

# Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

Жижин Петр

Higher School of Economics

Жижин Петр 2019

# План доклада

1 Примеры Reinforcement learning

2 Необходимые знания

3 Мультиагентный RL

4 Идея статьи

5 Эксперименты

Список литературы

# Что такое Reinforcement Learning?

- Ищем хорошие действия
- Закрепляем их и повторяем в будущем

# Примеры

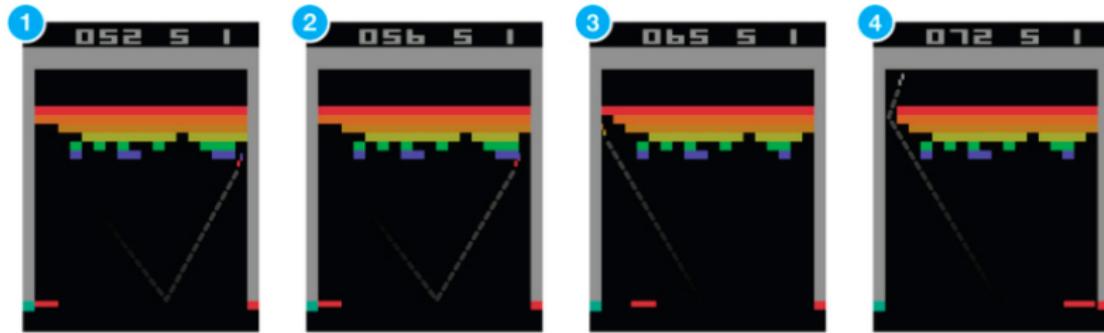


Рис.: Playing Atari with Deep Reinforcement Learning [5]

## Примеры

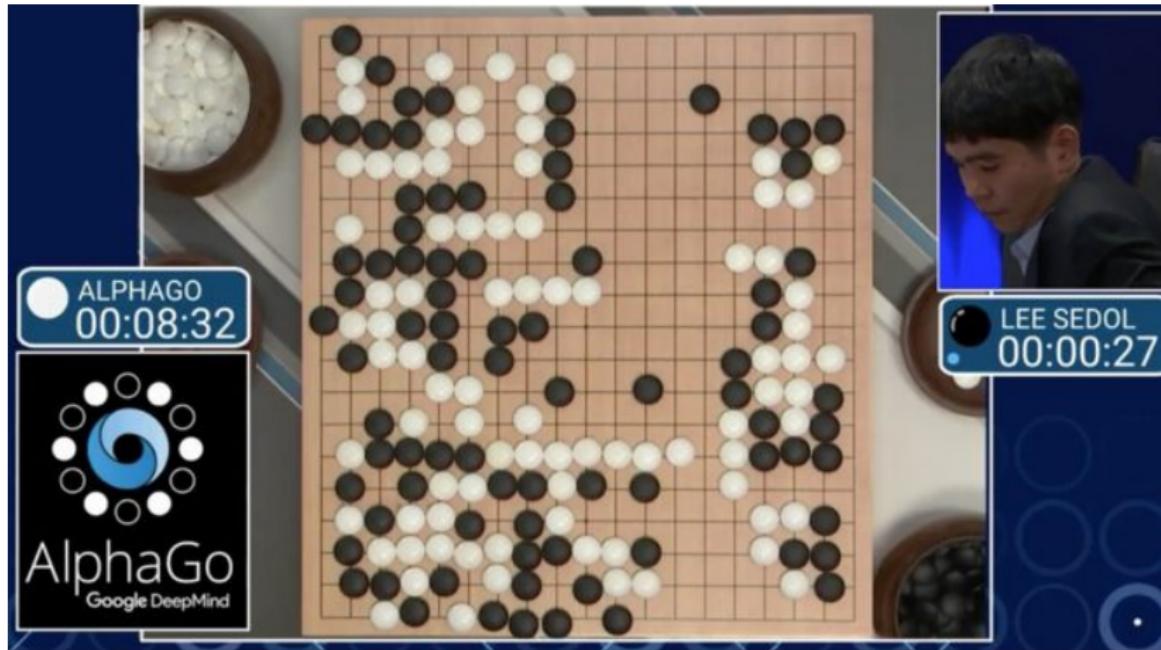


Рис.: Mastering the game of Go without human knowledge [8]

# Примеры

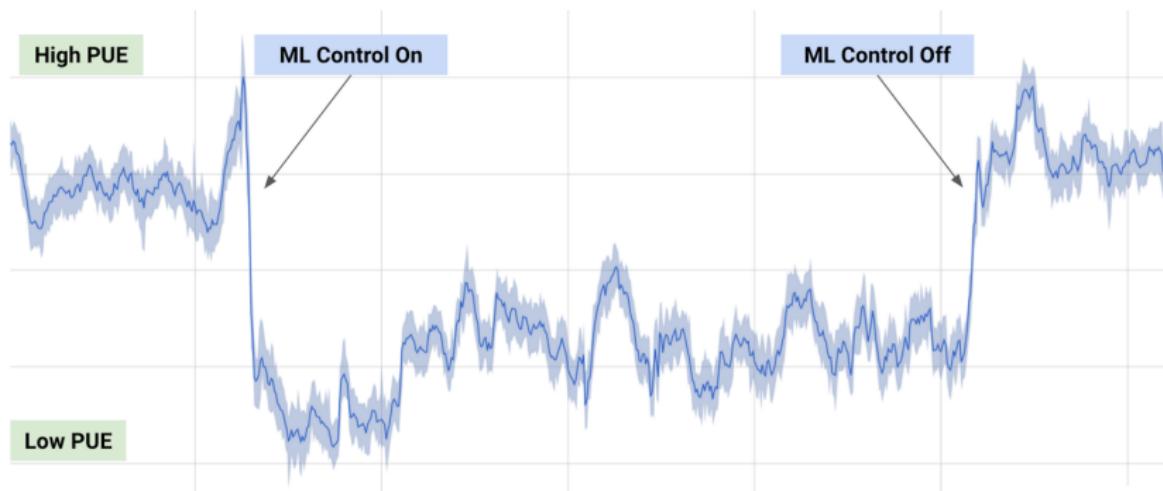


Рис.: DeepMind AI Reduces Google Data Centre Cooling Bill by 40%  
[2]

# Примеры

- ❶ Модель предсказывает на какую рекламу кликнут
- ❷ Показываем наиболее вероятные баннеры
- ❸ Проходит время, модель уже не работает
- ❹ Учим модель заново на новых данных

# Примеры

- ❶ Модель предсказывает на какую рекламу кликнут
- ❷ Показываем наиболее вероятные баннеры
- ❸ Проходит время, модель уже не работает
- ❹ Учим модель заново на новых данных

# Примеры

- ➊ Модель предсказывает на какую рекламу кликнут
- ➋ Показываем наиболее вероятные баннеры
- ➌ Проходит время, модель уже не работает
- ➍ Учим модель заново на новых данных

# Примеры

- ① Модель предсказывает на какую рекламу кликнут
- ② Показываем наиболее вероятные баннеры
- ③ Проходит время, модель уже не работает
- ④ Учим модель заново на новых данных

# Примеры

- ① Модель предсказывает на какую рекламу кликнут
  - ② Показываем наиболее вероятные баннеры
  - ③ Проходит время, модель уже не работает
  - ④ Учим модель заново на новых данных
- } RL!

## Примеры мультиагентных систем

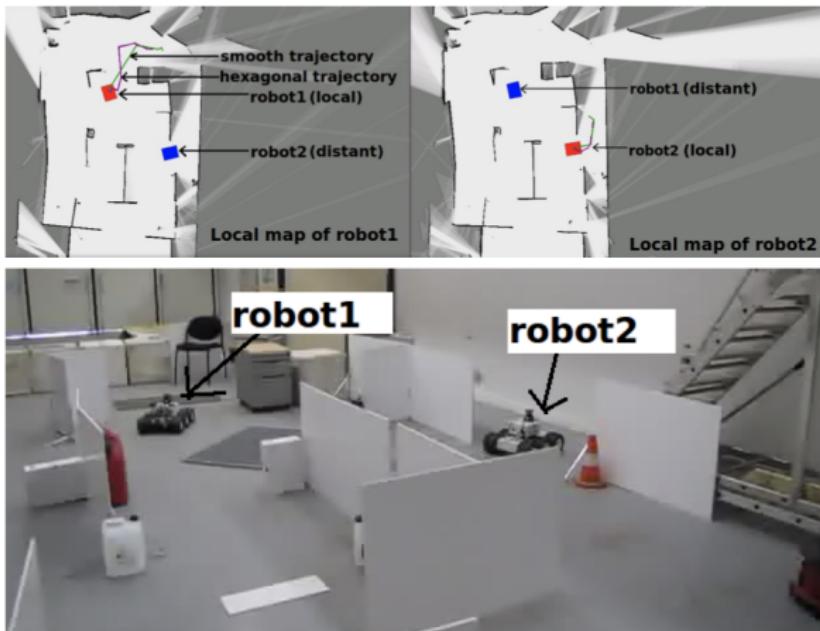
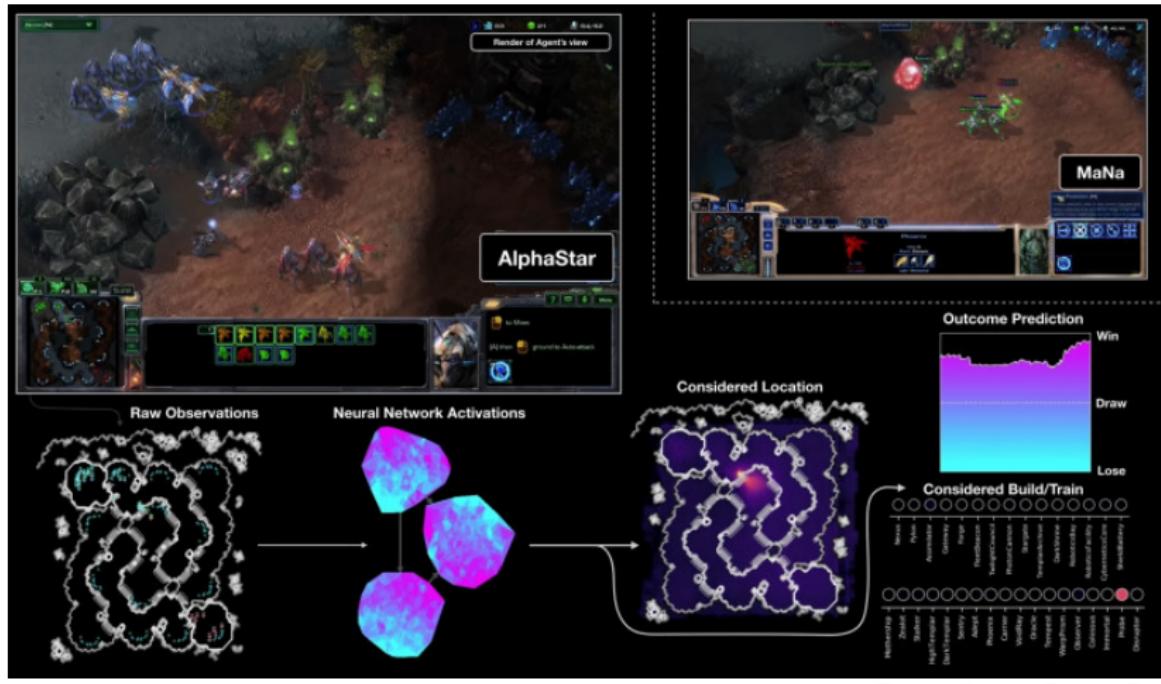


Рис. · Coordinated Multi-Robot Exploration [4]  
Жижин Петр

# Примеры мультиагентных систем



# Примеры мультиагентных систем

Scene 4: Team Zoning Mid Push

ACTIONS   OBSERVATIONS

Observed Units

Unit Type	Count
Controlled Hero	10
Neutral Creep	10
Controlled Creep	10
Neutral Minion	10
Controlled Minion	10

Team Dire

Stat	Value	Max	
Health	686 / 794	Attack	113
Armor	11	Distance	0
Level	9	Mana	108 / 531

Items   Abilities

Item	Ability
Health Pot	Heal
Mana Pot	Mana
Booster	Booster
Wards	Wards
Gold	Gold
Experience	Experience
Buyback	Buyback
Phasing	Phasing

Modifiers

Modifier	Value
Health	686 / 794
Attack	113
Armor	11
Distance	0
Level	9
Mana	108 / 531

On units of type Controlled Hero we also observe: absolute position; health over last 12 frames; attacking or attacked by hero; projectiles time to impact; movement, attack, and regeneration speed; current animation; time since last attack; number of deaths; using or phasing an ability; nearby terrain traversability; height, and creep occupancy; and buyback availability, cost, and cooldown.



Рис.: OpenAI Five [6]

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1)$ ,  $R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1)$ ,  $R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1), R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1)$ ,  $R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1)$ ,  $R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1), R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Основные понятия

- ❶  $\mathcal{S}$  — множество состояний системы
  - Пиксели с экрана
  - Знания о мире из API (StartCraft 2, Dota 2)
- ❷  $\mathcal{A}$  — множество возможных действий
  - Куда ткнуть на экране (StartCraft 2, Dota 2)
  - Какой двигатель активировать и на сколько (роботы)
- ❸  $\rho(s_0)$  — распределение на начальные состояния
- ❹  $\rho(s'|s, a)$  — распределение на следующее состояние из состояния  $s$  и при действии  $a$ .
- ❺  $r(s, a)$  — распределение на награду из  $s$  при  $a$
- ❻  $\pi(a|s)$  или  $\mu(s) \in \mathcal{A}$  — политика (как правильно играть в игру, управлять роботом)
- ❼  $\gamma \in (0; 1)$ ,  $R_i = \sum_{t=0}^T \gamma^t r_t$  — дисконтированный выигрыш (оптимизируем)

# Необходимые уравнения

- ➊ Наша политика играет и получает какой-то выигрыш
- ➋  $Q(s, a)$  — Q-функция (какой будет выигрыш, если из  $s$  сделать  $a$ )
- ➌ Политика оптимальна (получает наибольший возможный выигрыш) тогда и только тогда:

$$Q^*(s, a) = r(s, a) + \max_{a' \in \mathcal{A}} Q^*(s', a')$$

— основа Q-learning алгоритмов [9]

# Необходимые уравнения

- ❶ Наша политика играет и получает какой-то выигрыш
- ❷  $Q(s, a)$  — Q-функция (какой будет выигрыш, если из  $s$  сделать  $a$ )
- ❸ Политика оптимальна (получает наибольший возможный выигрыш) тогда и только тогда:

$$Q^*(s, a) = r(s, a) + \max_{a' \in \mathcal{A}} Q^*(s', a')$$

— основа Q-learning алгоритмов [9]

# Q-learning [9]

- ➊ Строим сеточку, которая предсказывает  $Q(s, a)$
- ➋ Играем и собираем  $\mathcal{D} \sim (s, a, r, s')$  – Experience replay
- ➌ Учим loss:

$$\mathbb{E}_{(s, a, r, s') \sim \mathcal{D}} \left[ Q(s, a) - r(s, a) - \max_{a' \in \mathcal{A}} Q(s', a') \right]^2 \rightarrow \min$$

# Policy gradient [9]

1

$$\mathcal{J}_\theta(s) = \mathbb{E}_{a \sim \pi_\theta(s)} Q(s, a) = \int_{a \in \mathcal{A}} \pi_\theta(a|s) Q(s, a) da$$

2

$$\nabla_\theta \mathcal{J}_\theta(s) = ???$$

# Policy gradient [9]

1

$$\mathcal{J}_\theta(s) = \mathbb{E}_{a \sim \pi_\theta(s)} Q(s, a) = \int_{a \in \mathcal{A}} \pi_\theta(a|s) Q(s, a) da$$

2

$$\nabla_\theta \mathcal{J}_\theta(s) = ???$$

# Policy gradient [9]

1

$$\mathcal{J}_\theta(s) = \mathbb{E}_{a \sim \pi_\theta(s)} Q(s, a) = \int_{a \in \mathcal{A}} \pi_\theta(a|s) Q(s, a) da$$

2

$$\nabla_\theta \mathcal{J}_\theta(s) = ???$$

3 Log-derivative trick:  $\nabla_\theta \log \pi(a|s) = \frac{\nabla_\theta \pi(a|s)}{\pi(a|s)}$ :

$$\begin{aligned} \nabla_\theta \mathcal{J}_\theta(s) &= \int_{a \in \mathcal{A}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q(s, a) da = \\ &= \underbrace{\mathbb{E}_{a \sim \pi_\theta(s)}}_{\text{Актор}} \underbrace{\nabla_\theta \log \pi_\theta(a|s) Q(s, a)}_{\text{Критик}} \end{aligned}$$

# Actor-Critic

- ➊ Строим сеточку, которая предсказывает  $Q(s, a)$
- ➋ Строим сеточку, которая предсказывает  $\pi(a|s)$
- ➌ Играем и собираем  $\mathcal{D} \sim (s, a, r, s')$  – Experience replay
- ➍ Учим Q-функцию:

$$\mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[ Q(s, a) - r(s, a) - \max_{a' \in \mathcal{A}} Q(s', a') \right]^2 \rightarrow \min$$

- ➎ Учим политику, поднимаясь по градиенту:

$$\mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)$$

## Deterministic policy gradient (DPG) [7]

- ➊ Когда вместо  $\pi(a|s)$  (распределения на действия) есть  $\mu(s)$  (просто действие)
- ➋ Тогда:

$$\nabla_{\theta} \mathcal{J}_{\theta}(s) = \mathbb{E}_{a \sim \rho^{\mu}} [\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q(s, a)|_{a=\mu_{\theta}(s)}]$$

# Мультиагентный RL

- $N$  агентов
- Вместо  $\mathcal{S}$  теперь  $\mathcal{O}_1 \times \dots \times \mathcal{O}_N$  — наблюдения индивидуальных агентов
- Вместо  $\mathcal{A}$  теперь  $\mathcal{A}_1 \times \dots \times \mathcal{A}_N$  — действия каждого агента
- Вместо  $R$  теперь  $R_1 \times \dots \times R_N$  — у каждого своя награда

# Почему бы не использовать стандартный подход?

- Давайте переназначим:
  - $\mathcal{S} = \mathcal{O}_1 \times \dots \times \mathcal{O}_N$
  - $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$
- Проблемы?

# Почему бы не использовать стандартный подход?

- Давайте переназначим:
  - $\mathcal{S} = \mathcal{O}_1 \times \dots \times \mathcal{O}_N$
  - $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$
- Проблемы?

# Почему бы не использовать стандартный подход?

- Давайте переназначим:
  - $\mathcal{S} = \mathcal{O}_1 \times \dots \times \mathcal{O}_N$
  - $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_N$
- Проблемы?
  - $R_1, \dots, R_N$  могут противоречить друг-другу
  - Одним из агентов может быть человек
  - Да, в принципе, можно (иногда)

## Отдельные агенты

- Давайте учить  $N$  отдельных агентов
- $Q_1, \dots, Q_N$  — отдельные  $Q$ -функции
- $\pi_1, \dots, \pi_N$  — отдельные политики
- Проблемы?

## Отдельные агенты

- Давайте учить  $N$  отдельных агентов
- $Q_1, \dots, Q_N$  — отдельные  $Q$ -функции
- $\pi_1, \dots, \pi_N$  — отдельные политики
- Проблемы?

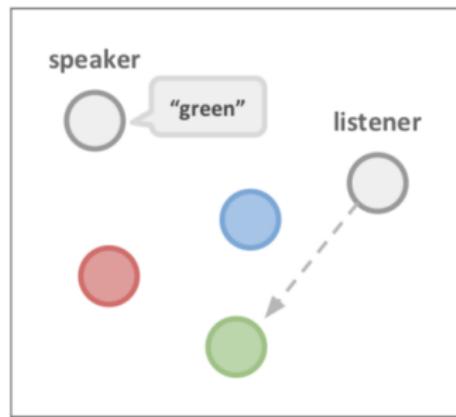


Рис.: Cooperative-Communication task [3]

## Почему так происходит? [3]

- $P(s'|s, a_i, \pi_1, \dots, \pi_N) \neq P(s'|s, a_i, \pi'_1, \dots, \pi'_N)$  – одни и те же действия агента могут приводить к разным результатам
- Игра с одним шагом.  $a_i \in \{0; 1\}$ . Выигрыш  $R(a_1, \dots, a_N) = I(a_1 = \dots = a_N = 1)$
- Пусть  $\nabla \mathcal{J}$  – настоящий policy gradient,  $\nabla \hat{\mathcal{J}}$  – policy gradient от одного агента
- Тогда  $P(\langle \nabla \mathcal{J}, \nabla \hat{\mathcal{J}} \rangle > 0) \propto (0.5)^N$

## Централизованный критик

- Вместо  $Q(s, a)$  для каждого агента учим  $Q_i(s, a_1, \dots, a_n)$
- Градиент лосса:

$$\nabla_{\theta_i} \mathcal{J} = \mathbb{E} [\nabla_{\theta_i} \log \pi_i(a_i|o_i) Q_i(s, a_1, a_2, \dots, a_n)]$$

- Утверждается, что лучше всего работает в случае детерминированных политик:

$$\nabla_{\theta_i} \mathcal{J} = \mathbb{E} [\nabla_{\theta_i} \mu_i(a_i|o_i) \nabla_{a_i} Q_i(s, a_1, a_2, \dots, a_n)|_{a_i=\mu_i(o_i)}]$$

- Учим  $N$   $Q$ -функций:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'} \left[ (Q_i(s, a_1, a_2, \dots, a_n) - y_i)^2 \right]$$

$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

# Схема

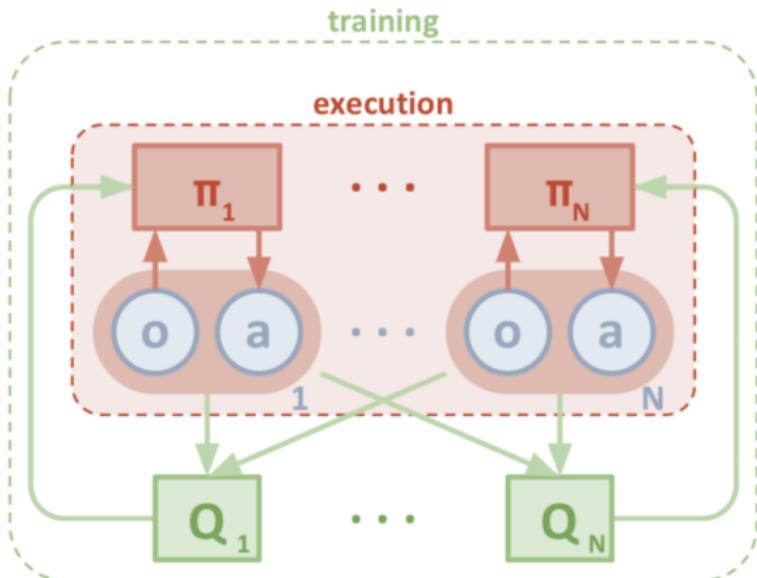


Рис.: Схема модели

## Предсказываем поведение других агентов

- Зачем это может быть нужно?
- 

$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

- Учим нейронку ( $\hat{\mu}_i$ ):

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ (\hat{\mu}_i(o_i) - \mu_i(o_i))^2 \right] + \underbrace{L_2(\phi_i)}_{\text{Регуляризатор}}$$

- Если политика детерминированная, то:

$$\mathcal{L}(\phi_i) = -\mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ \log \hat{\pi}_i^j(a_j | o_i) + \underbrace{\lambda H(\hat{\pi}_i^j)}_{\text{Энтропия-регуляризатор}} \right]$$

## Предсказываем поведение других агентов

- Зачем это может быть нужно?



$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

- Учим нейронку ( $\hat{\mu}_i$ ):

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ (\hat{\mu}_i(o_i) - \mu_i(o_i))^2 \right] + \underbrace{L_2(\phi_i)}_{\text{Регуляризатор}}$$

- Если политика детерминированная, то:

$$\mathcal{L}(\phi_i) = -\mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ \log \hat{\pi}_i^j(a_j | o_i) + \underbrace{\lambda H(\hat{\pi}_i^j)}_{\text{Энтропия-регуляризатор}} \right]$$

## Предсказываем поведение других агентов

- Зачем это может быть нужно?
- 

$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

- Учим нейронку ( $\hat{\mu}_i$ ):

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ (\hat{\mu}_i(o_i) - \mu_i(o_i))^2 \right] + \underbrace{L_2(\phi_i)}_{\text{Регуляризатор}}$$

- Если политика детерминированная, то:

$$\mathcal{L}(\phi_i) = -\mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ \log \hat{\pi}_i^j(a_j | o_i) + \underbrace{\lambda H(\hat{\pi}_i^j)}_{\text{Энтропия-регуляризатор}} \right]$$

## Предсказываем поведение других агентов

- Зачем это может быть нужно?
- 

$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

- Учим нейронку ( $\hat{\mu}_i$ ):

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ (\hat{\mu}_i(o_i) - \mu_i(o_i))^2 \right] + \underbrace{L_2(\phi_i)}_{\text{Регуляризатор}}$$

- Если политика детерминированная, то:

$$\mathcal{L}(\phi_i) = -\mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ \log \hat{\pi}_i^j(a_j | o_i) + \underbrace{\lambda H(\hat{\pi}_i^j)}_{\text{Энтропия-регуляризатор}} \right]$$

## Предсказываем поведение других агентов

- Зачем это может быть нужно?
- 

$$y_i = r_i + \gamma Q(s', \mu_1(s'), \dots, \mu_N(s'))$$

- Учим нейронку ( $\hat{\mu}_i$ ):

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ (\hat{\mu}_i(o_i) - \mu_i(o_i))^2 \right] + \underbrace{L_2(\phi_i)}_{\text{Регуляризатор}}$$

- Если политика детерминированная, то:

$$\mathcal{L}(\phi_i) = -\mathbb{E}_{(x,a,r,s') \sim \mathcal{D}} \left[ \log \hat{\pi}_i^j(a_j | o_i) + \underbrace{\lambda H(\hat{\pi}_i^j)}_{\text{Энтропия-регуляризатор}} \right]$$

- В статье написано, в препринте ошибка, в коде нету

# Борьба с переобучением

- Пусть агенты играют против друг-друга (StarCraft)
- Переобучение: агент выигрывает, так как подстроился под стратегию другого
- Как решать?

# Борьба с переобучением

- Пусть агенты играют против друг-друга (StarCraft)
- Переобучение: агент выигрывает, так как подстроился под стратегию другого
- Как решать?

# Борьба с переобучением

- Пусть агенты играют против друг-друга (StarCraft)
- Переобучение: агент выигрывает, так как подстроился под стратегию другого
- Как решать?

## Борьба с переобучением

- Пусть агенты играют против друг-друга (StarCraft)
- Переобучение: агент выигрывает, так как подстроился под стратегию другого
- Как решать?
- $K$  сетей для каждого агента, случайно выбирать на  $M$  шагов одну
- В AlphaStar[1] строится "лига агентов" с рейтинговой системой

# Итого

- $N \cdot K$  сетей для каждого агента, акторы
- $N$  сетей для каждого агента, критики
- $N$  сетей для предсказания поведения других
- Итого  $N \cdot (K + 2)$  сетей

# Итого

- $N \cdot K$  сетей для каждого агента, акторы
- $N$  сетей для каждого агента, критики
- $N$  сетей для предсказания поведения других
- Итого  $N \cdot (K + 2)$  сетей

# Итого

- $N \cdot K$  сетей для каждого агента, акторы
- $N$  сетей для каждого агента, критики
- $N$  сетей для предсказания поведения других
- Итого  $N \cdot (K + 2)$  сетей

# Итого

- $N \cdot K$  сетей для каждого агента, акторы
- $N$  сетей для каждого агента, критики
- $N$  сетей для предсказания поведения других
- Итого  $N \cdot (K + 2)$  сетей

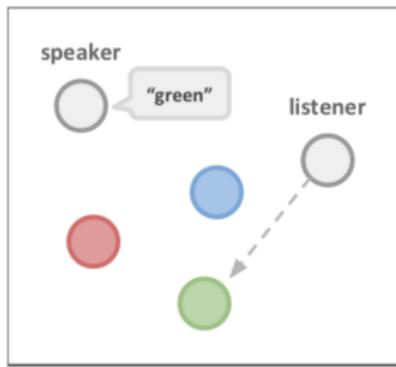
## Итого

- $N \cdot K$  сетей для каждого агента, акторы
- $N$  сетей для каждого агента, критики
- $N$  сетей для предсказания поведения других
- Итого  $N \cdot (K + 2)$  сетей

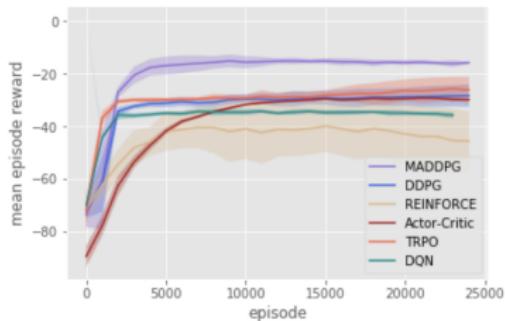


## Cooperative communication

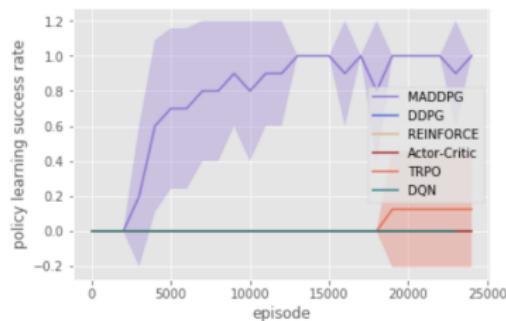
- Два агента: speaker, listener
- Три возможных цели, speaker знает куда идти
- Speaker умеет говорить, listener умеет ходить
- Оба получают выигрыш в конце: чем ближе к цели, тем лучше



# Результаты



(a) Выигрыш разных моделей

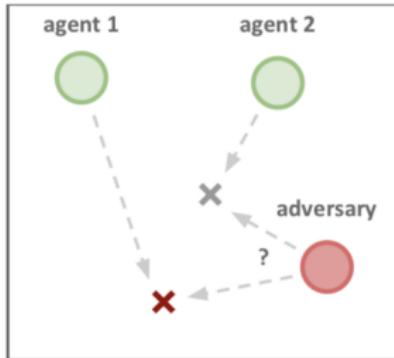


(b) Количество успехов

Рис.: Результаты на Cooperative Communication

# Physical Deception

- $N$  агентов одной команды, один противник
- $K$  возможных целей, только первая команда знает настоящую
- Все  $N + 1$  агентов награждаются за достижение цели



# Результаты

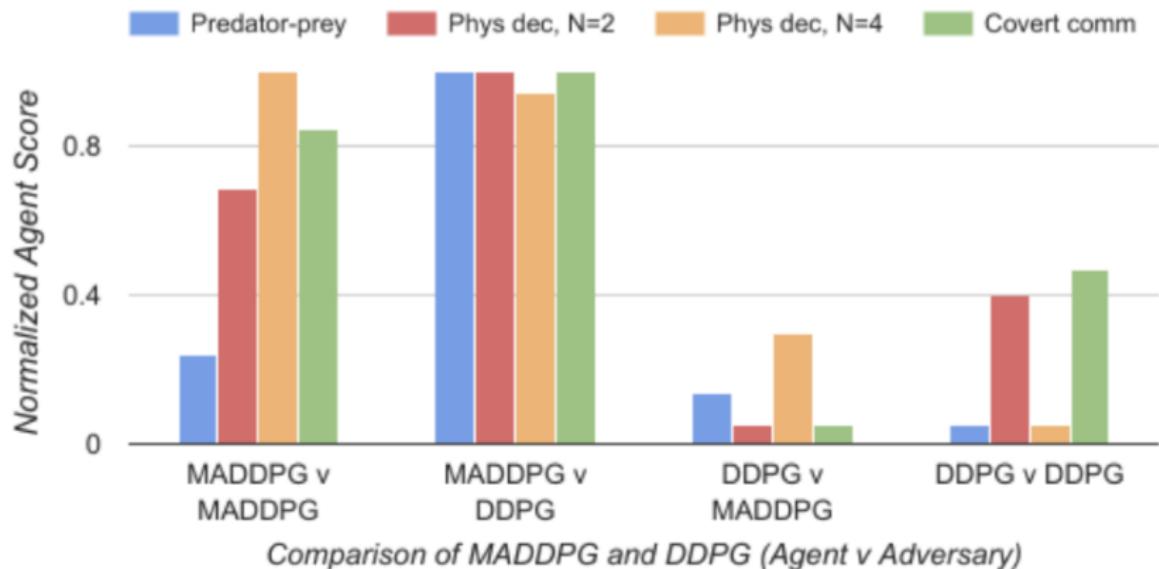


Рис.: Результаты на Physical Deception



*AlphaStar: Mastering the Real-Time Strategy Game StarCraft II.* URL:

<https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>.



*DeepMind AI Reduces Google Data Centre Cooling Bill by 40%.* URL: <https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/>.



*Ryan Lowe и др. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.* 2017. eprint: [arXiv:1706.02275](https://arxiv.org/abs/1706.02275).



Laetitia Matignon, Laurent Jeanpierre и  
Abdel-Illah Mouaddib. *Coordinated Multi-Robot  
Exploration Under Communication Constraints Using  
Decentralized Markov Decision Processes.* 2012. URL:  
[https://www.aaai.org/ocs/index.php/AAAI/AAAI12/  
paper/view/5038/5366](https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5038/5366).



Volodymyr Mnih и др. “Playing Atari With Deep  
Reinforcement Learning”. В: *NIPS Deep Learning  
Workshop*. 2013.



OpenAI. *OpenAI Five*. Сент. 2018. URL:  
<https://blog.openai.com/openai-five/>.

-  David Silver и др. “Deterministic Policy Gradient Algorithms”. B: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML’14. Beijing, China: JMLR.org, 2014, с. I-387—I-395. URL:  
<http://dl.acm.org/citation.cfm?id=3044805.3044850>.
-  David Silver и др. “Mastering the game of Go without human knowledge”. B: *Nature* 550.7676 (2017), с. 354.
-  Richard S. Sutton и Andrew G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262193981.