Обучение с подкреплением в мини играх StarCraft II

Viktor Romanenko

April 2022

Ключевые слова

A2C, StarCraft II, Reinforcement Learning

Содержание

1 Аннотация

Обучение с подкреплением является одной из важнейших составляющих машинного обучения. Данная работа представляет из себя исследование обучения агента в среде мини игр StarCraft II с помощью Advantage Actor Critic(A2C)

2 Введение

Используя обучение с подкреплением с алгоритмом A2C и свёрточные нейронные сети, исследуется возможность обучения мини игр в игре StarCraft II и сравнение с результатами, полученными в подобной работе DeepMind[1]

3 Обучение с подкреплением

Обучение с подкреплением базируется на Марковском процессе принятия решений (MDP[2]) и отличается от других подразделов машинного обучения способом получения данных. В начальный момент времени никакой обучающей выборки не дано, агент выполняет некоторые действия и постепенно получает информацию о среде. В процессе обучения максимизируется функция награды, предоставляемая средой или задающаяся искусственно (модификация MDP). В качестве базовой входной информации подаются состояния (набор данных отражающий текущее положение в среде).

Марковский процесс принятия решений представляется как набор из 4 элементов (S, A, P_a, R_a), где S - набор состояний, A - набор действий, тогда A_s это набор действий доступный в состоянии s, $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ - вероятность того что действие a в состоянии s в момент времени t приведёт в состояние s' в момент времени $t + 1$, $R_a(s, s')$ - награда при переходе из состояния s в s' с помощью действия a.

В процессе обучения агент выучивает «политику» (стратегию), а также находит баланс между исследованием среды и эксплуатацией знаний о среде (чаще баланс задаётся искусственно).

Выбор действий агента можно представить как:

$$\pi : S \times A \to [0, 1]$$

$$\pi(a|s) = P(a_t = a | s_t = s)$$

где π - вероятностная политика (стратегия) выбора действий, S - состояния, A - действия.

4 Описание метода решения

В данном разделе будет описан алгоритм использованный для решения данной задачи.

4.1 A2C

Advantage Actor Critic(A2C) является синхронной версией алгоритма Asynchronous Advantage Actor Critic(A3C[3]) Алгоритм содержит в себе политику $\pi(a_t|s_t,\theta)$ и оценивающую функцию $V(s_t,\theta_v)$. Алгоритм использует одни и те же состояния среды полученный в течение n шагов для обновления политики и оценивающей функции. Политика и оценивающая функция обновляются каждые t_{max} шагов или при достижении терминального состояния. Градиент политики можно представить в виде $\nabla_{\theta'}\log\pi(a_t|s_t,\theta')A(s_t,a_t;\theta,\theta_v) + \eta\nabla_{\theta'}H(\pi(s_t,\theta'))$, где $A(s_t,a_t;\theta,\theta_v)$ оценка функции выгоды вычисляемая как $\sum_{i=0}^{k-1}(\gamma^i r_{t+i}) + \gamma^k V(s_{t+k},\theta_v) - V(s_t,\theta_v)$, где k меняется от шага к шагу и ограничена сверху t_{max}, γ коэффициент дисконтирования, $H(\pi(s_t,\theta'))$ энтропия, вычисляемая как $\sum\limits_a \pi(a|s_t,\theta')\log\pi(a|s_t,\theta')$, а гиперпараметр η регулирует силу энтропии. Градиент оценивающией функции представляется в виде $\delta(A(s_t,a_t;\theta,\theta_v)^2)/\delta\theta_v$ Полный градиент A2C:

$$\nabla_{\theta'}\log\pi(a_t|s_t,\theta')A(s_t,a_t;\theta,\theta_v) + \eta\nabla_{\theta'}H(\pi(s_t,\theta')) + \beta\delta(A(s_t,a_t;\theta,\theta_v)^2)/\delta\theta_v$$

где β гиперпараметр отвечающий за силу градиента оценивающией функции.

4.2 Политика

В связи с тем, что в StarCraft II действия требуют список a содержащий само действие a^0, а также набор аргументов, и тем, что большинство действий требуют пространственные параметры включающие позицию в пикселях на экране, наивное представление политики $\pi_\theta(a|s)$ могло бы потребовать миллионы значений для представления всех взаимосвязей между a, даже для низкого разрешения. Взамен можно представить политику следующим способом:

$$\pi_\theta(a|s) = \prod_{l=0}^{L} \pi_\theta(a^l, s)$$

Благодаря чему можно выбирать действие a^0 и набор аргументов независимо друг от друга. Стоит заметить, что в зависимости от выбранного действия a^0 количество аргументов L может отличаться.

4.3 Агент

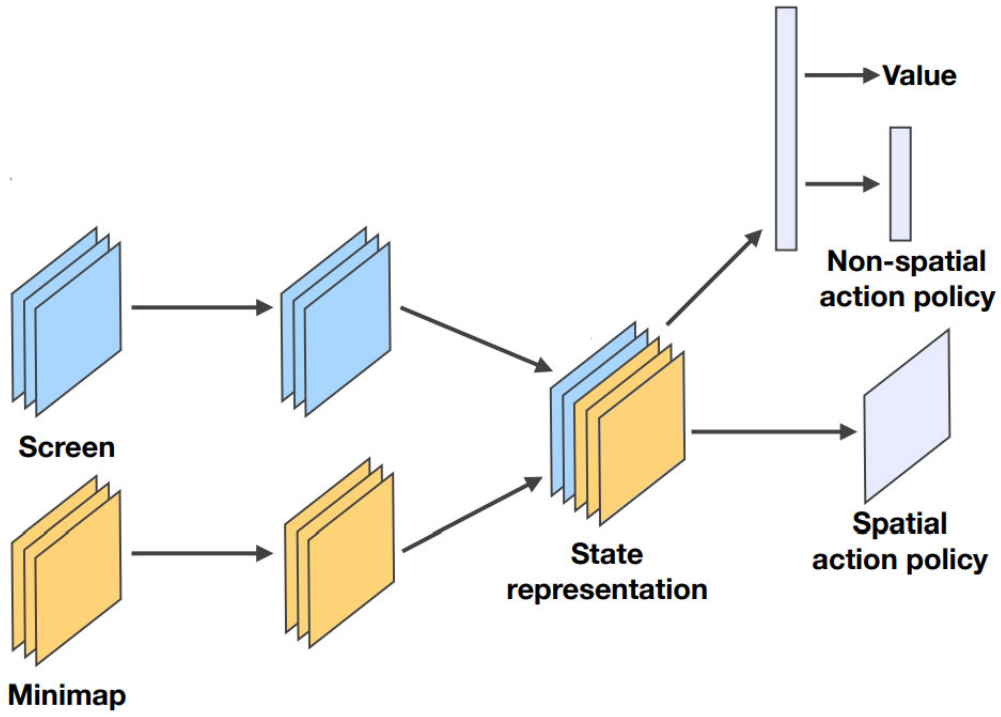

Рис. 4.3.1: Архитектура сети

Все агенты обучались с фиксированными значениями $\beta = 0.7$, $\eta = 10^{-3}$, $\gamma = 0.99$, оптимизатором RMSProp с learning rate $= 7 \times 10^{-7}$, и $t_{max} = 40$

5 Результаты

Обучение происходило на 3 мини играх предоставленных библиотекой pysc2: MoveToBeacon, CollectMineralShards, DefeatRoaches

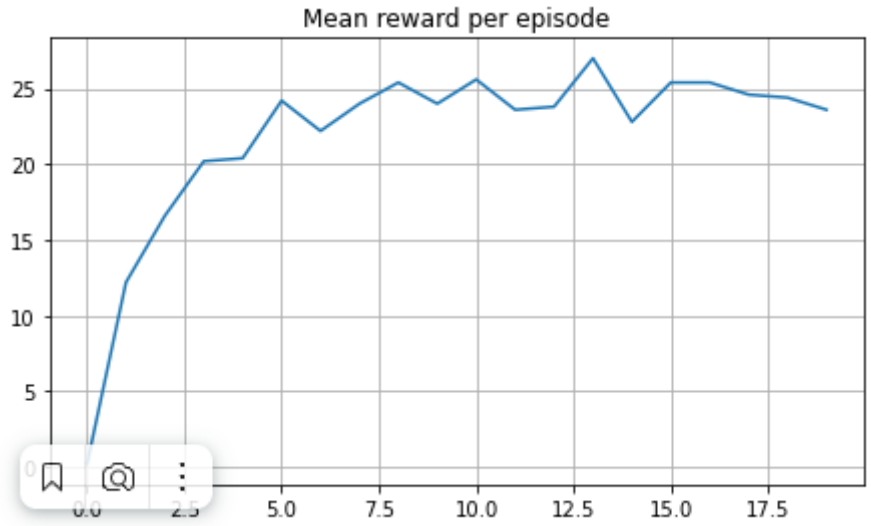

Рис. 5.0.1: Награда мини игры MoveToBeacon

Рис. 5.0.2: Награда мини игры DefeatRoaches

Рис. 5.0.3: Награда мини игры CollectMineralShards

Также было проведено обучение модели на всех 3 мини играх сразу, что, в теории, могло бы позволить лучше выделять ближайших юнитов, чтои требуется во всех 3 мини играх. Награда считалась по игре CollectMineralShards, так как она сложнее чем MoveToBeacon и менее рандомная, чем DefeatRoaches

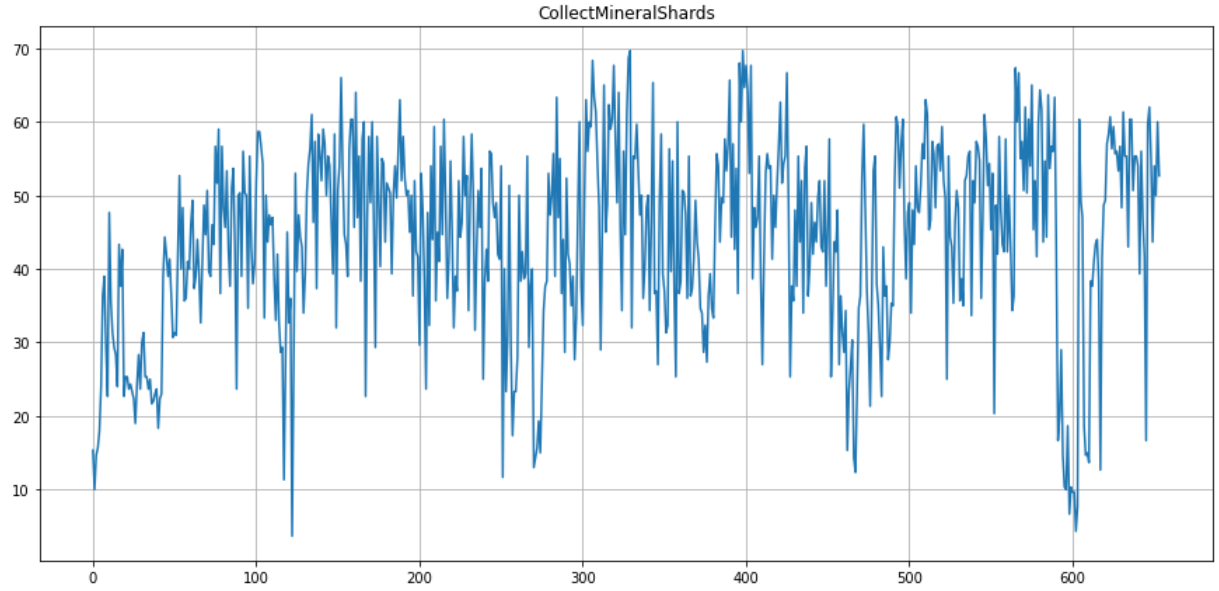

Рис. 5.0.4: Награда мини игры CollectMineralShards в обучении на 3 мини играх сразу

Данный эксперимент не принёс ожидаемых результатов и награда оказалась такой же, как при обычном обучении на меньшем числе шагов.

Итоговые результаты в сравнении с результатами DeepMind:

Agent	MoveToBeacon	CollectMineralShards	DefeatRoaches
Рандомная политика	1	17	1
Игрок любитель DeepMind	26	133	41
DeepMind FullyConv	26	103	100
Результаты	26	64	33

Таблица 5.0.1: Сводная таблица сравнения полученных резултатов с результатами DeepMind

Итоговая награда считалась как средняя награда по 25 играм, округлённая по правилам математического округления.

6 Выводы

Обучение самой лёгкой мини игры проходит быстро, в то время как на других не получается быстро добиться хороших результатов. Итоговый результат получился лучше чем рандомная политика, но хуже чем имплементация алгоритма от DeepMind и хуже игрока любителя.

7 Дальнейшая работа

Проблемы с длительностью обучения, связаны с долгим взаимодействием со средой, следовательно алгоритм A3C с возможностью взаимодействовать с множеством сред, может помочь решить данную проблему. Также сеть не опирается на предыдущие состояния, в то время как, в рамках данной игры, они могут быть полезны, LSTM или буфер фреймов могут помочь решить данную проблему.

Список литературы

[1] Oriol Vinyals, Timo Ewalds, Sergey Bartunov et al.; StarCraft II: A New Challenge for Reinforcement Learning. — 2017. — URL: https://arxiv.org/abs/1708.04782.

[2] Markov decision process. — URL: https://en.wikipedia.org/wiki/Markov$_d$*ecision*$_p$*rocess*.

[3] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza et al.; Asynchronous Methods for Deep Reinforcement Learning // *ICML*. — 2016. — URL: https://arxiv.org/abs/1602.01783v2.