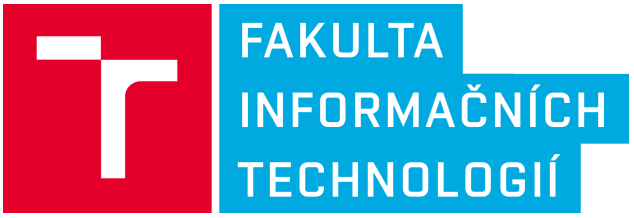


Vysoké učení technické v Brně

SFC

Soft Computing

Praktická úloha řešená sítí BP - predikce

Řešení úlohy Frozen Lake pomocí metody DQN

Autor:
Petr Buchal

Login:
xbucha02

1. Prosinec, 2019

1 Hluboké Q-učení

Problém, který si tato práce klade za úkol vyřešit spadá do oblasti zpětnovazebního učení. Z tohoto okruhu jsem si vybral metodu hluboké Q-učení, jejíž středobodem je neuronová síť. Zpětnovazební učení je oblast strojového učení, zabývající se chováním agentů v různých prostředích. Cílem zpětnovazebního učení je, co nejlépe vytrénovat agenta k maximalizaci kumulativní odměny za akce, které v prostředí provádí. Prostředí v jakých se agenti pohybují bývají většinou formulovány jako Markovův rozhodovací proces. Zjednodušeně se jedná o diskrétní stochastický proces, ve kterém se agent nachází ve stavu s a následným provedením jedné z možných akcí a se dostane do stavu s' a obdrží odměnu r [1].

Algoritmus 1 DQN [3]

Inicializuj paměť D s kapacitou N
Inicializuj neuronovou síť Q s náhodnými váhami
for $epizoda = 1, M$ **do**
  Inicializuj prostředí a pozoruj počáteční stav s_t
  for $krok = 1, T$ **do**
    S pravděpodobností ϵ vyber náhodnou akci a_t
    jinak $a_t = argmax(Q(s_t))$
    Proveď akci a_t, pozoruj odměnu r_t a nový stav s_{t+1}
    Ulož vzpomínku (s_t, a_t, r_t, s_{t+1}) do paměti D
    Vezmi náhodný vzorek vzpomínek (s_i, a_i, r_i, s_{i+1}) z paměti D
    $l_i = \begin{cases} r_i & \text{pro stav } s_i, \text{ který je koncový} \\ r_i + \ * \max(Q(s_{i+1})) & \text{pro stav } s_i, \text{ který není koncový} \end{cases}$
    Za použití s_i jako trénovacích dat a l_i jako štítků trénuj Q
    $s_t = s_{t+1}$
  end for
end for

DQN vychází z algoritmu Q-učení. Při použití Q-učení se agent snaží pohybovat v prostředí na základě předpovědí tzv. Q-funkce. Ta předpovídá očekávaný užitek z provedení konkrétní akce v konkrétním stavu tzv. Q-hodnotu. Akci s největší Q-hodnotou poté agent provede. Problémem Q-učení je, že pro každý stav a v něm možnou akci musí mít uložený záznam. Jejich počet činní jen pro piškvorky o velikosti 3x3 skoro 27 000 záznamů. Když se poté vezme v úvahu ještě to, že by agent měl při učení všech 27 000 záznamů aktualizovat, stane se z Q-učení nepříliš použitelná metoda pro složitější prostředí. DQN problém s ukládáním záznamů řeší tak, že je neukládá. Místo toho veškeré Q-hodnoty aproximuje pomocí neuronové sítě. Ta se

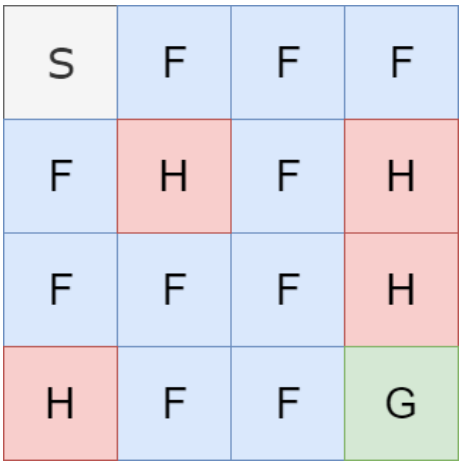
učí z paměti se vzpomínkami, ve které jsou uloženy stavy, kterými agent prošel [2]. Pseudokód algoritmu DQN se nachází v algoritmu 1.

1.1 Parametry učení

Váhy neuronové sítě jsou na začátku trénování inicializované náhodně. Učící konstanta je nastavena na hodnotu 0.01. Proměnná ϵ má na začátku trénování hodnotu 1. Její rozklad probíhá lineárně odečítáním hodnoty 0.001 v každém kroku, dokud ϵ neklesne na hodnotu 0.1, od této hranice zůstává hodnota ϵ konstantní. Paměť vzpomínek D má velikost 1 000 záznamů.

2 Neuronová síť

Prostředí na které budu algoritmus DQN aplikovat je relativně jednoduché a proto jsem vybral zjednodušenou architekturu sítě z práce [1], která se zabývá řešením podobných problémů. V práci se používá třívrstvá dopředná neuronová síť, kde se jako aktivační funkce používají ReLu s výjimkou poslední vrstvy, kde je aktivační funkce lineární. Já používám dvouvrstvou neuronovou síť, kde první vrstva má 16 neuronů a ReLu jako aktivační funkci a druhá vrstva má 4 neurony a lineární aktivační funkci.



Obrázek 1: Vizualizace prostředí FrozenLake - start (S), led (F), díra (H), cíl (G).

3 Prostředí Frozen Lake

Prostředí ve kterém se agent bude pohybovat je zamrzlé jezero s dírami, kde je cílem agenta dostat se z jednoho rohu do druhého (Obrázek 1). Agent dostává za stoupnutí do díry odměnu -1 a za dosažení cílového pole odměnu 1, jinak 0. Stoupnutím do díry či dostáním se do cíle nastává konec hry. Stav ve kterém se agent nachází je popsán vektorem 16 čísel (každé představující jedno pole), kde pole na kterém je agent má hodnotu 1, jinak 0.

4 Aplikace

Aplikace má dva módy - trénovací a testovací. V trénovacím módu probíhá trénování dopředné neuronové sítě pomocí algoritmu backpropagation. V tomto módu je možné zobrazit tři rozdílné vizualizace učení sítě. První vizualizací je mapa prostředí, která zobrazuje aktuální pozici agenta (Obrázek 2). Druhou vizualizací je mapa prostředí, která zobrazuje aktuální odhad kvality akcí neuronové sítě, tato mapa neobsahuje aktuální polohu hráče (Obrázek 3). Třetí vizualizací je pak výpis zobrazující informace o testovacím průchodu prostředím a jeho úspěšném/neúspěšném konci (Obrázek 4). V testovacím módu se načte natrénovaný model a provede se jeden průchod prostředím pro ověření funkčnosti modelu.

Obrázek 2: Vizualizace průchodu agenta prostředím.

		0			1			2			3	
		-0.639			-0.569			-0.520			-0.482	
0	-0.636	S	-0.487	-0.556	F	-0.486	-0.511	F	-0.521	-0.562	F	-0.532
		-0.486			-0.843			-0.596			-0.725	
		-0.545			-0.501			-0.581			-0.574	
1	-0.571	F	-0.915	-0.610	H	-0.597	-0.568	F	-0.577	-0.559	H	-0.494
		-0.463			-0.589			-0.548			-0.693	
		-0.505			-0.624			-0.587			-0.558	
2	-0.566	F	-0.470	-0.586	F	-0.425	-0.561	F	-0.624	-0.544	H	-0.526
		-0.792			-0.608			-0.555			-0.691	
		-0.495			-0.525			-0.519			-0.517	
3	-0.589	H	-0.558	-0.574	F	-0.666	-0.590	F	-0.507	-0.596	G	-0.479
		-0.630			-0.582			-0.682			-0.536	

Obrázek 3: Vizualizace Q-hodnot pro všechny dvojice stav-akce.

```
Episode: 0; Epsilon: 0.99; Test outcome: CYCLE
Episode: 1; Epsilon: 0.98; Test outcome: CYCLE
Episode: 2; Epsilon: 0.97; Test outcome: CYCLE
Episode: 3; Epsilon: 0.97; Test outcome: CYCLE
Episode: 4; Epsilon: 0.95; Test outcome: CYCLE
Episode: 5; Epsilon: 0.95; Test outcome: CYCLE
Episode: 6; Epsilon: 0.94; Test outcome: CYCLE
Episode: 7; Epsilon: 0.93; Test outcome: CYCLE
Episode: 8; Epsilon: 0.93; Test outcome: CYCLE
Episode: 9; Epsilon: 0.9; Test outcome: CYCLE
Episode: 10; Epsilon: 0.89; Test outcome: CYCLE
Episode: 11; Epsilon: 0.88; Test outcome: CYCLE
Episode: 12; Epsilon: 0.87; Test outcome: CYCLE
Episode: 13; Epsilon: 0.87; Test outcome: CYCLE
Episode: 14; Epsilon: 0.85; Test outcome: WIN in 6 moves
[SUCCESSFUL RUN]
```

Obrázek 4: Výpis zobrazující informace o testovacích průchodech prostředím na konci každé epizody trénování.

4.1 Spuštění aplikace

Aplikace je spustitelná na referenčním serveru Merlin a dále v prostředích, která obsahují knihovny třetích stran Numpy a Pickle.

4.1.1 Parametry a příklady použití

-mode {train,test}	slouží pro výběr módu aplikace
-r_mode {weights,map,stats}	slouží pro výběr vizualizace v trénovacím módu
-model MODEL	slouží pro výběr modelu v testovacím módu

python3 q_learning.py -mode train -r_mode map Příkaz spustí aplikaci v trénovacím režimu s vizualizací pozice agenta.
python3 q_learning.py -mode test -model model Příkaz spustí aplikaci v testovacím režimu s modelem jménem model.pkl.

4.2 Metriky kódu

- Počet souborů: 6
- Počet řádků zdrojového kódu: 513
- Celková velikost souborů (bez dokumentace): 20 kB

4.2.1 Soubory

- **environment.py** – soubor obsahuje třídu FrozenLake, která implementuje prostředí popsané v kapitole 3
- **helper.py** – soubor obsahuje pomocné metody využívající se v ostatních částech aplikace
- **model.pkl** – soubor obsahuje parametry natrénované neuronové sítě
- **model.py** – soubor obsahuje implementaci neuronové sítě
- **q_learning.py** – soubor obsahuje implementaci algoritmu Q-učení
- **manual.pdf** – dokumentace projektu

Reference

[1] Petr Buchal. Hraní her pomocí neuronových sítí. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2018.

[2] Petr Buchal. Hraní her pomocí neuronových sítí. Excel@Fit 2018, May 2018. https://goo.gl/FHbELw.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, December 2013. https://arxiv.org/pdf/1312.5602.pdf.