Jednoczesne planowanie i uczenie się

Jednoczesne planowanie i uczenie się

- Poprzednio nauczyliśmy się jak wygląda rozwiązanie problemu uczenia się gdy agent uczy się bezpośrednio ze swojego doświadczenia.
- Pokazaliśmy też w jaki sposób agent może zaplanować swoje zachowanie znając z góry model.
- Jak można połączyć obie metody?

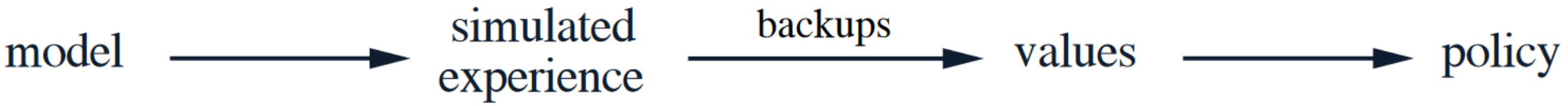
Model

- Zastanówmy się najpierw czym jest model.
- Model $\mathcal{M}$ definiujemy jako aproksymację procesu decyzyjnego Markowa (S, A, P, R, β) *parametryzowaną przez* θ*:*

$$\mathcal{M} = (S_\theta, A_\theta, P_\theta, R_\theta, \beta_\theta)$$

  - Dla uproszczenia często zakładamy, że S i A są znane:

$$\mathcal{M} = (S, A, P_\theta, R_\theta, \beta_\theta)$$



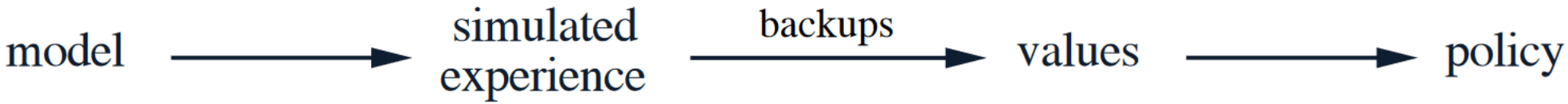
Model

- Zastanówmy się najpierw czym jest model.
- Model $\mathcal{M}$ definiujemy jako aproksymację procesu decyzyjnego Markowa (S, A, P, R, β) *parametryzowaną przez* θ*:*

$$\mathcal{M} = (S_\theta, A_\theta, P_\theta, R_\theta, \beta_\theta)$$

  - Dla uproszczenia często zakładamy, że S i A są znane:

$$\mathcal{M} = (S, A, P_\theta, R_\theta, \beta_\theta)$$



- Oznacza to, że problem znalezienia modelu możemy sprowadzić do problemu *uczenia nadzorowanego.*

Model

- Po wyestymowaniu modelu $\mathcal{M}$ problem staje się prosty – wystarczy rozwiązań proces decyzyjny Markowa: $(S, A, P_\theta, R_\theta, \beta_\theta)$ za pomocą metod programowania dynamicznego lub poprzez planowanie oparte na próbkowaniu (*sample-based planning*).

Planowanie oparte na próbkowaniu

- Idea jest prosta – mając gotowy model $\mathcal{M}$ generujemy za jego pomocą sample, które następnie wykorzystujemy do uczenia za pomocą metod bezmodelowych (Monte-Carlo, SARSA, etc.).

Model

- Co jednak gdy estymacja modelu jest niedokładna?
- Uczenie będzie **nieefektywne**, agent będzie w stanie nauczyć się optymalnej strategii dla modelu, ale nie dla rzeczywistego procesu.
- Jak temu przeciwdziałać?

Model

- Co jednak gdy estymacja modelu jest niedokładna?
- Uczenie będzie **nieefektywne**, agent będzie w stanie nauczyć się optymalnej strategii dla modelu, ale nie dla rzeczywistego procesu.
- Jak temu przeciwdziałać?
- Zrezygnować z uczenia opartego o model.

Model

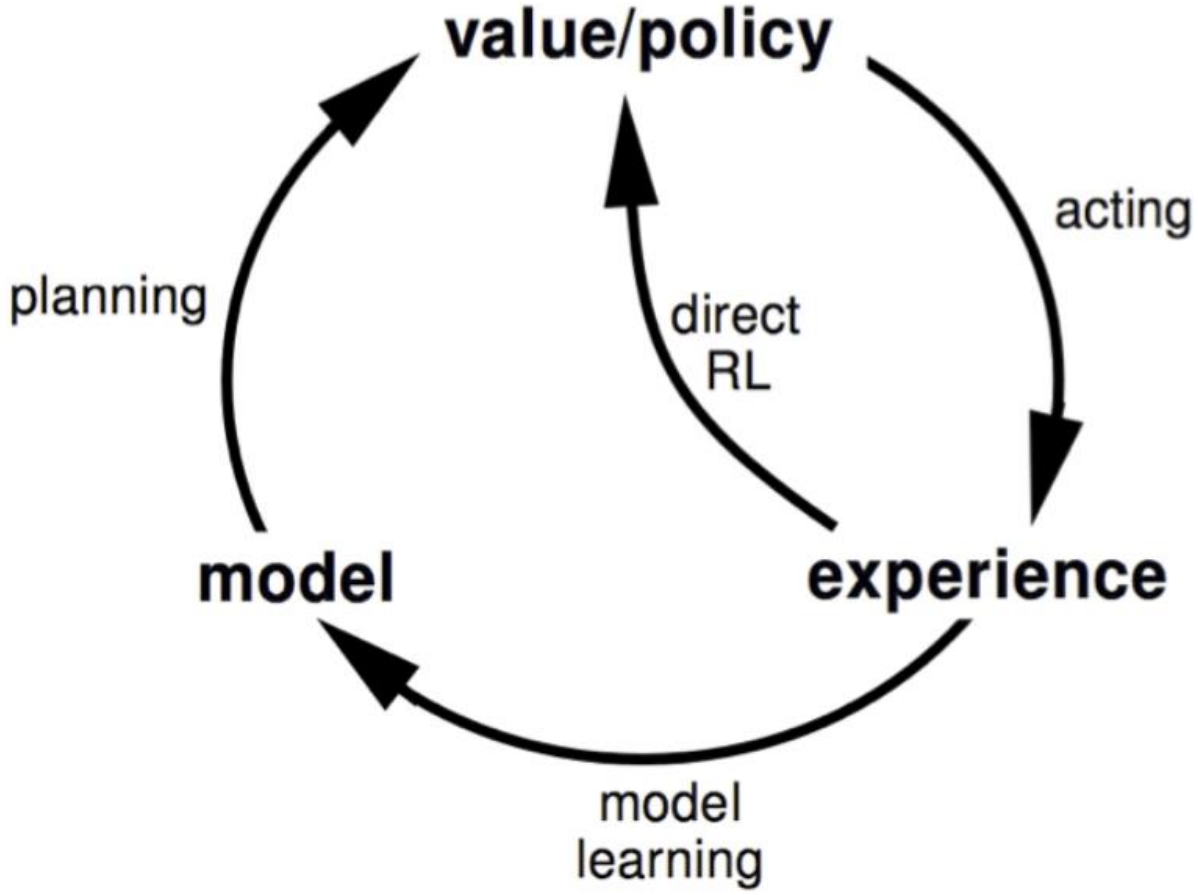
- Co jednak gdy estymacja modelu jest niedokładna?
- Uczenie będzie **nieefektywne**, agent będzie w stanie nauczyć się optymalnej strategii dla modelu, ale nie dla rzeczywistego procesu.
- Jak temu przeciwdziałać?
- Zrezygnować z uczenia opartego o model.
- Na bieżąco oceniać i poprawiać model.

Dyna-Q

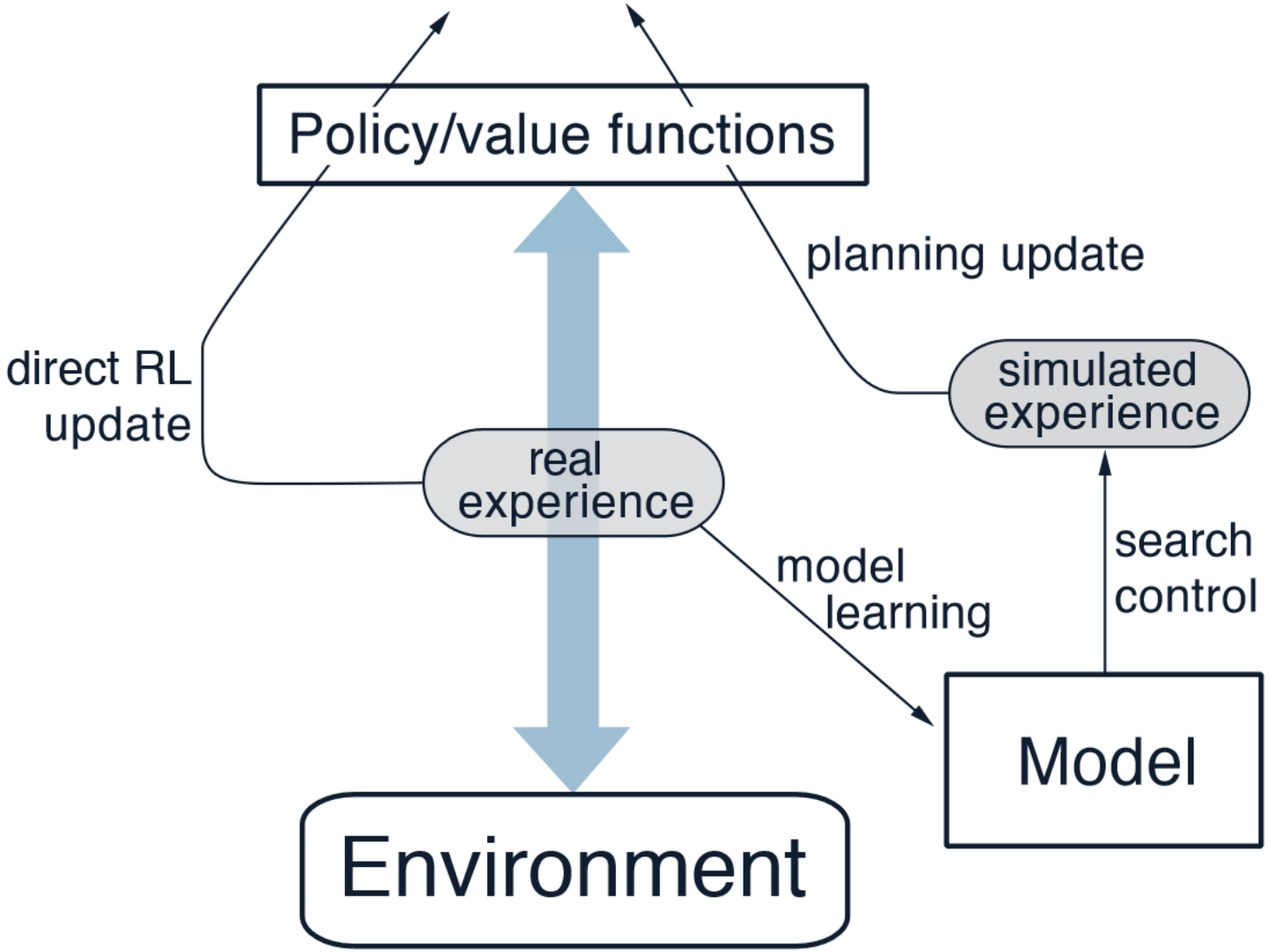
Dyna-Q

- Jest to hybrydowa metoda uczenia.
- Agent jednocześnie czerpie wiedzę z rzeczywistego doświadczenia, jak i z symulowanego na postawie modelu.

Dyna-Q


value/policy
acting
direct
RL
planning
model
experience
model
learning

Dyna-Q


Policy/value functions
planning update
direct RL
update
simulated
experience
real
experience
model
learning
search
control
Model
Environment

Dyna-Q

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S,Q)

(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

$S \leftarrow$ random previously observed state

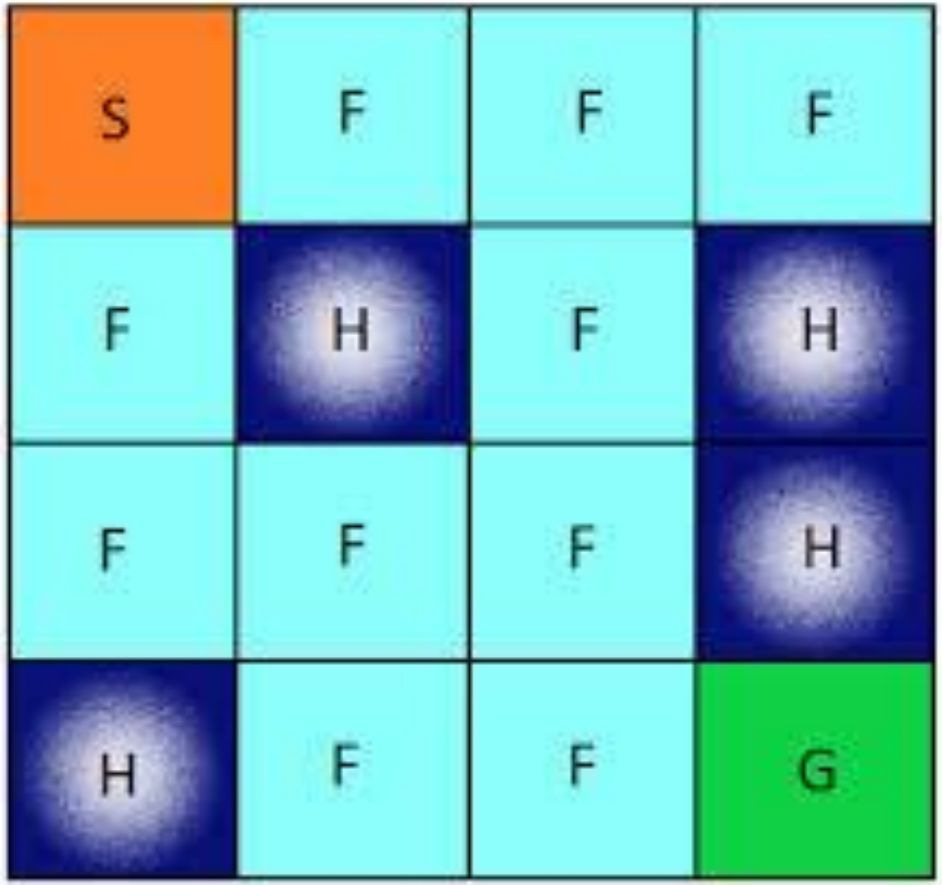
$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Przykład

Frozen Lake cd.



- Celem agenta jest przejść z punktu S do Punktu G.
- Agent może iść po lodzie (pola oznaczone literą F), musi unikać wpadnięcia do przerębli (pola oznaczone jako H).
- Lód jest śliski; idąc przed siebie z pewnym prawdopodobieństwem p może się poślizgnąć i przesunąć w lewo lub w prawo w stosunku do swojej wyjściowej pozycji.

Dyna-Q+

- Dyna-Q w swojej podstawowej formie jest w stanie efektywnie rozwiązywać deterministyczne problemy.
- Nie radzi sobie jednak z problemami stochastycznymi.
- Wynika to z omawianego już wcześniej problemu wyboru pomiędzy eksploracją a eksploatacją przez agenta.
- Rozwiązaniem jest prosta modyfikacja algorytmu, która umożliwia mu skorzystanie z odpowiedniej heurystyki wspomagającej wybór decyzji.

Dyna-Q+

- W przypadku bazowego algorytmu problematyczne jest to, że istnieje wiele par (s, a), które od dawna nie są odwiedzane i dla których oszacowanie funkcji Q może być błędne.
- Konieczne jest wymuszenie na agencie ponownych odwiedzin w takiej parze.

Dyna-Q+

- W algorytmie Dyna-Q+ agent dla każdej pary (s, a) przechowuje informację na temat tego jak dawno temu ją odwiedził.
- Ta informacja służy do wyliczania wartości prostej heurystyki, która motywuje agenta do odwiedzania dawno nieodwiedzanej pary (s, a).

Dyna-Q+

- Mając przejście (s, a, R, s'), które ostatni raz było odwiedzone τ kroków temu możemy wyznaczyć wartość nagrody jako:

$$R + \kappa\sqrt{\tau}$$

dla małego $\kappa > 0$

  - Wyrażenie $\kappa\sqrt{\tau}$ możemy traktować jako specjalną nagrodę za odwiedzenie dawno nieodwiedzanego pola. Im wyższe κ tym efekt jest silniejszy.

- Tak wyznaczoną nagrodę możemy wykorzystać do uaktualnienia wartości funkcji Q na etapie planowania:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(R + \kappa\sqrt{\tau} + \beta \max_a Q(s', a) - Q(s, a))$$