Uczenie ze wzmocnieniem

Na podstawie: AIMA ch21

Wojciech Jaśkowski

Instytut Informatyki,
Politechnika Poznańska

7 maja 2015

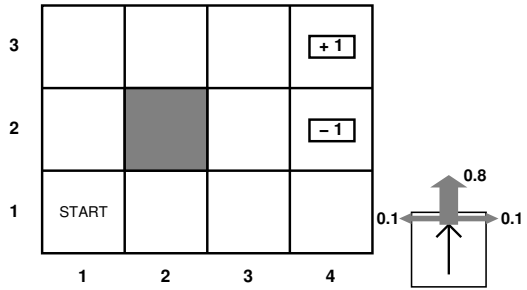
Uczenie ze wzmocnieniem
Na podstawie: AIMA ch21

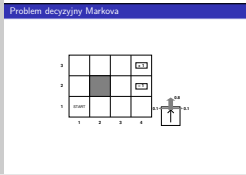
Wojciech Jaśkowski

Instytut Informatyki,
Politechnika Poznańska

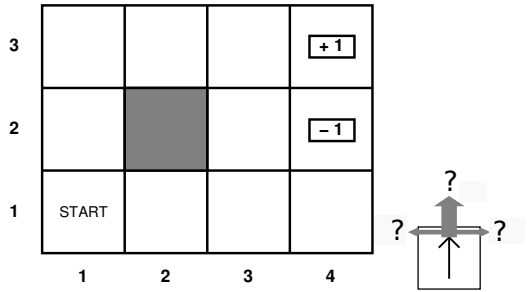
7 maja 2015

Problem decyzyjny Markova

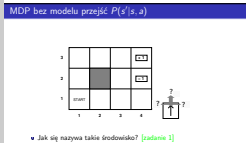




MDP bez modelu przejść $P(s'|s,a)$

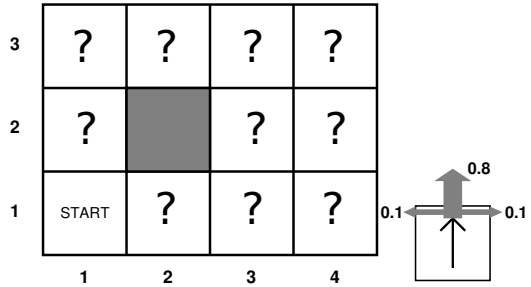


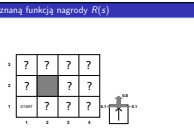
- Jak się nazywa takie środowisko? [zadanie 1]



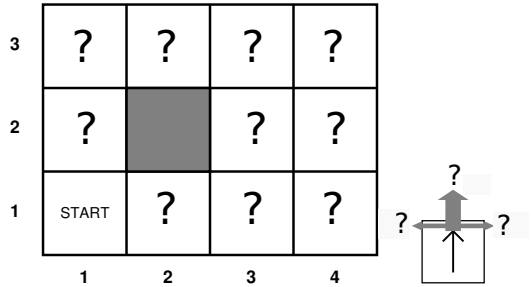
1. Środowisko jest nieznane

MDP z nieznaną funkcją nagrody $R(s)$

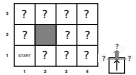




Nieznane MDP



Nieznane MDP



Uczenie ze wzmocnieniem (RL)

Problem uczenia ze wzmocnieniem

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

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

1. W skrócie: wyobraź sobie grę, której zasad nie znasz. Grasz, a po 100 ruchach sędzia mówi: „przegrałeś”. To jest uczenie ze wzmocnieniem.

Wzmocnienie

- Bez żadnej informacji ze środowiska agent nie ma podstaw, aby decydować, który ruch wykonać:
  - Musi wiedzieć, że coś dobrego się stało, gdy wygrał albo wykonał dobry ruch
    $\rightarrow$ **nagroda** (reward), **wzmocnienie** (reinforcement)
- **Wzmocnienie**:
  - szachy: tylko na końcu gry,
  - ping pong: za każde odbicie,
  - nauka pływania: za przesuwanie się do przodu.
- Cel: **optymalna polityka** (racjonalny agent)

Wzmocnienie
- Bez żadnej informacji ze środowiska agent nie ma podstaw, aby decydować, który ruch wykonać:
  - Musi wiedzieć, że coś dobrego się stało, gdy wygrał albo wykonał dobry ruch
    → **nagroda** (reward), **wzmocnienie** (reinforcement)
- **Wzmocnienie**:
  - szachy: tylko na końcu gry,
  - ping pong: za każde odbicie,
  - nauka pływania: za przesuwanie się do przodu.
- Cel: **optymalna polityka** (racjonalny agent)

1. Przypomnienie: racjonalny agent maksymalizuje oczekiwaną (zdyskontowany) sumę nagród (pod warunkiem posiadanej wiedzy).

Aplikacje i przykłady

- W wielu domenach RL jest najlepszą drogą postępowania, aby automatycznie uzyskać efektywnego agenta:
  - agent grający w grę (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`

Aplikacje i przykłady
- W wielu domenach RL jest najlepszą drogą postępowania, aby automatycznie uzyskać efektywnego agenta:
  - agent grający w grę (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

1. Co jest stanem środowiska? Jakie akcje wykonuje? Za co dostaje wzmocnienie?

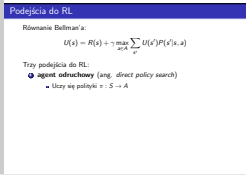
Podejścia do RL

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 $\pi : S \to A$



1. Równanie Bellman'a: teraźniejszość $R(s)$ + (najlepsza) przyszłość.
2. Polityka π agenta odruchowego mapuje bezpośrednio stany na akcje.
3. utility-based agent musi posiadać model środowiska, żeby podejmować decyzje.
4. AIMA nazywa tę funkcję *utility function*, ale w literaturze przyjęła się nazwa *value function* $V(s)$.
5. Q-learning agent nie musi posiadać modelu środowiska, ale przez to nie może wnioskować na więcej niż jeden ruch do przodu (bo nie wie w jakim stanie będzie).
6. Tylko ten z funkcją U.

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 $\pi : S \to A$
2. **agent z funkcją użyteczności** U
   - uczy się f. użyteczności stanu $U(s)$. Jego polityka jest zachłanna ze względu na U.

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 $\pi : S \to A$
2. agent z funkcją użyteczności U
   - uczy się f. użyteczności stanu $U(s)$. Jego polityka jest zachłanna ze względu na U.

1. Równanie Bellman'a: teraźniejszość $R(s)$ + (najlepsza) przyszłość.
2. Polityka π agenta odruchowego mapuje bezpośrednio stany na akcje.
3. utility-based agent musi posiadać model środowiska, żeby podejmować decyzje.
4. AIMA nazywa tę funkcję *utility function*, ale w literaturze przyjęła się nazwa *value function* $V(s)$.
5. Q-learning agent nie musi posiadać modelu środowiska, ale przez to nie może wnioskować na więcej niż jeden ruch do przodu (bo nie wie w jakim stanie będzie).
6. Tylko ten z funkcją U.

Podejścia do RL

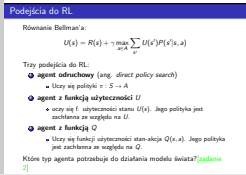
Równanie Bellman'a:

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

Trzy podejścia do RL:

1. **agent odruchowy** (ang. *direct policy search*)
   - Uczy się polityki $\pi : S \to A$
2. **agent z funkcją użyteczności** U
   - uczy się f. użyteczności stanu $U(s)$. Jego polityka jest zachłanna ze względu na U.
3. **agent z funkcją** Q
   - Uczy się funkcji użyteczności stan-akcja $Q(s,a)$. Jego polityka jest zachłanna ze względu na Q.

Które typ agenta potrzebuje do działania modelu świata?[zadanie 2]



1. Równanie Bellman'a: teraźniejszość $R(s)$ + (najlepsza) przyszłość.
2. Polityka π agenta odruchowego mapuje bezpośrednio stany na akcje.
3. utility-based agent musi posiadać model środowiska, żeby podejmować decyzje.
4. AIMA nazywa tę funkcję *utility function*, ale w literaturze przyjęła się nazwa *value function* $V(s)$.
5. Q-learning agent nie musi posiadać modelu środowiska, ale przez to nie może wnioskować na więcej niż jeden ruch do przodu (bo nie wie w jakim stanie będzie).
6. Tylko ten z funkcją U.

Typy uczenia ze wzmocnieniem

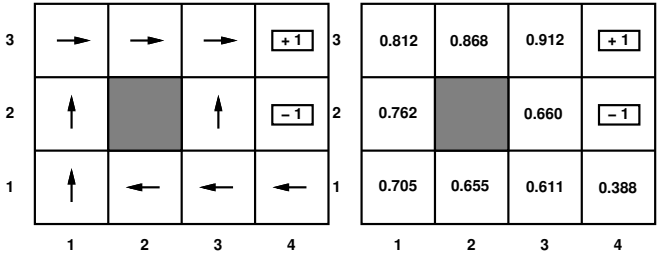
Typy uczenia ze wzmocnieniem:

- **pasywne (problem predykcji).** Polityka π jest dana.
  - Uczymy się tylko użyteczności stanów $U^\pi(s)$ lub użyteczności par stan-akcja $Q^\pi(s, a)$
- **aktywne (problem sterowania).** Musimy znaleźć optymalną politykę π (zwykle robimy to poprzez znalezienie U i Q, a polityka jest zachłanna ze względu na U lub Q).
  - Konieczna eksploracja...

Typy uczenia ze wzmocnieniem

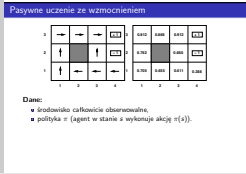
Typy uczenia ze wzmocnieniem:
- **pasywne (problem predykcji).** Polityka π jest dana.
  - Uczymy się tylko użyteczności stanów $U^\pi(s)$ lub użyteczności par stan-akcja $Q^\pi(s, a)$
- **aktywne (problem sterowania).** Musimy znaleźć optymalną politykę π (zwykle robimy to poprzez znalezienie U i Q, a polityka jest zachłanna ze względu na U lub Q).
  - Konieczna eksploracja...

Pasywne uczenie ze wzmocnieniem

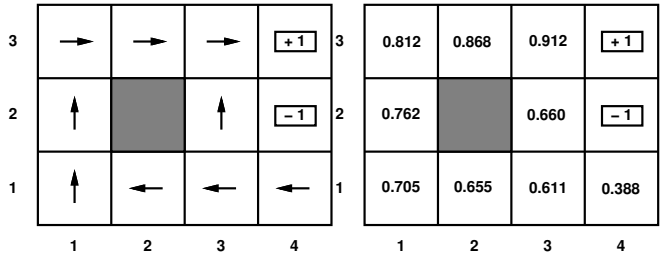


Dane:

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



Pasywne uczenie ze wzmocnieniem

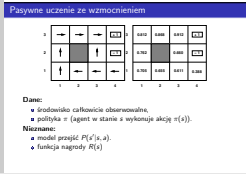


Dane:

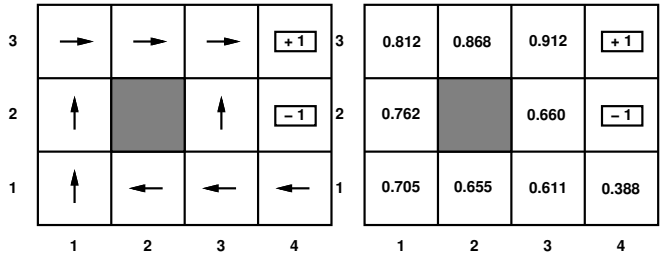
- środowisko całkowicie obserwowalne,
- polityka π (agent w stanie s wykonuje akcję $\pi(s)$).

Nieznane:

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



Pasywne uczenie ze wzmocnieniem



Dane:

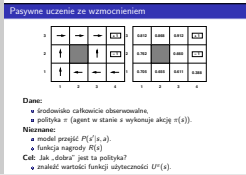
- środowisko całkowicie obserwowalne,
- polityka π (agent w stanie s wykonuje akcję $\pi(s)$).

Nieznane:

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

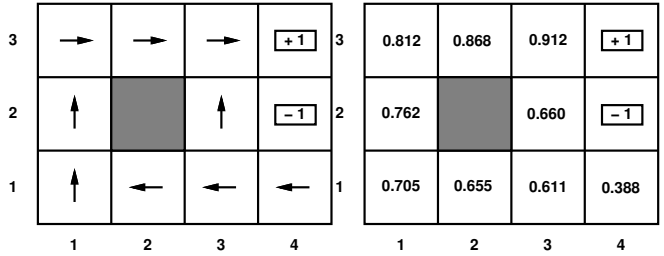
Cel: Jak „dobra" jest ta polityka?

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



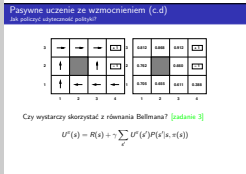
Pasywne uczenie ze wzmocnieniem (c.d)

Jak policzyć użyteczność polityki?



Czy wystarczy skorzystać z równania Bellmana? [zadanie 3]

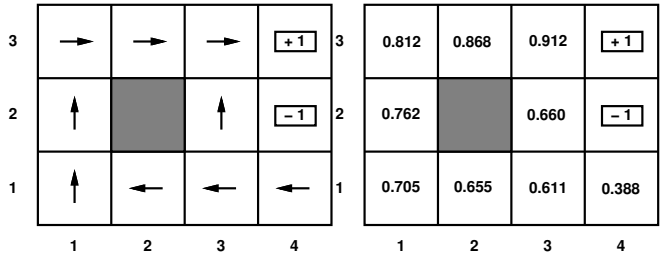
$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} U^{\pi}(s') P(s'|s, \pi(s))$$



1. Nie, bo nie znamy modelu świata oraz R. Algorytm iteracji polityki.
2. Do przemyślenia: czym więc różni się uczenie pasywne od ewaluacji polityki z poprzedniego rozdziału?

Pasywne uczenie ze wzmocnieniem (c.d)

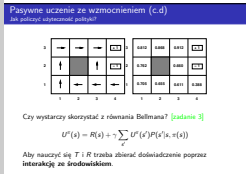
Jak policzyć użyteczność polityki?



Czy wystarczy skorzystać z równania Bellmana? [zadanie 3]

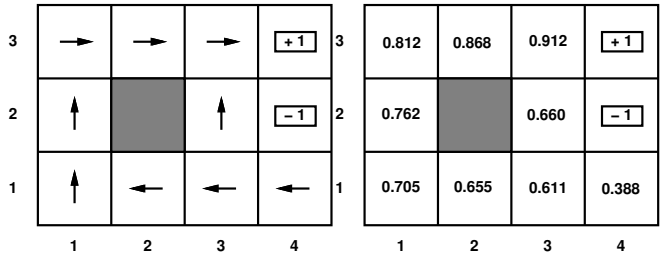
$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} U^{\pi}(s') P(s'|s, \pi(s))$$

Aby nauczyć się T i R trzeba zbierać doświadczenie poprzez **interakcję ze środowiskiem**.



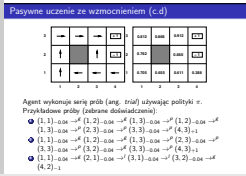
1. Nie, bo nie znamy modelu świata oraz R. Algorytm iteracji polityki.
2. Do przemyślenia: czym więc różni się uczenie pasywne od ewaluacji polityki z poprzedniego rozdziału?

Pasywne uczenie ze wzmocnieniem (c.d)



Agent wykonuje serię prób (ang. *trial*) używając polityki π.
Przykładowe próby (zebrane doświadczenie):

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



1. W p. 3 jest błąd. Zgodnie z modelem, który przyjmowaliśmy, z $(2,1)$ idąc w lewo nie można się dostać do $(3,1)$. Ale to nic strasznego.

Pasywne uczenie ze wzmocnieniem (c.d)

Przypomnienie
Użyteczność polityki w stanie s:

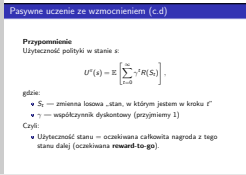
$$U^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right],$$

gdzie:

- S_t — zmienna losowa „stan, w którym jestem w kroku t"
- γ — współczynnik dyskontowy (przyjmiemy 1)

Czyli:

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



1. reward-to-go to empiryczna wartość użyteczności stanu.

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

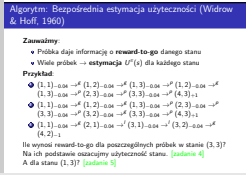
Zauważmy:

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

Przykład:

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

Ile wynosi reward-to-go dla poszczególnych próbek w stanie $(3,3)$?
Na ich podstawie oszacujmy użyteczność stanu. [zadanie 4]
A dla stanu $(1,3)$? [zadanie 5]



1. Stan ten odwiedzono 3 razy. Możemy więc estymować jego użyteczność jako $U^{\pi}(3,3) = (0.88 + 0.96 + 0.96)/3 \approx 0.93$.
2. $U^{\pi}(1,3) = (0.80 + 0.80 + 0.88)/3 \approx$
3. ang. *Direct Utility Estimation*
4. Ten algorytm można też nazwać *Monte Carlo Policy Evaluation* (wg Sutton i Barto, 1998, str. 112)

Bezpośrednia estymacja użyteczności (c.d.)

1. Sprowadza problem predykcji (pasywne uczenie się) do problemu uczenia nadzorowanego (regresja):
   - zbiór przykładów typu ⟨stan, reward-to-go⟩

Bezpośrednia estymacja użyteczności (c.d.)
Sprowadza problem predykcji (pasywne uczenie się) do problemu uczenia nadzorowanego (regresja):
zbiór przykładów typu ⟨stan, reward-to-go⟩

1. Ale zbiega, i to niechybnie.

Bezpośrednia estymacja użyteczności (c.d.)

1. Sprowadza problem predykcji (pasywne uczenie się) do problemu uczenia nadzorowanego (regresja):
   - zbiór przykładów typu ⟨stan, reward-to-go⟩
2. 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 → algorytm zbiega wolno.

Bezpośrednia estymacja użyteczności (c.d.)
Sprowadza problem predykcji (pasywne uczenie się) do problemu uczenia nadzorowanego (regresja):
zbiór przykładów typu ⟨stan, reward-to-go⟩
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 → algorytm zbiega wolno.

1. Ale zbiega, i to niechybnie.

Adaptatywne Programowanie Dynamiczne (ADP)

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

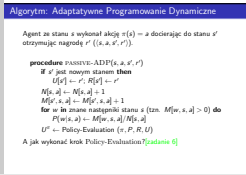
Adaptatywne Programowanie Dynamiczne (ADP)
1. Bierze pod uwagę zależności pomiędzy użytecznościami stanów.
2. Bezpośrednio uczy się:
   - modelu przejść $P(s'|s,a)$
   - funkcji nagrody $R(s)$

Algorytm: Adaptatywne Programowanie Dynamiczne

Agent ze stanu s wykonał akcję $\pi(s) = a$ docierając do stanu s' otrzymując nagrodę r' ($\langle s, a, s', r' \rangle$).

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**
    $P(w|s, a) \leftarrow M[w, s, a]/N[s, a]$
  $U^{\pi} \leftarrow$ Policy-Evaluation (π, P, R, U)

A jak wykonać krok Policy-Evaluation?[zadanie 6]



1. Algorytm po prostu uaktualnia wiedzę o modelu na podstawie próbek uczących. A no końcu, znając model świata oblicza U^{π} (Naturalnie (w pasywnym uczeniu się) nie ma konieczności uaktualniania U^{π} po każdym kroku, skoro z U^{π} nie korzystamy).
2. Jeśli obliczamy U^{π} w każdym kroku, to można użyć „starego" U^{π} jako punku początkowego w iteracji wartości, co bardzo przyspieszy algorytm.

Algorytm: Adaptatywne Programowanie Dynamiczne

Agent ze stanu s wykonał akcję $\pi(s) = a$ docierając do stanu s' otrzymując nagrodę r' ($\langle s, a, s', r' \rangle$).

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**
    $P(w|s, a) \leftarrow M[w, s, a]/N[s, a]$
  $U^\pi \leftarrow$ Policy-Evaluation (π, P, R, U)

A jak wykonać krok Policy-Evaluation?[zadanie 6]
Znamy model (T, R), więc układ równań Bellman'a lub iteracja wartości.

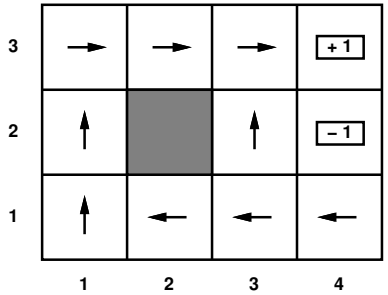
Algorytm: Adaptatywne Programowanie Dynamiczne
Agent ze stanu s wykonał akcję $\pi(s) = a$ docierając do stanu s' otrzymując nagrodę r' ($\langle s, a, s', r' \rangle$).
procedure PASSIVE-ADP(s, a, s', r')
if s' jest nowym stanem **then**
$U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$
$N[s, a] \leftarrow N[s, a] + 1$
$M[s', s, a] \leftarrow M[s', s, a] + 1$
for w **in** znane następniki stanu s (tzn. $M[w, s, a] > 0$) **do**
$P(w|s, a) \leftarrow M[w, s, a]/N[s, a]$
$U^\pi \leftarrow$ Policy-Evaluation (π, P, R, U)
A jak wykonać krok Policy-Evaluation?[zadanie 6]
Znamy model (T, R), więc układ równań Bellman'a lub iteracja wartości.

1. Algorytm po prostu uaktualnia wiedzę o modelu na podstawie próbek uczących. A no końcu, znając model świata oblicza U^π (Naturalnie (w pasywnym uczeniu się) nie ma konieczności uaktualniania U^π po każdym kroku, skoro z U^π nie korzystamy).
2. Jeśli obliczamy U^π w każdym kroku, to można użyć „starego" U^π jako punku początkowego w iteracji wartości, co bardzo przyspieszy algorytm.

ADP — Przykład

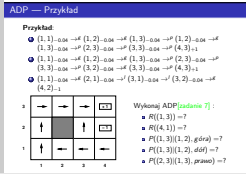
Przykład:

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



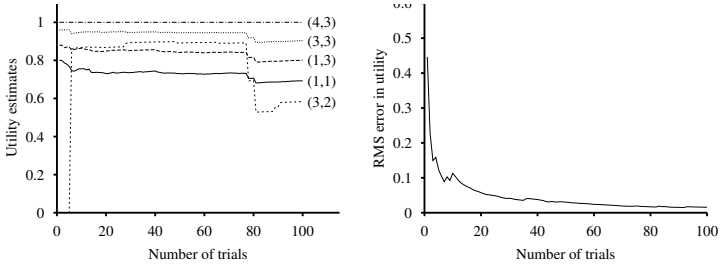
Wykonaj ADP[zadanie 7] :

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



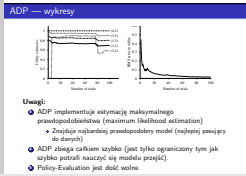
1. -0.04
2. Null
3. 3/3=1
4. Null
5. 2/3=0.66

ADP — wykresy



Uwagi:

1. ADP implementuje estymację maksymalnego prawdopodobieństwa (maximum likelihood estimation)
   - Znajduje najbardziej prawdopodobny model (najlepiej pasujący do danych)
2. ADP zbiega całkiem szybko (jest tylko ograniczony tym jak szybko potrafi nauczyć się modelu przejść).
3. Policy-Evaluation jest dość wolne.



1. Po lewej: znaczy wzrost skuteczności po 78 przebiegach (wtedy po raz pierwszy agent trafił na stan (4,2) z wartością -1. Po prawej: średnia ze 20 runów (100 przebiegów każdy)

Uczenie różnicowe (TDL)

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

Pomysł:

- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

Uczenie różnicowe (TDL)
ang. temporal difference (TD) learning (TDL)
Pomysł:
- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

1. $\alpha \in (0, 1]$ to współczynnik uczenia. Mówi o ile zwiększymy niepoprawną wartość w kierunku poprawnej.
2. Zauważmy: przykładowe przejście nie zawsze ma miejsce (!), bo czasem przejdę do pola $(1, 2)$ a czasem zostanę na polu $(1, 3)$. Ale prawd. tych przejść (a więc też aktualizacje!) będą odpowiadały modelowi. Częściej aktualizacja będzie zgodnie z $U^{\pi}(2, 3)$ niż z pozostałymi.

Uczenie różnicowe (TDL)

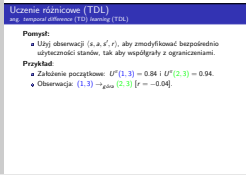
ang. *temporal difference* (TD) *learning* (TDL)

Pomysł:

- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

Przykład:

- Założenie początkowe: $U^\pi(1,3) = 0.84$ i $U^\pi(2,3) = 0.94$.
- Obserwacja: $(1,3) \rightarrow_{góra} (2,3)$ $[r = -0.04]$.



1. $\alpha \in (0, 1]$ to współczynnik uczenia. Mówi o ile zwiększymy niepoprawną wartość w kierunku poprawnej.
2. Zauważmy: przykładowe przejście nie zawsze ma miejsce (!), bo czasem przejdę do pola $(1,2)$ a czasem zostanę na polu $(1,3)$. Ale prawd. tych przejść (a więc też aktualizacje!) będą odpowiadały modelowi. Częściej aktualizacja będzie zgodnie z $U^\pi(2,3)$ niż z pozostałymi.

Uczenie różnicowe (TDL)

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

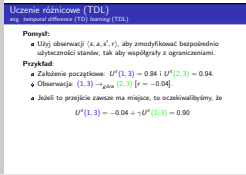
Pomysł:

- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

Przykład:

- Założenie początkowe: $U^\pi(1,3) = 0.84$ i $U^\pi(2,3) = 0.94$.
- Obserwacja: $(1,3) \rightarrow_{góra} (2,3)$ $[r = -0.04]$.
- Jeżeli to przejście zawsze ma miejsce, to oczekiwalibyśmy, że

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



1. $\alpha \in (0,1]$ to współczynnik uczenia. Mówi o ile zwiększymy niepoprawną wartość w kierunku poprawnej.
2. Zauważmy: przykładowe przejście nie zawsze ma miejsce (!), bo czasem przejdę do pola $(1,2)$ a czasem zostanę na polu $(1,3)$. Ale prawd. tych przejść (a więc też aktualizacje!) będą odpowiadały modelowi. Częściej aktualizacja będzie zgodnie z $U^\pi(2,3)$ niż z pozostałymi.

Uczenie różnicowe (TDL)

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

Pomysł:

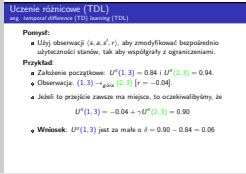
- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

Przykład:

- Założenie początkowe: $U^\pi(1,3) = 0.84$ i $U^\pi(2,3) = 0.94$.
- Obserwacja: $(1,3) \rightarrow_{góra} (2,3)$ $[r = -0.04]$.
- Jeżeli to przejście zawsze ma miejsce, to oczekiwalibyśmy, że

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

- **Wniosek**: $U^\pi(1,3)$ jest za małe o $\delta = 0.90 - 0.84 = 0.06$



1. $\alpha \in (0,1]$ to współczynnik uczenia. Mówi o ile zwiększymy niepoprawną wartość w kierunku poprawnej.
2. Zauważmy: przykładowe przejście nie zawsze ma miejsce (!), bo czasem przejdę do pola $(1,2)$ a czasem zostanę na polu $(1,3)$. Ale prawd. tych przejść (a więc też aktualizacje!) będą odpowiadały modelowi. Częściej aktualizacja będzie zgodnie z $U^\pi(2,3)$ niż z pozostałymi.

Uczenie różnicowe (TDL)

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

Pomysł:

- Użyj obserwacji $\langle s, a, s', r \rangle$, aby zmodyfikować bezpośrednio użyteczności stanów, tak aby współgrały z ograniczeniami.

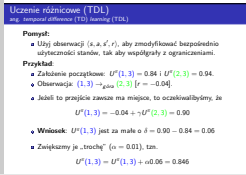
Przykład:

- Założenie początkowe: $U^\pi(1,3) = 0.84$ i $U^\pi(2,3) = 0.94$.
- Obserwacja: $(1,3) \rightarrow_{góra} (2,3)$ $[r = -0.04]$.
- Jeżeli to przejście zawsze ma miejsce, to oczekiwalibyśmy, że

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

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

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

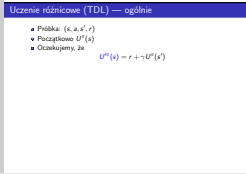


1. $\alpha \in (0, 1]$ to współczynnik uczenia. Mówi o ile zwiększymy niepoprawną wartość w kierunku poprawnej.
2. Zauważmy: przykładowe przejście nie zawsze ma miejsce (!), bo czasem przejdę do pola $(1,2)$ a czasem zostanę na polu $(1,3)$. Ale prawd. tych przejść (a więc też aktualizacje!) będą odpowiadały modelowi. Częściej aktualizacja będzie zgodnie z $U^\pi(2,3)$ niż z pozostałymi.

Uczenie różnicowe (TDL) — ogólnie

- Próbka: (s, a, s', r)
- Początkowo $U^\pi(s)$
- Oczekujemy, że

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



Uczenie różnicowe (TDL) — ogólnie

- Próbka: (s, a, s', r)
- Początkowo $U^\pi(s)$
- Oczekujemy, że
$$U'^\pi(s) = r + \gamma U^\pi(s')$$
- 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" $U^\pi(s)$:
$$U^\pi(s) \leftarrow U^\pi(s) + \alpha\Delta$$

Uczenie różnicowe (TDL) — ogólnie
Próbka: (s, a, s', r)
Początkowo $U^\pi(s)$
Oczekujemy, że
$U'^\pi(s) = r + \gamma U^\pi(s')$
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" $U^\pi(s)$:
$U^\pi(s) \leftarrow U^\pi(s) + \alpha\Delta$

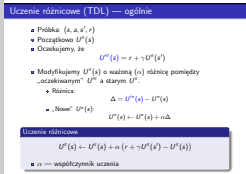
Uczenie różnicowe (TDL) — ogólnie

- Próbka: (s, a, s', r)
- Początkowo $U^\pi(s)$
- Oczekujemy, że
$$U'^\pi(s) = r + \gamma U^\pi(s')$$
- 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" $U^\pi(s)$:
$$U^\pi(s) \leftarrow U^\pi(s) + \alpha\Delta$$

Uczenie różnicowe

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

- α — współczynnik uczenia



Uczenie różnicowe (TDL) — algorytm

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

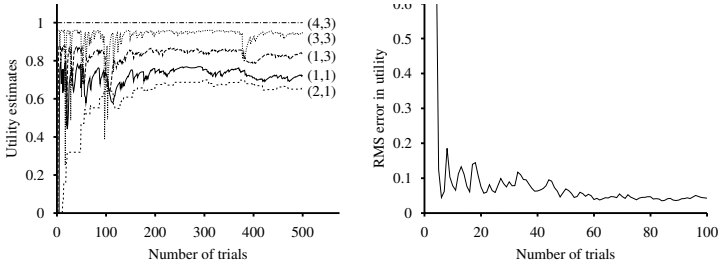
Uwagi:

1. Aktualizacja $U[s]$ nie uwzględnia akcji dostępnych i modelu przejść, ale to się odpowiednio uśredni.
2. TDL nie potrzebuje modelu środowiska, aby uaktualniać użyteczności stanów.
3. Jeżeli α w odpowiedni sposób zmniejsza się w czasie, to TDL gwarantuje zbieżność do optimum globalnego.

Uczenie różnicowe (TDL) — algorytm
procedure PASSIVE-TD(s, a, s', r')
if s' jest nowym stanem then
$U[s'] \leftarrow r'$
$U[s] \leftarrow U[s] + \alpha(R[s] + \gamma U[s'] - U[s])$
Uwagi:
1. Aktualizacja $U[s]$ nie uwzględnia akcji dostępnych i modelu przejść, ale to się odpowiednio uśredni.
2. TDL nie potrzebuje modelu środowiska, aby uaktualniać użyteczności stanów.
3. Jeżeli α w odpowiedni sposób zmniejsza się w czasie, to TDL gwarantuje zbieżność do optimum globalnego.

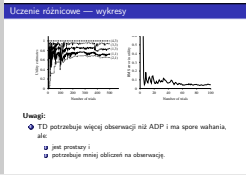
1. Obserwacja: jest to uczenie gradientowe
2. Uczenie się następuje w każdej próbce (vide. *stochastic gradient descent*)

Uczenie różnicowe — wykresy



Uwagi:

1. TD potrzebuje więcej obserwacji niż ADP i ma spore wahania, ale:
   1. jest prostszy i
   2. 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)

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)

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

- 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.

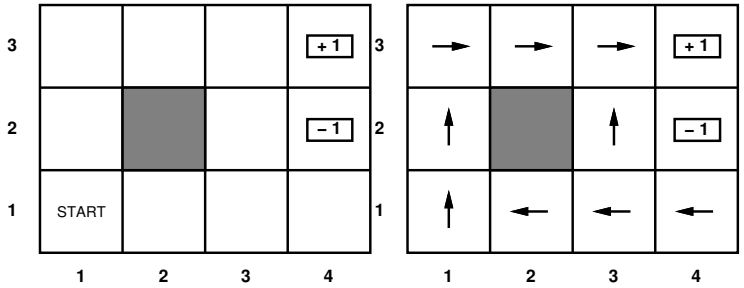
Stąd: możliwe są rozwiązania pośrednie:

- 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?

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.

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 (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 $\pi[s']$

1. Algorytm tutaj zwraca akcję do wykonania.

ADP dla uczenia aktywnego

Jak zmodyfikować ADP?

1. Musimy nauczyć się **całego modelu przejść**, a nie tylko przejść określonych przez (aktualne) π.
   - ADP nie ma z tym problemu: zbiera co tylko otrzymuje.

ADP dla uczenia aktywnego
Jak zmodyfikować ADP?
1. Musimy nauczyć się **całego modelu przejść**, a nie tylko przejść określonych przez (aktualne) π.
   - ADP nie ma z tym problemu: zbiera co tylko otrzymuje.

1. Czy wybierać w każdym kroku „aktualnie optymalną akcję” (zachłanną ze względu na U)? Nie. To nie jest mądre, bo nie ma eksploracji.

ADP dla uczenia aktywnego

Jak zmodyfikować ADP?

1. Musimy nauczyć się **całego modelu przejść**, a nie tylko przejść określonych przez (aktualne) π.
   - ADP nie ma z tym problemu: zbiera co tylko otrzymuje.
2. Agent **nie ma danej polityki**, więc Policy-Evaluation musi skorzystać z pełnego równania Bellman'a, żeby brać pod uwagę najlepsze możliwe akcje:

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

   - Można użyć Iteracji Wartości albo (Zmodyfikowanej) Iteracji Polityki

ADP dla uczenia aktywnego
Jak zmodyfikować ADP?
Musimy nauczyć się całego modelu przejść, a nie tylko przejść określonych przez (aktualne) π.
ADP nie ma z tym problemu: zbiera co tylko otrzymuje.
Agent nie ma danej polityki, więc Policy-Evaluation musi skorzystać z pełnego równania Bellman'a, żeby brać pod uwagę najlepsze możliwe akcje:
$U(s) = R(s) + \gamma \mathrm{max_a} \sum_{s'} \mathrm{P(s'|s,a)U(s')}$
Można użyć Iteracji Wartości albo (Zmodyfikowanej) Iteracji Polityki

1. Czy wybierać w każdym kroku „aktualnie optymalną akcję" (zachłanną ze względu na U)? Nie. To nie jest mądre, bo nie ma eksploracji.

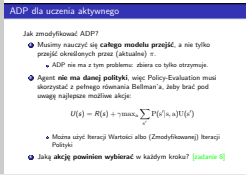
ADP dla uczenia aktywnego

Jak zmodyfikować ADP?

1. Musimy nauczyć się **całego modelu przejść**, a nie tylko przejść określonych przez (aktualne) π.
   - ADP nie ma z tym problemu: zbiera co tylko otrzymuje.
2. Agent **nie ma danej polityki**, więc Policy-Evaluation musi skorzystać z pełnego równania Bellman'a, żeby brać pod uwagę najlepsze możliwe akcje:

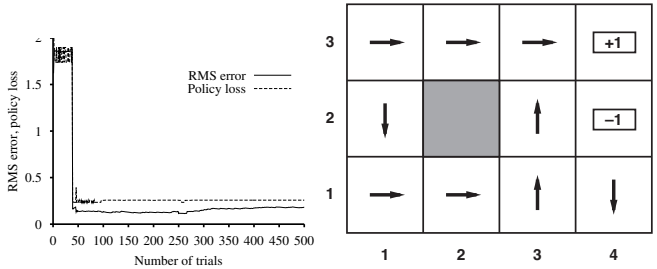
$$U(s) = R(s) + \gamma \mathrm{max_a} \sum_{s'} \mathrm{P(s'|s,a)U(s')}$$

   - Można użyć Iteracji Wartości albo (Zmodyfikowanej) Iteracji Polityki
3. Jaką **akcję powinien wybierać** w każdym kroku? [zadanie 8]

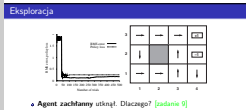


1. Czy wybierać w każdym kroku „aktualnie optymalną akcję" (zachłanną ze względu na U)? Nie. To nie jest mądre, bo nie ma eksploracji.

Eksploracja

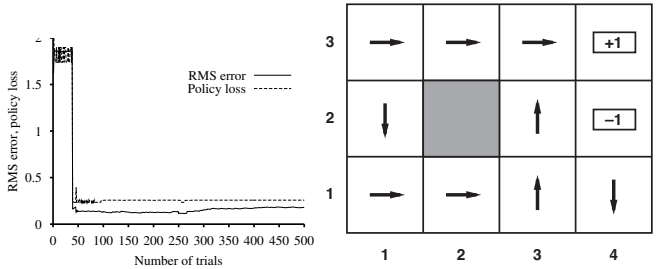


- **Agent zachłanny** utknął. Dlaczego? [zadanie 9]

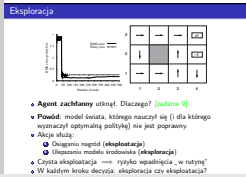


1. Agent zachłanny zawsze wybiera „aktualnie optymalna politykę”
2. Jest OK czy zmienić pracę? Większa wiedza -> potrzeba mniej eksploracji

Eksploracja

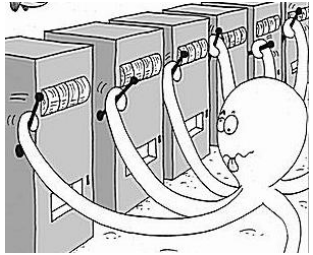


- **Agent zachłanny** utknął. Dlaczego? [zadanie 9]
- **Powód**: model świata, którego nauczył się (i dla którego wyznaczył optymalną politykę) nie jest poprawny.
- Akcje służą:
  1. Osiąganiu nagród (**eksploatacja**)
  2. Ulepszaniu modelu środowiska (**eksploracja**)
- Czysta eksploatacja $\implies$ ryzyko wpadnięcia „w rutynę"
- W każdym kroku decyzja: eksploracja czy eksploatacja?



1. Agent zachłanny zawsze wybiera „aktualnie optymalna politykę"
2. Jest OK czy zmienić pracę? Większa wiedza -> potrzeba mniej eksploracji

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 (wszystkich modeli $P(s'|s,a)$) jest najlepsza.

Problem wielorękiego bandyty
n automatów do gry.
gra → 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 (wszystkich modeli P(s'|s, a)) jest najlepsza.

Gittins index

- rozwiązania są zwykle obliczeniowo bardzo trudne (→ **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 nie działa

Gittins index
- rozwiązania są zwykle obliczeniowo bardzo trudne (→ **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 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

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**:**

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

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

Ciekawość i optymistyczna f. użyteczności

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

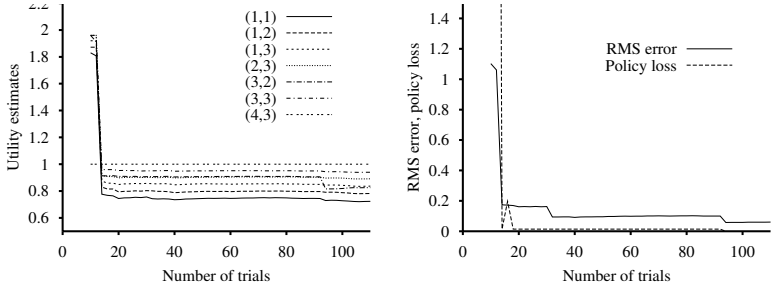
$$U^+(s) \leftarrow R(s) + \gamma \mathrm{max_a f}\left(\sum_{\mathrm{s'}} \mathrm{P(s'|s,a)U^+(s')}, \mathrm{N(s,a)}\right),$$

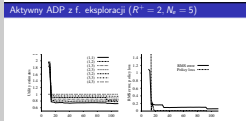
gdzie funkcja eksploracji waży **użyteczność** stanu i **„ciekawość"** (niewiedzę)

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

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:

1. Musimy **uczyć się modelu** $P(s'|s,a)$ tak jak ADP
2. Potrzebna jakaś funkcja eksploracji do wyboru akcji.

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

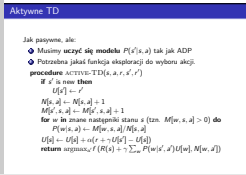
1. Uwaga: algorytm zwraca akcję do wykonania w następnym kroku.

Aktywne TD

Jak pasywne, ale:

1. Musimy **uczyć się modelu** $P(s'|s,a)$ tak jak ADP
2. 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 $\text{argmax}_{a'} f\left(R(s) + \gamma \sum_w P(w|s',a')U[w], N[w,a']\right)$



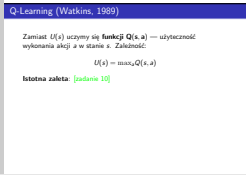
1. Uwaga: algorytm zwraca akcję do wykonania w następnym kroku.

Q-Learning (Watkins, 1989)

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

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

Istotna zaleta: [zadanie 10]



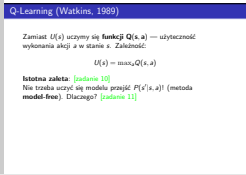
1. Ponieważ model nie jest potrzebny do wyboru najlepszej akcji.

Q-Learning (Watkins, 1989)

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

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

Istotna zaleta: [zadanie 10]
Nie trzeba uczyć się modelu przejść $P(s'|s, a)$! (metoda **model-free**). Dlaczego? [zadanie 11]



1. Ponieważ model nie jest potrzebny do wyboru najlepszej akcji.

Q-Learning (Watkins, 1989)

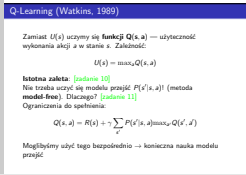
Zamiast $U(s)$ uczymy się **funkcji Q(s, a)** — użyteczność wykonania akcji a w stanie s. Zależność:

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

Istotna zaleta: [zadanie 10]
Nie trzeba uczyć się modelu przejść $P(s'|s, a)$! (metoda **model-free**). Dlaczego? [zadanie 11]
Ograniczenia do spełnienia:

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

Moglibyśmy użyć tego bezpośrednio → konieczna nauka modelu przejść



1. Ponieważ model nie jest potrzebny do wyboru najlepszej akcji.

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:

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

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:

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

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:

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

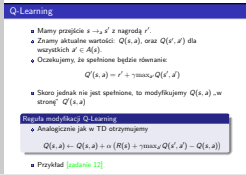
- Skoro jednak nie jest spełnione, to modyfikujemy $Q(s,a)$ „w stronę" $Q'(s,a)$

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 12]



Algorytm (TD) Q-Learning

procedure ACTIVE-Q(s, a, r, s', r')
  if s jest stanem terminalnym **then**
    $Q[s, None] \leftarrow r'$
  $N[s, a] \leftarrow N[s, a] + 1$
  $Q[s, a] \leftarrow Q[s, a] + \alpha\,(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
  return $\operatorname{argmax}_{a'} f\,(Q[s', a'], N[s', a'])$

Czy algorytmy TD-learning i Q-learning można zastosować do **znanego** MDP? [zadanie 13]

Algorytm (TD) Q-Learning
procedure ACTIVE-Q(s, a, r, s', r')
if s jest stanem terminalnym then
$Q[s, None] \leftarrow r'$
$N[s, a] \leftarrow N[s, a] + 1$
$Q[s, a] \leftarrow Q[s, a] + \alpha\,(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
return $\operatorname{argmax}_{a'} f\,(Q[s', a'], N[s', a'])$
Czy algorytmy TD-learning i Q-learning można zastosować do znanego MDP? [zadanie 13]

1. Jeśli, używamy ϵ-zachłannej eksploracji, to $N[s, a]$ nie jest potrzebne.
2. Tak. Można! I często się to robi ze względu, choćby, na prostotę tych algorytmów. W takim przypadku uczenie się funkcji U (klasyczny TD-Learning) upraszcza się, ponieważ nie trzeba nam wyznaczać modelu świata P (czyli sytuacja jest podobna jak w uczeniu pasywnym).

SARSA (State-Action-Reward-State-Action)

Podobne do Q-Learning, ale uczenie jest wykonywane po krotce doświadczenia $\langle s, a, r, s', a' \rangle$.
Q-Learning:

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

SARSA (State-Action-Reward-State-Action)
Podobne do Q-Learning, ale uczenie jest wykonywane po krotce doświadczenia $\langle s, a, r, s', a' \rangle$.
Q-Learning:
$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right).$

1. Dla strategii zachłannej względem Q algorytmy te są identyczne.
2. SARSA zbiegnie do optimum, jeśli każda akcja jest wykonywana nieskończenie wiele razy a polityka zbiega do polityki zachłannej względem Q. Np. jeśli polityka jest ϵ-zachłanna, gdzie $\epsilon = 1/t$ a t jest krokiem algorytmu.

SARSA (State-Action-Reward-State-Action)

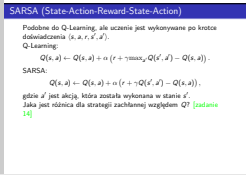
Podobne do Q-Learning, ale uczenie jest wykonywane po krotce doświadczenia $\langle s, a, r, s', a' \rangle$.
Q-Learning:

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

SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a) \right),$$

gdzie a' jest akcją, która została wykonana w stanie s'.
Jaka jest różnica dla strategii zachłannej względem Q? [zadanie 14]



1. Dla strategii zachłannej względem Q algorytmy te są identyczne.
2. SARSA zbiegnie do optimum, jeśli każda akcja jest wykonywana nieskończenie wiele razy a polityka zbiega do polityki zachłannej względem Q. Np. jeśli polityka jest ϵ-zachłanna, gdzie $\epsilon = 1/t$ a t jest krokiem algorytmu.

SARSA (State-Action-Reward-State-Action)

Podobne do Q-Learning, ale uczenie jest wykonywane po krotce doświadczenia $\langle s, a, r, s', a' \rangle$.
Q-Learning:

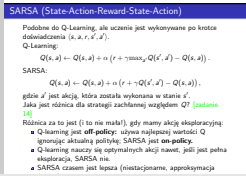
$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right).$$

SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q(s',a') - Q(s,a) \right),$$

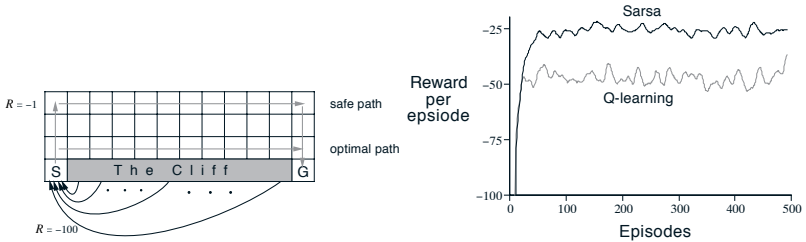
gdzie a' jest akcją, która została wykonana w stanie s'.
Jaka jest różnica dla strategii zachłannej względem Q? [zadanie 14]
Różnica za to jest (i to nie mała!), gdy mamy akcję eksploracyjną:

- Q-learning jest **off-policy:** używa najlepszej wartości Q ignorując aktualną politykę; SARSA jest **on-policy.**
- Q-learning nauczy się optymalnych akcji nawet, jeśli jest pełna eksploracja, SARSA nie.
- SARSA czasem jest lepsza (niestacjonarne, approksymacja



1. Dla strategii zachłannej względem Q algorytmy te są identyczne.
2. SARSA zbiegnie do optimum, jeśli każda akcja jest wykonywana nieskończenie wiele razy a polityka zbiega do polityki zachłannej względem Q. Np. jeśli polityka jest ϵ-zachłanna, gdzie $\epsilon = 1/t$ a t jest krokiem algorytmu.

SARSA vs. Q-Learing



(Sutton & Barto, 1998, str. 150)

- Wejście na klif powoduje -100 i powrót na start.
- $\epsilon = 0.1$
- Wykres pokazuje wyniki „online”.
- Q-learning nauczył się optymalnej ścieżki, ale „online” ma gorszy wynik, bo częściej spada.