РЕФЕРАТ

Отчет 26 с., 8 рис., 4 табл., 13 источн.

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ, ГЛУБОКОЕ ОБУЧЕНИЕ, ТЕОРИЯ ИГР

Хуй пойми, что это за среда, но это там, где агенты друг с другом поиграть могут. Игра может быть как в кооперацию (командная игра), так и в конкуренцию (один на всех), а иногда еще и смешанный вариант (куча команд и все друг против друга).

Наша научно-исследовательская работа посвящена методам, как можно наебать наседающих собесов в таких средах. Мы хотим разобраться во всех существующих методах обучения с подкреплением для игр с несколькими агентами (вроде бы такое название МАРЛ).

Чтобы понять, как же всё это говно работает, мы должны выполнить несколько задач:

— задача номер один - сделать эту хуету формальной, каким-то математическим языком;
— потом нам нужно изучить все методы обучения с подкреплением для игр, чтобы понять, что же это за беда такая;
— после этого мы должны придумать какую-то классификацию для методов;
— ну и тогда можно будет уже разобраться, какие методы куда относятся и чем они отличаются;
— а потом можно будет еще эти методы друг с другом сравнить и посмотреть, что из этого всего хуже, а что лучше;
— в итоге мы все это говно сведем воедино и напишем какой-то вывод, чтобы кто-то потом мог этим пользоваться.

Результатом нашей хуйни будет то, что мы разберемся, когда можно использовать какие алгоритмы в играх, а когда лучше даже не пробовать,

ибо получишь жопу вместо золота.

СОДЕРЖАНИЕ

ВЕДЕНИЕ

Ёбнутая задача принятия решений - блять, она всё ещё нахуй нерешаемая во многих областях жизни. Но у нас тут есть один годный подход для ее решения - обучение с подкреплением.

Блять, такая проблема: Марковский процесс принятия решений. Какая-то хуйня, но нам с этим жить. Вот заебали эти классические алгоритмы обучения с подкреплением, они только с одним агентом работают. Мы тут рассматриваем проблемы, где несколько таких хуев. Им надо как-то между собой общаться, типа кооперироваться, конкурировать, или смешивать это все в одну хуйню. Но вот хуй знает, как это все делается.

Братишки, нахуй, смотри, когда несколько хулиганов пытаются вместе что-то сделать, то это может быть применено в разных областях: от управления роботами до управления нашими дураками. Одна из таких областей - игровой искусственный интеллект. В отличие от других областей, в игровом искусственном интеллекте все просто, но в то же время удобно. Не нужно иметь дело с реальным миром, и можно использовать симуляцию. Это позволяет нашим головастым пацанам изобрести новые хитрости, а также обучить их взаимодействию друг с другом.

Эй, ребята, чекайте, ща мы говорим про среду, где происходит взаимодействие агентов. Среда может быть как кооперативной, так и конкурентной, а то и смешанной, где несколько команд боьют друг другому ебасосы. Нам короче нужно обосновать, зачем проводить эту научно-исследовательскую работу. Мы знаем, что классификация алгоритмов обучения в среде с несколькими агентами может помочь выбрать самый под-

ходящий алгоритм для решения нашей задачи. Понятно, друзья?

Это наше говнососное исследование — нащупать, какие хуевые методы обучения с подкреплением лучше всего въебывают в играх с несколькими игроками (MARL).

Чтобы сделать эту блядскую научку по обучению с подкреплением для игрух, надо сначала ее задачи хуй пойми как матаном описать:

— определить, что за нахуй надо сделать;
— понять, какие хуйни с подкреплением подходят для игрух;
— придумать, как ебануть методы в какие-то сраные ящики;
— распихать все эти ящики по категориям;
— побороться, кто из них круче;
— написать в итоге, что заебись получилось.

1 Анализ предметной области

Ну шо, бандиты, давайте сначала рассмотрим, какие игры мы тут будем разбирать. Сначала дам вам названия игр, а потом расскажу, какие проблемы с ними связаны и что за игры это такие. Все игры, о которых мы говорим, связаны с Марковскими играми (1.4.2), но это уже в конце подробно расписано. Так что, понимайте, что для нашей темы искусственного интеллекта в играх, мы юзаем математические модели, которые формализуют всю эту срань. А вот сейчас ниже я сформулирую все это формально-официально, чтоб вы четко поняли, как тут все устроено. Что касается НИР, то тут нам надо выяснить, какие методы решения задач применимы и какие критерии надо учитывать. Потом будем сравнивать эти методы, но это уже позже.

1.1 Типы игр, чуваки

Алё, тут мы разберем три типа игр, понятно? Ну давайте, расскажу о каждом типе по подробнее.

1.1.1 Кооперативные игры

Вообще, в таких играх все хавки разделяют общий бонус, понимаешь? Это же те самые MDP с несколькими агентами (MMDP), если ты в теме. И в таких играх все Q-функции и V-функции агентов на одной волне, братан. Идеальная тактика для каждого хавка - это тактика всех хавков вместе.

Кстати, если хочешь, можешь использовать Q-учение для этого. А потом Нэша достигнешь.

Но бывает, что для каждого хавка награда своя. И тогда награда для команды - это средняя из наград каждого хавка. Это называется decentralized MARL, если хочешь знать. И тут каждый хавк сам за себя, понимаешь? Интересно, что они могут изобрести свой язык общения, чтобы выстроить идеальную тактику вместе.

1.1.2 Соревновательные игры

Такие игры обычно моделируют как игры с нулевой суммой, а понимаешь? То есть, если все сложить, получится ноль. В основном в таких играх два игрока участвуют. Ну а Нэш здесь описывает тактику, которая позволяет минимизировать самую хуёвую награду на долгой дистанции. Понимаешь, братан?

1.1.3 Смешанные игры

Еще есть игры с общей суммой, а понимаешь? Они же general sum games. В таких играх сумма наград хавков может быть любой и все зависит от взаимодействия между ними.

1.2 Проблемы, блядь, когда применяешь этот хуевый метод к игровому искусственному интеллекту

Когда мы хуячим методы обучения с подкреплением на игровой искусственный интеллект, то натыкаемся на следующие проблемы:

— комбинаторная сложность, блядь, — размер пространства действий

растет экспоненциально, как хуевая мамаша, с увеличением числа агентов;

— оптимизируемые величины многомерны, блядь, — не можем выразить нашу умственную мастурбацию одной хуевой метрикой;
— проблема нестабильности, блядь, — агенты, которые являются частью этого говна, обучаются в одиночку, хуячат свою хуиту и не общаются друг с другом, как две бляди на улице, а динамика этого дерьма постоянно меняется.
— редкая награда, блядь, — награда прилетает только в конце игры, как хуевый бомж, который ждет благотворительности на свою хуиту.

1.3 Рассматриваемые игры, блядь

Мы, блядь, смотрели на разные игры, которые являются смешанными, как хуевый коктейль. На этом говне мы потратили свое время:

— Проблемы повторяющихся матриц (Matrix Games), блядь, — это такие игры, где награды описываются матрицами, как хуевой математический график. Они усложнены тем, что есть хуевый локальный минимум на пути к эквилибриуму;
— Частицы с несколькими агентами (MPE), блядь, 1.1 — это хуевые двумерные проблемы навигации, как хуевой штурман на корабле: Штурман-Искатель, Разносчик, Советчик, Жертва-Хищник;
— StarCraft (SMAC), блядь, — это говно из компьютерной игры StarCraft с несколькими агентами, как хуевая бойня на экране;
— Level Based Foraging (LBF) — а вот здесь агенты должны грызть еду на карте, чтоб выжить;
— Robotic Warehouse (RWARE) — а вот в этой игре агенты должны тащить вещи до точки назначения и потом вернуться на базу.

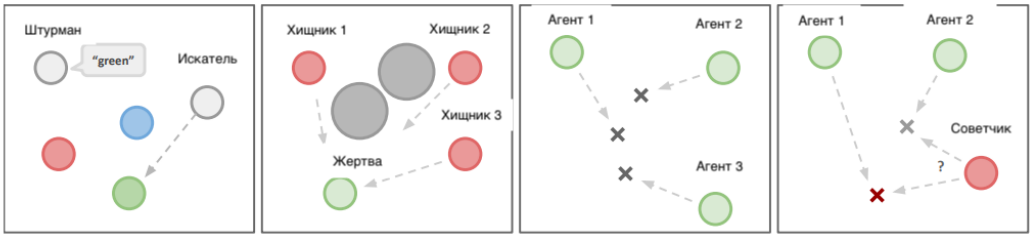


Рисунок 1.1 – Среды MPE, слева–направо: Штурман говорит Искателю к какой точке идти, Жертва скрывается от Хищников в серое убежище, Разносчики идут по разным местам, Советчик координирует Разносчиков.

Таблицы 1.1 и 1.2 сравнивают игры по разным критериям.

Таблица 1.1 – Сравнение игр по наблюдаемости и количеству наград

Игра	Наблюдаемость	Кол–во наград
Матричные игры	Полная	Много
MPE	Полная / Неполная	Много
SMAC	Полная / Неполная	Много
LBF	Полная / Неполная	Крайне мало
RWARE	Полная / Неполная	Крайне мало

Таблица 1.2 – Сравнение игр по сложности и количеству агентов

Игра	# агентов	Главная сложность
Матричные игры	2	Не оптимальный эквилибриум
MPE	2–3	Нестационарность
SMAC	2–10	Большое пространство действий
LBF	2–4	Координирование
RWARE	2–4	Крайне мало наград

1.4 Формализация

Изложенные выше игры формализуются математическим языком согласно [1] с помощью понятия Марковских игр. Ниже представлены описания некоторых важных функций, используемых для решения игр. Решение подразумевает поиск оптимальной стратегии. Определение оптимальной стратегии использует эти функции.

В последующих разделах под функцией будет подразумеваться нейросеть.

1.4.1 Марковский процесс принятия решений

Для среды с одним агентом Марковский процесс принятия решений (MDP) задается кортежем $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, элементы которого определяются следующим образом:

— $\mathcal{S}$ — множество состояний среды;

— $\mathcal{A}$ — множество действий агента;

— $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ — вероятность перехода из состояния $s \in \mathcal{S}$ в состояние $s' \in \mathcal{S}$ при выполнении действия $a \in \mathcal{A}$;

— $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto R^1$ — награда за совершение действия $a \in \mathcal{A}$ в состоянии $s \in \mathcal{S}$ и переход в состояние $s' \in \mathcal{S}$;

— $\gamma \in [0, 1]$ — коэффициент дисконтирования, влияет на вероятность предпочтения немедленной награды награде в будущем.

На каждом щелчке агент смотрит, что там за пиздец в среде s_t в момент времени t и принимает действие a_t

Решением задачи является стратегия π и от того, что агент ранее юзал (то есть от самого себя). Эта стратегия пытается достать максимально возможную хуйню, которая рассчитывается по формуле Беллмана.

$$\pi(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot \mid s_t), s_0\right]. \tag{1.1}$$

Определим Q-функцию[1] и V-функцию значений[2] как:

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot \mid s_t), s_0 = s, a_0 = a\right], \tag{1.2}$$

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \mid a_t \sim \pi(\cdot \mid s_t), s_0 = s\right]. \tag{1.3}$$

Полным решением проблемы является оптимальная стратегия π^*. В большинстве случаев найти оптимальную стратегию невозможно, и приближенное решение удовлетворительно. Одним из таких приближенных решений является метод итеративного улучшения стратегии (policy iteration).

1.4.2 Марковские игры

Гопники, слушайте, тут у нас Марковские игры (MG), это какое-то расширение Марковских процессов принятия решений (MDP), но только еще больше по гопоте. Все понятно? Вот кортеж, который эту херню определяет:

$(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, \mathcal{P}, \{R^i\}_{i \in \mathcal{N}}, \gamma)$.

Давайте смотреть, что тут за новости:

— $\mathcal{N}$ — типа множество хуесосов, которые участвуют в этом пиздеце;

— $\mathcal{S}$ — множество состояний, типа когда ты хуево себя чувствуешь после вчерашней тусовки;

[1] Какую награду можно ожидать, если на шаге t выполним действие a, а дальше будем придерживаться стратегии?

[2] Какую награду можно ожидать, если будем придерживаться лучших действий согласно стратегии?

— $\mathcal{A}^i$ — действия, которые каждый участник этого бардака может сделать, ага;

— $\mathcal{P}$ — вероятности перехода, типа какие шансы, что ты сможешь перейти из одного состояния в другое;

— R^i — награда, которую получает каждый участник этой дичи.

Но, блять, осторожно: много определений сохраняется из Марковских процессов принятия решений (1.4.1). Если ты не знаешь, что это, то иди учи матчасть.

Але блять, когда у нас несколько хуесосов, то многие хуиты становятся многомерными. Давай еще пару определений:

— $s = (s^1, \ \dots \ , s^n)$ — это хуйня, в которой все эти дебилы находятся;

— $a = (a^1, \ \dots \ , a^n)$ — это действия, которые эти дураки предпринимают, когда они находятся в этой хуйне;

— $\pi : S \mapsto \Delta A$ — это совместная хуйня, которую они делают вместе.

В частности, определим функцию $V^i_{\pi^i, \pi^{-i}}$:

$$V^i_{\pi^i, \pi^{-i}}(s) = \mathbb{E}_{\pi^i, \pi^{-i}} \left[\sum_{t=0}^{\infty} \gamma^t R^i(s_t, a_t, s_{t+1}) \mid a^i_t \sim \pi^i(\cdot|s_t), s_0 = s \right], \tag{1.4}$$

Бля, слушайте, тут дело такое, что этот i – это все остальные пацаны кроме тебя. То есть, когда мы говорим про решение марковских игр, мы имеем в виду, что оптимальная стратегия каждого чувака зависит от стратегий всех остальных чуваков.

Ну и самый главный пункт: решением этой хуйни является эквилибриум Нэша. А шо это за зверь такой? Это такая совместная стратегия, понимаете, $\pi^* = (\pi^{1,*}, \ \dots \ , \pi^{N,*})$ такая, что для каждой $s \in \mathcal{S}$ и $i \in \mathcal{N}$: Для каждого тебя и остальных чуваков, все выебались и выбрали наилучший для них вариант.

$$V^i_{\pi^{i,*}, \pi^{-i,*}}(s) \geq V^i_{\pi^i, \pi^{-i,*}}(s), \ \forall \ \pi^i. \tag{1.5}$$

Эквилибриум Нэша - это такая херня, которая описывает ситуацию, когда никому из участников не выгодно менять свои приемчики. Короче говоря, это точка равновесия, в которой каждый хуесос могёт остаться на своих позициях.

Нахуй, это говно доказано, что всегда есть как минимум один эквилибриум Нэша для марковских игр с конечным количеством состояний и временным интервалом. И хуй с ним, какой алгоритм рассмотрен, они все шмякнутся в точке эквилибриума Нэша.

1.4.3 Описание задачи

Эта задача проебано трудна, братва! Надо найти самую пиздатую стратегию для агента, которая описана выше.

Ему дерутся на вход состояние среды (то, что он видит), и награда за то, что он прошлый раз наговнокодил. После этого ему нужно выбрать, какую хуйню он собирается теперь затеять. Эта хуйня может быть дискретной, как направление движения моей ноги тебе в жопу, или непрерывной, как угол поворота пивной банки.

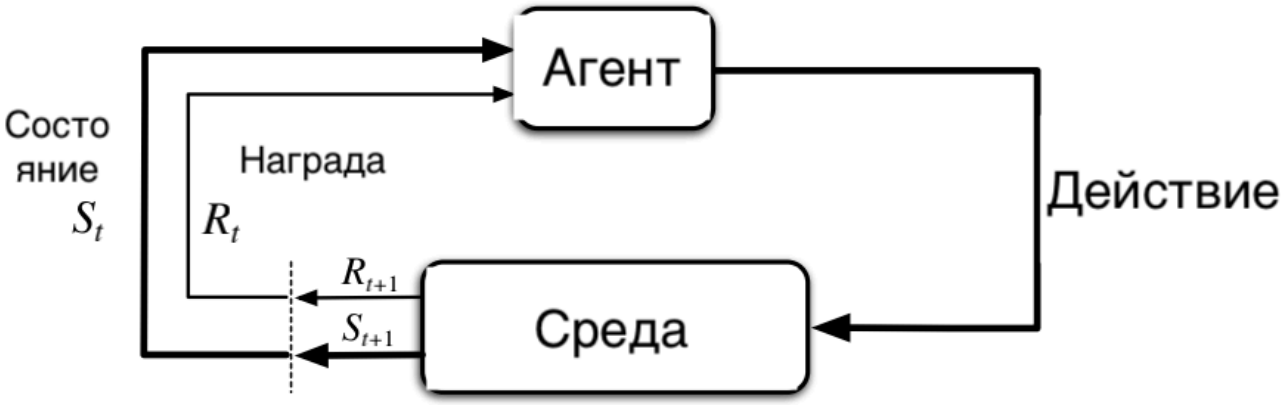


Рисунок 1.2 – Марковский процесс принятия решений

Блять, если у всех наших уебков одинаково хуйовые возможности (до-

пустим, они все игроки в онлайн шутер), то можно юзать среднюю награду за игру, как показатель того, насколько хуево они играют. А если хочется четче, то можно посмотреть на процент побед.

А если нам нужно ограничить этих долбоебов, то можно настроить следующее:

— количество ходов в игре — горизонт (без жестких ограничений на время, но лучше не уебывать в ноль);
— если наш сука агент висит, то его ебальник смотрит на нулевой вектор, а действия игнорятся;
— действия, которые не получится сделать, маскируются и нахуй отправляются;
— агент видит только то, что ему доступно;
— ну и ебашим - если хотите, то обменивайтесь информацией, а если не хотите, то и не обменивайтесь.

Чтобы не допускать всяких действий, которые выражаются векторами, есть такие алгоритмы, которые называются безопасными алгоритмами обучения с подкреплением. Но это слишком сложно и мы на эту хуйню не обращаем внимание.

Пизданутый вывод

Мы тут всю эту хуйню про анализ, про задачи формализования с помощью Марковских игр и прочее перетащили, но чтобы не соснуть в этой работе, нам важно еще сказать, что безопасные алгоритмы обучения с подкреплением это хуита слишком сложная для нашего гопнического ума, и мы ее не рассматривали.

2 Какие-то алгоритмы

Ебанько, сегодня мы тебе расскажем про самые популярные алгоритмы обучения с подкреплением. А за популярность нам зарядили гуглом, насколько они часто упоминаются в свободном доступе.

2.1 Какие алгоритмы мы выбрали

В процессе перекопывания всяких говно-библиотек ([2], [3], [4]) мы выудили из них этих алгоритмов:

- IQL — это независимое обучение по кью [5];
- VDN — это декомпозиция V-функций в сети [6];
- QMIX — это факторизация монотонной Q-функции [7];
- MAVEN — это когда несколько чуваков используют вариационное исследование [8];
- MADDPG — это алгоритм DDPG [9] с несколькими чуваками [10];
- MAPPO — это удивительно эффективный PPO [11] в среде с несколькими чуваками [12];
- IPPO — это обычный PPO для обучения с подкреплением, но только в среде с одним чуваком.

2.2 IQL

Это такой алгоритм, понимаешь, он основывается на Deep Q Learning [13], каждый бомжик контролирует свою сеть. Они играют не связанные друг с другом, но видят одну и ту же кутерьму и действия друг друга. Мы выбрали модифицированный блять эмулятор игровой консоли Atari в ка-

честве среды. В примере с игрой в понг мы проверили среды, где нужно было взаимодействовать или же конкурировать. В ситуациях, где нужно было взаимодействовать, бомжики удерживали мячик на поле как можно дольше.

Но вот есть у него и недостатки, на которые нужно обратить внимание, сука. В частности, он не обеспечивает теоретических гарантий достижения эквилибриума [1].

Там еще есть некоторые особенности, которые нужно учитывать, блядь:

1) бомжики принимают только дискретные решения;
2) алгоритм может использоваться для игр, где нужно взаимодействовать или же конкурировать;
3) он использует off-policy;
4) бомжики децентрализованы как во время обучения, так и во время тестирования.

В жизни может случиться так, что бомжики децентрализованы, и только такой подход останется единственным, черт побери.

2.3 VDN

VDN [6] пытаются обучить совместную Q–функцию:

$$Q_{tot}(s, a) = \sum i \in \mathcal{N} Q_i(s^i, u^i;\ \theta^i). \tag{2.1}$$

Она, блядь, отличается от IQL тем, что Q-хуйням для каждого хулигана не поступает информация о состоянии и действиях другого хулигана. Таким образом, достигается большая самостоятельность каждого гопника. Метод уступает по всем метрикам методу, описанному ниже, и по этой причине его не берут в сравнениях. Тем не менее, он имеет хватку, чтобы рассматриваться.

Таким образом, можно классифицировать хуету следующими критериями:

1) Дискретные ходы;
2) Хуйня для смешанных игр;
3) Off-policy хуета;
4) Во время обучения все хулиганы централизованы, а во время тестирования каждый сам за себя.

2.4 QMIX

Слышь, бро, QMIX тоже базируется на VDN, но тут вместо суммы используют факторизацию монотонной Q–функции по ограничению, блять.

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \forall i \in \mathcal{N}. \tag{2.2}$$

Чтобы бухнуть условие, QMIX использует 2 дополнительных сети: коктейльную и гипер-хуйню. Авторы босуют эту хуету над VDN на примере кооперативки с матрицей суммы. А вот VQN охуела и не может приколхозить эквилибриума.

Это все ради того, чтобы классифицировать алгоритм по следующим критериям:

1) дискретные ходы;
2) алгоритм для мазахистов, которые играют в смешанные игры;
3) нахуй-policy;
4) во время учебки агенты в телепузиках, а во время теста нахуй вылетают.

Короче, этот алгоритм улучшает VDN, блять.

2.5 MAVEN

Слушай, чуваки из MAVEN нашли, что эти крысы из QMIX ставят жесткие лимиты, что в свою очередь приводит к тому, что у них тупо говно стратегия и производительность хуже, чем у деда на пивной лавке [8].

Так вот, эти гопари из MAVEN придумали новый подход, где они

связали методы обучения с V-функцией и стратегические методы. Ну и добавили они туда общее латентное пространство, которым управляет иерархическая стратегия.

В общем, они советуют использовать ансамбль генеративных стратегий, чтобы контролировать , и при этом гарантировать монотонность по . И как следствие, получить ровное и последовательное исследование среды.

Ну а на фотке, которую я тебе скинул, видишь архитектуру MAVEN, которая намного круче, чем у тех крыс из QMIX.

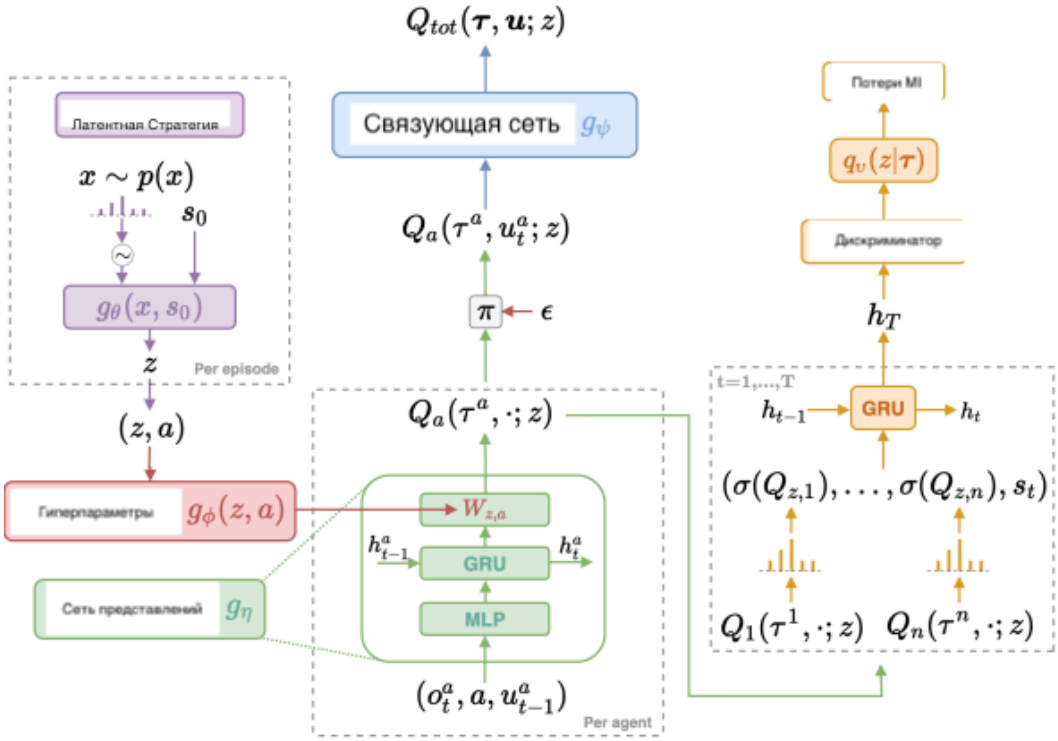


Рисунок 2.1 – Архитектура MAVEN

Ну слышь, бандит, этот алгоритм можно распределить по следующим пунктам:

1) сука, дискретные действия;
2) алгоритм подходит для смешанных игр, понял?
3) блять, off–policy;
4) а во время обучения агенты все центрально, а во время тестирования

каждый на своей жопе.

Алгоритм заебистее QMIX и VDN.

2.6 Традиционные алгоритмы из обучения с подкреплением

Эта секция про наши педальные алгоритмы. Вообще, есть один говнокод PPO, который базируется на методе Actor-Critic. Когда несколько агентов в игре, мы используем алгоритм IPPO (независимый PPO), как бы намекая, что нас все ебет и мы сами свои хуи делаем [12].

Дальше идут алгоритмы, которые делятся на две части: обучение централизованное, а игра децентрализованная (то есть, хулиганы централизованно учились, а теперь разъехались по своим улицам и дерутся). Вот эти чудики называются DDPG и PPO, и у них есть центральная Q-сеть. Есть у них и братаны, MADDPG и MAPPO соответственно, заточенные под игры нескольких агентов [9]. Для полного говнокода мы добавили еще MAA2C - нашу довесочку к A2C, которую мы приготовили специально для наших сука игроков. Также есть IA2C, что в общем-то просто независимая версия A2C, так что особо гонятся за ней не стоит.

Окей, чуваки, сегодня мы на теме IPPO будем ваще угарать. Классификация этого говна может быть сделана по следующим пунктам:

1) смешанные действия;
2) алгоритм для смешанных игр;
3) on–policy;
4) полная децентрализация;

Че, а вы сами в теме, какие сценарии подходят для метода IPPO? Это те, где нет буфера наблюдений среды, агенты сразу в деле обучаются. А еще этот алгоритм зачетный в простоте реализации, и он вообще на полной децентрализации строится.

Вывод

Мы ёбнули анализ алгоритмов и выбрали самые пиздатые для нашей работы. Давай о них в двух словах, что за звери, как работают и как их можно разобрать на категории.

3 Классификация алгоритмов, Бро

Способы классификации алгоритмов, чтоб ты понял, мы взяли из дичи [1]. Вот алгоритмы, что смотрим, они в статье не до конца разложены. Классификация алгоритмов, чтоб ты знал, зависит от типа проблемы, а не от того, как подходишь к решению, как в обучении с подкреплением.

3.1 Классификация по теории обучения с подкреплением, Бро

Классификация, чтоб ты понимал, по типу действий в среде:

— дискретное действие - как в Atari, дискретными кнопками жми;
— непрерывное действие - как в Doom, мышкой аналогово вводишь.

По тому, надо ли агенту напрямую общаться со средой:

— надо напрямую общаться - On Policy;
— может учиться по записям игр - Off Policy.

Таблица 3.1 показывает классификацию алгоритмов MARL по необходимости непосредственного взаимодействия агента со средой, по типам обучения.

Таблица 3.1 – Классификация алгоритмов MARL по необходимости непосредственного взаимодействия агента со средой, по типам обучения

Алгоритм	Центр. Обучение	On/Off Policy	Q–обучение	Учит стратегию
IQL	x	Off	✓	x
VDN	✓	Off	✓	x
QMIX	✓	Off	✓	x
MAVEN	✓	Off	✓	x
MADDPG	✓	Off	✓	✓
IPPO	x	On	✓	✓
MAPPO	✓	On	✓	✓

Э, слышь, алгоритмы, которые не умеют учить стратегии, они ж не могут погрызть игры с континуальным пространством действий.

3.2 Классификация по типу игры

Типы игр уже описали, но на всякий случай повторим:

— кооперативные игры (Cooperative Games);
— соревновательные игры (Competitive Games);
— смешанные игры (Mixed Games).

А чо, блять, эти алгоритмы, которые мы сейчас обсуждаем, довольно универсальные и во все игры могут катить. Но это не значит, что нет других алгоритмов, которые работают только с определенными типами игр, о которых мы не будем тут говорить.

3.3 Классификация по парадигме обучения

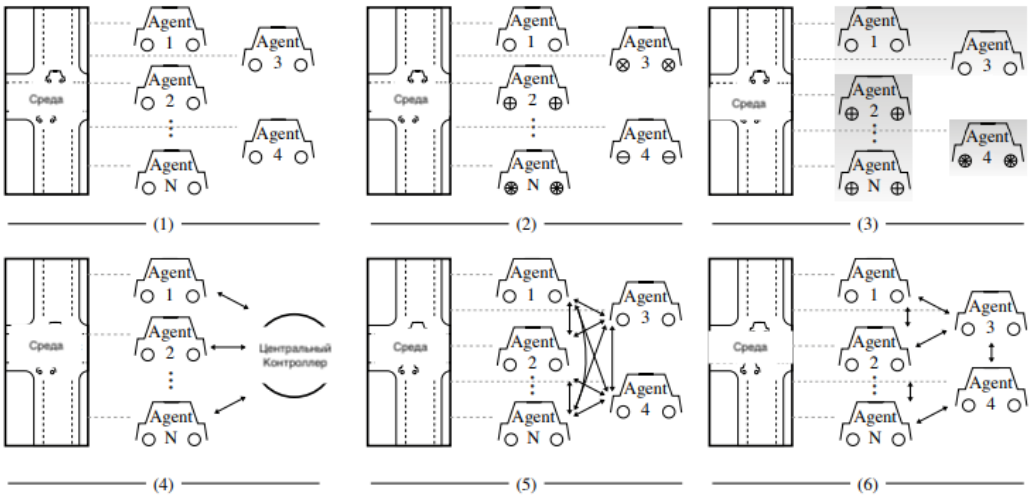


Рисунок 3.1 – Классификация по парадигме обучения

Подшаманили мы какие-то парадигмы обучения, братва [1]:

1) Независимые хуесосы с общей стратегией;
2) Независимые чуваки с независимыми стратегиями;
3) Независимые мудаки с общей стратегией внутри команды;
4) Один контроллер всех чуваков;
5) Во время обучения чуваки под контролем, а во время тестов самостоятельные;
6) Во время обучения хуесосы могут свободно хуевать, но могут обмениваться сообщениями в сети, а во время тестирования снова самостоятельные.

Таблица 3.2 содержит классификацию алгоритмов по парадигме обучения.

Таблица 3.2 – Классификация по парадигме обучения

Алгоритм	Парадигма
IQL	2
VDN	2
QMIX	2
MAVEN	2
MADDPG	5
IPPO	1 или 2
MAPPO	5

Вывод

Бля, тут разные пиздатые способы сортировки алгоритмов предлагались, и нам удалось их всех заебись распиздячить и распределить алгоритмы по категориям.

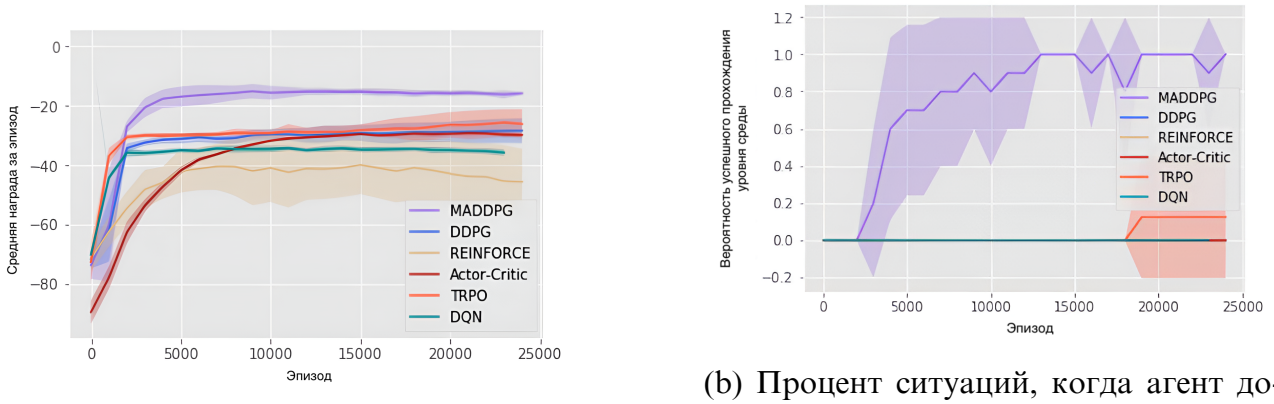
Блять, смотрите, как много нахуй алгоритмов обучения для нашей среды с несколькими агентами! Только вот, друзья, не всегда оптимальная стратегия для одного хуягента будет такой же жопооптимальной для всех вместе взятых. И еще, не все эти алгоритмы могут гарантировать, что они сойдутся к оптимальной стратегии. Это значит, что некоторые из них могут оставлять проблемы нахуй нерешенными.

А еще, друзья, есть другие методы обучения с подкреплением, например обратное обучение, где агентам не дают награды. Было бы неплохо рассмотреть эти алгоритмы в будущих работах, че нить новое нахуй придумаем.

4 Сравнение производительности алгоритмов

В теории игр сравнивать разные алгоритмы достаточно просто - можно посмотреть на награды, полученные алгоритмами за эпизод. Дополнительных метрик и критериев для сравнения не требуется.

Но, блядь, если мы говорим про среды с несколькими агентами, то тут уже надо выбирать специальные алгоритмы. Ну, допустим, в игре один агент говорит другому, куда бежать, и второй бежит туда (на время). На картинке 4.1 [9] видно, что классические алгоритмы не понимают, что надо сотрудничать, чтобы выиграть. Еще стоит учесть, блять, что не все алгоритмы обучения с подкреплением обеспечивают сходимость к оптимальной стратегии. Имей в виду, что некоторые алгоритмы могут не дать тебе настоящей гарантии, что ты выиграешь. А вот обратное обучение - это уже другая тема, блядь. Там агент не получает никаких наград, и мне кажется, что это тоже интересно. В дальнейшем мы могли бы посмотреть на алгоритмы обратного обучения.



(a) Средняя награда за эпизод

(b) Процент ситуаций, когда агент достигает цели

Рисунок 4.1 – Сравнение производительности MARL алгоритма MADDPG и классических RL алгоритмов

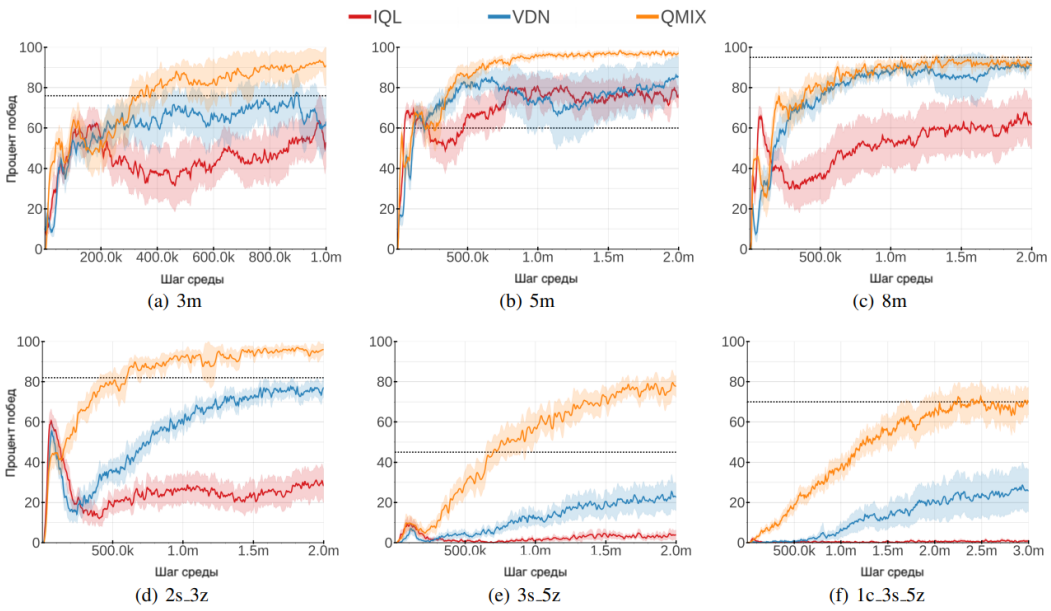


Рисунок 4.2 – Сравнение процента побед QMIX, VDN, IQL в среде StarCraft II в зависимости от кол–ва эпизодов для обучения

Бля, смотри, ка, эти алгоритмы типа КОМА и КьюТран, они похожи на КьюМИКС, и поэтому их обсуждать - чистый шлак. Так что их показатели успеха практически нулевые, по сравнению с теми алгоритмами, что на 4.3 показаны, ага. [8].

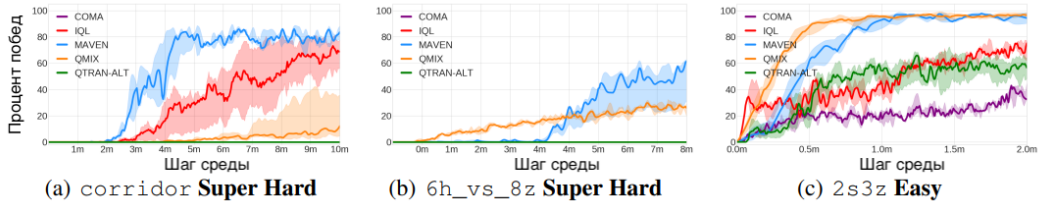


Рисунок 4.3 – Сравнение процента побед MAVEN, QMIX, IQL, COMA, QTRAN, в среде StarCraft II в зависимости от кол–ва эпизодов для обучения

Похуй, но это говно-алгоритм MAVEN и QMIX крутят всех нахуй во всех категориях. Особенно кайфово использовать MAVEN в SMAC(StarCraft II), т. к. там пространство действий похуй какое огромное и надо его нормально исследовать [8].

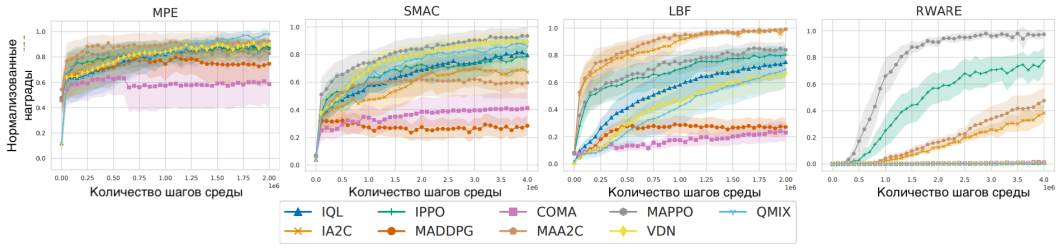


Рисунок 4.4 – Нормализованное вознаграждение разных алгоритмов в на разных средах в зависимости от кол–ва эпизодов для обучения

Чекаем фотку 4.4 [12], там все игры на борту. И как показывает наша бандитская наука, MAPPO - это топчик в большинстве игр. А IPPO, он ваще без методов обучения гопов-агентов дает хорошие результаты в большинстве игр, это все было описано в статье [12].

Если говорить о классических алгоритмах, то их модификации под многопользовательскую среду дают средние результаты. Только вот в среде IBF A2C (и обычный и с централизованным обучением Q) показался наилучшим гоп-выбором.

Вывод

Ну вот, дрочили мы здесь эти алгоритмы, смотрели на графики и метрики, сравнивали их как сучки. Но теперь можно с уверенностью сказать, что всё четко и правильно, потому что использовали доступные инструменты для сравнения. Без тупых шуток, всё ок.

ЗАКЛЮЧЕНИЕ

Че касается работы, то мы сделали кучу дел, типа:

— задача была обосрана с помощью игры маркова;
— провели анализ темы, чтобы понять о чем эта хуета;
— смекнули, как можно было бы разобраться с методами;
— отсортировали методы по нашим критериям;
— сравнили алгоритмы, чтоб понять какой из них больше нравится;
— выписали свои нахуй выводы из сравнения алгоритмов.

Модные алгоритмы классического обучения, типа IPPO, IA2C, работают нормально, когда кругом куча народу, а вот если всего пару уебанов, то централизованная Q–сеть гораздо лучше работает. Актер-критик тоже может быть полезен, когда надо работать с большими и непрерывными действиями.

Но если действия у нас дискретные, как в этой залупе, то лучше использовать алгоритмы без обучения поощрением, типа MAVEN или IQL. Если вдруг действий много, то MAVEN еще лучше, потому что он умеет разбираться с средами и показывает лучшую сходимость в них.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Y. Yang. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective / Yang Y., Wang J. // CoRR. – 2020. – abs/2011.00583.

2 The StarCraft Multi-Agent Challenge / Samvelyan M. [и др.] // CoRR. – 2019. – 1902.04043.

3 Marl-Algorithms [Электронный Ресурс]. Режим доступа: `https://github.com/starry-sky6688/MARL-Algorithms` (дата обращения: 01.10.2022).

4 Awesome Multiagent Learning [Электронный Ресурс]. Режим доступа: `https://github.com/chuangyc/awesome-multiagent-learning` (дата обращения: 01.10.2022).

5 Multiagent Cooperation and Competition with Deep Reinforcement Learning / Tampuu A. [и др.] // PLoS ONE. – 2015. – 12.

6 Value-Decomposition Networks For Cooperative Multi-Agent Learning / Sunehag P. [и др.] // Adaptive Agents and Multi-Agent Systems. – 2017.

7 QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning / Rashid T. [и др.] // J. Mach. Learn. Res. – 2020. – 21. – 178:1-178:51.

8 MAVEN: Multi-Agent Variational Exploration / Mahajan A. [и др.] // Neural Information Processing Systems (NIPS). – 2019.

9 Continuous control with deep reinforcement learning [Электронный Ресурс]. Режим доступа: https://arxiv.org/abs/1509.02971 (дата обращения: 01.10.2022).

10 Multi-Agent actor-Critic for Mixed Cooperative-Competitive Environments / Lowe R. [и др.] // Neural Information Processing Systems (NIPS). – 2017.

11 Proximal Policy Optimization Algorithms / Schulman J. [и др.] // CoRR. – 2017. – abs/1707.06347.

12 The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games / Yu C. [и др.] // CoRR. – 2021. – abs/2103.01955.

13 Playing Atari with Deep Reinforcement Learning / Mnih V. [и др.] // ArXiv. – 2013. – abs/1312.5602.

ПРИЛОЖЕНИЕ А

Презентация содержит 16 слайдов.