Применение техник обучения с подкреплением по учебной программе и самостоятельной игры для конкурентных сред в казахских национальных играх

June 12, 2024

Динара Жусупова
Руководитель:Елена Кантонистова
Машинное обучение и высоконагруженные системы, НИУ Высшая школа экономики

О выполнении работы

Цель работы

Применение методов для обучения с подкреплением агентов и на основе проведения экспериментов получение на достаточно хорошем уровне играющих игроков, способных выигрывать и человека в игре «Тогызкумалак».

Задачи

Для достижения цели были выполнены следующие задачи:

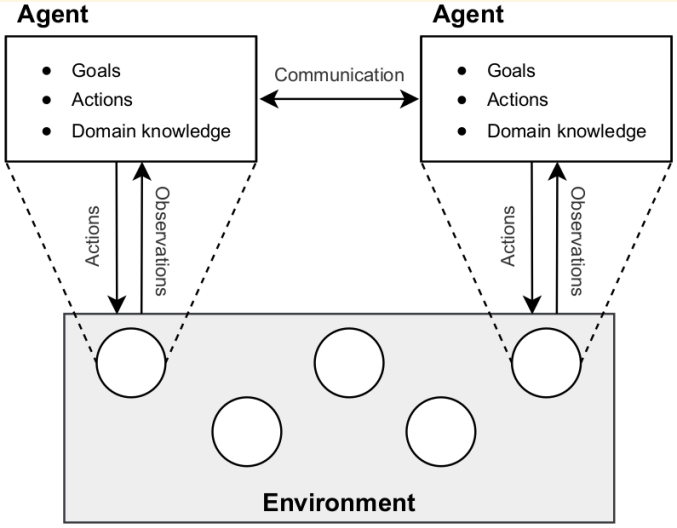
- изучение постановки задачи
- обзор литературы и техник обучения
- изучение фреймворков для написания среды и обучения агентов
- построение конкурентной мультиагентной среды
- обучение агентов
- проведение сравнительных экспериментов по обучению и выбор лучших моделей

Методы

RL алгоритмы:

- Double Deep Q-Network
- Rainbow
- Soft actor critic

Мультиагентная среда



Алгоритмы обучения с подкреплением могут обучать агентов, которые решают проблемы в сложных, интересных средах.

RL

RL – это эффективный инструмент, который помогает системам искусственного интеллекта (ИИ) достигать оптимальных результатов в полно/частично наблюдаемых средах.

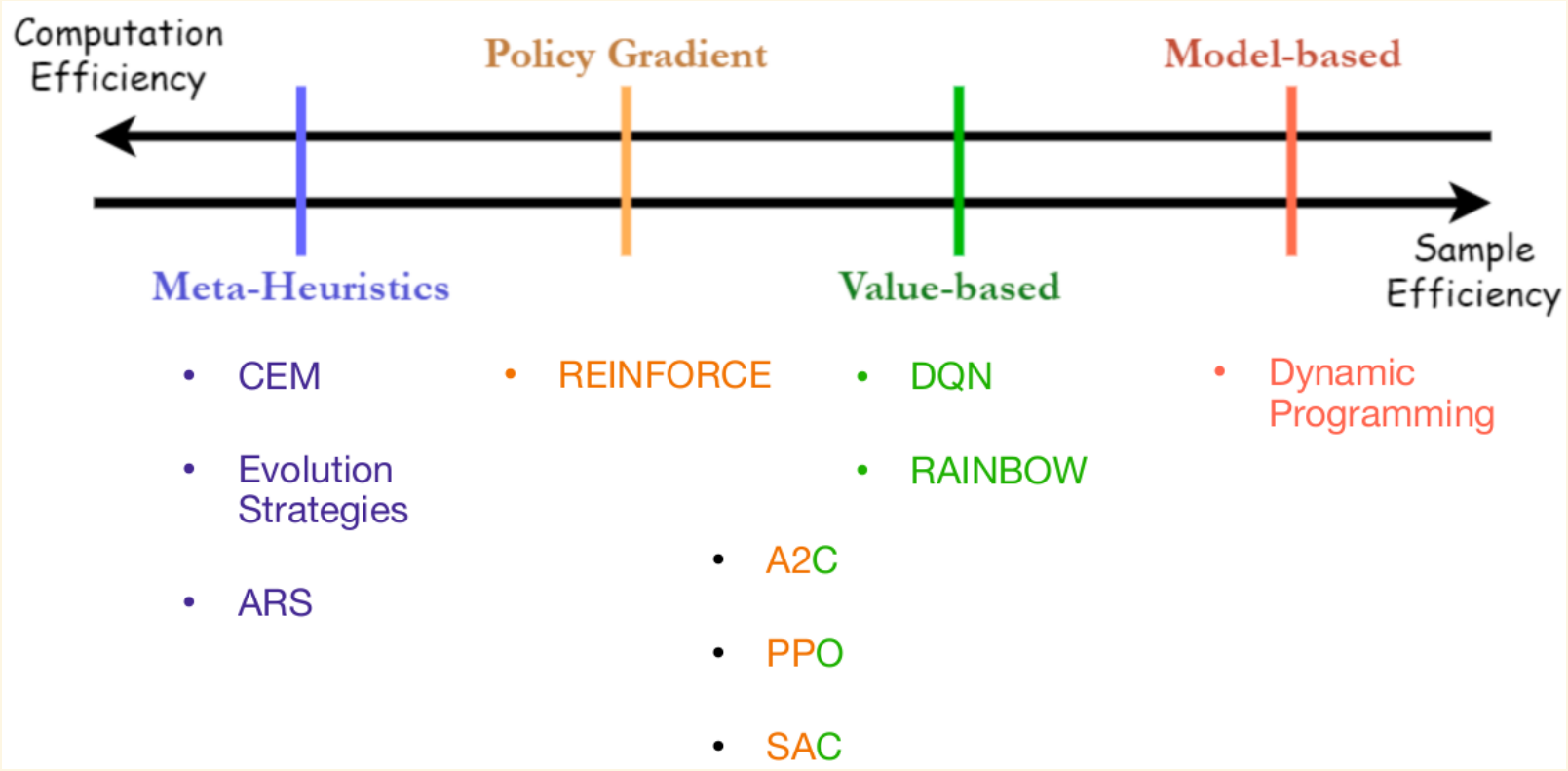
MARL

Мультиагентная среда или система состоит из среды и множества агентов, принимающих решения, которые взаимодействуют в среде для достижения определенных целей.

Self-play (самостоятельная игра)

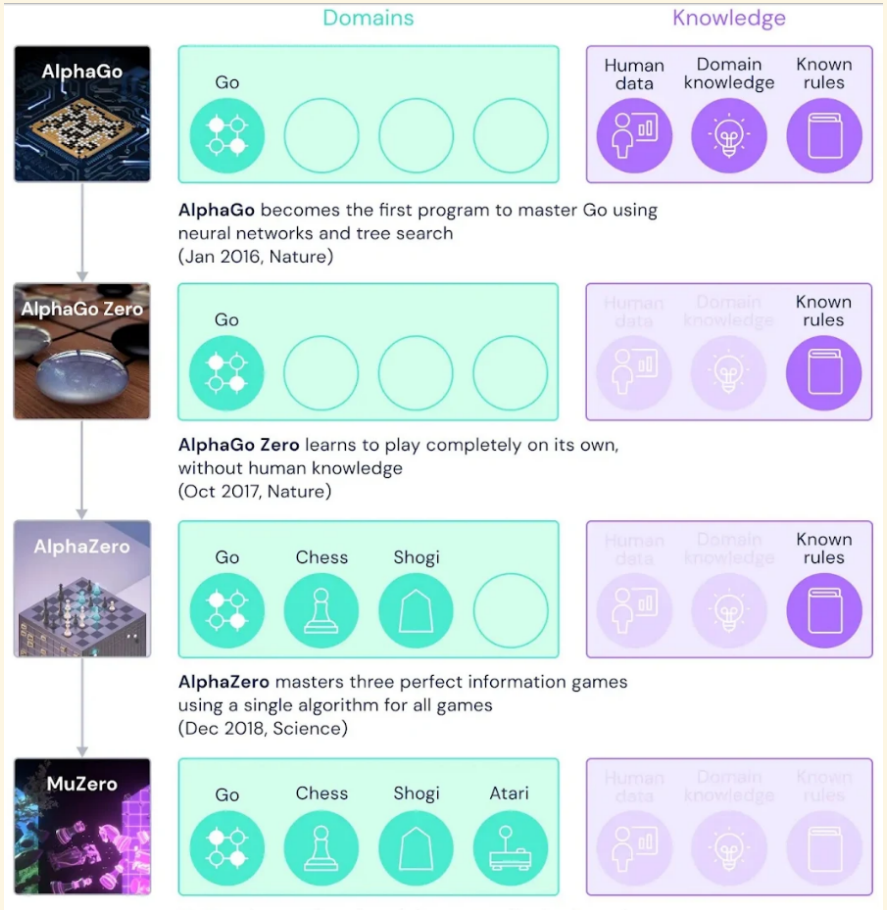
В конкурентной многоагентной среде агенты обучаются с помощью техники self-play.

Алгоритмы RL



Обучение агентов в мультиагентных средах

Обученные агенты



AlphaZero

Chess

В шахматных партиях AlphaZero против Stockfish 8 каждая программа имела по одной минуте времени на ход. Из 100 игр с нормального начального положения AlphaZero выиграл 25 партий белыми, 3 чёрными и свёл вничью оставшиеся 72.

Shogi

В ста играх против Elmo AlphaZero выиграл девяносто раз, восемь раз проиграл и две партии завершились вничью.

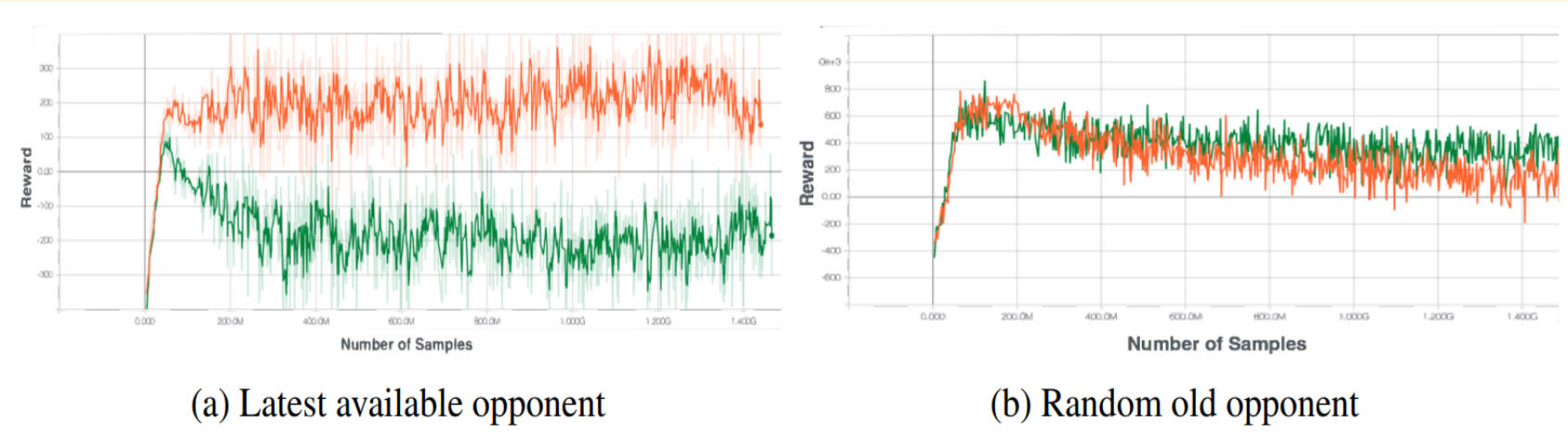
Go

После 8 часов самостоятельного обучения игры в Go, в матчах против предыдущей версии AlphaZero, AlphaZero выиграл шестьдесят игр и проиграл сорок.

Обучение конкурентных агентов. Сэмплирование оппонентов

Навыки противников, с которыми сталкиваются во время обучения, могут оказать существенное влияние на обучение агентов.

Обучение агентов против самого последнего противника приводит к дисбалансу в обучении, когда один агент становится более опытным, чем другой агент, на ранних этапах обучения, а другой агент не может восстановиться [1].



(a) Latest available opponent (b) Random old opponent

[1] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever и Igor Mordatch. "Emergent Complexity via Multi-Agent Competition". В: CoRR abs/1710.03748 (2017). arXiv: 1710. 03748. url: http://arxiv.org/abs/1710.03748.

Метод Q-Learning

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

$$\mathrm{Q}(\mathrm{S_t},\mathrm{A_t}) \longleftarrow \mathrm{Q}(\mathrm{S_t},\mathrm{A_t}) + \alpha\left[\mathrm{R_{t+1}} + \gamma\max_{\mathrm{a}}\mathrm{Q}(\mathrm{S_{t+1}},\mathrm{a}) - \mathrm{Q}(\mathrm{S_t},\mathrm{A_t})\right].$$

Алгоритм 1 Q-Learning

Параметры алгоритма $\alpha \in [0,1]$ и $\varepsilon > 0$ малое число
Инициализировать $Q(s,a)$ для всех a и s произвольно, за исключением того, что $Q(terminal,\cdot) = 0$
for episode **do**
Выбрать A из S, используя политику, полученную из Q (например, «ε-жадно»)
Выполнить выбранное на предыдущем шаге действие A и получить наблюдение S' и вознаграждение R
Обновить Q-значение:

$$Q(S,A) = Q(S,A) + \alpha\left[R + \gamma\max_a Q(S',A) - Q(S,A)\right]$$

Повторить, пока не будет достигнуто терминальное состояние
end for

Метод DQN Deep Q-network

Deep Q-Learning использует глубокую нейронную сеть для аппроксимации различных значений Q(S, A) для каждого возможного действия A в состоянии S (оценка функции значения)

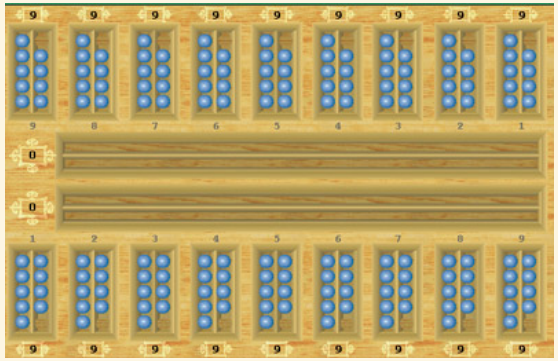
Алгоритм 2 DQN

Инициализировать размер Replay Buffer
Инициализировать функцию значения действия Q рандомными весами θ
Инициализировать целевую функцию $\hat{Q}$ с весами $\theta^- = \theta$
for episode **do** Инициализировать последовательность $s_1 = x_1$ и последовательность препроцессинга $\phi_1 = \phi(s_1)$
  for t **do** Выбрать действие a_t с вероятностью ε или с помощью формулы $a_t = argmax_a Q(\phi(s_t), a; \theta)$
    Выполнить действие a_t и получить награду и следующее состояние
    Присвоить $s_{t+1} = s_t$, a_t и подготовить препроцессинг $\phi_{t+1} = \phi(s_{t+1})$
    Сохранить в Replay Buffer D кортеж $(\phi_t, a_t, r_t, \phi_{t+1})$
    Сэмплировать минибатч с кортежами $(\phi_t, a_t, r_t, \phi_{t+1})$ из D
    Присвоить

$$y_j = \begin{cases} r_j & \text{если эпизод заканчивается на шаге } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a_j; \theta^-\right) & \text{иначе} \end{cases}$$

    Реализовать шаг градиентного спуска для $\left(y_j - Q\left(\phi_{j+1}, a_j; \theta\right)\right)^2$ квадратичного отклонения относительно параметра θ
    Через каждые C шагов сбрасывать $\hat{Q} = Q$
  end for
end for

Об игре «Тогызкумалак» (каз. Тоғызқұмалақ)



Описание игры

Тогызкумалак – одна из традиционных казахских национальных настольных игр. В 2020 году ЮНЕСКО включило эту игру в репрезентативный список нематериального наследия. Игра «Тогызкумалак» проводится между двумя игроками на специальной доске. Игровая доска состоит из 2 лунок накопительных, 18 лунок игровых и 162 камней. Игрок, совершивший первый ход, называется «Бастауши» (Начинающий), игрок, совершивший ответный ход «Костауши» (Продолжающий). Иногда используются выражения «белая сторона» для начинающего и «черная сторона» для продолжающего.

- **Количество игроков:** 2
- **Исходная позиция:** Перед началом игры все камни разложены в 18 лунок по 9 камней. У каждого игрока есть накопительная ячейка, в которую складываются выигранные камни. Игрок владеет ячейками напротив него.

Основные правила игры

Описание хода игрока

- Игрок берет все камни кроме одного из одной лунки и раскладывает по одному камню в каждую ячейку против часовой стрелки. Ход делается только с непустой лунки, если все лунки пусты, то это положение называется Атсырау.
- Если в последней лунке оказалось четное количество камней и эта лунка на стороне противника, игрок выигрывает эти камни и складывает в свою накопительную лунку.
  Исключение. Если в текущей лунке только один камень, то этот камень перекладывается в соседнюю лунку справа.
  Исключение. Нельзя ходить с лунки, завоеванной противником, т.е. с туздыка.

Получение Туздыка

Туздык — завоеванная ячейка у противника. Случаи, когда игрок получает Туздык:

1. если последний камень попадает не в последнюю ячейку соперника и чило камней становится равно 3-м, то игрок получает Туздык
2. не допускается получения туздыка номер, которого совпадает с туздыком соперника (например, если соперник владеет ячейкой номер 3, то нельзя получить с таким же номером ячейку).

Атсырау

Положение в игре, когда игрок не может сделать ход, так как в его игровых лунках нет ни одного камня.

Победа в игре

Если игрок собирает в накопительной лунке больше 81 камня, он побеждает. Если соперник достиг Атсырау, то все камни игрока, которые находятся в его игровых ячейках

Стек

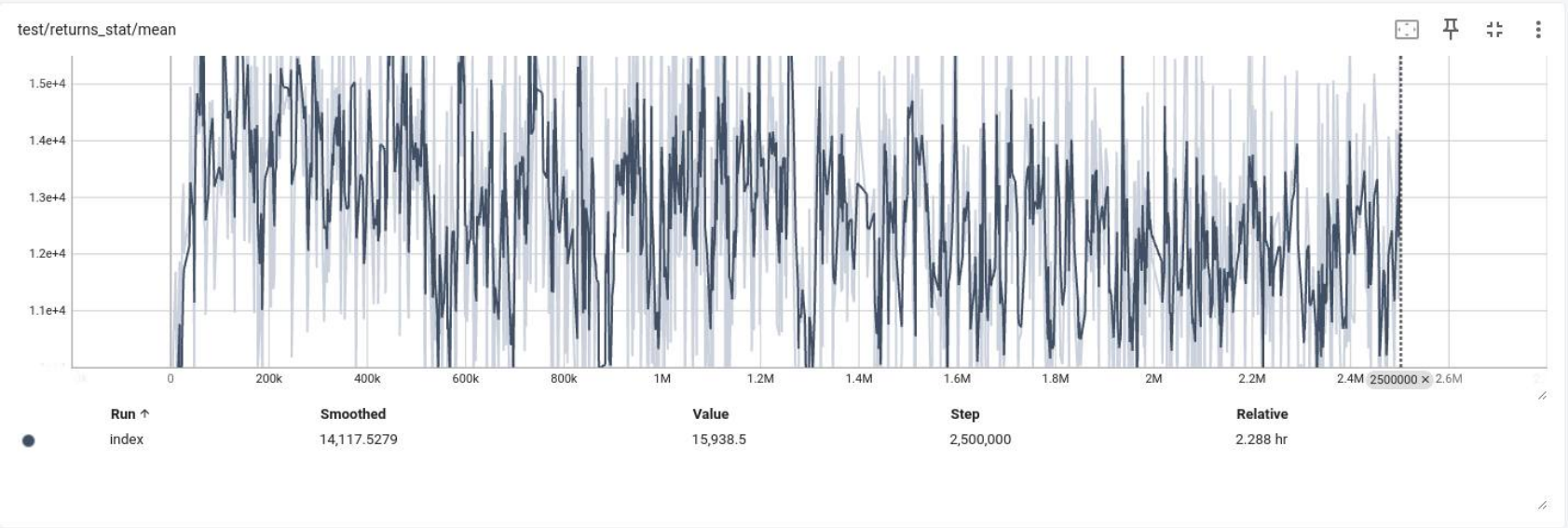
- **Для создания мультиагентной среды:** PettingZoo

- **Для обучения конкурентных агентов:** Tianshou

- **Для создания веб приложения игры:** Dash

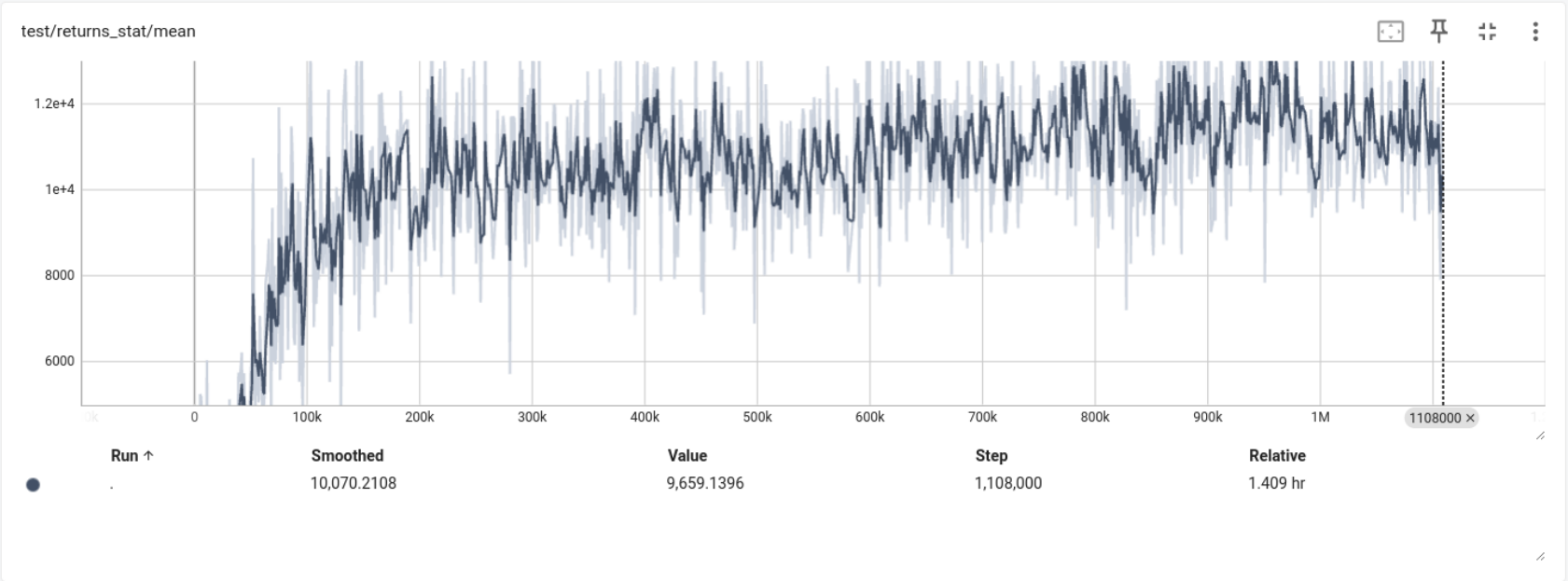
Начальные эксперименты с обычным вознаграждением

Обучение игрока Бастауши (неудачная попытка)



Эксперименты: подход zero-sum (игра Тогызкумалак как антагонистическая)

Обучение игрока Бастауши



Детали обучения:

- алгоритм - Double Deep Q-Network
- количество эпох - 3 000
- размер батча: 256
- обновление таргета - 352
- количество шагов в 1 эпохе - 1000
- количество эпизодов при тестировании - 50
- lr = 3e-04
- размер ReplayBuffer - 100 000

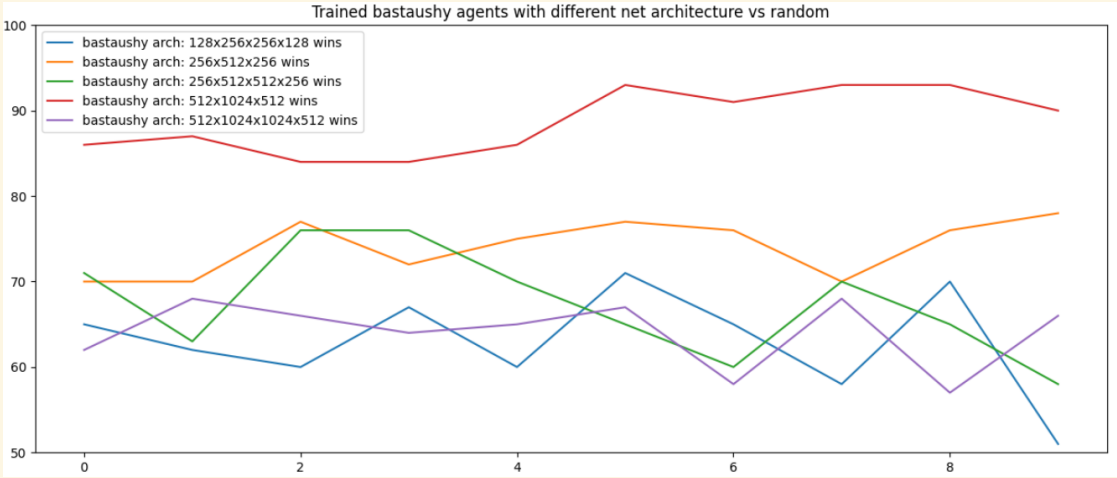
После обучения игрок побеждает рандомную политику в 95–98% случаях.

Выбор архитектуры DQN

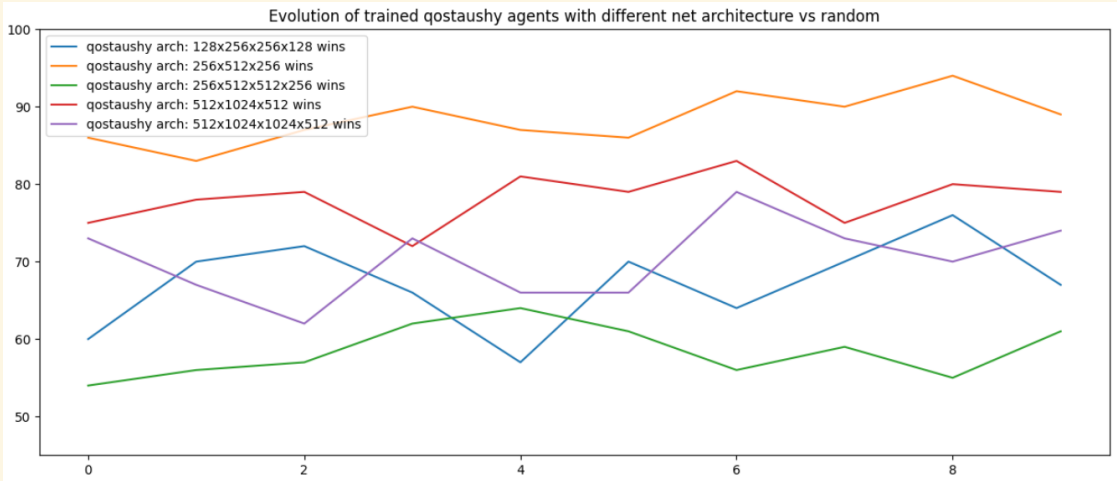
Потенциальные MLP архитектуры
NET_ARCHS = {[128,256,256,128],
[256,512,256],
[256,512,512,256],
[512,1024,512],
[512,1024,1024,512]}.
Исследование и последующий выбор осуществлялся при помощи обучения нейронных сетей с рандомными политиками в роли противника.
Детали обучения:

- алгоритм – Double Deep Q-Network
- количество эпох – 800
- размер батча – 256
- обновление таргета – 352
- количество шагов в 1 эпохе – 1000
- количество эпизодов при тестировании – 50
- lr = 3e-04
- размер ReplayBuffer – 100 000

Результаты обучения игрока Бастауши



Результаты обучения игрока Костауши



Дизайн вознаграждения

Для улучшения результатов в системе вознаграждения были введены следующие изменения:

zero-sum

согласно правилам антагонистической игры, при выигрыше камней одним игроком, у противника соответственно уменьшалась награда на такое же количество камней

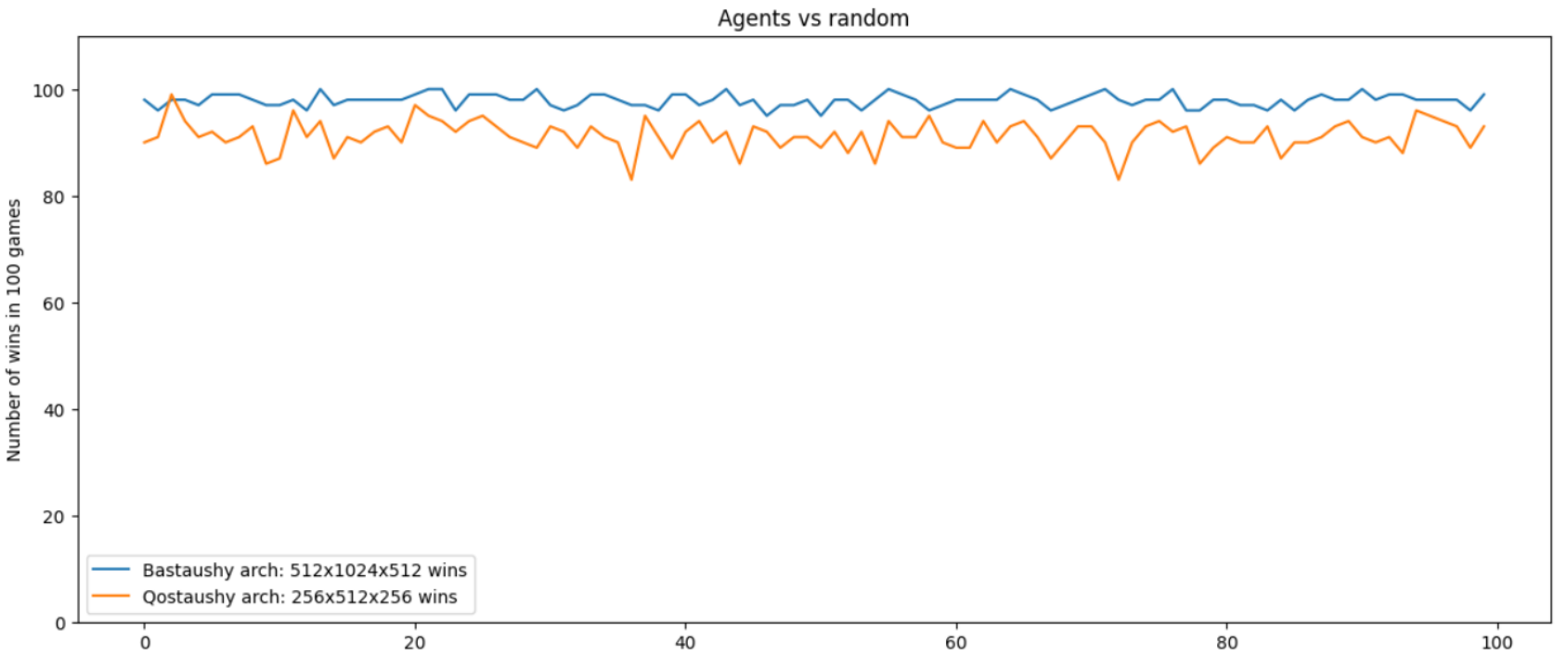
wins reward

награда за победу присваивалась в размере 500 камней

tuzdyq

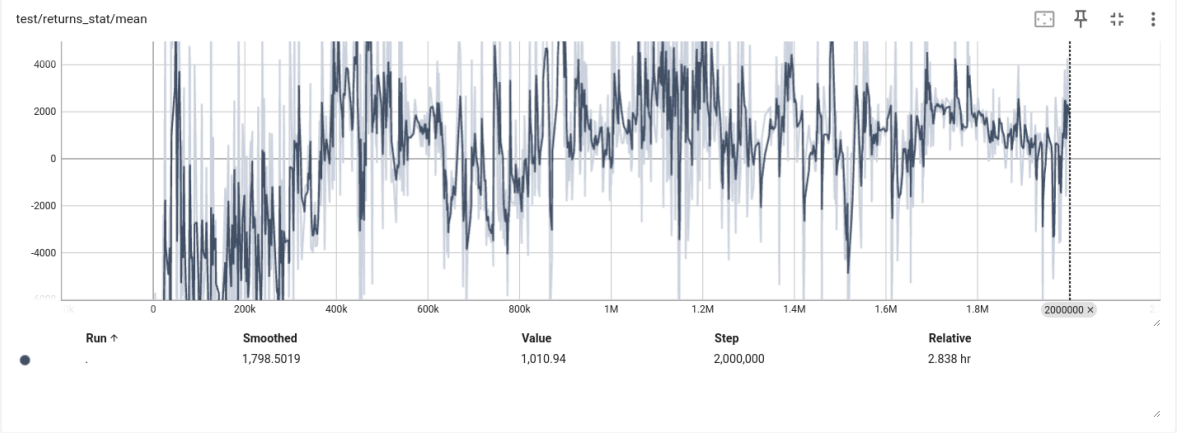
дополнительная награда за завоевание Туздыка была равна 50 камням.

Выигрыши обученных агентов против рандомной политики

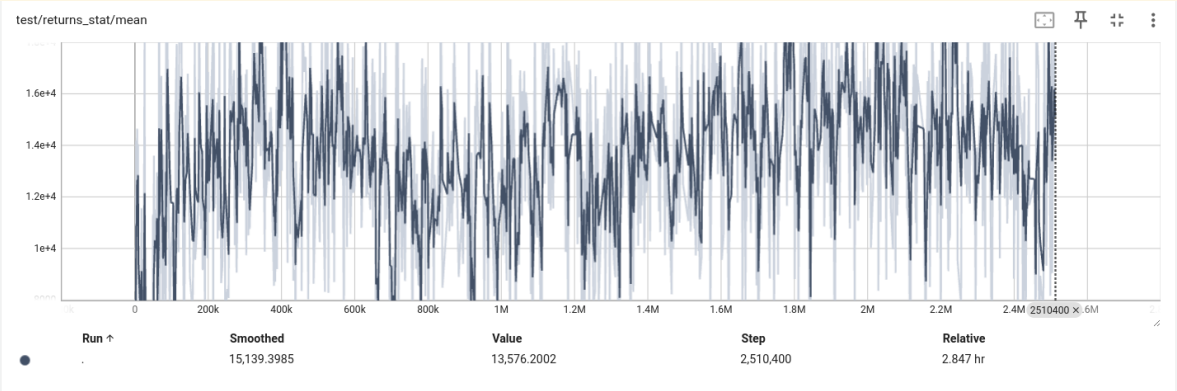


Self-play

Результаты обучения с обученным оппонентом и $\varepsilon = 0.0$-жадной политики



Результаты обучения с обученным оппонентом и $\varepsilon = 0.5$-жадной политики



Результаты партий с человеком

Агенты	1	2	3	4
Бастауши/Человек	0:1	0:1	0:1	1:0
Человек/Костауши	1:0	1:0	0:1	1:0

Веб приложение игры

Проект: https://github.com/zhus-dika/togyz-qumalaq-agent.git

Запуск приложения: python app/app.py