Go-Explore: a New Approach for Hard-Exploration Problems

Adrien Ecoffet Joost Huizinga Joel Lehman Kenneth O. Stanley* Jeff Clune*
Uber AI Labs
San Francisco, CA 94103
adrienle,jhuizinga,joel.lehman,kstanley,jeffclune@uber.com
*Co-senior authors

【强化学习 41】Go-Explore

张楚珩
清华大学 交叉信息院博士在读

Go-Explore是uber团队开发的算法，直观的意思是走到最好的状态（Go），然后从这个状态开始探索（Explore）。

原文传送门

Ecoffet, Adrien, et al. "Go-Explore: a New Approach for Hard-Exploration Problems." arXiv preprint arXiv:1901.10995 (2019).

特色

Go-Explore旨在解决探索困难（hard-exploration）的问题，这类问题通常奖励稀疏（sparse）并且会有误导性的奖励（deceptive）。这篇文章通过一系列算法设计，使得它在 Montezuma's Revenge 和 Pitfall 游戏上的表现相比于之前的算法有了质的飞跃。

过程

1. 任务困难之处

文章中讲到的探索困难的问题主要包括两个难点。

- 第一个难点在于其任务给的奖励很稀疏，即需要作出特定的一连串动作才可能得到一个非零的奖励，对于每一步都随机探索的很多算法来说，很可能在整个探索过程中都遇不到一个非零的奖励。比如在 Montezuma's Revenge 里面需要走很多步去获取钥匙或者进入新的房间才会有一个奖励。
- 第二个难点在于任务给的奖励具有误导性。比如在 Pitfall 这个游戏里面不仅奖励很稀疏，而且很多小动作会导致负奖励，这会使得智能体在学习到如何获取奖励之前，反而由于这些负奖励而原地不动、停止探索。

在此之前的算法为了更加有效地探索，通常使用使用 intrinsic motivation （IM）或者 intrinsic reward（IR）技术，除了尽量获取更多的奖励之外，也尽可能激励智能体去探索没有遇到过的状态空间。这方面的算法可以参考本专栏的其他文章（比如 RND、VIME、ICM 等）。

但是 IM 类方法具有 detachment 和 derailment 的缺点。前者指的是，算法虽然鼓励智能体去探索未

知的状态空间，但是当前状态到被探索状态空间的边界之间隔着很多被探索过的状态，算法并不能激励智能体越过这些 IR 很小的状态走到边界上再去探索。因此，当前状态到未被探索过的状态之间是“分离”的，这限制了有效的探索。那么自然有一个想法就是让智能体先走到已被探索过的状态的边界上，然后再去探索新的状态，但是现有的算法会在走到边界的半路上“边走边探索”，最后偏离的轨迹，无法到达边界，这就是 derailment 所描述的问题。

2. 算法大致流程

如下图所示，算法分为两个部分。其中第一部分（Phase 1）的最终目的是通过在环境中探索生成一系列高奖励的轨迹；第二部分（Phase 2）使用第一部分生成的轨迹做 imitation learning 并获得最终的策略。

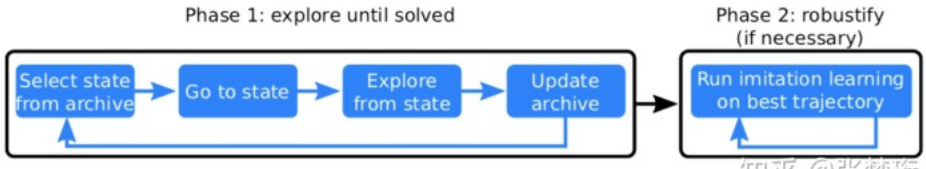


Figure 2: **A high-level overview of the Go-Explore algorithm.**

第一部分限定在确定性（deterministic）并且可以重启（resettable）的环境中进行，这样的好处是记录下行动序列就可以确定性地到达之前到达的状态上。第一部分的主要思想是维护一个存档（archive），保存着从起始状态到达各个已经探索到状态的路径。每一轮都选择一个已经达到过的状态，然后利用存档中的信息走到这个状态上，接着从这个状态出发做探索，如果探索到了新的状态或者有用的路径，就记录下来加入存档中。注意到这个部分纯粹是做探索并且记录状态和行动序列，这里面并没有使用神经网络。（该部分具体细节见后）

第二部分使用第一部分生成的轨迹做训练。这不仅仅是因为第一部分其实压根没有学习到概括性的策略，而且也是因为第一部分实在确定性的环境中探索的，其策略没法适应真实的随机性环境。文中使用了一种 learning from demostration 的方法 Backward Algorithm。大致上来说，拿到一条高奖励的轨迹之后，先把智能体放在离轨迹末端较近的位置，让智能体能够学习到如何从这个离目标很近状态走到目标，然后再逐步把智能体放在轨迹上离目标更远一些的地方。通过这种方式智能体就能逐步学习到如何从初始位置走到目标位置。其学习的算法还是使用的通用的强化学习算法，比如 PPO 等。这个过程也可以看做是一个课程学习（curriculum learning），使用第一部分生成的反向轨迹来作为由易到难设定的课程。

3. 算法细节

【状态的表示】

文章中算法的输入是游戏视频，这是一个高维度的状态表示，如果把每个不同的高维度状态表示都放到存档里面显然是不可能的，因为不同的状态太多了。文章通过转化为灰度图像和下采样的方法把这些高维度的状态都转化为了状态单元（cell），如下图所示。算法第一部分的所说的各种状态都是针对这个状态单元（cell）而言的。

Figure 3: **Example cell representation without domain knowledge, which is simply to downsample each game frame.** The full observable state, a color image, is converted to grayscale and downscaled to an 11×8 image with 8 possible pixel intensities.

同时文章还测试了加上与任务特定知识有关的信息（domain knowledge）。他们在上面提取的状态的基础上还使用编写的程序从图中提取一些与游戏相关的信息作为状态特征，比如智能体的x, y坐标，当前房间的位置，当前处于的关卡数等。

【从存档中选择要前往的状态】

显然我们更希望找出那些离未探索区域更近的状态，因为从这些状态出发能够更容易找到未探索过的状态。文章中使用了一些启发式的评价指标，并且以更大的概率选择那些具有更高得分的状态。文章中使用的评价指标主要包括三类：1）该状态是否更少被访问过，并且之前对它的访问都有较好的结果；2）该状态附近是否有更多未被探索过的状态；3）该状态是否处于更高的关卡中。

【从初始状态出发前往探索的起点】

从存档中选择了要前往的状态之后，就需要开始行动前往这个状态。由于在算法的第一部分中环境都是确定性并且可以重启的，因此这一部分就变得很容易了，只需要把存档中的行动序列读出来然后按照原来的行动序列行动就可以到达指定的状态了。

注意到为了更加充分地利用存档中储存的信息，轨迹中途经的各个状态也会被记录下来，如果要前往这些状态也可以复用前半部分的轨迹。

【从这个状态出发探索】

这个探索的过程相对来说比较直接，就是做 $k=100$ 步的随机行动采样，并且有95%的概率重复上一步的动作，然后看看能够到达什么状态。

【更新存档】

存档会记录以下信息不仅会记录到达某状态的轨迹，而且还会记录下所获得的奖励和轨迹长度。

当遇到以下两种情况之一的时候会更新存档。一种是遇到了之前没有遇到过的状态；另一种遇到了已经存在于存档中的状态，但是其走到该状态的路径更优。这里说的更优指的是获得了更多的奖励或者获得了相同的奖励但是轨迹长度更短。这样原本的状态对应的轨迹就会被更新成新的轨迹。

假如，之前从 A 到 B 的轨迹被更新了，同时存档中还存在一条从 A 经 B 到 C 的轨迹，后面这条轨迹不会被更新，因为这里讲的状态 B 是很多可能的状态集合起来的状态单元，前面一条轨迹中的 B 可能和后者中的 B 不一样。

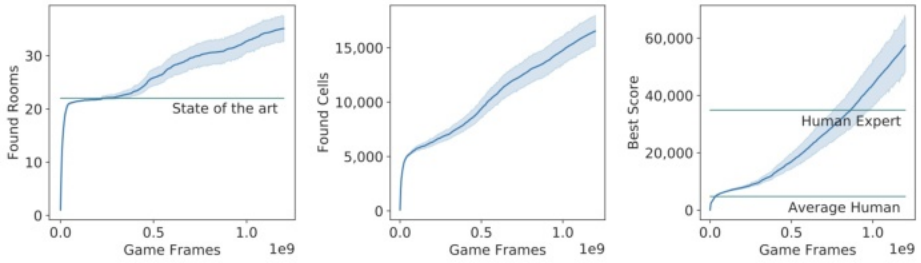
4. 算法核心思想

文章总结了 Go-Explore 算法成功的三大核心思想，文章称这三大核心思想是该算法成功的关键。

- 记录下之前访问过的状态；
- 优先考虑返回更有可能探索到新状态的状态，返回的过程中采用确定性环境，这样一定能够返回到特定的状态；
- 现在确定性的环境中探索出可能比较脆弱的高奖励轨迹，然后再利用该轨迹学习到鲁棒的策略。

实验结果

这里只放 Montezuma's Revenge 上的结果，Pitfall 上的结果类似。不使用 domain knowledge 的结果就超越了人类专家的水平。



(a) Number of rooms found (b) Number of cells found (c) Maximum score in archive

Figure 4: **Performance of the exploration phase of Go-Explore with downscaled frames on Montezuma's Revenge.** Lines indicating human and the algorithmic state of the art are for comparison, but recall that the Go-Explore scores in this plot are on a deterministic version of the game (unlike the post-Phase 2 scores presented in this section).

不使用 domain knowledge就在 Montezuma's Revenge 游戏上超过人类专家的水平

下图展示了 Go-Explore 算法和其他算法得分的比较，可以看到该算法相比于其他算法也有了很大的提升。

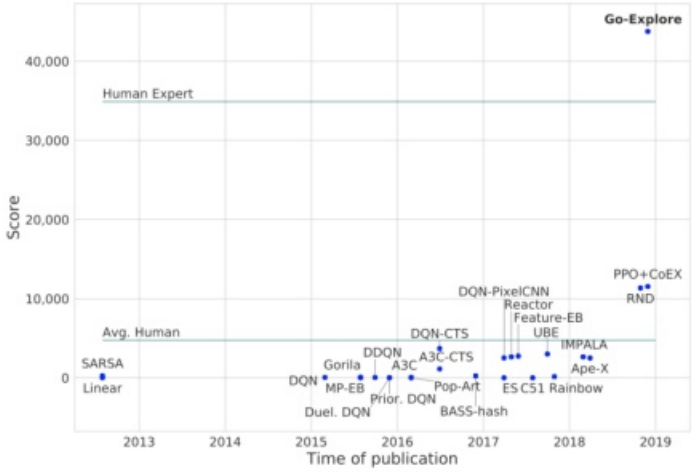
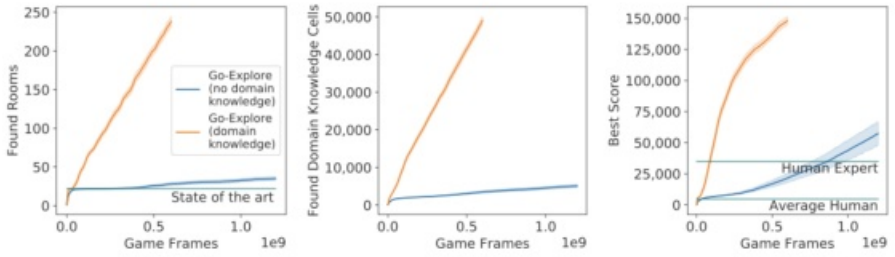


Figure 6: **History of progress on Montezuma's Revenge vs. the version of Go-Explore that does not harness domain knowledge.** Go-Explore significantly improves on the prior state of the art. These data are presented in tabular form in Appendix A.9.

使用了专家知识之后，效果又提升了一大截，通过这个也说明如果能够使用更强的 representation 将给算法效果带来更大的提升。



(a) Number of rooms found (b) Number of cells found (c) Maximum score in archive

Figure 7: **Performance on Montezuma's Revenge of Phase 1 of Go-Explore with and without domain knowledge.** The algorithm finds more rooms, cells, and higher scores with the easily provided domain knowledge, and does so with a better sample complexity. For (b), we plot the number of cells found in the no-domain-knowledge runs according to the more intelligent cell representation from the domain-knowledge run to allow for an equal comparison.

评述

该工作在实验效果上取得了巨大的成就，不过目前看来有以下的不足之处。

- 算法的第一部分把高维度的状态都做了离散化，这是为了确切地区分某两个状态是相同的还是不同的，并且把不同的状态都存在存档里面。这样的做法不方便 scale up，可能会限制该算法解决更困难的问题。另外，文章中把游戏图像下采样的方法虽然说是没有用到游戏的 domain knowledge，但是这其实还是利用了一个先验，即这个游戏都是“格子”类的游戏，通过下采样刚好反映了各个格子上的状态。这样的方法能不能用于更为困难的游戏值得商榷。
- 算法及其依赖于确定性的训练环境，当无法获取一个确定性的训练环境的时候算法将失效（这一点文章也提到了）。当然作者也说了，这种情况下可以再学一个 goal-conditioned policy，让智能体返回特定的状态，不过感觉这么操作就有点复杂了。下一步可能需要研究如何简化这个过程，并且保证相当的返回特定状态的概率。
- 同时，个人感觉，当环境极其复杂的时候，单一确定性环境中训练得到的轨迹们可能无法有效表征对应随机环境的特征，这种情况下第二部分算法能否通过这些轨迹有效地学习值得进一步实验观察。
- 本文的思路可能更适合文章实验的这两款以探索为目的的游戏，对其他游戏不一定能试用。毕竟文章中也提到了，Go-Explore 解决的是 hard-exploration。目前还有一类比较困难的游戏，比如即时战略类游戏，其难点可能在于 temporal abstract。

不过不管怎样，文章中提出的思路非常有价值，即需要直接走到最前沿的某个状态，然后再探索，它确实能有效解决目前强化学习探索上的痛点。这个思路也很符合人类平时探索的思维模式。

发布于 2019-03-02

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