Replicator dynamics and reinforcement learning

陈宇

2023 年 1 月 14 日

目录

摘要

这个 note 从解决多智能体的问题出发，介绍了两种流派：基于遗传算法的动力学和现代流行的强化学习之间的一些联系，以及回顾了一下为啥这两种途径可以殊途同归。

1 简介

鉴于现在对于 decision making 这个领域，各个方向都有自己的一套数学建模的语言，我们这边则不再去严格地定义整个过程，也不去列一个表格去链接这些术语之间的差别，而只是用一套简单但是不那么严谨，不过足够说明问题的符号来记录。对于一个序列决策过程，假设我们有 N 个决策者，每个决策者可以依据某种策略 $\pi(a|s)$ 基于目前所观测到的信息 s, 采取一些 action, 随之决策者们会各自获得一些奖励 r。

2 博弈论简介

我们这边用几个简单的例子，介绍一下博弈论以及其中比较出名的 Nash Equilibrium(NE)。所谓的纳什均衡，我们可以简单地理解为当决策者的策略集合达到如下一种状态：对于游戏中任何一个玩家来说，他都没有办法修改其自己的决策，来提高自己所能获得的奖励。但是我们需要知道的是，

- 纳什均衡真的只是达到了一种“势均力敌”的状态，并不代表从上帝视角去看，每个玩家都在做最正确的决定。
- 纳什均衡不一定是唯一的。

下面我们简单介绍两个例子猎鹿博弈和囚徒困境。

2.1 猎鹿博弈

假设有两个猎人 A 和 B, 他们要一起去打猎，如果他们两个合作一起击杀鹿，则每个人可以获得 5 的奖励；如果他们两个人合作一起击杀兔子，则每个人可以获得 3 的奖励，但是如果 A 选择击杀鹿，B 选择击杀兔子，则只会有 B 获得 3 的奖励，反之依然。因此，我们可以用下面这个表格来描述整个游戏的关系。

	鹿	兔
鹿	(4,4)	(1,3)
兔	(3,1)	(3,3)

我们可以简单针对这种情况算一下，假设 A 认为 B 会以 p 的概率选择鹿，$1-p$ 的概率选择兔子。那么这时候 A 该怎么优化自己的决策策略 $(q, 1-q)$ 呢？可以简单计算到 A 以 $(q, 1-q)$ 能获得的收益为，

$$r = 4pq + q(1-p) + 3(1-q) = (3p-2)q + 3$$

因此我们可以看到三种情况 $p > \frac{2}{3}, p = \frac{2}{3}, p < \frac{2}{3}$。三种情况各自优化，我们可以看出来对于这样的收益矩阵，存在三个纳什均衡点：

- 两个人同时选择鹿
- 两个人同时选择兔子
- 两个人以 $\frac{2}{3}$ 的概率选择鹿，$\frac{1}{3}$ 的概率选择兔子

2.2 囚徒困境

囚徒困境和上面那个例子比较相似，只是囚徒困境将两个人互相合作和欺骗之间的 gap 变得更加的大，例如

	合作	欺骗
合作	(3,3)	(0,5)
欺骗	(5,0)	(1,1)

对于这种情况，我们也可以像上面做一些简单的计算，当然也可以直接从收益矩阵看出来，对于 A 来说无论 B 采取什么策略，A 选择欺骗的收益永远高于合作，因此在囚徒困境里面，纳什均衡则是每个人选择欺骗，两人分别获得收益 $(1, 1)$。然而，从上帝视角看，这样的纳什均衡明显不是最优的，两个人如果能够坦诚相待，相互合作，那么会获得更高的收益。

3 Replicator Dynamics

Replicator Dynamics 则是从遗传算法或者生物进化的角度来看待序列决策问题。每个可供赋值的可以是一个 action，也可以是一种 general 的策略。我们这边则直接考虑稍微一般一点的情况，每个可供复制的个体代表了一种复杂的策略。一开始我们考虑简单有限的离散情况。假设我们有 n 个策略，当前整个种群中个体数目为 N，使用第 i 种策略的个体所占总体的比例为 x_i，我们定义向量 $x = (x_1, \cdots, x_n), \sum_{i=1}^{N} x_i = 1$ 为描述整个种群状

态的状态向量。如果现在我们有两个种群 x 和 y，那么我们用 $U(x,y)$ 表示在对方使用混合策略 y 时，我方使用混合策略 x 所能获得的收益。

有了这些记号，我们可以定义 ESS(Evolutionarily stable state),

Definition 3.1 *我们称状态 x 为 ESS，如果对于任意的状态 $y \neq x$, 在邻域内构造一个小偏离的状态 $\tilde{x} = (1-\epsilon)x + \epsilon y$, 其中 ϵ 是一个足够小的数，满足下面的条件，*

$$U(y, \tilde{x}) < U(x, \tilde{x}) \tag{1}$$

从这个定义出发我们可以看到

Property 3.1 *ESS 是一个纳什均衡*

建立在前面这些假设条件下，我们可以使用一个简单的线性模型来模拟整个动力学过程。通俗一点来说，如果当前我们使用单一策略 e_i 对抗策略 x, 可以获得收益 $U(e_i, x)$, 我们可以知道 e_i 对于 x 的超额收益则是,

$$U(e_i, x) - \bar{U}(x), \bar{U}(x) = x^T U x \tag{2}$$

在这种情况，直觉上的线性更新则是，如果超额收益越大，我们给予更高的比例，因此动力学方程为，

$$\frac{dx_i}{dt} = \alpha x_i (U(e_i, x) - \bar{U}(x)) \tag{3}$$

这边 α 可以是一个归一化因子, 也可以某个认为设置的超参数等等。现在我们拥有了一个动力学方程, 那么在数学上我们也会使用一些手段去判断对于一个动力学方程来说，整个动力学过程是不是稳定，以及存不存在一个稳态解等等。那么我们可以猜想一下，这个动力学方程对应的一些稳定性和之前提到的 ESS 中的 stability 是不是会有某种联系呢? 为了方便分析, 我们线性化这个动力学系统或者认为收益函数是一个线性函数，即 $U(x,y) = x^T U y$, 这边的 U 是一个 $n \times n$ 的矩阵。因此如果有两种策略对抗的话，两个策略的动力学方程可以如下:

$$\frac{dx_i}{dt} = \alpha x_i (e_i^T U_A y - x^T U_A y), \frac{dy_i}{dt} = \alpha y_i (x^T U_B e_i - x^T U_B y) \tag{4}$$

这边我们代码画出了囚徒困境和猎鹿的动力学过程的相图，

可以看出来对于囚徒困境拥有一个平衡点 $(0,0)$, 猎鹿博弈拥有三个平衡点: $(0,0), (1,1), (\frac{2}{3}, \frac{2}{3})$ 其中最后一个平衡点并不是很稳定，稍有一些微扰，则会转而收敛其余两个点。这边得出来的结论，和纯用博弈论的结果是一样的。

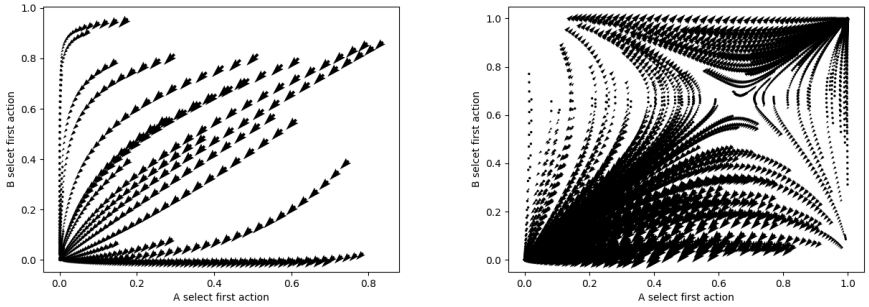

图 1: a. 左边是囚徒困境的相图；右边是猎鹿博弈相图, b. 横坐标为选手 A 选择第一个动作的概率，纵坐标为选手 B 选择第一个动作的概率。

3.1 Regular ESS

上面的分析和例子，让我们对 ESS 有了一个直观的了解，下面从数学的角度，仔细分析 ESS 到底在描述怎样一件事情。我们将 Eq.(1) 在 x 附近展开看看，

$$U(y,\tilde{x}) - U(x,\tilde{x}) = U(y,x) + \epsilon\nabla_2 U|^T_{(y,x)}(y-x) - U(x,x) - \epsilon\nabla_2 U|^T_{(x,x)}(y-x) + o(\epsilon) \tag{5}$$

$$= U(y,x) - U(x,x) + \epsilon(y-x)^T(\nabla\nabla^T)U|_{(x,x)}(y-x) + o(\epsilon) \tag{6}$$

这边 $\nabla = (\partial_1, \partial_2, \cdots)^T$。从上面的表达试我们可以看到 ESS 分成两种情况。

- $U(y,x) < U(x,x)$
- $U(y,x) = U(x,x), (y-x)^T\nabla\nabla^T U(y-x) < 0$

为了更好地表述上面两种情况代表着什么意思，我们引入下面两个定义，

Definition 3.2 ((平衡态)Equilibrium State) 我们称一个策略 x 处于一个平衡态，如果任何纯策略 $e_i, \forall i,$ 对抗他的收益都是一样的，即 $U(e_i,x) = U(e_j,x), \forall i \neq j$。

Definition 3.3 (Support) 对于任意的策略 x, 我们定义其 *Support* 为 $supp(x) = \{i : x_i \neq 0\}$

关于 support 的定义，直觉上可以类比线性空间中的 support。

使用这两个定义，我们可以一个 ESS 满足下面的条件：

Property 3.2 (ESS 条件) *ESS 需要满足的条件：*

- $U(e_i|x) < U(x,x), \forall e_i \notin supp(x)$
- $U(e_i|x) = U(x,x), \forall e_i \in supp(x)$
- $(y-x)^T \nabla \nabla^T U(y-x) < 0, \forall supp(y) \in supp(x)$

上面的分析，都是基于一个我们没有说明的隐藏假设，U 函数拥有良好的分析性质，因此，学术界也常常会将满足上面三个条件的策略称之为 *regular ESS*。

Definition 3.4 *一个平衡态 x 称之为 regular ESS 如果，*

- $U(e_i, x) < U(x,x), \forall e_i \notin supp(x)$
- $z^T \nabla^T \nabla U z < 0, supp(z) \in supp(x), z \neq 0, \sum_i z_i = 0$

显然，我们可以发现，*regular ESS* 是一个 *ESS*。反过来，从严格的数学意义上是不成立的。但是，现实应用的过程我们几乎可以都可以在一个不 regular 的 U 函数的附近找到一个 regular 的函数使之很接近，代替原生的效用函数。之所以强调函数的正则性，是因为函数的正则性，会给数学上的证明带来很大的便利性。比如下面的著名定理：[3]

Theorem 3.1 *如果 x 是一个* regular ESS*, 那么 x 则会是动力学过程 Eq.(3) 的一个严格稳定的平衡点。*

这个定理给了我们上面想要讨论的 ESS 和动力学过程的关系一个终极的结论。

3.2 Reinforcement learning

3.2.1 Cross Learning

Cross learning 是最早提出来的无状态 (stateless) 的强化学习算法之一 [1]，其核心逻辑的非常的简单，如果采取了某个动作，获得了正的收益，那

么我们就将提高该动作的概率，因此, 当决策采取动作 j 可以简化表达式：

$$\pi(i) \to \pi(i) + \theta r(1 - \pi(i)), i = j \tag{7}$$

$$\pi(i) \to \pi(i) - \theta r \pi(i), \forall i \neq j. \tag{8}$$

这边的 θ 可以理解为一个归一化的因子, 为了保证整个 reward 是被归一化，这样才能保证算法的收敛。容易证明的是 Eq.(7,8) 在连续的极限下等价于 Replicator Daynamics Eq.(3), 数学证明见附录 (A)

3.2.2 Boltzmann Q Learning

强化学习中的 Q learning 是 value based 的算法之一，其核心只依赖于价值函数 Q 的更新，他的策略也是因 Q 而生，当然他也衍生了出很多其他有效的算法。这边只是分析最原始的 Q learning 和 Replicator Dynamics 之间的关系，即采取的更新方式如下：

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha(r + \gamma \max_{a'} Q_t(s',a') - Q_t(s,a)) \tag{9}$$

文章 [4] 中证明了该更新方式的 Q learning 有一个与之对应的 Replicator Dynamics，并且其动力学方程有如下的形式：

$$\frac{\frac{dx_i}{dt}}{x_i} = \alpha\tau \left(r_i - \sum_j x_j r_j \right) + \alpha \sum_j x_j \ln \frac{x_j}{x_i} \tag{10}$$

我们证明使用了更加简洁一点的证明思路如附录 *(B)* 所示，有兴趣的可以参考。同时，按照我的理解，这个结果严格意义上只对无状态的游戏是成立，但是原文并没有特别强调这一点，不知道是不是我理解有问题。

这边我们可以看到 Eq.(10) 的第一部分和 Cross Learning 或者传统的基于简单的遗传算法的 Replicator dynamics 是一模一样的，但是有意思的 Q learning 多出来的一项。直觉上，我们可以将这意向理解为希望新的 x 的更新在老的 x 附近一定范围，但是又鼓励在该范围的尽力去探索。现代的深度强化学习算法很多都直接加入了这一项作为 reward，例如 deepmind 做 Stratego 的 AI[5]，OpenAI 做的 RLHF[2]。

A Cross Learning 和 Replicator Dynamics 的等价性

根据 Eq.(7, 8) 我们可以容易地计算出 cross learning 算法对应的策略改变的数学期望为，

$$\begin{aligned}\mathbb{E}[\Delta\pi(i)] &= \theta\pi(i)\mathbb{E}[r|i](1-\pi(i)) - \theta\sum_{j\neq i}\pi(j)\mathbb{E}[r|j]\pi(i) \\ &= \theta\pi(i)\left[\mathbb{E}[r|i] - \sum_j \pi(j)\mathbb{E}[r|j]\right]\end{aligned}$$

短时间我们将 π_t 更新到 $\pi_{t+\delta} := \pi_t + \delta\mathbb{E}[\Delta\pi]$, 因此连续时间的极限，

$$\frac{d\pi(i)}{dt} = \lim_{\delta\to 0}\frac{\pi_{t+\delta}(i) - \pi_t(i)}{\delta} = \theta\pi(i)\left[\mathbb{E}[r|i] - \sum_j \pi(j)\mathbb{E}[r|j]\right] \tag{11}$$

将这个方程和 Eq.(3) 对比我们发现，θ 类比 α。而 $U(e_i, x), U(x, x)$ 则是和 $\mathbb{E}[r|i], \sum_j \pi(j)\mathbb{E}[r|j]$ 是一回事，只是不同领域使用了不同的记号而已。

B Replicator Dynamics of Q learning

该证明我将省略 condition on state 的标记。一是为了避免大量重复缩短公式长度，二是我觉得证明中对于 stateful 是不成立，我也会指出来。对于 Q learning 来说，其策略分布如下：

$$x_i(t) := \frac{e^{\tau Q_t(i)}}{\sum_j e^{\tau Q_t(j)}} \tag{12}$$

我们的目标是找到 $\frac{dx_i(t)}{dt}$ 的表达式，当然我们采取的策略依然是离散到连续的极限近似，即：

$$\frac{dx_i(t)}{dt} := \lim_{k\delta\to t}\lim_{\delta\to 0}\frac{x_i((k+1)\delta) - x_i(k\delta)}{\delta} \tag{13}$$

鉴于 Boltzmann 系统的特性，我们这边采用统计物理常有的一个 trick 来计算这个极限，即：

$$\frac{d\ln x_t}{dt} = \lim_{\delta\to 0}\frac{x_{t+\delta} - x_t}{\delta x_t} = \lim_{\delta\to 0}\frac{1}{\delta}\ln\frac{x_{t+\delta}}{x_t} \tag{14}$$

因此为了计算 Eq.(13), 我们只需要计算 $\frac{1}{\delta}\ln\frac{x_i((k+1)\delta)}{x_i(k\delta)}$, 带入 Eq.(12), 我们发现

$$\lim_{k\delta\to t}\lim_{\delta\to 0}\frac{1}{\delta}\ln\frac{x_i((k+1)\delta)}{x_i(k\delta)} = \frac{1}{\delta}\ln\frac{e^{\tau\Delta Q_i(k\delta)}}{\sum_j x_j e^{\tau\Delta Q_j(k\delta)}} \tag{15}$$

$$= \lim_{k\delta\to t}\lim_{\delta\to 0}\frac{1}{\delta}\left(\tau\Delta Q_i(k\delta) - \tau\sum_j x_j\Delta Q_j(k\delta)\right) \tag{16}$$

$$= \tau\left(\frac{dQ_i(t)}{dt} - \sum_j x_j\frac{dQ_j(t)}{dt}\right) \tag{17}$$

这边我们还是使用了 $\sum_j x_j = 1, \lim_{x\to 0}\ln(1+x) = x$ 因此我们获得了第一个很重要的方程,

$$\frac{dx_i(t)}{dt} = x_i(t)(\frac{dQ_i(t)}{dt} - \sum_j x_j\frac{dQ_j(t)}{dt}) \tag{18}$$

注意观察 Eq.(18) 是不是和 Eq.(3) 有一些接近了，那么下面我们只需要计算 $\frac{dQ_i(t)}{dt}$ 了。那么知道在离散的情况下，Q learning 的更新为 Eq.(9)。我们采用的同样的方式，计算连续极限情况，

$$\frac{dQ_i(t)}{dt} := \lim_{k\delta\to t}\lim_{\delta\to 0}\frac{Q_i((k+1)\delta) - Q_i(k\delta)}{\delta}. \tag{19}$$

从 Eq.(9), 我们可以知道，

$$Q_i((k+1)\delta) - Q_i(k\delta) = \alpha\left(r_i(k\delta) + \gamma\max_j Q_j((k+1)\delta) - Q_i(k\delta)\right)((k+1)\delta - k\delta) \tag{20}$$

因此，我们得到第二个重要的方程:

$$\frac{dQ_i(t)}{dt} = \alpha(r_i + \gamma\max_j Q_j - Q_i) \tag{21}$$

合并 Eq.(18) 和 Eq.(21) 并使用条件 $\max_j Q_j$ 为一个常数（按照我的理解该条件只有在 stateless 的环境里才成立，对一般的环境应该是没有，$\max_a Q(s,a) = \max_a Q(s',a)$, 的），我们可以得到，

$$\frac{\frac{dx_i(t)}{dt}}{x_i(t)} = \tau\alpha(r_i - \sum_j x_j r_j + \sum_j(Q_j - Q_i)) \tag{22}$$

利用 $\frac{x_j}{x_i} = \frac{e^{\tau Q_j}}{e^{\tau Q_i}}$, 我们可以将最后一项化简成，

$$\frac{\frac{dx_i(t)}{dt}}{x_i(t)} = \tau\alpha(r_i - \sum_j x_j r_j) + \alpha \sum_j x_j \frac{x_j}{x_i} \tag{23}$$

证毕。

参考文献

[1] Tilman Börgers and Rajiv Sarin. Learning through reinforcement and replicator dynamics. *Journal of economic theory*, 77(1):1–14, 1997.

[2] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[3] Peter D Taylor and Leo B Jonker. Evolutionary stable strategies and game dynamics. *Mathematical biosciences*, 40(1-2):145–156, 1978.

[4] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 693–700, 2003.

[5] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.