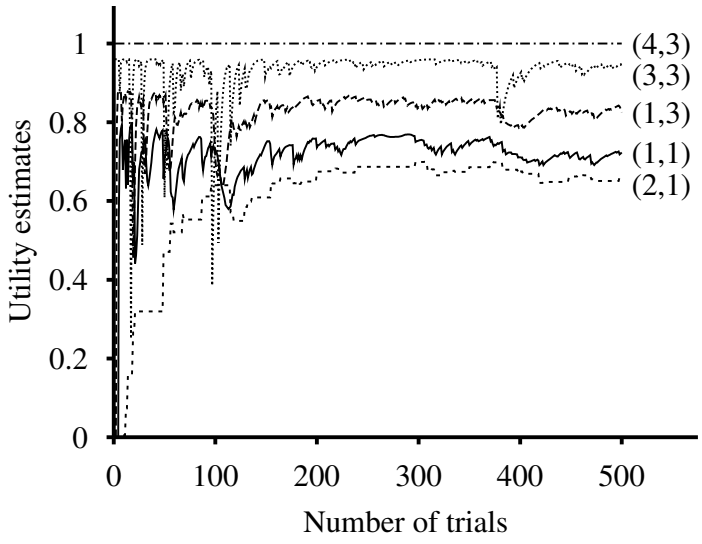
Uczenie różnicowe (TDL)

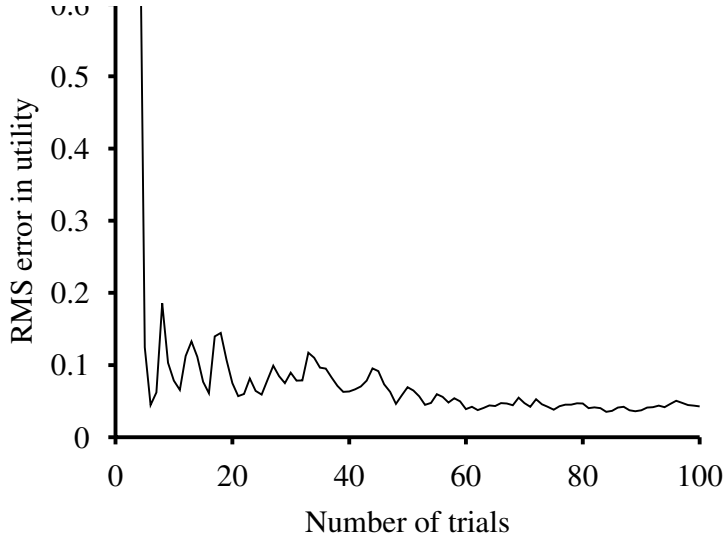
procedure PASSIVE-TD(s, a, s', r')
  if s' jest nowym stanem **then**
    $U[s'] \leftarrow r'$
  $U[s] \leftarrow U[s] + \alpha(R[s] + \gamma U[s'] - U[s])$
  return $\pi[s']$

Uwagi:

- Aktualizacja $U[s]$ nie uwzględnia akcji dostępnych i modelu przejść, ale to się uśredni się.
- TDL nie potrzebuje modelu przejść, aby uaktualniać użyteczności stanów (rodzina metod ***model-free***).
- jeżeli α w odpowiedni sposób zmniejsza się w czasie, to *TDL* gwarantuje zbieżność do optimum globalnego.

Uczenie różnicowe — algorytm





Uwagi:

- TD potrzebuje więcej obserwacji niż ADP i ma spore wahania, ale:
- TD jest prostszy i potrzebuje mniej obliczeń na obserwację

TD vs. ADP

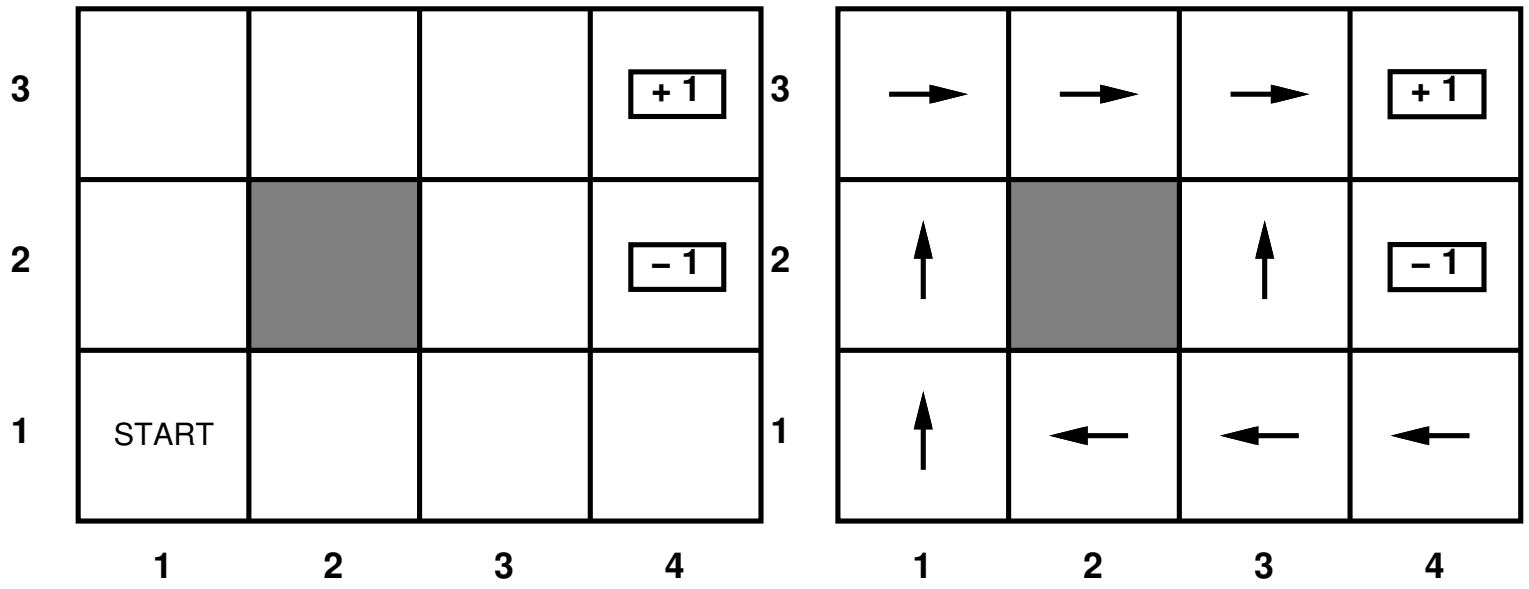
- TD i ADP są podobne: oba dokonują lokalnych zmian, po to, aby użyteczność stanu z jego następnikami „zgadzały się".
- **Różnica 1:**
  - TD bierze pod uwagę tylko jednego następnika
  - ADP bierze pod uwagę wszystkich następników (waży ich prawdopodobieństwami)
- **Różnica 2:**
  - TD zmienia tylko jedną wartość użyteczności na obserwację
  - ADP zmienia użyteczności tylu stanów, ile potrzeba, aby równania się zgadzały
- $\Longrightarrow$ TD można traktować jako aproksymację ADP.
- Z p. widzenia TD, ADP używa **pseudodoświadczenia** wygenerowanego na podstawie aktualnej wiedzy o środowisku.

TD vs. ADP c.d.

Stąd: możliwe są rozwiązania pośrednie:

- np. TD, który generuje pewne pseudodoświadczenia (czyli aktualizuje więcej użyteczności stanów)
- lub ADP, który nie aktualizuje wszystkich użyteczności
  - **Prioritized sweeping** (Moore i Atkeson, 1993)— aktualizuj użyteczności tylko niektórych stanów (tych, które prawd. najbardziej tego wymagają)
  - Sens: skoro i tak model nie jest poprawny, to po co dokładnie liczyć użyteczności?

Aktywne uczenie ze wzmocnieniem



- Polityka π jest nieznana.

ADP (przypomnienie)

procedure PASSIVE-ADP(s, a, s', r')

if s' jest nowym stanem **then**

$U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$

$N[s, a] \leftarrow N[s, a] + 1$

$M[s', s, a] \leftarrow M[s', s, a] + 1$

for w **in** znane następniki stanu s (tzn. $M[w, s, a] > 0$) **do**

$P(w|s, a) \leftarrow M[w, s, a]/N[s, a]$

$U \leftarrow$ Policy-Evaluation (π, P, U)

return $\pi[s']$

ADP dla uczenia aktywnego

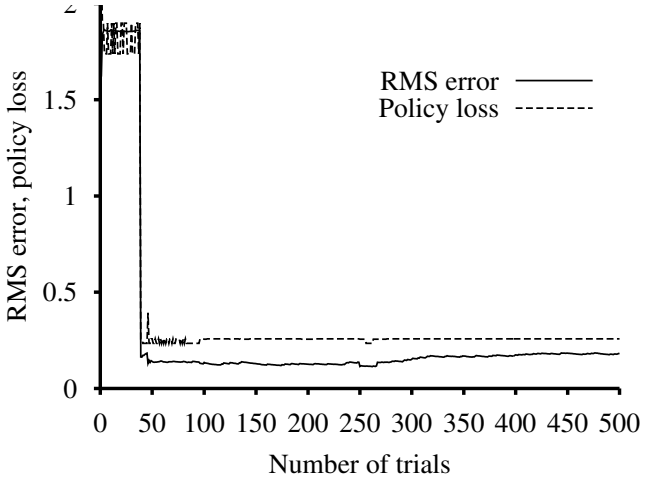
Jak zmodyfikować ADP?

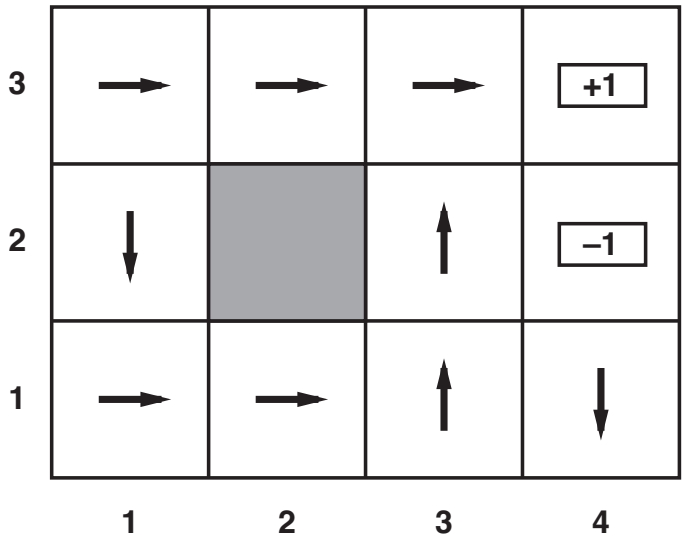
1. Musimy nauczyć się **całego modelu przejść**, a nie tylko przejść określonych przez π.
   - ADP sobie poradzi
2. Agent **nie ma danej polityki**, więc Policy-Evaluation musi skorzystać z pełnego wzoru Bellman'a:

$$U(s) = R(s) + \gamma \mathrm{max_a} \sum_{\mathrm{s'}} \mathrm{P(s'|s,a)U(s')}$$

   - Można użyć Iteracji Wartości (albo Iteracji Polityki)
3. Jaką **akcję powinien wybierać** w każdym kroku?

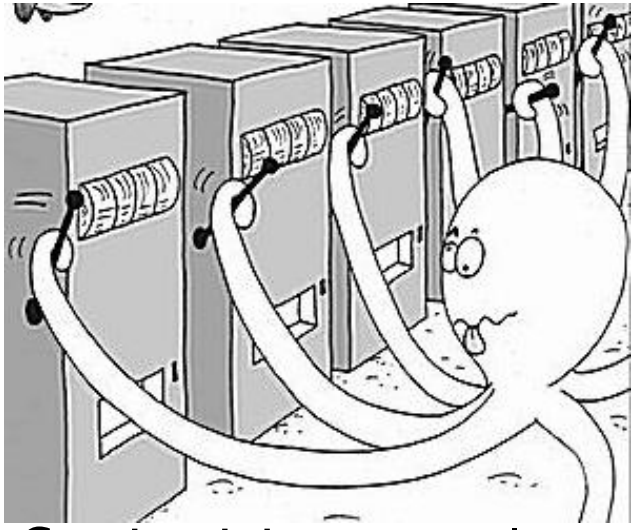
Eksploracja





- **Agent zachłanny** utknął.
- **Powód**: model świata, dla którego wyznaczył optymalna politykę nie jest modelem prawdziwym.
- Akcje służą:
  - Osiąganiu nagród (**eksploatacja**)
  - Ulepszaniu modelu środowiska (**eksploracja**)
- Czysta eksploatacja $\Longrightarrow$ ryzyko wpadnięcia „w rutynę”
- W każdym kroku decyzja: eksploracja czy eksploatacja?

Problem wielorękiego bandyty



- n automatów do gry.
- gra $\rightarrow$ możliwa wypłata
- próbować inne automaty czy eksploatować ten, który daje rozsądne wyniki?

Czy istnieje optymalna metoda eksploracji?

- co to znaczy **optymalny**?
  - oczekiwana wartość dla wszystkich możliwych światów ($P(s'|s,a)$) jest najlepsza

Gittins index

- rozwiązania są zwykle obliczeniowo bardzo trudne ($\rightarrow$ **statystyczna teoria decyzji**)
- jeśli wypłaty są niezależne od siebie i są dyskontowane w czasie, to rozwiązaniem jest **Gittins index**.
  - Określa jak wartościowy jest wybór danej maszyny
  - Dla sekwencyjnych problemów decyzyjnych Gittins indeks nie działa

Metoda ϵ-zachłanna

- Rozwiązanie „rozsądne” zapewniają, że każda akcja z każdego stanu jest wykonywana nieograniczoną liczbę razy.
  - $\Longrightarrow$ gwarancja, że użyteczność $U(s)$ zbiegnie w granicy do „prawdziwej” użyteczności stanów.
- Prostym przykładem jest metoda ϵ-**zachłanna**:
  - z prawd. $1 - \epsilon$ użyj „optymalnej” (zachłannej) akcji
  - z prawd. ϵ użyj losowej akcji

Ciekawość i optymistyczna f. użyteczności

- Powyższe się zbiegnie, ale jest wolne. Lepiej w praktyce: użyj prostej **funkcji eksploracji** i **optymistycznej wersji f. użyteczności** np:

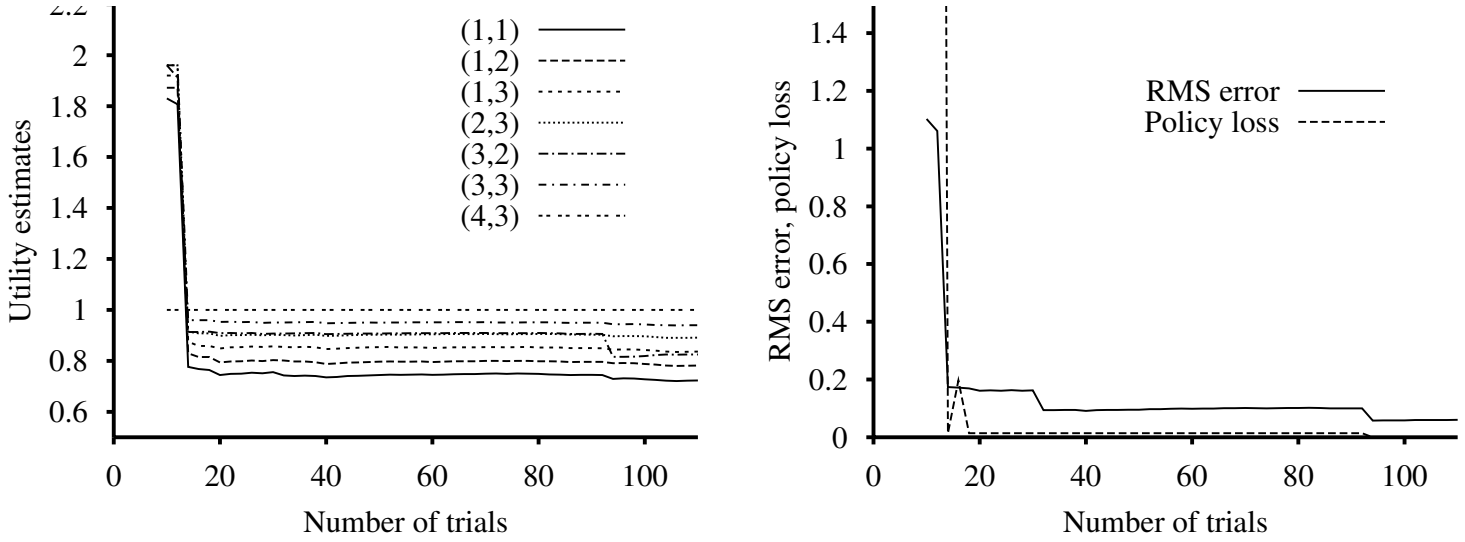
$$U^+(s) \leftarrow R(s) + \gamma \mathrm{max_a f}\left(\sum_{\mathrm{s'}} \mathrm{P(s'|s,a)U^+(s'), N(s,a)}\right),$$

gdzie funkcja eksploracji waży **użyteczność** stanu i **„ciekawość”** (niewiedzę)

$$f(u,n) = \begin{cases} R^+ & n < N_e \\ u & \text{w przeciwnym wypadku} \end{cases}$$

R^+ — optymistyczna nagroda (np. $R^+ = \max_s R(s)$)
N_e — stała

Aktywny ADP z f. eksploracji ($R^+ = 2, N_e = 5$)



Aktywne TD

- Jak pasywne, ale:
  - Musimy **uczyć się modelu** $P(s'|s,a)$ tak jak ADP
  - Potrzebna jakaś funkcja eksploracji do wyboru akcji.

procedure ACTIVE-TD(s, a, r, s', r')

  if s' is new **then**

    $U[s'] \leftarrow r'$

  $N[s,a] \leftarrow N[s,a] + 1$

  $M[s',s,a] \leftarrow M[s',s,a] + 1$

  for w **in** znane następniki stanu s (tzn. $M[w,s,a] > 0$) **do**

    $P(w|s,a) \leftarrow M[w,s,a]/N[s,a]$

  $U[s] \leftarrow U[s] + \alpha(r + \gamma U[s'] - U[s])$

  return $\mathrm{argmax}_{a'} f\left(R(t) + \gamma \sum_w P(w|s',a')U[w], N[w,a']\right)$

Q-Learning (Watkins, 1989)

- Zamiast $U(s)$ uczymy się **funkcji Q(s, a)** — użyteczność wykonania akcji a w stanie s. Zależność:

$$U(s) = \max_a Q(s, a)$$

- **Istotna zaleta**: Nie trzeba uczyć się modelu przejść $P(s'|s, a)$! (metoda **model-free**)
- Ograniczenia do spełnienia:

$$Q(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

- Moglibyśmy użyć tego bezpośrednio $\rightarrow$ konieczna nauka modelu przejść

Q-Learning

- Mamy przejście $s \rightarrow_a s'$ z nagrodą r'.
- Znamy aktualne wartości $Q(s, a)$ oraz $Q(s', a')$ dla wszystkich $a' \in A(s)$.
- Oczekujemy, że spełnione będzie równanie:

$$Q'(s, a) = R(s) + \gamma \max_{a'} Q(s', a')$$

- Skoro jednak nie jest spełnione, to modyfikujemy $Q(s, a)$ „w stronę" $Q'(s, a)$

Reguła modyfikacji Q-Learning

- Analogicznie jak w TD otrzymujemy

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(R(s) + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

- Przykład [zadanie 6]

(TD) Q-Learning — algorytm

procedure ACTIVE-TD(s, a, r, s', r')

  if s jest stanem terminalnym **then**

    $Q[s, None] \leftarrow r'$

  $N[s, a] \leftarrow N[s, a] + 1$

  $Q[s, a] \leftarrow Q[s, a] + \alpha\left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a]\right)$

  return $\mathrm{argmax}_{a'} f\left(Q[s', a'], N[s', a']\right)$