Reinforcement Learning

Petr Salajka[1]

1 Úvod

Metody strojového učení se často dělí na učení s a bez učitele. Zapomíná se ale, že mezi těmito dvěma třídami existuje ještě třetí. Je jí skupina metod Reinforcement Learning. Tento příspěvek si klade za cíl především popularizaci a rozšíření povědomí o těchto metodách.

Metody Reinforcement Learning nabízejí možné řešení v případech kdy známe cíl našeho snažení, ale hodnocení správnosti jednotlivých kroků je obtížné. Příkladem může být návrh umělé inteligence, která bude umět hrát šachy. Na konci hry lze objektivně (a snadno) rozhodnout, který z hráčů vyhrál. Hodnotit jednotlivé tahy je na oproti tomu velmi obtížné.

Základem učení metodami Reinforcement Learning je interakce s prostředím. Umělá inteligence se zde obvykle označuje jako agent. Ten sleduje prostředí okolo sebe a pokud je to možné, dostává také informaci o tom, jestli je pro nás takové pozorování žádoucí. Na základě pozorování agent volí akci a tou ovlivňuje prostředí. Cyklus se opakuje, dokud ho nepřerušíme.

Jednou z oblíbených metod Reinforcement Learning je algoritmus Q-Learning. Základní metoda předpokládá existenci Q-tabulky, jejíž řádky reprezentují stavy (odvozené z pozorování prostředí) a sloupce reprezentují možné akce. Agent tedy na základě pozorování prostředí určí v jakém se nachází stavu a následně vybere optimální akci. Takovou u které na základě znalostí obsažených v Q-tabulce předpokládá nejvyšší celkovou odměnu. Provedení zvolené akce způsobí změnu prostředí, tím pádem i možnou změnu stavu, a agent provede úpravu tabulky tak, aby pro příště lépe reflektovala prožitou zkušenost.

Jde o tzv. offline variantu, což znamená, že agent předpokládá, že v budoucnu vždy zvolí optimální akci. Během učení ale vždy optimální akce volena není. Jde o princip označovaný angl. jako exploration and explotation. Na začátku je znalost agenta o prostředí nulová a pokud by vždy volil optimální akci (explotation), mohlo by se stát, že se zasekne v některém ze suboptimálních řešení. Proto se zavádí koeficient ϵ, který definuje pravděpodobnost, s jakou se vybere akce náhodná. Tím agent získává zkušenost z alternativních řešení a za určitých podmínek je schopen nalézt řešení optimální. Při použití offline varianty se hodnota ϵ postupně snižuje až k nule. Lehce odlišným přístupem je online varianta často označovaná jako Sarsa. Při této variantě neklesá hodnota ϵ až k nule nebo dokonce zůstává po celou dobu (behem učení i během provozu) neměnná. Jak ukazuje rovnice (1), metoda to reflektuje a pro výpočet nepoužívá optimální, ale skutečně zvolenou akci a. Q označuje Q-tabulku, S_t daný stav, A_t zvolenou akci, α je koeficient rychlosti učení, R_t obdržená odměna, γ je koeficient vlivu budoucí (očekávané) odměny, S_{t+1} stav, který nastal vlivem provedení akce A_t ve stavu S_t a A_{t+1} v něm zvolená akce.

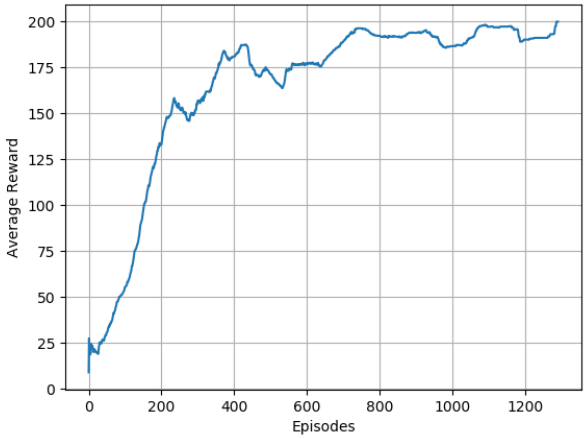
$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha\left(R_{t+1} + \gamma Q\left(S_{t+1}, A_{t+1}\right) - Q\left(S_t, A_t\right)\right) \tag{1}$$

[1] student doktorského studijního programu Aplikované vědy a informatika, obor Kybernetika, e-mail: salajka@kky.zcu.cz

2 Experiment

Následující experiment ukazuje použití algoritmu Sarsa pro řízení (udržení ve vzpřímené poloze) inverzního kyvadla na vozíku. Pro simulaci je použita knihovna OpenAI Gym. Jde o knihovnu připravených prostředí, která přijímají akci a a vrací pozorování o a odměnu r. Programátor se tak může zcela věnovat pouze návrhu agenta. V tomto případě jde o prostředí CartPole-v0. Prostředí poskytuje pozorování o čtyřech hodnotách: pozice vozíku, jeho rychlost, náklon kyvadla a jeho úhlová rychlost. Simulace probíhá v diskrétních časových krocích 0,02 s a v každém kroku musí agent vybrat jednu ze dvou možných akcí: postrčení zleva, nebo postrčení zprava. Jednotlivé simulace jsou rozděleny na epizody přičemž epizoda končí, pokud vozík překročí povolenou vzdálenost od středu (selhání), kyvadlo překročí povolený náklon (selhání), nebo čas simulace dosáhne 200 kroků (úspěch). Za každý krok simulace obdrží agent odměnu 1 a podle pokynů autorů se úloha považuje za vyřešenou, pokud agent dosáhne průměrné odměny alespoň 195 ve 100 po sobě jdoucích epizodách.

Převod pozorování na stav řeším jednoduchou kvantizací. Pro každou hodnotu pozorování si zaznamenám extrémy, kterých dosahuje. Tento interval následně dělím symetricky podle středu na 10 stavů. Celý algoritmus je velmi jednoduchý nicméně zcela zásadní se ukázalo doplnění penalizace za selhání. Vývoj hodnot průměrné odměny ve 100 po sobě jdoucích epizodách ukazuje obr. 1. Slouží pouze pro ilustraci úspěšného řešení úlohy.



Obrázek 1: Vývoj hodnot průměrné odměny

Protože každé pozorování má 4 hodnoty a agent volí v každém stavu ze 2 akcí, má výsledná Q-tabulka $2 \cdot 10^4 = 20000$ hodnot. Z toho si lze představit, že řešení složitějších reálných úloh pomocí metod Reinforcement Learning nebylo z důvodů výpočetní náročnosti dlouho možné. To ovšem zcela změnilo nahrazení Q-tabulky a jí odpovídajících struktur hlubokými neuronovými sítěmi a následující vznik pojmu Deep Reinforcement Learning.

Literatura

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.