Uczenie ze wzmocnieniem

Uczenie się systemu - każda autonomiczna zmiana w systemie zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania.

Rodzaje uczenia:

Z nauczycielem

Z krytykiem

Samoorganizacja

Uczenie ze wzmocnieniem

Literatura:

- Paweł Cichosz, ***Systemy uczące się***, Wydawnictwa Naukowo-Techniczne, Warszawa 2000, str. 712-792.
- Richard Sutton, Andrew G. Barto, ***Reinforcement Learning: An Introduction***, MIT Press, Cambridge, MA, 1998.
  http://www.cs.ualberta.ca/~sutton/book/the-book.html
- Stuart J.Russel, Peter Norvig, ***Artificial Intelligence***, Prentice-Hall, London, 2003, str. 598-645.

Plan wykładu

- Wieloetapowe procesy decyzyjne - typy procesów i środowisk
- Programowanie dynamiczne a metoda Monte Carlo
- Metoda różnic czasowych – podstawowy algorytm
- Eksploatacja a eksploracja
- Metody przyspieszania zbieżności - ślady aktywności
- Aproksymacja funkcji wartości stanów
- Metody kodowania stanów
- Przykłady zastosowań

Środowisko

Cechy środowiska w sztucznych systemach uczących się:

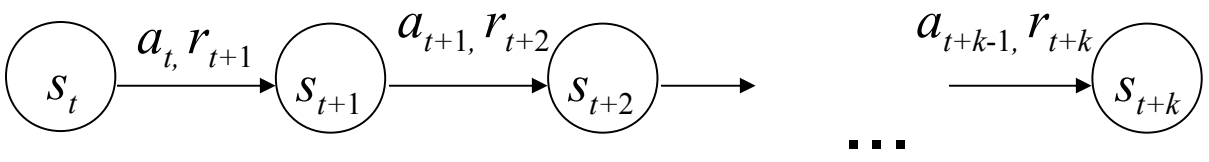
- przydziela nagrody i wyznacza bieżący stan
- jest niezależne od ucznia, czyli oznacza wszystko to, na co uczeń nie ma wpływu

Typy środowisk:

- stacjonarne / niestacjonarne (zmienne w czasie)
- deterministyczne / niedeterministyczne - taka sama akcja może spowodować przejście do różnych stanów, a przy przejściu do takiego samego stanu można uzyskać różne nagrody z tym, że wartości oczekiwane nagród i prawdopodobieństwa przejść są stałe
- niedeterministyczne o znanym / nieznanym modelu
- o parametrach ciągłych / dyskretnych
- o pełnej informacji o stanie / o niepełnej informacji o stanie

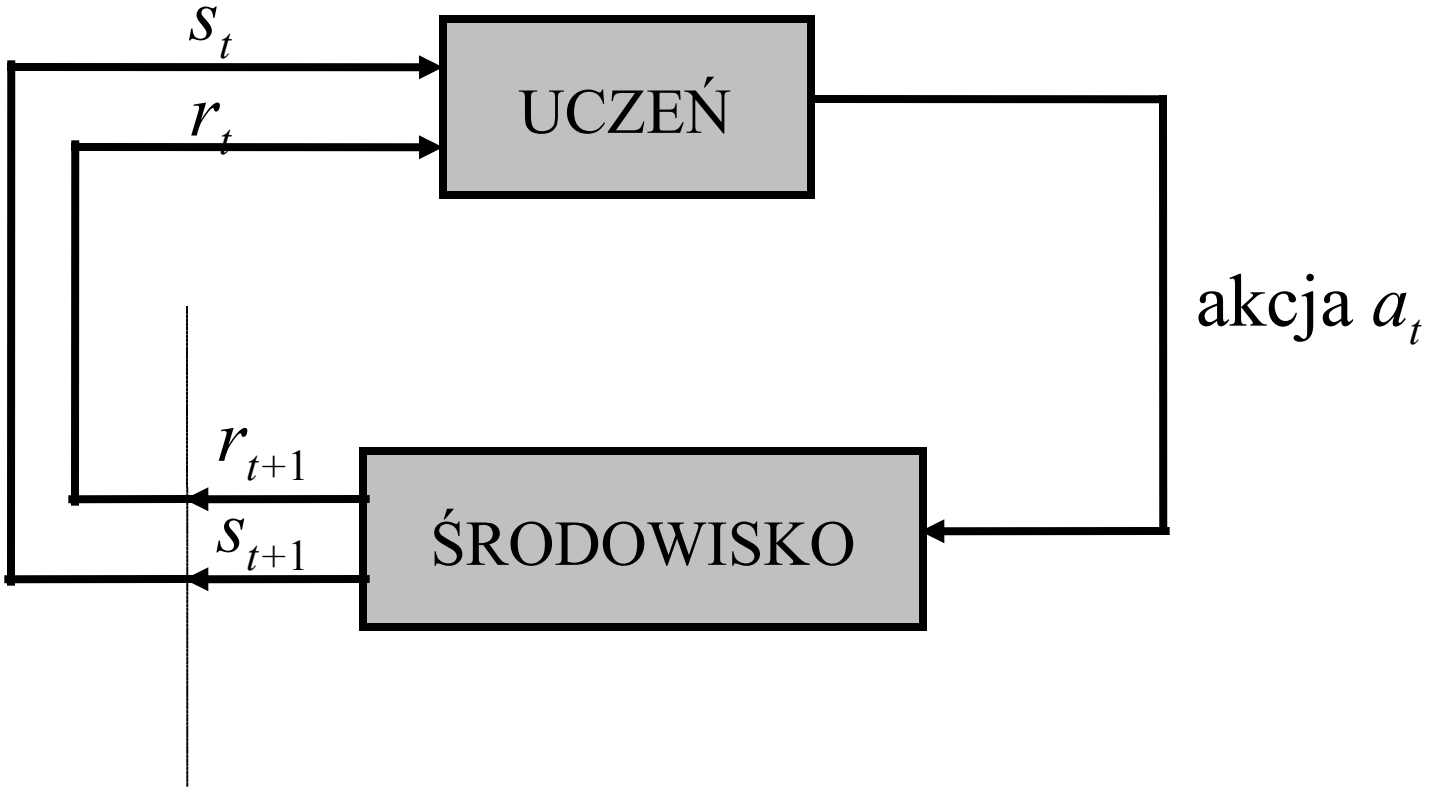
Wieloetapowe procesy decyzyjne

- Procesy polegające na wielokrotnej interakcji ucznia (agenta) ze środowiskiem. W wyniku podjęcia jednej z możliwych akcji a_t w danym stanie s_t, środowisko przechodzi do nowego stanu s_{t+1} i zwraca nagrodę r_{t+1}



- Celem uczenia jest maksymalizacja nagród uzyskanych w ciągu całego procesu, niezależnie od stanu początkowego
- Wniosek: należy szukać optymalnej strategii (*policy*) zachowania ucznia (wyboru odpowiedniej akcji w każdym ze stanów)

Ogólny schemat uczenia się w interakcji ze środowiskiem



Proces decyzyjny Markowa

Proces decyzyjny Markowa można zdefiniować jako czwórkę (S, A, ψ, δ):

S - skończony zbiór stanów

A - skończony zbiór akcji

$\psi(s,a)$ - funkcja wzmocnienia - zmienna losowa o wartościach rzeczywistych oznaczająca nagrodę po wykonaniu akcji a w stanie s

$\delta(s,a)$ - funkcja przejść stanów - zmienna losowa o wartościach ze zbioru S oznaczająca następny stan po wykonaniu akcji a w stanie s

w każdym kroku t nagroda r_{t+1} jest realizacją zmiennej losowej $\psi(s_t,a_t)$ a stan s_{t+1} jest realizacją zmiennej losowej $\delta(s_t,a_t)$, prawd. przejścia oraz średnia nagroda:

$$P_{ss'}^{a} = \Pr(\delta(s,a) = s')$$

$$R_{s}^{a} = E(\psi(s,a))$$

Zadanie optymalizacji w procesach epizodycznych

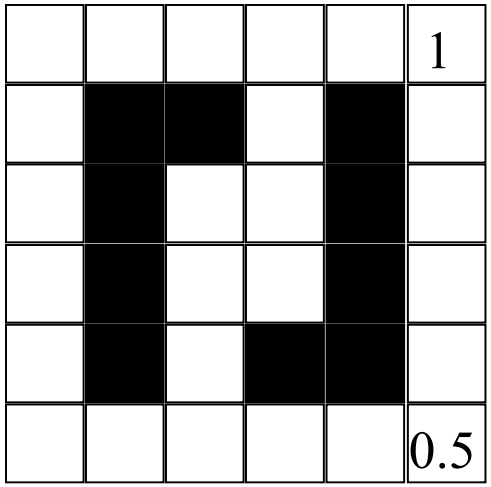
Cel maksymalizacji:

$$E\left[\sum_{k=0}^{n}\gamma^{k}r_{t+k+1}\right]=E[r_{t+1}+\gamma r_{t+2}+\gamma^{2}r_{t+3}+\ldots+\gamma^{n}r_{t+n+1}]$$

gdzie r_t - nagroda w kroku t, γ - współczynnik dyskontowania, $0\leq\gamma\leq 1$, reguluje ważność krótko i długoterminowych nagród.

Zastosowanie współczynnika dyskontowania wynika z pewnych praktycznych spostrzeżeń: nagrody warto zdobywać jak najszybciej (zadania *do-sukcesu*), kary jak najdłużej odwlekać (zadania *do-porażki*)

Przykład GRID-6



$A = \{\rightarrow, \leftarrow, \downarrow, \uparrow\},$

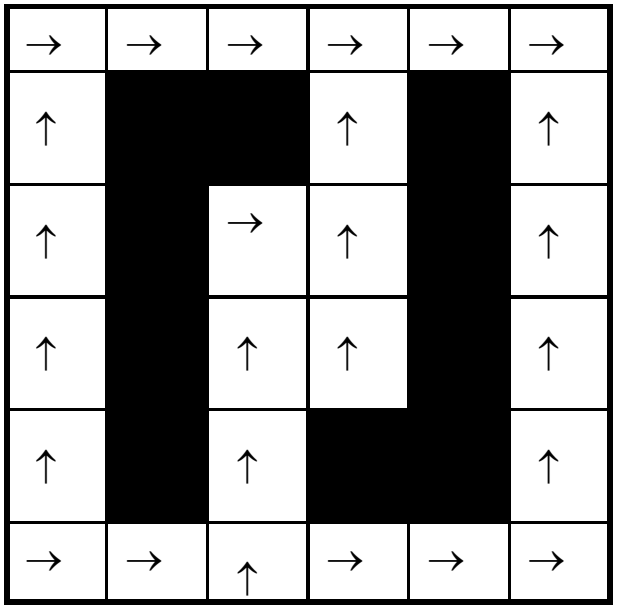
$S = \{(0,0),(0,1),...,(5,5)\}$

$P_{ij} = 1$ gdy akcja nie prowadzi w pole niedostepne

$$R = \begin{cases} 1 & \text{dla } s' = (0,5) \\ 0.5 & \text{dla } s' = (5,5) \\ 0 & \text{dla pozostalych} \end{cases}$$

Przykład GRID-6 – przykładowe strategia

π_1



Funkcje wartości

Funkcja użyteczności (wartości) stanu s_t przy strategii π :

$$V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{n} \gamma^{k} r_{t+k+1} \mid s = s_t\right]$$

Funkcja wartości pary [*stan*,*akcja*]: (s_t , a_t) przy strategii π :

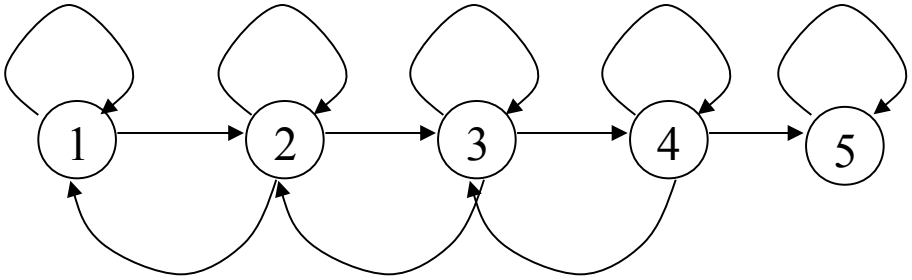
$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{n} \gamma^{k} r_{t+k+1} \mid s = s_t, a = a_t\right]$$

Przy danej strategii π dla każdego stanu *s* zachodzi równanie:

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$$

Przykład GRAF-5

S = {1,2,3,4,5}, A={0,1}



$$P_{s,s+1}(a) = \begin{cases} 0.2 \text{ dla } s < 5, a = 0 \\ 0.8 \text{ dla } s < 5, a = 1 \\ 0 \quad \text{dla } s = 5 \end{cases}$$

$$P_{s,s}(a) = \begin{cases} 0.4 \text{ dla } 1 < s < 5, a = 0 \\ 0.1 \text{ dla } 1 < s < 5, a = 1 \\ 0.8 \text{ dla } s = 1, a = 0 \\ 0.2 \text{ dla } s = 1, a = 1 \\ 1 \text{ dla } s = 5 \end{cases}$$

$$P_{s,s-1}(a) = \begin{cases} 0.4 \text{ dla } 1 < s < 5, a = 0 \\ 0.1 \text{ dla } 1 < s < 5, a = 1 \\ 0 \quad \text{dla } s = 1 \text{ lub } s = 5 \end{cases}$$

Nagroda za akcję a w stanie s:

$$R(s,a) = \begin{cases} 0.2 \text{ dla } s = 4, a = 0 \\ 0.8 \text{ dla } s = 4, a = 1 \\ 0 \text{ dla pozostalych} \end{cases}$$

Funkcja wartości a strategia

Strategia π' jest lepsza od strategii π jeśli dla każdego s lub (s,a):

$$V^{\pi'}(s) \geq V^{\pi}(s) \quad \text{lub} \quad Q^{\pi'}(s,a) \geq Q^{\pi}(s,a)$$

oraz istnieje takie s lub (s,a), że zachodzi:

$$V^{\pi'}(s) > V^{\pi}(s) \quad \text{lub} \quad Q^{\pi'}(s,a) > Q^{\pi}(s,a)$$

Zachłanny wybór strategii na podstawie przybliżonych wartości V lub Q:

$$\pi(s) = \arg\max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V(s'))$$

$$\pi(s) = \arg\max_a Q(s,a)$$

$P^a_{ss'}$ - prawdopodobieństwo przejścia od stanu s do s' przy wykonaniu akcji a

$R^a_{ss'}$ - średnia nagroda przy przejściu od s do s' dzięki a

Porównanie funkcji *V* oraz *Q*

- Użycie funkcji wartości stanu *V(s)* w przypadku niedeterministycznego środowiska wymaga każdorazowej symulacji wykonania jednego kroku naprzód w celu znalezienia akcji optymalnej lub stworzenia oddzielnej funkcji akcji (met. *Actor-Critic*)
- Użycie funkcji *Q(s,a)* wymaga stosowania większych tablic lub bardziej złożonych aproksymatorów funkcji, ale ułatwia wybór akcji

Strategia optymalna

Strategia π^* jest optymalna jeśli dla każdej strategii π oraz dla każdego stanu s:

$$V^{\pi^*}(s) \geq V^{\pi}(s)$$

lub dla każdej akcji w każdym stanie:

$$Q(s, \pi^*(s)) \geq Q(s, \pi(s))$$

Zachłanna metoda wyboru akcji:

$$\pi^*(s) = \arg\max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V^{\pi^*}(s'))$$

$$\pi^*(s) = \arg\max_a Q^{\pi^*}(s, a)$$

$P^a_{ss'}$ - prawdopodobieństwo przejścia od stanu s do s' przy wykonaniu akcji a

$R^a_{ss'}$ - średnia wartość nagrody przy przejściu od s do s' dzięki akcji a

Zachłanna metoda wyboru akcji z użyciem funkcji użyteczności obliczonej dla strategii optymalnej jest realizacją strategii optymalnej

Metody szukania optymalnej strategii

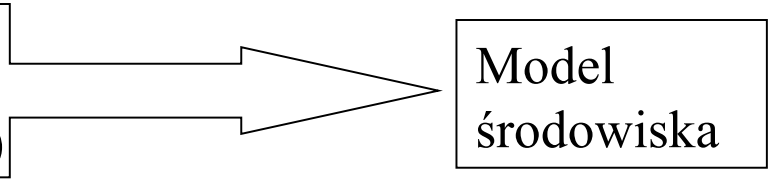
- Programowanie dynamiczne
- Metoda Monte Carlo
- Metoda różnic czasowych (TD)

Programowanie dynamiczne

Prawdopodobieństwo przejścia ze stanu *s* do *s'* po wykonaniu akcji *a*, oraz średnia wartość nagrody związanej z tym zdarzeniem:

$$P^a_{ss'} = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$$

$$R^a_{ss'} = E(r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s')$$



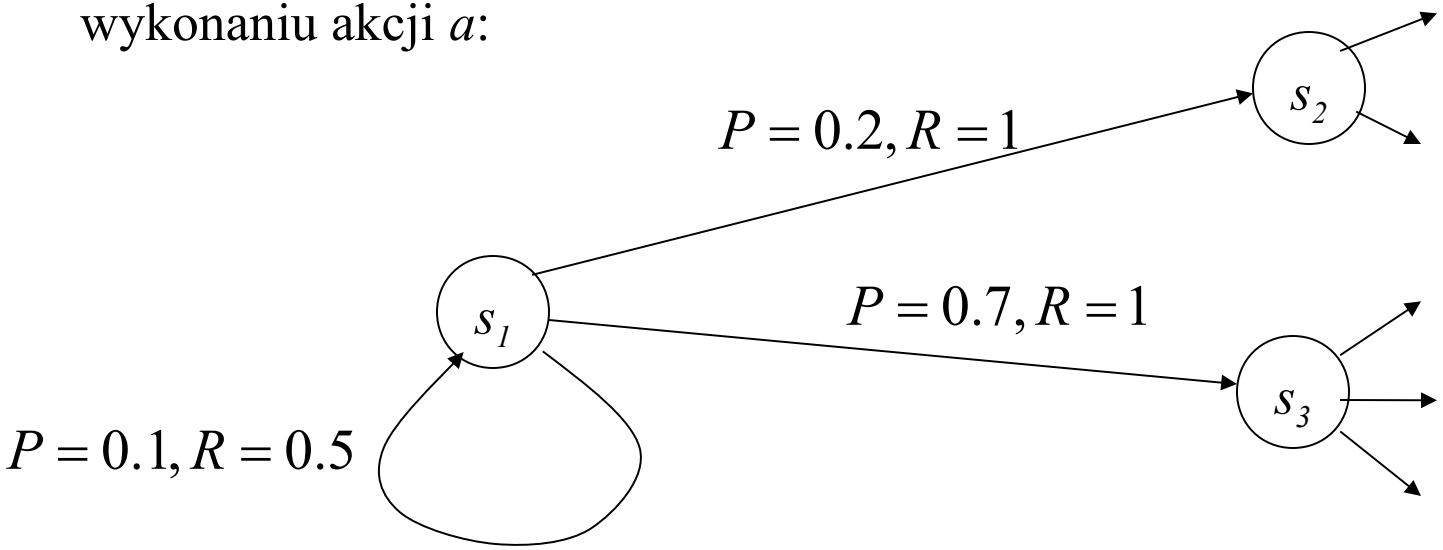
Równania równowagi Bellmana dla reprezentacji [*stan*] oraz [*stan*,*akcja*] i strategii π, ($\pi(s)$ - akcja w stanie s zgodna ze strategią π):

$$V^\pi(s) = \sum_{s'} P^{\pi(s)}_{ss'} [R^{\pi(s)}_{ss'} + \gamma V^\pi(s')]$$

$$Q^\pi(s,a) = \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma Q^\pi(s', \pi(s'))]$$

Programowanie dynamiczne

Przykładowy graf przejść ze stanu $s=s_1$ do $s' \in \{s_1, s_2, s_3\}$, po wykonaniu akcji a:



$$V^{\pi}(s) = \sum_{s'} P_{ss'}^{\pi(s)} [R_{ss'}^{\pi(s)} + \gamma V^{\pi}(s')]$$

$$V(s_1) = 0.2(1 + \gamma V(s_2)) + 0.7(1 + \gamma V(s_3)) + 0.1(0.5 + \gamma V(s_1))$$

stąd:

$$V(s_1) = \frac{0.95 + 0.2\gamma V(s_2) + 0.7\gamma V(s_3)}{1 - 0.1\gamma}$$

Programowanie dynamiczne

Wyprowadzenie równania równowagi dla funkcji wartości stanu s:

$$V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \mid s = s_t\right] =$$

$$= E_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \mid s = s_t\right] =$$

$$= \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \mid s' = s_{t+1}\right]] =$$

$$= \sum_{s'} P_{ss'}^{\pi(s)}[R_{ss'}^{\pi(s)} + \gamma V^{\pi}(s')]$$

Programowanie dynamiczne

Równania optymalności Bellmana dla reprezentacji [*stan*] oraz [*stan*,*akcja*]:

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')]$$

$$Q^*(s,a) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \max_{a'} Q^*(s',a')]$$

$V^*(s), Q^*(s,a)$ - wartości odpowiadające strategii optymalnej

Programowanie dynamiczne

Metody wyznaczania wartości V lub Q dla danej strategii:

- Rozwiązanie układu równań o $|S|$ (lub $|S \times A|$ w przypadku reprezentacji [*stan,akcja*]) niewiadomych
- Iteracyjne na podstawie równań równowagi Bellmana (o udowodnionej zbieżności)

Metody wyznaczania optymalnej strategii:

- Iteracja strategii - naprzemienne obliczanie przybliżonych wartości $V^{\pi}(s)$ lub $Q^{\pi}(s,a)$ dla wszystkich stanów przy danej (początkowo losowej) strategii π oraz wyznaczanie lepszej strategii π' dla $V^{\pi}(s)$ lub $Q^{\pi}(s,a)$ do momentu, gdy w kolejnych dwóch iteracjach strategia π pozostanie niezmienna
- Iteracja wartości - obliczanie $V(s)$ lub $Q(s,a)$ stosując zachłanną metodę wyboru akcji do momentu, gdy wartości $V(s)$ lub $Q(s,a)$ przestaną się zmieniać

Iteracyjne obliczanie wartości stanów

obliczanie wartości stanów dla strategii π :

mając dane: π, P^{π}, R^{π}

powtarzaj dla wszystkich s:

$$\widetilde{V}^{\pi}_{i+1}(s) \leftarrow \sum_{s'} P^{\pi(s)}_{ss'} [R^{\pi(s)}_{ss'} + \gamma \widetilde{V}^{\pi}_{i}(s')]$$

aż nastąpi w kroku i $\quad \max_{s \in S} \left| \widetilde{V}^{\pi}_{i}(s) - \widetilde{V}^{\pi}_{i-1}(s) \right| \leq \varepsilon$

Iteracja strategii dla reprezentacji [*stan*]

naprzemienne wyznaczanie strategii (początkowo losowej) oraz wartości stanów dotąd aż strategie w 2 kolejnych cyklach iteracji przestaną się różnić: $\pi_{k-1} = \pi_k$

$$\pi_1 \rightarrow \tilde{V}^{\pi_1} \rightarrow \pi_2 \rightarrow \tilde{V}^{\pi_2} \rightarrow \ldots \rightarrow \pi_k \rightarrow \tilde{V}^{*}$$

obliczanie wartości stanów dla strategii π :

iteracyjne obliczanie wartości stanów dla strategii π
lub
metodą rozwiązywania układu równań

wyznaczanie nowej strategii π':

dla wszystkich *s:*

$$\pi'(s) = \arg\max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma \tilde{V}^{\pi}(s'))$$

Iteracja wartości dla reprezentacji [*stan*]

mając dane: P^{π}, R^{π}

powtarzaj dla wszystkich *s:*

$$\widetilde{V}_{k+1}(s) \leftarrow \max_{a} \sum_{s'} P^{a}_{ss'}[R^{a}_{ss'} + \gamma \widetilde{V}_{k}(s')]$$

aż nastąpi w kroku k $\max_{s \in S} \left| \widetilde{V}_{k}(s) - \widetilde{V}_{k-1}(s) \right| \leq \varepsilon$

Programowanie dynamiczne - wady i zalety

Wady:

- konieczność znajomości modelu środowiska (prawdopodobieństw przejść pomiędzy stanami dla wszystkich możliwych akcji i oczekiwanych wartości nagród)
- taki sam nakład obliczeń dla stanów mniej i częściej odwiedzanych w trakcie eksploatacji

Zalety:

- pewność znalezienia rozwiązania w przypadku metody analitycznej oraz zbieżność metod iteracyjnych przy odpowiedniej eksploracji

Metody Monte Carlo

Obliczanie funkcji wartości stanów lub par [*stan*, *akcja*] dla pewnej strategii π metodą uśredniania nagród z wielu epizodów.

$$\widetilde{V}^{\pi}(s) = \frac{\sum_{e=1}^{L}\sum_{i=0}^{n_e}\gamma^{i} r_{e,t+i+1}}{L},$$

gdzie L - liczba epizodów, n_e – liczba kroków e-tego epizodu

Wyznaczanie strategii optymalnej: np. metodą iteracji strategii

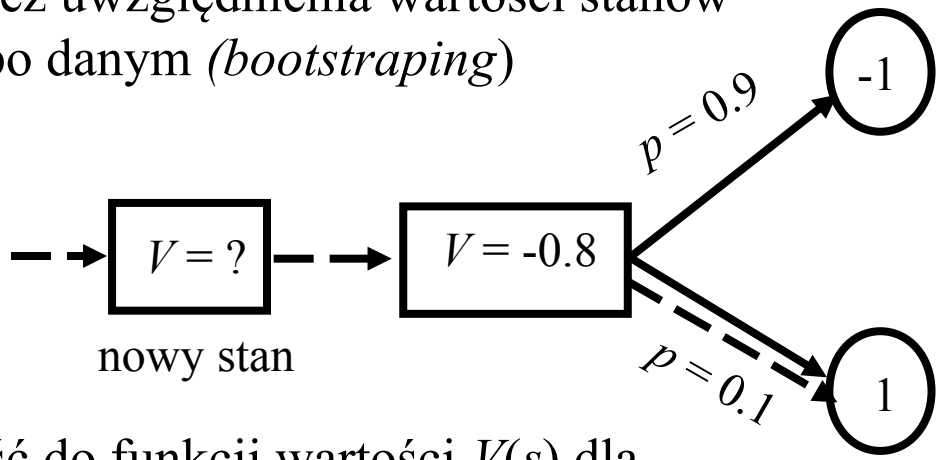
$$\pi_1 \to \widetilde{V}^{\pi_1} \to \pi_2 \to \widetilde{V}^{\pi_2} \to \ldots \to \pi_k \to \widetilde{V}^{*}$$

lub metodą iteracji wartości

Metody Monte Carlo - wady i zalety

Wady:

- Wymóg epizodyczności zadań
- Wymagana duża eksploracja
- Powolna zbieżność - obliczenie funkcji wartości nowego stanu bez uwzględnienia wartości stanów następujących po danym *(bootstraping*)



Zalety:

- Pewna zbieżność do funkcji wartości $V(s)$ dla ustalonej strategii przy odpowiedniej eksploracji
- Nie jest wymagana znajomość modelu środowiska

Metody różnic czasowych - ogólny algorytm

Zainicjuj $Q(s,a)$ lub $V(s)$
Repeat (dla kolejnych epizodów):
  Zainicjuj s
  Repeat (dla kolejnych kroków epizodu):
    obserwuj aktualny stan s_t;
    wybierz akcję a_t do wykonania w stanie s_t;
    wykonaj akcję a_t;
    obserwuj wzmocnienie r_{t+1} i następny stan s_{t+1};
    ucz się na podstawie doświadczenia
      $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$;
  until s jest stanem końcowym
until spełniony warunek końca

Metody różnic czasowych – cechy charakterystyczne

- Uczenie z krytykiem (bez informacji o właściwych decyzjach)
- Nie są znane optymalna strategia, metoda dojścia do optymalnej strategii ani model środowiska
- Uczenie się na zasadzie „prób i błędów'' – potrzebna jest eksploracja, co może wiązać się z kosztami
- Uczenie się oraz wykonywanie zadań (eksploatacja systemu) odbywają się jednocześnie
- Skupienie się na „obiecujących'' rejonach przestrzeni rozwiązań

Metody różnic czasowych – TD(0)

Najbardziej popularna metoda uczenia:

- Q-learning

Zalety metod TD:

- nie jest wymagany model środowiska
- możliwość uczenia w czasie rzeczywistym (*online-learning*)
  - zastosowanie w przypadku niestacjonarnego środowiska
  - duża uniwersalność zastosowań np. w środowiskach niestacjonarnych (gry planszowe)
- dobra zbieżność

Algorytm Q-learning

Algorytm Q-learning z aktualizacją wartości par [*stan*,*akcja*] niezależną od aktualnej strategii wyboru akcji (*off-policy*)

Zainicjuj $Q(s,a)$
Repeat (dla kolejnych epizodów):
  Zainicjuj s
  Repeat (dla kolejnych kroków epizodu):
    1.) Wybierz akcję z przyjętą metodą eksploracji,
     np:z prawdop. $1-\varepsilon$ wykonaj akcję a w stanie s
        o najwyższej wartości Q lub akcję losową
        z prawdop. ε przechodząc do stanu s'
    2.) Zmodyfikuj wartość akcji a w stanie s:

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

  until s jest stanem końcowym
until spełniony warunek końca

Eksploatacja i eksploracja

Przykłady strategii wyboru akcji w trakcie uczenia:

- maksimum
- losowa
- ε-zachłanna
- softmax

Strategia uczenia ε-zachłanna :

- z prawdopodobieństwem ε wybierz akcję losowo
- z prawdopodobieństwem 1-ε wybierz akcję: $a = \arg\max_a Q(s,a)$

Strategia uczenia softmax - wybór akcji zgodnie z rozkładem Bolzmanna (prawdopodobieństwo wylosowania akcji proporcjonalne do jej funkcji wartości):

$$P(a) = \frac{\exp\left(\frac{Q(s,a)}{T}\right)}{\sum_{b \in A(s)} \exp\left(\frac{Q(s,b)}{T}\right)}$$

Aproksymacja i kodowanie

Aproksymacja funkcji wartości – przedstawienie funkcji wartości stanów lub par [*stan*,*akcja*] w postaci modelu parametrycznego funkcji (struktury) o odpowiednio dobranych (nauczonych) wartościach parametrów

$$\vec{\theta}_t = (\theta_t(1), \theta_t(2), ..., \theta_t(n))$$

Kodowanie stanów – transformacja parametrów stanów do nowej przestrzeni cech

$$s \rightarrow \vec{\phi}_s$$

Wydobywanie cech - kodowanie

Przekształcenie wektorów z pierwotnej przestrzeni stanów $s = [s_1, s_2,..., s_N]$ (np. układu figur na szachownicy) do przestrzeni cech istotnych dla określenia wartości stanu:

$$s \to \vec{\phi}(s) = \vec{\phi}_s = [\phi_s(1), \phi_s(2),..., \phi_s(M)]$$

Cele:

- Uzyskanie cech istotnych dla określenia wartości stanów (wykorzystanie wiedzy o problemie w celu redukcji złożoności problemu)
- Uzyskanie większej liczby cech by ułatwić aproksymację funkcji użyteczności (wartości)
- Zwiększenie uogólniania poprzez agregację stanów o podobnej wartości

Aproksymatory funkcji

Przykłady:

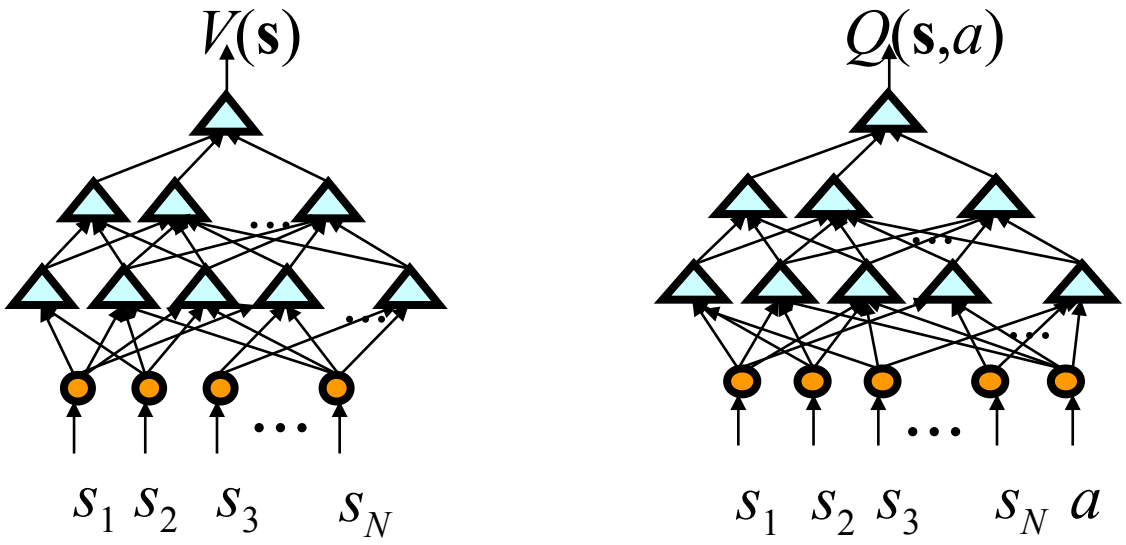
- Aproksymator liniowy $V(s) = \vec{\theta}^T \vec{\phi}_s = \sum_{i=1}^{M} \theta(i)\phi_s(i)$
- Sieci o podstawie radialnej (*Radial Basis Functions* – RBF)
- Wielomiany stopnia > 1
- Sztuczne sieci neuronowe (SNN)
- Systemy rozmyte

Zalety:

- Oszczędność miejsca przy dużych zbiorach stanów lub par [*stan*,*akcja*]
- Możliwość uogólniania wiedzy dla stanów pośrednich
- Brak dyskretyzacji w przypadku ciągłych parametrów stanów lub akcji

Aproksymator SSN

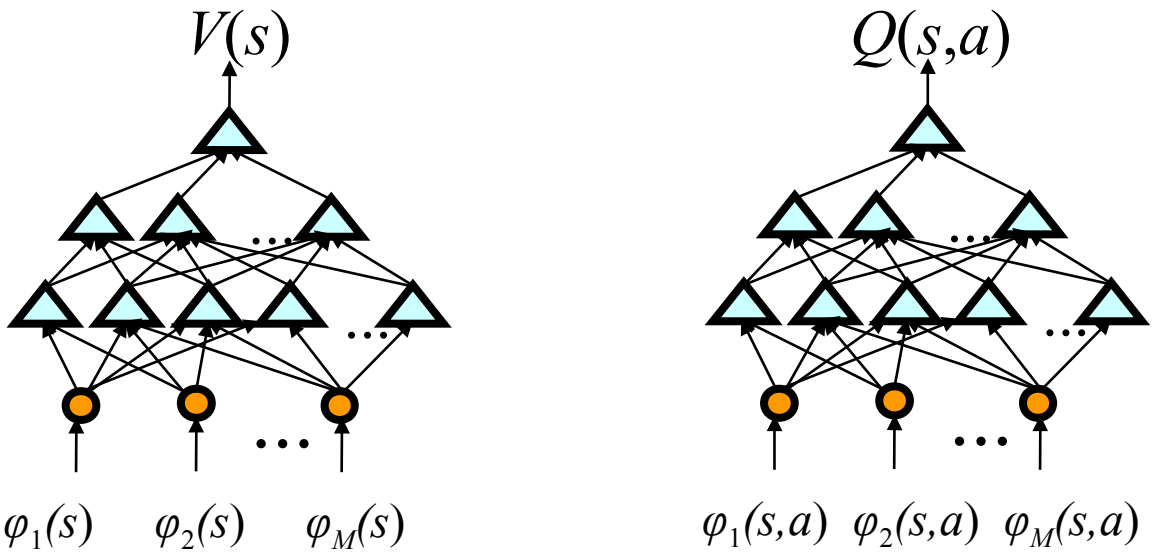
Bez kodowania parametrów stanu i akcji:



- Wektorowi parametrów modelu odpowiada wektor wag sieci: $\vec{\theta} = \vec{w}$
- Poprawa wartości oceny odbywa się poprzez zmianę wektora wag w kierunku największego spadku funkcji błędu – gradientu. Gradient funkcji błędu względem wag oblicza się zwykle metodą propagacji wstecznej błędu.

Aproksymator SSN

Z kodowaniem parametrów stanu i akcji:



- Wektorowi parametrów modelu odpowiada wektor wag sieci: $\vec{\theta} = \vec{w}$
- Poprawa wartości oceny odbywa się poprzez zmianę wektora wag w kierunku największego spadku funkcji błędu – gradientu. Gradient funkcji błędu względem wag oblicza się zwykle metodą propagacji wstecznej błędu.

Aproksymatory funkcji - definicje

Wartości stanów lub par [*stan*,*akcja*] reprezentowane są za pomocą funkcji zależnej od wektora parametrów θ:

$$V(s) = F_{\vec{\theta}}(s)$$

$$Q(s,a) = F_{\vec{\theta}}(s,a)$$

Wektor parametrów:

$$\vec{\theta}_t = (\theta_t(1), \theta_t(2), ..., \theta_t(n))$$

Kryterium optymalizacji:

$$MSE(\vec{\theta}_t) = \sum_{s \in S} \left(V^{\pi}(s) - V_t(s) \right)^2,$$

$V^{\pi}(s)$ – poszukiwana wartość stanu s dla strategii π
$V_t(s)$ – aktualna wartość stanu s

Gradientowa metoda aproksymacji funkcji wartości stanów

$$\vec{\theta}_{t+1} = \vec{\theta}_t - \frac{1}{2}\alpha\nabla_{\vec{\theta}_t}\left(V^{\pi}(s_t) - V_t(s_t)\right)^2$$
$$= \vec{\theta}_t + \alpha\left(V^{\pi}(s_t) - V_t(s_t)\right)\nabla_{\vec{\theta}_t} V_t(s_t)$$

parametry funkcji wartości modyfikowane są w kierunku maksymalnego spadku funkcji błędu *MSE*

gdzie gradient $\nabla_{\vec{\theta}_t} f(\vec{\theta}_t) = \left(\frac{\partial f(\vec{\theta}_t)}{\partial \vec{\theta}_t(1)}, \frac{\partial f(\vec{\theta}_t)}{\partial \vec{\theta}_t(2)}, \ldots, \frac{\partial f(\vec{\theta}_t)}{\partial \vec{\theta}_t(n)}\right)$

Przyjmując przybliżenie:

$$V^{\pi}(s_t) \cong r_{t+1} + \gamma V_t(s_{t+1})$$

Otrzymujemy algorytm aktualizacji wartości stanu:
(następny slajd)

Gradientowa metoda aproksymacji funkcji wartości stanów - TD(0)

Zainicjuj $V(s) = F(\vec{\theta}, s) = F_{\vec{\theta}}(s)$

Repeat (dla kolejnych epizodów):

  Zainicjuj s

  Repeat (dla kolejnych kroków epizodu):

    Wybierz i wykonaj akcję a w stanie s zgodnie ze

    strategią określoną przez $V(s) = F_{\vec{\theta}}(s)$ i eksploracją

$$\delta \leftarrow r + \gamma V(s') - V(s)$$

$$\vec{\theta} \leftarrow \vec{\theta} + \alpha \delta \nabla_{\vec{\theta}} V(s)$$

$$s \leftarrow s'$$

  until s jest stanem końcowym

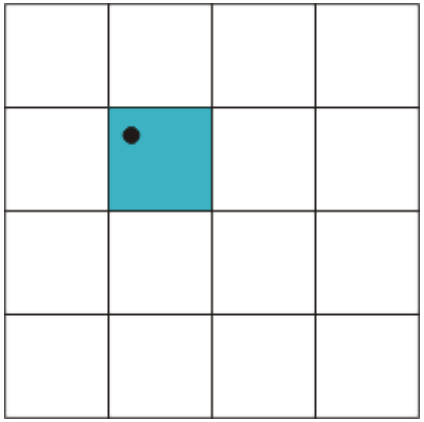
until spełniony warunek końca

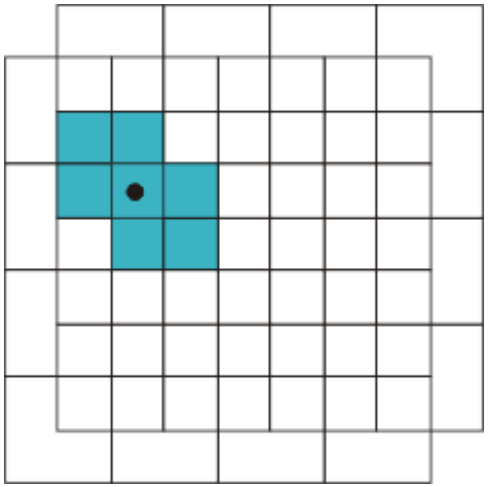
Metody kodowania stanów o parametrach ciągłych

Metody kodowania (obliczania cech):

- Kodowanie metodą pokryć (CMAC, *tile coding*)
- Kodowanie przybliżone (*coarse coding*)
- Kodowanie prototypowe (*Kanerva coding*)

Kodowanie metodą pokryć





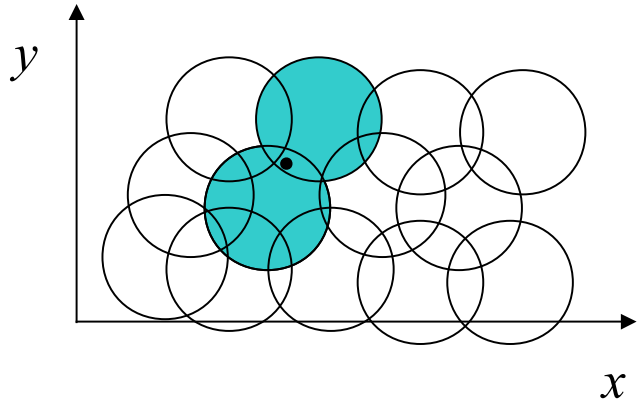
aproksymacja liniowa funkcji wartości stanu:

$$V(s) = \vec{\theta}^T \vec{\phi}_s = \sum_{i=1}^{M} \theta(i)\phi_s(i) \qquad \vec{\phi}_s \quad \text{- wektor cech stanu}$$

gradient funkcji wartości: $\nabla_{\vec{\theta}} V(s) = \vec{\phi}_s = [\phi_s(1), \phi_s(2), ..., \phi_s(M)]$

Kodowanie przybliżone

Kodowanie przybliżone dla 2-wymiarowej przestrzeni stanów - każde pole jest związane z jedną cechą binarną, równą 1 jeśli stan znajduje się wewnątrz pola:



Licząc po kolejnych wierszach
od lewej do prawej wektor cech:

$$\vec{\phi}_s = [0,1,0,0,0,1,0,0,0,0,0,0,0]$$

Przykładowe zastosowanie: aproksymator liniowy z wykorzystaniem zbioru cech:

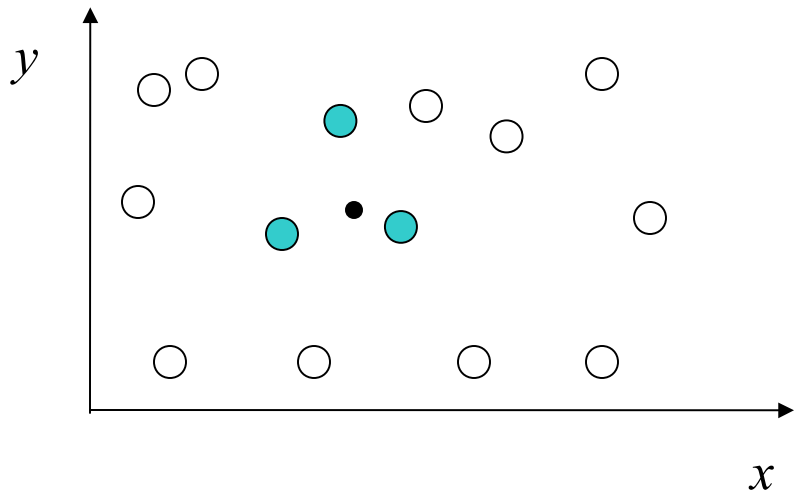
$$V(s) = \vec{\theta}^T \vec{\phi}_s = \sum_{i=1}^{M} \theta(i)\phi_s(i)$$

$\vec{\phi}_s$ - wektor cech stanu

gradient funkcji wartości: $\nabla_{\vec{\theta}} V(s) = \vec{\phi}_s = [\phi_s(1), \phi_s(2), ..., \phi_s(M)]$

Kodowanie prototypowe (przybliżone, rozproszone)

Kodowanie prototypowe dla przykładowej 2-wymiarowej przestrzeni stanów - każdy prototyp stanu jest związany z jedną cechą binarną, równą 1 jeśli spełnione jest kryterium odległości np. euklidesowej:



Licząc po kolejnych wierszach od lewej do prawej, nowy wektor cech:

$$\vec{\phi}_s = [0,0,1,0,0,0,1,0,1,0,0,0,0,0]$$

Prototypowe stany lub pary [*stan, akcja*] są początkowo wybierane losowo. Dodatkowo, w bardziej zaawansowanych metodach mogą być przemieszczane w celu większego ich skupienia w ważniejszych obszarach przestrzeni stanów

Kodowanie prototypowe (kodowanie Kanervy)

Prototypy przedstawione są w postaci wektorów cech (najczęściej binarnych). Każdy stan również przedstawiany jest w postaci wektora cech, który jest następnie porównywany do prototypów pod względem stopnia zgodności (np. odwrotność odległości Hamminga). W nowej przestrzeni cech, których liczba odpowiada liczbie prototypów, cecha *n*-ta przyjmuje wartość 1, wtedy i tylko wtedy gdy stopień zgodności wektora cech stanu z *n*-tym prototypem przekracza ustaloną wartość progową. Prototypy mogą zawierać wartości dowolne (możliwość uogólniania).
Przykład:

prototypy:
0001010100101110
1100101101010001
0011010100001100
1010111000111101
1111111110000011
0000000000001111
0000000000000001

wektor cech stanu:
0001010100001110

próg zgodności = 80%

wektor cech w nowej przestrzeni:
$\vec{\phi}_s = [1,0,1,0,0,0,0]$