Article

# Grandmaster level in StarCraft II using multi-agent reinforcement learning

https://doi.org/10.1038/s41586-019-1724-z
Received: 30 August 2019
Accepted: 10 October 2019
Published online: 30 October 2019

Oriol Vinyals[1,3]*, Igor Babuschkin[1,3], Wojciech M. Czarnecki[1,3], Michaël Mathieu[1,3], Andrew Dudzik[1,3], Junyoung Chung[1,3], David H. Choi[1,3], Richard Powell[1,3], Timo Ewalds[1,3], Petko Georgiev[1,3], Junhyuk Oh[1,3], Dan Horgan[1,3], Manuel Kroiss[1,3], Ivo Danihelka[1,3], Aja Huang[1,3], Laurent Sifre[1,3], Trevor Cai[1,3], John P. Agapiou[1,3], Max Jaderberg[1], Alexander S. Vezhnevets[1], Rémi Leblond[1], Tobias Pohlen[1], Valentin Dalibard[1], David Budden[1], Yury Sulsky[1], James Molloy[1], Tom L. Paine[1], Caglar Gulcehre[1], Ziyu Wang[1], Tobias Pfaff[1], Yuhuai Wu[1], Roman Ring[1], Dani Yogatama[1], Dario Wünsch[2], Katrina McKinney[1], Oliver Smith[1], Tom Schaul[1], Timothy Lillicrap[1], Koray Kavukcuoglu[1], Demis Hassabis[1], Chris Apps[1,3] & David Silver[1,3]*

## 【强化学习 99】AlphaStar

张楚珩
清华大学 交叉信息院博士在读



给大家带来最新出炉的重磅 paper，在星际争霸 II 上打败人类选手的 AlphaStar。这篇工作在 10 月 30 日作为封面文章发表在 Nature 上。

### 原文传送门

Vinyals, Oriol, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning." Nature (2019): 1-5.

### 特色

之前也有玩星际的 AI，但是它们都简化了游戏或者使用了一些人工设计的子系统。这里提出了一个端到端的训练方法，最后训练出来的 AlphaStar 打败了职业星际玩家，超过了 99.8% 的人类玩家。在训练方法上，使用人类数据和智能体对弈数据，使用了多智能体强化学习方法；特别地，设计了若干策略池（league）来连续地学习策略和反制策略。

从下面的封面也可以看出，该工作代表了目前人工智能领域的最高水平。


## 过程

### 一、面临困难

- 如果采用自博弈（自己和自己玩）的方法来学习策略的话，随着学习的进行，可能会出现循环。例如，学习到的 B 策略打败了之前的 A 策略，接下来学习到的 C 策略打败了 B 策略，最后又重新学习到了 A 策略发现它能打败 C 策略。
- 如果纯使用自博弈来学习，学习到的策略可能不能有效对抗人类策略。
- StarCraft 本身的困难：
  - 动作空间组合数目较多，每一个动作都需要先选择一个对象（比如农民），选择动作的类型，然后可能还需要从地图中选择作用的位置（比如走到某个位置），最后还需要选择什么时候进行下一个动作。
  - 一局游戏需要几万步决策，但是最后只有一个稀疏的奖励；
  - 不完全信息；
  - 对于操作速度（action per minute，APM）有限制，并且也受到网络延迟和计算延时的影响。

### 二、MDP 建模


## 三、神经网络架构

策略网络 $\pi_\theta(a_t|s_t,z)$ ，其中状态包括历史上所有的 observation 和 action，即 $s_t=(o_{1:t},a_{1:t-1})$ 。

- Observation 输入之后经过 self-attention 处理。
- 组合图像和非图像的信息，使用 scatter connection。
- 处理 POMDP，使用 LSTM。
- 组里结构化的组合行动空间，使用 auto-regressive policy 和 recurrent pointer network。

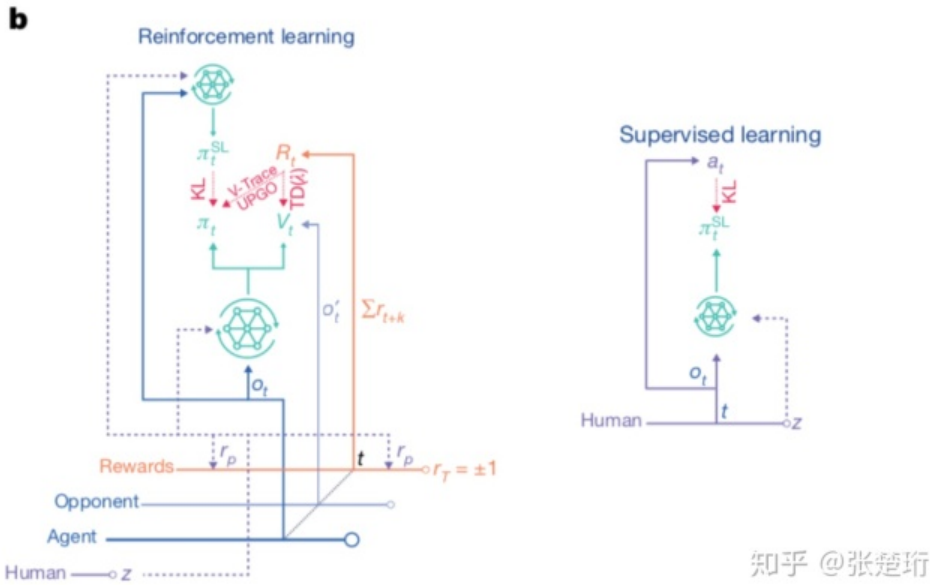
## 四、训练过程

第一步，使用人类的数据来做有监督学习，学习目标就是给定一个状态，预测下一步的动作。

第二步，使用 RL 来最大化胜率，对手的选择比较关键，下面将会重点讲这一部分。其他使用的一揽子技术如下：

- RL 算法使用 A3C，并且使用经验池，用 V-trace 技术来处理 off-policy 数据；
- 针对奖励稀疏的问题，使用 TD(lambda) 结合学习的一个 critic 来减小 variance；
- 使用模仿学习 UPGO，尽量选择能够达到收益较高轨迹的行动。

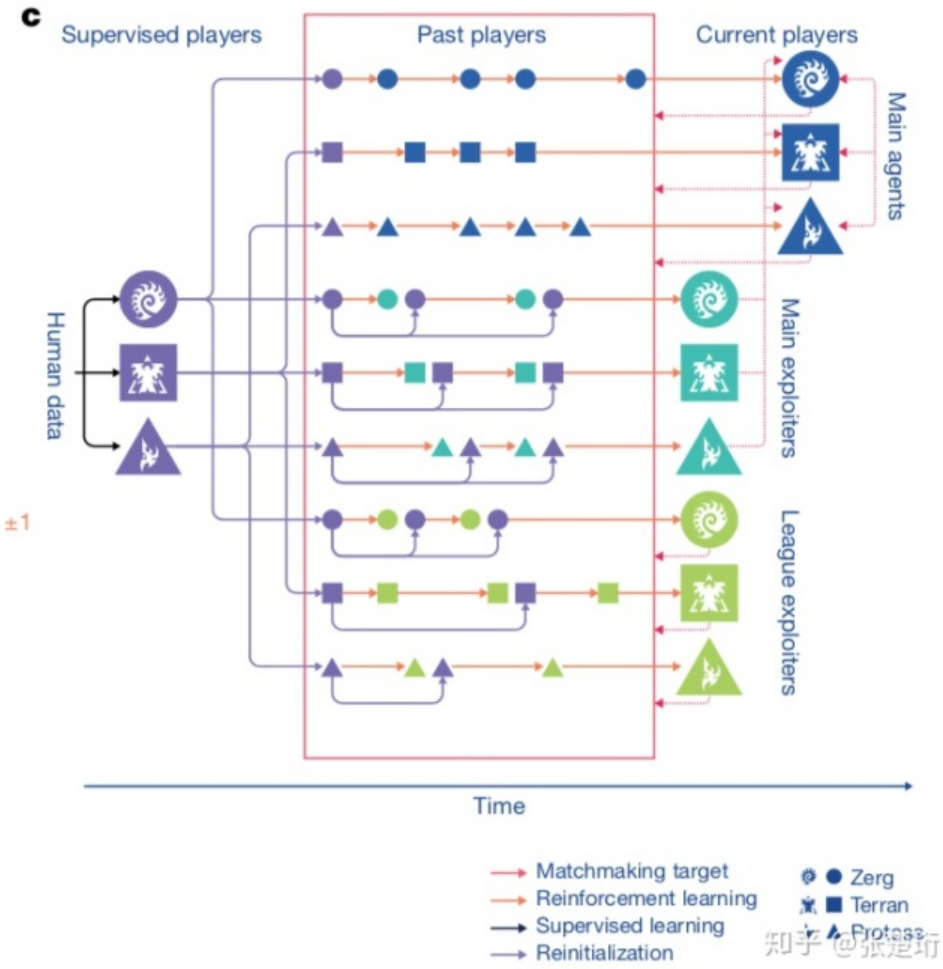
这些技术的示意图如下




最关键的技术在于这里同时训练了三个策略集合：main agents、main exploiters、league exploiters。

- Main agents 使用 prioritized fictitious self-play（PFSP）技术。其中的 fictitious 含义是不仅仅像 AlphaGo 里面那样找较近的得到的 agent 去和它对弈，而是希望找到一个能够对抗历史上智能体某个分布的策略，在两个玩家的零和游戏中该分布最后会趋向纳什均衡。其中 prioritized 的含义是以更高的概率去对抗那些历史上对其胜率低的智能体。Main agents 的对手从所有的三个策略集合中选择（包含历史上的策略）。
- Main exploiters 的对手只是当前的 main agents，主要目的是找到当前策略集合的弱点。
- League exploiters 也使用 PFSP 方法，对手为 main agents 的历史，目标是发现系统性弱点。
- Main exploiters 和 league exploiters 都会隔一段时间重置为有监督学习得到的智能体，避免三个策略集合相对于人类的策略『走偏了』以增加对抗人类策略的稳定性。
- 每个种族使用一个 main agents 集合、一个 main exploiters 集合和两个 league exploiters 集合。

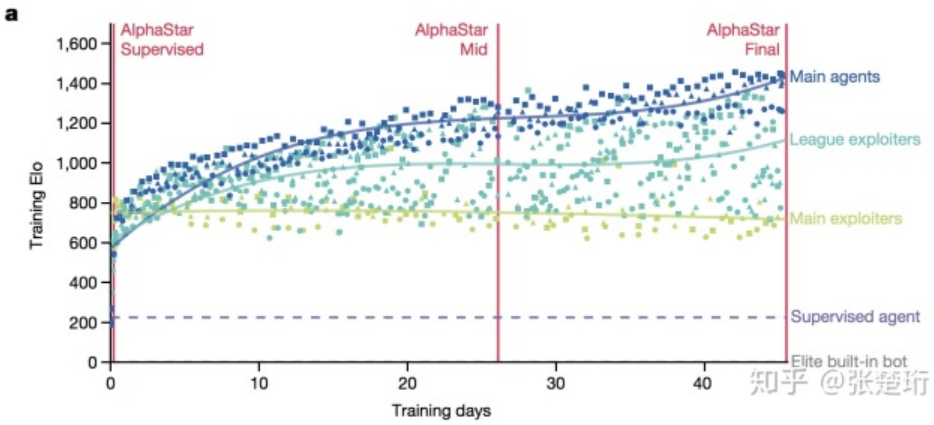
该方法概括为下图。


c
Supervised players
Past players
Current players
Main agents
Human data
Main exploiters
League exploiters
±1
Time
Matchmaking target
Reinforcement learning
Supervised learning
Reinitialization
Zerg
Terran
Protoss
知乎 @张楚珩

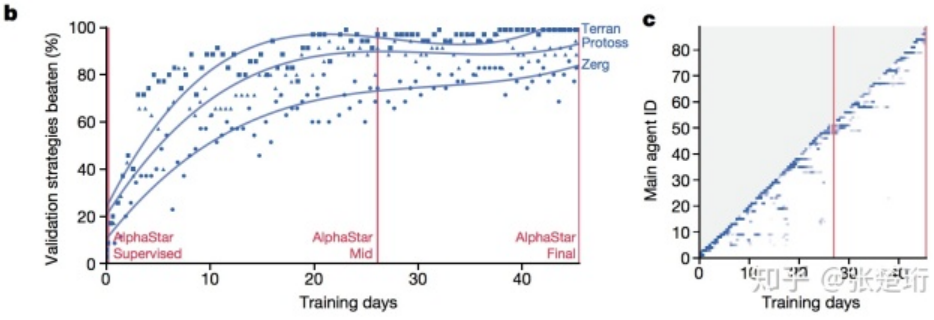
五、实验结果

实验在 32 块 TPU 上跑了 44 天。文章测试了三个版本：有监督学习之后的版本、学习了一半的版本 AlphaStar Mid（学习了 27 天）、学习完成之后的版本 AlphaStar Final（学习了 44 天）。测试方式为匿名地在 Battle.net 上随机玩，AlphaStar Mid 每个种族各玩 10 局（一共 30 局），AlphaStar Final 每个种族玩 20 局（一共 60 局）。最后得分超过了 99.8% 的人类玩家。

随着学习的进行，智能体的 Elo score 变化如图所示。该分数是通过和策略集合历史上的策略做循环赛得到的分数。反映了训练使得策略在慢慢变好。

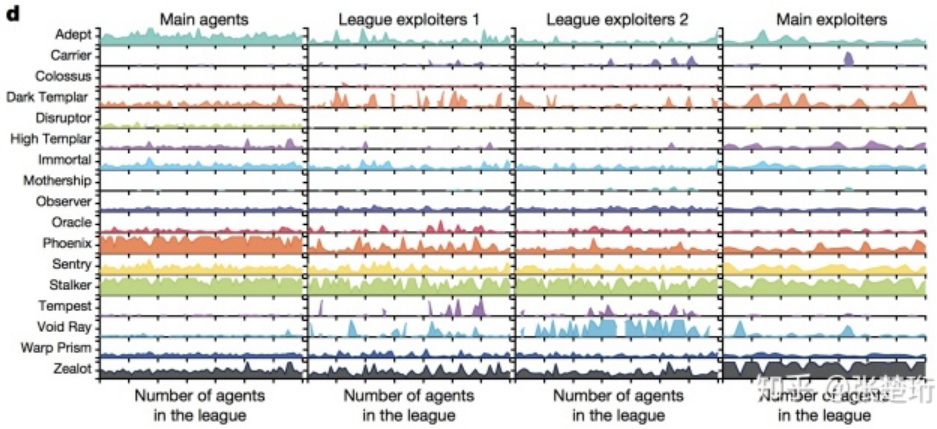



文章还弄了一个遵循特殊策略的 held-out 策略集合，main agents 和该集合中策略比赛的结果如图 b 所示。这反映了策略学习的绝对效果。图 c 反映了不同 agent 对应的不同纳什均衡分布，这个图有点不好理解：比如 ID=40 的 agent 在大概第 25 天被造出来之后，在大概之后的五天里面都还会经常能够战胜后面新学到的策略，但是大概在 30 天之后，新产生的所有策略都完全战胜 ID=40 的策略。这说明了整个策略集合的学习过程中，新的策略能够完全战胜以前的所有策略，而没有出现前面提到的训练过程中的循环（不稳定）的情况。



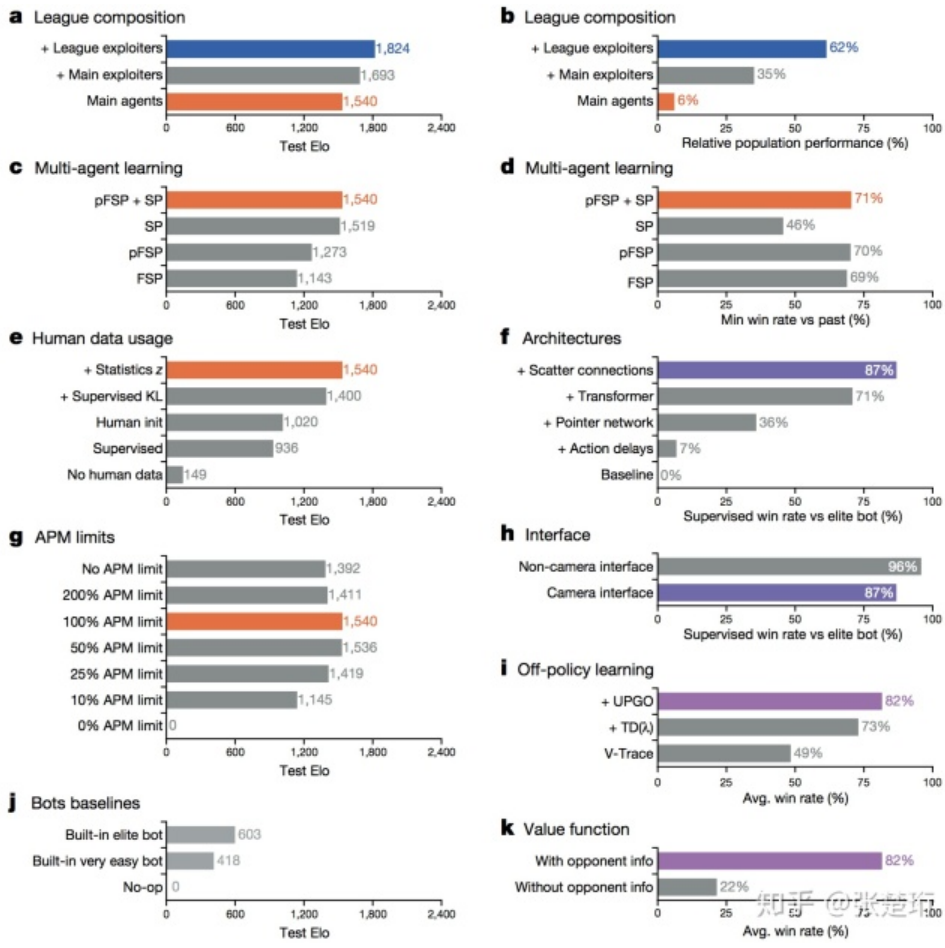

文章为了说明学习到的策略集合中的策略具有比较高的多样性，画出来各个策略使用不同游戏元素

的分布。同时可以注意到，exploiter 的三个集合中，不同策略的多样性更高。




## 六、各个元素的作用

文章还做了 ablation study 来定量分析各中不同设计的作用，如下图所示。



可以看到 League 的作用还是比较明显的（图 b）；相比于 self-play（SP），fictitious 的作用还是很明显的（图 d）；另外，神经网络结构的用处也是比较明显的（图 f）

---

**神经网络的输入中有一个 z，文章中称它是从人类玩家数据中得到的一个统计量，有监督学习和强化学习中都会 condition on z。它究竟是什么？**

z 是从不同人类玩家中抽取出来的信息，包括建造顺序等信息，比如单位、建筑、升级的顺序等。在强化学习和有监督学习的时候都会 condition on z，但是 10% 的有监督学习中会把它设置为 0。

---

为了保证学习到的策略不要太走偏，全程都在最小化和有监督版本智能体的 KL；同时，condition on z，也希望实际建造顺序和 z 对应建造顺序的 Hamming distance 相差不要太大，具体的做法是把它做成 pseudo-reward。

TD(lambda) 和 V-trace 本专栏之前都讲过了。这里说一下 upgoing policy update（UPGO）。在该方法中，策略梯度变成如下形式。

$$\rho_t(G_t^U - V_\theta(s_t, z))\nabla_\theta \log \pi_\theta(a_t \mid s_t, z)$$

where

$$G_t^U = \begin{cases} r_t + G_{t+1}^U & \text{if } Q(s_{t+1}, a_{t+1}, z) \geq V_\theta(s_{t+1}, z) \\ r_t + V_\theta(s_{t+1}, z) & \text{otherwise} \end{cases}$$

考虑一个奖励稀疏的情形，比如获胜之后只在最后一步有一个 +1 的奖励。遵循 UPGO，如果有一个好于平均的轨迹，那么最后的这个奖励将会容易一直传播到每一步上，而不会被轻易衰减掉。这实际上和 self-immitation 的想法比较类似。

编辑于 2019-11-20

星际争霸 2　DeepMind　强化学习 (Reinforcement Learning)



**文章被以下专栏收录**

**强化学习前沿**
读呀读paper

进入专栏