Zdroje

* <https://openai.com/research/openai-baselines-ppo>
* <https://libstore.ugent.be/fulltxt/RUG01/003/063/153/RUG01-003063153_2022_0001_AC.pdf>
  + Hra s neurčitostí – PPO porazilo další populární algoritmy
* <https://arxiv.org/abs/1707.06347>
* <https://www.youtube.com/watch?v=ovIykchkW5I&t=1456s>  
  <https://openai.com/research/emergent-tool-use>
  + openAI – hra na schovku kde se učili pomocí PPO
* <https://fse.studenttheses.ub.rug.nl/25709/1/mAI_2021_BickD.pdf>
* <https://lilianweng.github.io/posts/2018-02-19-rl-overview/#key-concepts>
  + Hlavní zdroj
* <https://lilianweng.github.io/posts/2018-04-08-policy-gradient/>
  + Hlavní zdroj
* <https://jeremykun.com/2015/04/06/markov-chain-monte-carlo-without-all-the-bullshit/>

<https://arxiv.org/abs/1506.02438>

<https://telin.ugent.be/telin-docs/windows/pytorch/>

https://docs.ray.io/en/latest/ray-core/tasks/using-ray-with-gpus.html

Myšlenky

Síť na odhadnutí kde se nachází pan X.

PPO

Předchůdce TRPO, ACER

Policy gradient methods are fundamental to recent breakthroughs in using deep neural networks for control, from video games, to 3D locomotion, to Go. But getting good results via policy gradient methods is challenging because they are sensitive to the choice of stepsize — too small, and progress is hopelessly slow; too large and the signal is overwhelmed by the noise, or one might see catastrophic drops in performance. – špatná efektivita

Exploitation exploration

* *LCLIP*(*θ*)=*E*^*t*​[*min*(*rt*​(*θ*))*A*^*t*​,*clip*(*rt*​(*θ*),1−*ε*,1+*ε*)*A*^*t*​)]  
  *θ* is the policy parameter
* *E*^*t*​ denotes the empirical expectation over timesteps
* *rt*​ is the ratio of the probability under the new and old policies, respectively
* *A*^*t*​ is the estimated advantage at time *t*
* *ε* is a hyperparameter, usually 0.1 or 0.2

<https://libstore.ugent.be/fulltxt/RUG01/003/063/153/RUG01-003063153_2022_0001_AC.pdf>  
  
  
dle tohoto zdroje PPO algoritmus dovedl vyhrát nad mone carlo algoritu ve hře s neúplnou informací. Je třeba prozkoumat co bude vhodné do Scotland yardu

**Proximal policy optimization (2017)**

Pomocí učení vzniká tzv. politika. Politika je matematická funkce, která definuje rozdělení pravděpodobnosti nad množinou akcí.  
Politika určuje strategii agenta v různých stavech a usiluje o maximální odměnu.

Jelikož politika určuje pouze pravděpodobnost zvolení akce je vždy možnost, že se nevybere statisticky nejlepší akce z pohledu agenta, a to má za důsledek lepší prozkoumávání možností a stabilnější učení. Tato politika se mění na základě trestů a odměn, jak je tomu v posilovaném učení zvykem, avšak algoritmus **PPO** zamezuje příliš náhlým a prudkým odchylkám od původní politky za využití ořezávání. Takže je přesně určeno o kolik se může politika maximálně změnit. Toto je žádoucí chování, jelikož je známo, že menší postupné změny v politice častěji konvergují k optimálnímu rozhodování.   
LCLIP(θ)=E^t[min(rt(θ))A^t,clip(rt(θ),1−ε,1+ε)A^t)]

* rt – poměrová funkce
  + pravděpodobnost

# Klíčové koncepty posilovaného učení

Agent existuje a jedná v určitém prostředí. To, jak prostředí zareaguje na určité akce agenta je definováno modelem, který agent může nebo nemusí znát. Agent je vždy v 1 z mnoha možných stavů prostředí. Agent na základě vypočítané pravděpodobnosti přejde do jiného stavu. A na základě tohoto přechodu mu prostředí přidělí odměnu. Odměna může být záporná či kladná.

**Model-based:**

Model je známý, nebo se ho algoritmus naučí explicitně (jsou dány přesné instrukce a pravidla)

**Model-free:**

Během učení není závislost na modelu

**On-policy:**

K trénování výsledné politiky používá samotnou trénovanou politiku.

**Off-policy:**

K trénování výsledné politiky, používá jiné politiky chování.

**Epizoda**

Interakce mezi agentem a prostředím zahrnuje sekvenci akcí a předvídatelnou odměnu v dohledném čase t. Během tohoto procesu agent získává vědomosti o prostředí, tím mění svou politiku. Tento proces se nazývá epizoda. Sekvence epizod končí terminálním stavem.

**Model**

Popisuje prostředí. Za pomocí modelu se můžeme naučit, nebo odvodit, jak bude prostředí interagovat s agentem a můžeme mu dát příslušnou zpětnou vazbu. Model má 2 hlavní funkce. Pravděpodobností funkci přechodu P a odměnovou funkci R. Přechod je definován n-ticí (s,a, s',r)

**Politika**

Pomocí učení vzniká tzv. politika. Politika je matematická funkce, která definuje rozdělení pravděpodobnosti nad množinou akcí.

Politika určuje chování agenta v různých stavech a usiluje o maximální celkovou odměnu. Je buď deterministická nebo stochastická.  
  
Stochastická politika je matematická funkce, která definuje rozdělení pravděpodobnosti nad množinou akcí. Tzv, výsledný přechod je „náhodný“ dle pravděpodobnosti.

Deterministická politika přesně definuje cílový stav přechodu na základě aktuálního stavu.

**Hodnotová funkce (value function)**

Vyhodnocuje, jak dobrý je stav tím, že predikuje budoucí odměnu. Čím vzdálenější odměna je, tím více je snížena, jelikož čím je vzdálenější tím více je nejistá, také neposkytuje okamžitou odměnu

A white paper with black text and black text

Popis byl vytvořen automaticky  
Je složena z hodnoty stavu a hodnoty akce

**Hodnota stavu – V value**

Očekávaný zisk, pokud by agent byl v daném stavu

**Hodnota akce – Q value**

Očekávaný zisk, pokud by agent zvolil danou akci. K výpočtu využívá hodnotu stavu.

Markov Decision Processes

Téměř všechny problémy posilovaného učení mohou být označeny jako Markov Decision Processes (MDPs). Tato teorie je založena na tom, že budoucí stavy záleží pouze na aktuálním stavu, nezajímají nás stavy minulé.

A diagram of a agent and environment

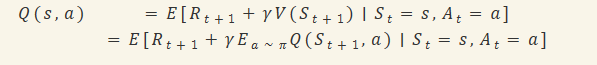
Popis byl vytvořen automaticky

Bellmanova rovnice

Rovnici hodnotové funkce rozkládá na okamžitou odměnu a snížené budoucí odměny.

A math symbols and equations

Popis byl vytvořen automaticky se střední mírou spolehlivosti



A group of math symbols

Popis byl vytvořen automaticky

Často ale nemůže aplikovat belmanovy rovnice na MDPs protože neznáme hodnoty p a R(s,a). Ale pokládá teoretické základy pro mnoho algoritmů posilovaného učení.

# Přístup k řešení posilovaného učení

**Dynamické programování:**

Model je plně známy a můžeme tedy použít Bellmanovy rovnice. Iterativně tedy vylepšujeme hodnotovou funkci a politiku.

**Evaluace politiky:**

Výpočet stavové hodnot pro danou politiku

**A math equation with black text

Popis byl vytvořen automaticky**

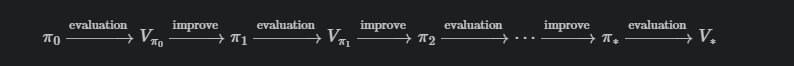
**Vylepšení politiky:**

Pomocí hodnotových funkcí, politika vygeneruje lepší politiku tím, že se bude chovat chamtivě.**A close-up of a white background

Popis byl vytvořen automaticky**

**Iterace politiky (generalized policy iteration) - GPI**

Kombinuje evaluaci politiky a vylepšení politiky.

****

Hodnotová funkce je dokola aproximována, aby byla blíže k aktuální politice, a politika je vylepšována aby se blížila optimálnosti. Tato metoda vždy konverguje k optimálnosti

**SARSA: on-policy TD control**

Stejný princip jako PI.



**Q-Learning: Off-poicy TD control**

Velký pokrok v posilovaném učení.

1. Inicializace t = 0; S = S0
2. Na kroku t, se chamtivě vybere akce na základě Q hodnot
3. Provede se vybraná akce s přechodem stavu a zhodnotí se odměna.
4. Aktualizuje se Q-value funkce  
   
5. t se zvětší o 1
6. Pokračuje se opět od kroku 2

**Deep Q-Network:**

Ve velkých stavových prostorech začínají být tyto metody velmi počítačově náročné. Proto se začaly Q hodnoty aproximovat. To má ale za následek nestabilitu a možnost divergování od optimálního řešení.

Deep Q-Network velmi zlepšuje stabilitu dvěma mechanismy

**Experience replay:**

Všechny episody SARS jsou uloženo do paměti D. Během aktualizace Q-value funkce je vzorek náhodně vybrán z této paměti a tím může být 1 vzorek vybrán vícekrát. To vede k lepší efektivitě dat, odstraňuje korelaci sekvencí a vyhlazuje změny.

Periodicky aktualizovaný cíl:

Q síť je naklonována, a změny se provádí na duplikátní verzi. Do hlavní cílové sítě se změny naklonují po C krocích. Tato modifikace dělá trénování více stabilní.

**Policy Gradient (gradient politiky):**

Všechny metody výše usilují o naučení state/value funkce a poté podle nich vybrat vhodnou akci.

Zato metody tohoto typu se snaží naučit politiku přímo s pomocí parametrizované funkce. Definice reward function:

A white background with black text

Popis byl vytvořen automaticky

A math equation with numbers and symbols

Popis byl vytvořen automaticky

Tato rovnice značí průchod Markov chain por pí0. Využívá toho, že pokud bychom procházeli Mark chain states navždy, šance na to že skončíme s daným stavem se nemění.

# Policy gradient theorem

Výpočet gradientu je složité protože závisí na výběru akce a distribuci stavů po výběru stavu. Jelikož je většinou prostředí neznáme, je složité odhadnout efekt na stavovou distribuci při aktualizaci politiky. Proto byl vymyšlen Policy gradient theorem. Ten zjednodušuje výpočet gradientu a odstraňuje závislost na derivaci stavové distribuce

A black background with white text

Popis byl vytvořen automaticky

TODO: do hloubky pochopit a popsat rovnice

# Exploration-Exploitation Dilemma

Bez prozkoumávání nemůžeme prozkoumat celé prostředí a bez využívání nemůžeme nejlépe optimalizovat odměnu za úkol.

# Policy gradient algorithms

## Monte-Carlo policy gradient