



Zpětnovazební učení

Michaela Walterová

Jednoocí slepým, 17. 4. 2019

V minulých dílech jste viděli

Tři paradigmatata strojového učení:

1) Učení s učitelem (supervised learning)

Trénovací data: vstup a požadovaný výstup, například obrázek a popis obrázku

Učení = optimalizace vnitřních parametrů modelu, gradientní metody

2) Učení bez učitele (unsupervised learning)

Model nezná požadovaný výstup

Shlukování na základě podobnosti dat, snaha správně zreprodukovat vstup

3) Zpětnovazební učení (reinforcement learning)

Hledání optimální strategie v daném prostředí

Teorie řízení, robotika, hraní deskových či počítačových her



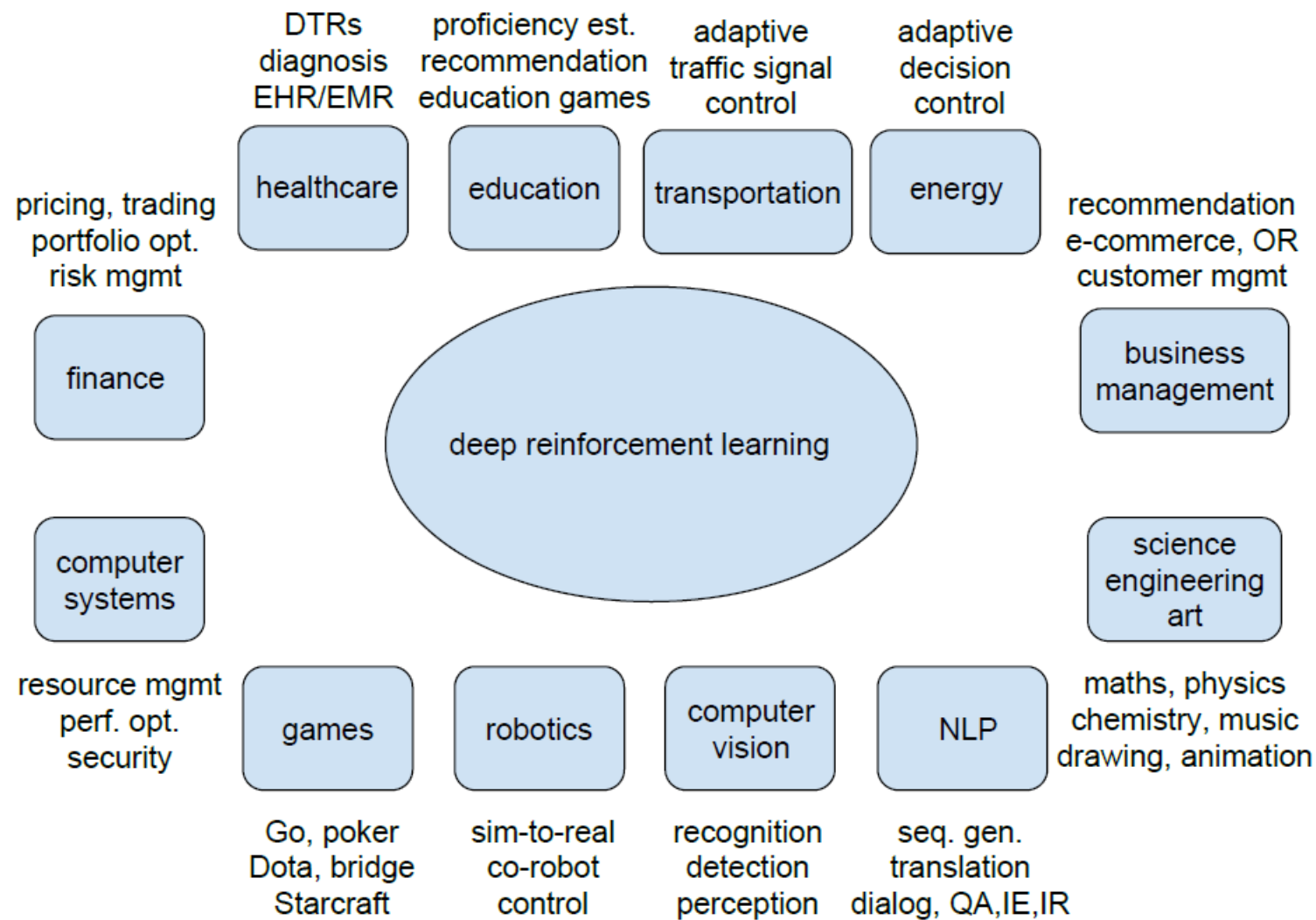
Hluboké zpětnovazební učení na matfyzu: <http://ufal.mff.cuni.cz/courses/npfl122>

(přednáší Milan Straka)

Volně dostupná učebnice: <http://incompleteideas.net/book/RLbook2018.pdf>

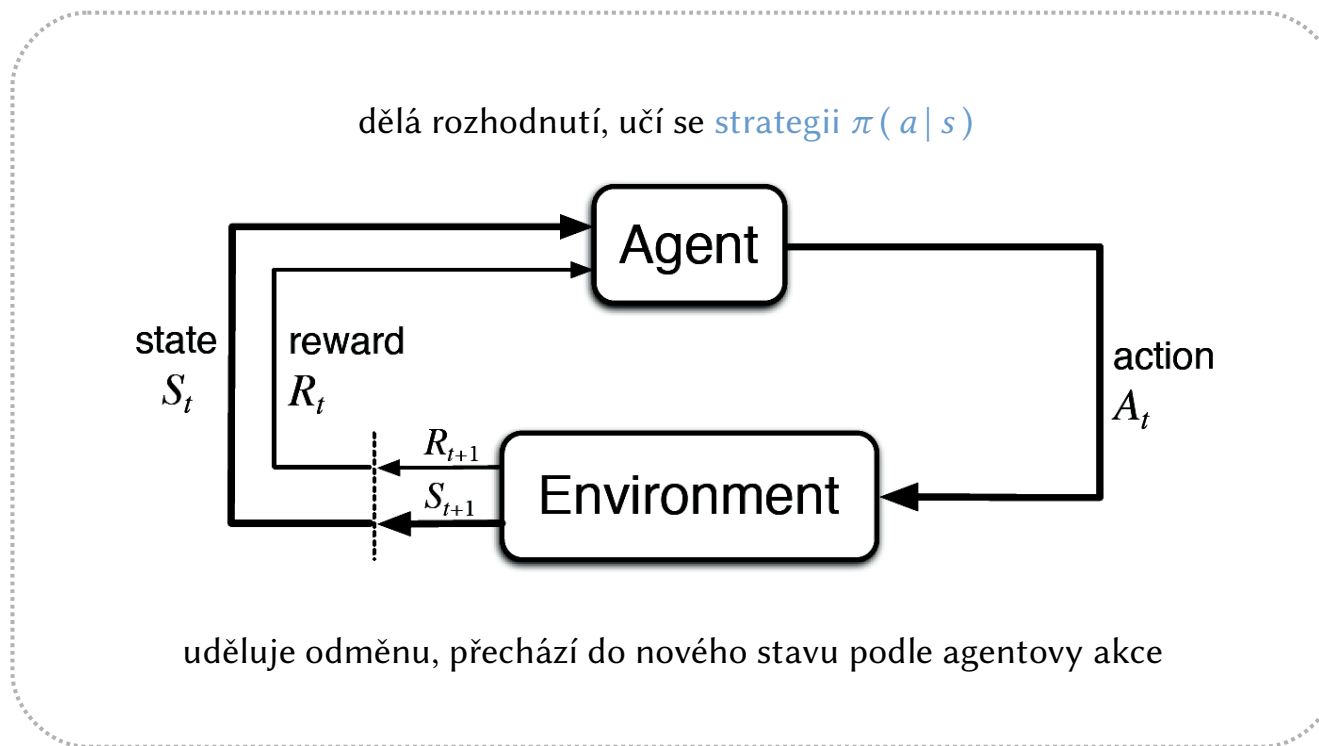
(Sutton, R.S. a Barto, A.G.)

Kde se používá zpětnovazební učení?



Markovův rozhodovací proces

Ústřední otázka: Jak se mám zachovat v konkrétní situaci dané vnějšími okolnostmi?



Dynamika prostředí (kterou agent nezná, ale učí se jí přizpůsobovat)

$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Trajektorie $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

Hodnotové funkce

Celkový zisk G_t odpovídající dané trajektorii (od času t) = součet odměn R_t

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

Agent hledá takovou strategii, která **maximalizuje celkový očekávaný zisk** G_0

Hodnotové funkce = nakolik je výhodné nacházet se v daném stavu (*state-value function*), nebo v něm provést danou akci (*action-value function*)

hodnota stavu s při následování strategie π

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = s] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

hodnota akce a ve stavu s při následování strategie π

$$q_{\pi}(s, a) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a] = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Optimální hodnota a optimální strategie

Pro každý stav (a akci) lze nalézt trajektorii, která maximalizuje celkový zisk → optimalizace hodnotové funkce

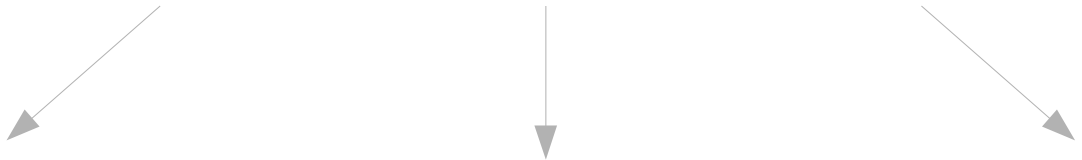
Optimální strategie π_* by tedy byla taková, která v každém stavu zvolí akci o nejvyšší (optimální) hodnotě

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s)$$

$$q_*(s, a) \doteq \max_{\pi} q_{\pi}(s, a)$$

Nejednoznačnost - například různé výherní strategie v šachách

Která akce má nejvyšší hodnotu? Která strategie je v daném prostředí optimální? To se musí agent naučit...



Dynamické programování
(Bellmanův princip optimality)

Temporální diference

Monte Carlo metody
(data)

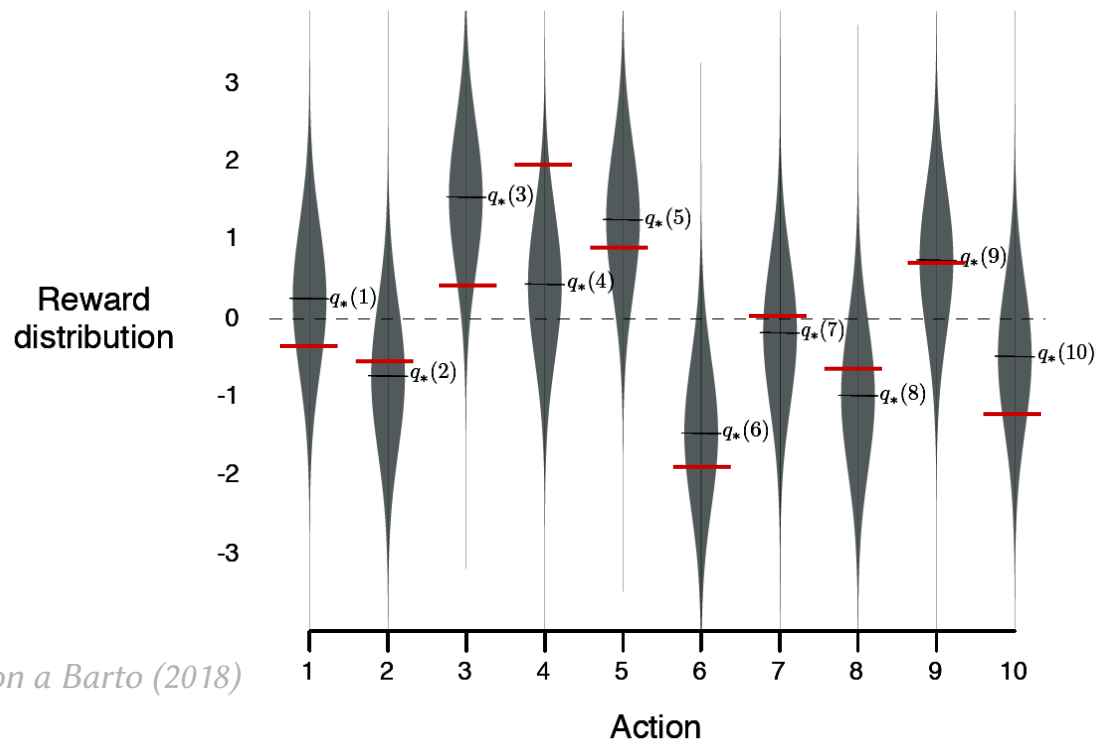
Při učení se typicky střídá zpřesňování odhadu hodnotové funkce a volba nové strategie, s níž bude vygenerována nová trajektorie

Hledání optimální strategie

Modelový příklad – **mnohoruký bandita** (stavy nezávislé na minulých akcích)

Jedna akce se po několika pokusech zdá lepší než všechny ostatní – znamená to, že je optimální a máme u ní zůstat?

Exploration vs. exploitation: **Vsadit na jistotu, nebo zkusit náhodnou akci?**



Sutton a Barto (2018)



ϵ -greedy strategie: s pravděpodobností ϵ volíme náhodnou akci, ve zbytku případů $A_t \doteq \arg\max_a Q_t(a)$

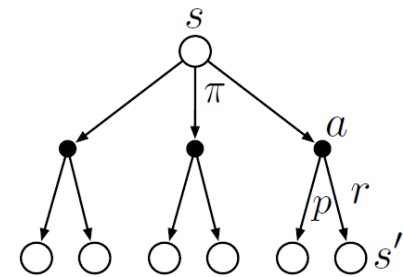
Dynamické programování

Richard Bellman, 50. léta – aplikace v ekonomii, teorii řízení a mnoha dalších oblastech

Rozdělení velké úlohy do podúloh, nalezení rekursivních vztahů, optimalizace po částech

Bellmanova rovnice pro hodnotovou funkci stavu

$$\begin{aligned} v_{\pi}(s) &\doteq \mathbb{E}_{\pi}[G_t \mid S_t = s] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \mathbb{E}_{\pi}[G_{t+1} | S_{t+1} = s'] \right] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \left[r + \gamma v_{\pi}(s') \right], \quad \text{for all } s \in \mathcal{S}, \end{aligned}$$



$$v_*(s) = \max_a \underbrace{\sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]}_{\text{střední hodnota zisku při akci } a \text{ ve stavu } s}$$

střední hodnota zisku při akci a ve stavu s

Obdobné vztahy lze psát také pro $q_{\pi}(s, a)$

Dynamické programování

Bellmanovy rovnice lze užít k **iterativnímu ohodnocení libovolné strategie π**

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_{\pi}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_k(s')] \end{aligned}$$

Diagram illustrating the Bellman equation for the value function $v_{k+1}(s)$ under a strategy π :

- $v_{k+1}(s)$: zpřesněný odhad očekávaného zisku dosažitelného ze stavu s
- $\pi(a|s)$: strategie
- $p(s', r | s, a)$: dynamika prostředí
- $r + \gamma v_k(s')$: očekávaný zisk po provedení akce a ve stavu s

Strategii můžeme zlepšit tak, že v každém stavu zvolíme akci, jejíž hodnota je vyšší nebo rovna hodnotě tohoto stavu

Postupné zlepšování strategie i odhadů hodnoty všech stavů (*policy iteration*) → **konvergence k optimalitě**

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Rychlejší alternativa: iterace rovnice pro optimální hodnotovou funkci (*value iteration*)

$$v_{k+1}(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_k(s')]$$

Metody Monte Carlo

Odhady hodnot $Q(s,a)$ nejsou založeny na Bellmanově rovnici, ale na datech – nepotřebujeme model prostředí

Běh velkého množství epizod a explicitní výpočet zisku G_t podle skutečně obdržených odměn R_t po provedení daných akcí v daných stavech $\rightarrow$ hodnotová funkce akce jako průměrný zisk

On-policy first-visit MC control (for ε -soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε -soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \dots, 0$:

$G \leftarrow G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg \max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Prozkoumání dostatečného počtu stavů a akcí může být docíleno tím, že epizody generujeme pomocí jiné strategie, než kterou se agent učí

Temporální diference

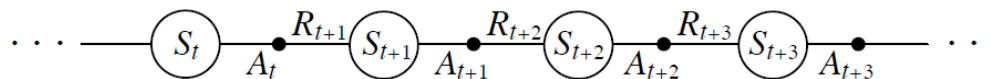
Podobnost s Monte Carlo metodami: nepotřebujeme znát model prostředí

Podobnost s dynamickým programováním: ke zpřesnění odhadu hodnoty akce nebo stavu použijeme odhad hodnoty následujícího stavu

Narozdíl od MC metod není třeba čekat do konce epizody, abychom mohli odhadnout celkový zisk

$$\text{MC: } V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)] \qquad \text{TD: } V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

SARSA: epizody generovány toutéž strategií, která je použita k odhadu hodnoty následující akce



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

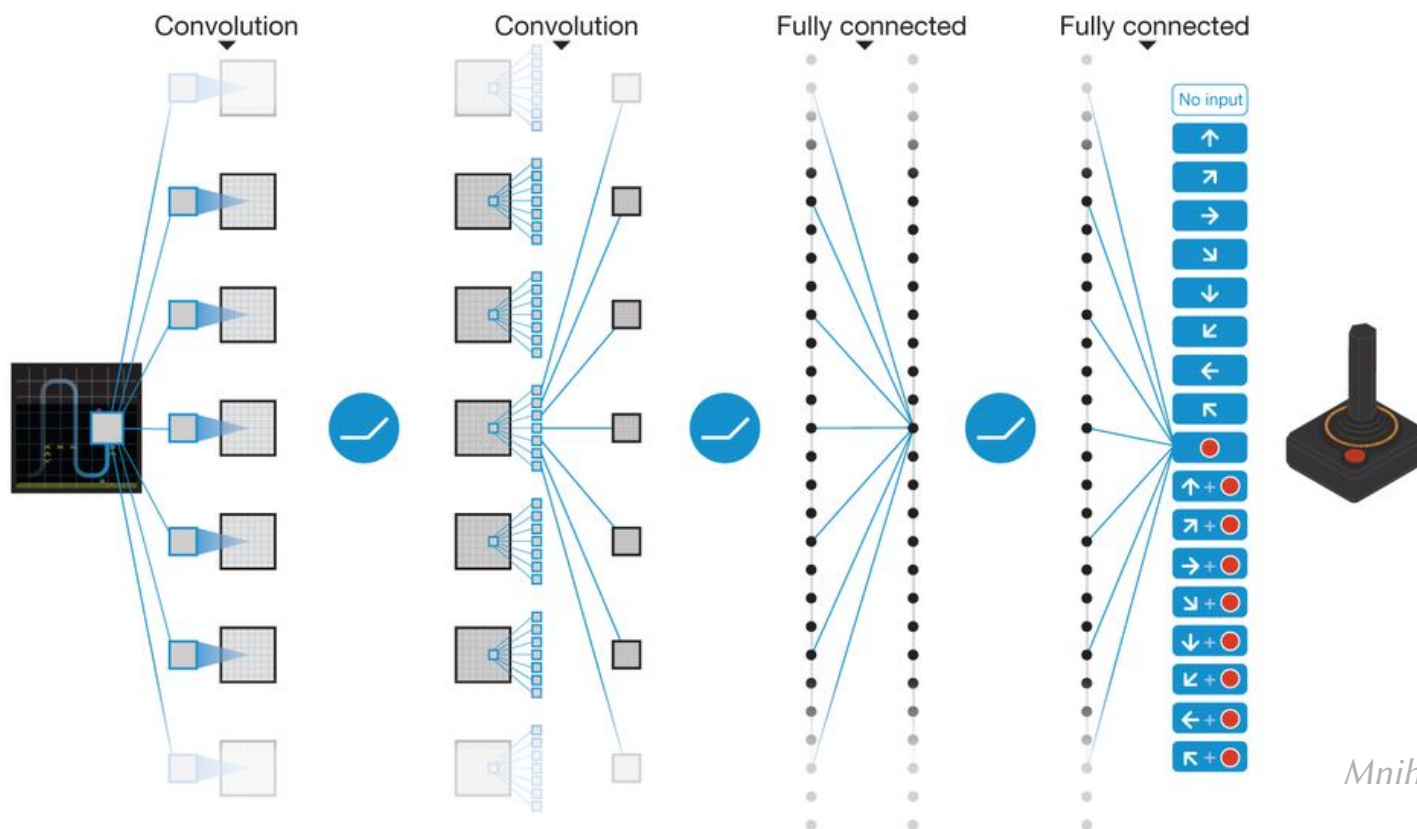
Q-learning: k odhadu hodnoty následující akce je použita strategie, která se jeví jako optimální

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Propojení s hlubokými sítěmi

Stavů může být (i nekonečně) mnoho, hodnotové funkce je pak vhodné parametrizovat a aproximovat například pomocí neuronových sítí (univerzální aproximátor)

Deep-Q Network: neuronová síť se učí předpovídat $Q(s, a; \theta)$ pro zadaná s a a
trénovací data: náhodně vylosované realizované dvojice (s, a) jako vstup, G_t jako výstup



Mnih a kol. (2015)

Co si z toho můžeme odnést?

Zpětnovazební učení umožňuje najít nejvhodnější strategii pomocí maximalizace celkové odměny

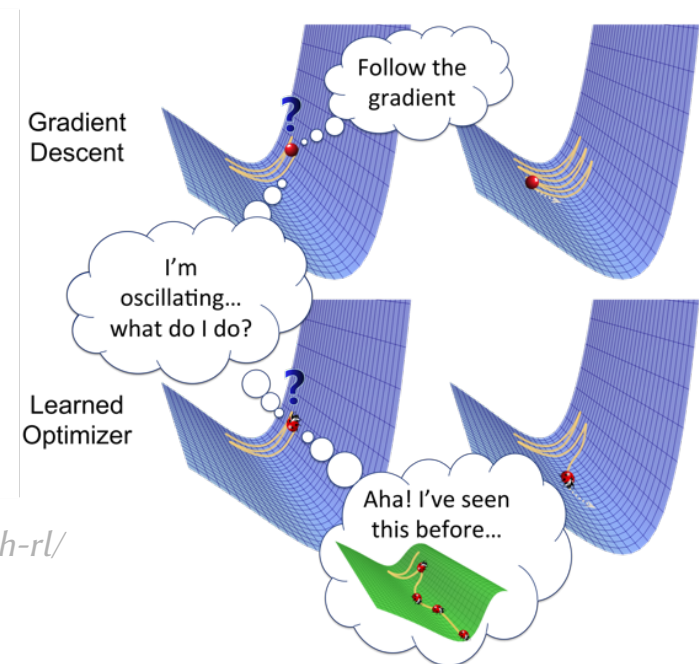
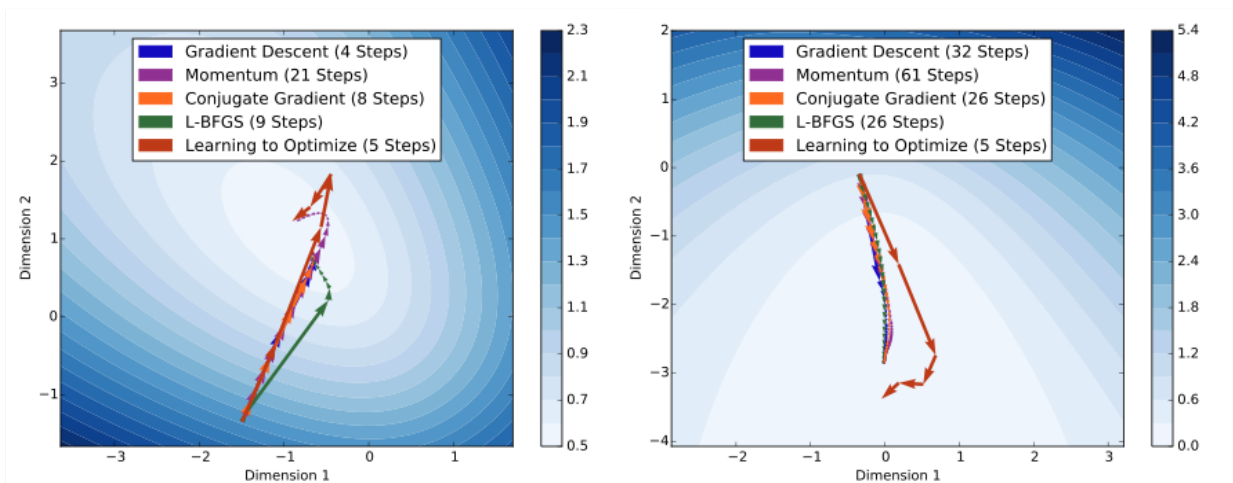
Dílčí odměny = naše komunikace s agentem

Není nutné znát model prostředí (Monte Carlo metody), stavy ani akce nemusí být diskrétní

Lze tuto metodu aplikovat v teoretické geofyzice?

Náměty:

- obrácené úlohy – automatizovaná volba kroku při minimalizaci misfitu (zobecněná gradientní metoda)



Ke Li (2019): <https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/>

- ???