2025 年 11 月 10 日至 11 月 16 日周报

何瑞杰
中山大学, 大湾区大学

1. 项目进展

1.1. 使用神经网络学习生命游戏的演化动力学

综合上周提出的总框架图（图 1），本周开始逐个考虑并使用代码实现其中的各关键部分。

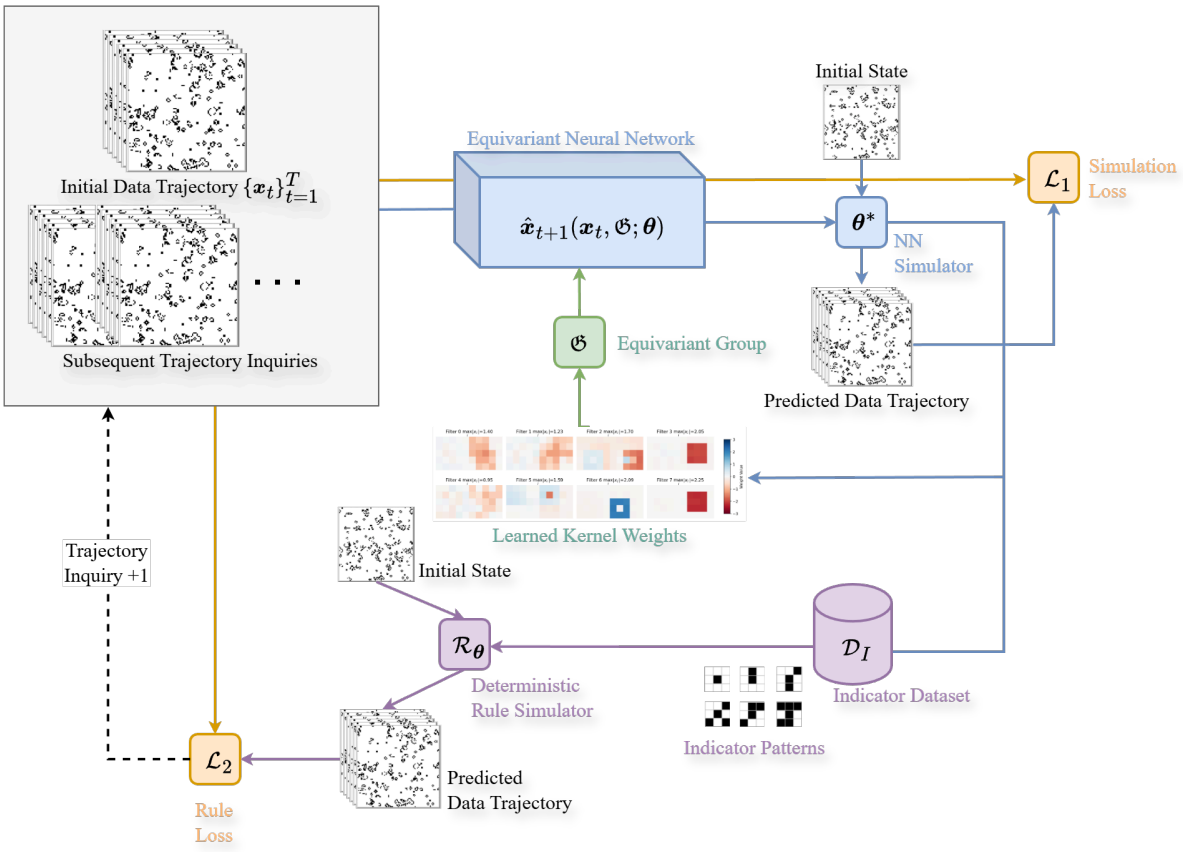

图 1 方法框架总览

1.2. 数据供应代理

asdasdasd

可以用现成数据集代替，但修改可见的演化轨道条数。在初始化时，令可见演化轨道条数为 1，每执行完毕一个主动学习循环，且需要继续循环时，可见演化轨道条数 +1，从而已收集好的新的一条轨道可以加入训练数据中。另外，还可以考虑将新的轨道赋予更大的权重，可以在 `__getitem__` 方法中添加一个输出，表示其权重，相应地将维护一个可见轨道条数改为维护一个可见轨道的有序列表，依据列表中轨道的先后赋予依次从低到高的权重。

1.3. 等变神经网络模块

神经网络训练模块有现成的代码，但是需要解决框架图中的施加等变约束的部分。注意刚开始的时候是只有平移不变性的，然后再加上其他的群等变约束，这意味着如果采用群等变卷积网络（`e2cnn`）包的话需要仔细研究其源码实现。一个暂行的替代方案是用普通的 CNN，然后加上不变性损失。例如对于晶体群 $p4$，可以令 3×3 卷积核的上下左右和四个对角的值分别相等，

可以采取强制相等，或者旋转损失的方式添加等变性约束。另外同时可以研究 e2cnn 的底层实现，来研究如何在不断扩充等变群元时最大程度保留上一次训练好的网络权重，避免需要再人为蒸馏一遍。

1.4. 指示数据集、规则提取器和生成器

先说指示数据集。虽然它生成起来非常简单：

```
w, h = 3, 3
ls = [[0, 1] for _ in range(w*h)]
indicator_dataset = None
for idx, i in enumerate(product(*ls)):
    arr = torch.tensor(list(i)).reshape(1, 3, 3)
    if indicator_dataset is None:
        indicator_dataset = arr
    else:
        indicator_dataset = torch.concatenate([indicator_dataset, arr])
```

但在 w 和 h 变大时，如果要遍历所有的 0-1 组合，其组合数将指数级增长（2^{wh}），带给生成和验证过程以很大的困难。另外，拿到指示数据集 indicator_dataset 和对应的网络预测值后，需要思考的是如何将其转化为确定性规则的问题。

在先前的想法中是选取网络的预测置信度高的转变对 $(x_t, N(x_t)) \to x_{t+1}$，但需要考虑两个问题，一是 pyseagull 的模拟器中是通过卷积来实现规则的，我们需要想办法将得到的转变对转化成卷积核的形式。换言之，我们需要将得到的这一堆转变对转换为显式的规则形式，我目前还没有什么好的想法。

2. 文献阅读

2.1. [续] Score-based generative modeling through SDE [1]

Yang Song et al. | https://arxiv.org/abs/2011.13456

2.1.1. 可控生成

在可控生成中，给定标签 $\boldsymbol{y}$，我们需要从假设的位置分布 $p_0(\boldsymbol{x}_0|\boldsymbol{y})$ 采样，即从对应于该类的先验分布 $p_N(\boldsymbol{x}_T|\boldsymbol{y})$ 开始采样。此时对应的对数梯度就变为

$$\begin{aligned}\nabla_{\boldsymbol{x}} p_t(\boldsymbol{x}|\boldsymbol{y}) &= \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x}) - \underbrace{\nabla_{\boldsymbol{x}} p_t(\boldsymbol{y})}_{=0} \\ &= \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x})\end{aligned} \tag{1}$$

因此得到的逆向 SDE 的形式就是

$$\mathrm{d}\boldsymbol{x} = \{\boldsymbol{f}(\boldsymbol{x},t) - g(t)^2[\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{y}|\boldsymbol{x})]\}\mathrm{d}t + g(t)\mathrm{d}\tilde{\boldsymbol{w}} \tag{2}$$

当 $\boldsymbol{y}$ 表示标签时，我们需要额外训练一个神经网络，以预测加噪过程中的 $p_t(\boldsymbol{y}|\boldsymbol{x}_t)$。

2.1.1.1. 图像修补

如果做图像修补任务，我们需要根据不全的图像点信息 $\Omega(\boldsymbol{y})$ 预测完整的图像信息 $\boldsymbol{y}$。如果记 $\bar{\Omega}$ 为图像的未知部分，并记 $\boldsymbol{f}_{\bar{\Omega}}$ 和 $\boldsymbol{G}_{\bar{\Omega}}$ 为相应函数在未知像素点 $\bar{\Omega}$ 的限制，图像修补问题就是从 $p\big(\bar{\Omega}(\boldsymbol{y})|\Omega(\boldsymbol{x}_0)=\boldsymbol{y}\big)$ 中采样。定义一个对 $\boldsymbol{z}(t)=\bar{\Omega}(\boldsymbol{x}(t))$ 的新的扩散过程：

$$\mathrm{d}\boldsymbol{z} = \boldsymbol{f}_{\bar{\Omega}}(\boldsymbol{z},t)\mathrm{d}t + \boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)\mathrm{d}\boldsymbol{w} \tag{3}$$

这对应的逆向条件 SDE 为

$$\begin{aligned}\mathrm{d}\boldsymbol{z} = \{&\boldsymbol{f}_{\bar{\Omega}}(\boldsymbol{z},t) - \nabla\cdot[\boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)\boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)^\top] \\ &-\boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)\boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)^\top \nabla_{\boldsymbol{z}} \log p_t(\boldsymbol{z}|\Omega(z(0))=\boldsymbol{y})\}\mathrm{d}t + \boldsymbol{G}_{\bar{\Omega}}(\boldsymbol{z},t)\mathrm{d}\bar{\boldsymbol{w}}\end{aligned} \tag{4}$$

注意上式中的 $p_t(\boldsymbol{z}|\Omega(z(0))=\boldsymbol{y})$ 无法计算，可以用下面的方法近似。记为 $\Omega(z(0))=\boldsymbol{y}$ 事件 A，有

$$\begin{aligned}p_t(\boldsymbol{z}|\Omega(z(0))=\boldsymbol{y}) = p_t(\boldsymbol{z}|A) &= \int p_t(\boldsymbol{z}|\Omega(z(t)),A)p_t(\Omega(\boldsymbol{x}(t))|A)\mathrm{d}\Omega(\boldsymbol{x}(t)) \\ &= \mathbb{E}_{p_t(\Omega(\boldsymbol{x}(t))|A)}[p_t(\boldsymbol{z}|\Omega(z(t)),A)] \approx \mathbb{E}_{p_t(\Omega(\boldsymbol{x}(t))|A)}[p_t(\boldsymbol{z}|\Omega(z(t)))] \approx p_t\big(\boldsymbol{z}|\hat{\Omega}(z(t))\big)\end{aligned} \tag{5}$$

其中 $\hat{\Omega}$ 是 $p_t(\Omega(\boldsymbol{x}(t))|A)$ 对于梯度，可以有下面的估计

$$\begin{aligned}\nabla_{\boldsymbol{z}} \log p_t(\boldsymbol{z}(t)|\Omega(\boldsymbol{z}(0)=\boldsymbol{y})) &\approx \nabla_{\boldsymbol{z}} p_t\big(\boldsymbol{z}(t)|\hat{\Omega}(\boldsymbol{x}(t))\big) \\ &= \nabla_{\boldsymbol{z}} p_t\Big(\big[\boldsymbol{z}(t);\hat{\Omega}(\boldsymbol{x}(t))\big]\Big)\end{aligned} \tag{6}$$

其中 $\big[\boldsymbol{z}(t);\hat{\Omega}(\boldsymbol{x}(t))\big]$ 表示一个图像向量 $\boldsymbol{u}(t)$，满足 $\Omega(\boldsymbol{u}(t))=\hat{\Omega}(\boldsymbol{x}(t))$ 以及 $\bar{\Omega}(\boldsymbol{u}(t))=\boldsymbol{z}(t)$，换句话说 $\boldsymbol{u}(t)$ 就是由 $\boldsymbol{z}(t)$ 和 $\hat{\Omega}(\boldsymbol{x}(t))$ 两部分拼起来的一个完整的图像向量。

2.1.1.2. 图像着色

对于图像着色，可以先用一个正交矩阵将灰度图向三个颜色方向做正交投影，然后对每个通道分别做图像补全。

2.1.1.3. 一般的反问题

正问题为从 $\boldsymbol{x}$ 根据条件分布 $p(\boldsymbol{x}|\boldsymbol{y})$ 生成 $\boldsymbol{y}$，相应地反问题需要从 $\boldsymbol{y}$ 根据反向条件分布 $p(\boldsymbol{y}|\boldsymbol{x})$ 生成 $\boldsymbol{x}$。我们可以考虑一系列用 SDE 扰动得到的扩散过程 $\{\boldsymbol{x}(t)\}_{t=0}^{T}$，和一个训练用于估计 $\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t))$ 的分数模型 $\boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{x}(t),t)$。只要我们能估计 $\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t)|\boldsymbol{y})$，我们就可以用逆向 SDE 得到 $p_0(\boldsymbol{x}(0)|y)$，也就是 $p(\boldsymbol{x}|\boldsymbol{y})$。首先有

$$\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t)|\boldsymbol{y}) = \nabla_{\boldsymbol{x}}\log\int p_t(\boldsymbol{x}(t)|\boldsymbol{y}(t),\boldsymbol{y})p(\boldsymbol{y}(t)|\boldsymbol{y})\mathrm{d}\boldsymbol{y}(t) \tag{7}$$

其中 $\boldsymbol{y}(t)$ 是 $\boldsymbol{x}(t)$ 关于前向过程 $p(\boldsymbol{y}(t)|\boldsymbol{x}(t))$ 生成的结果。

现在假定 $p(\boldsymbol{y}(t)|\boldsymbol{y})$ 是可达的，且 $p_t(\boldsymbol{x}(t)|\boldsymbol{y}(t),\boldsymbol{y})\approx p_t(\boldsymbol{x}(t)|\boldsymbol{y}(t))$（当 t 较小时，$\boldsymbol{y}(t)$ 和 $\boldsymbol{y}$ 相差无几；而当 t 较大时，$\boldsymbol{y}$ 对 $\boldsymbol{x}(t)$ 的影响可以忽略不计。）根据这两个条件，可以得到

$$\begin{aligned}
\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t)|\boldsymbol{y}) &\approx \nabla_{\boldsymbol{x}}\log\int p_t(\boldsymbol{x}(t)|\boldsymbol{y}(t))p(\boldsymbol{y}(t)|\boldsymbol{y})\mathrm{d}\boldsymbol{y}\\
&\approx \nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t)|\hat{\boldsymbol{y}}(t))\\
&\approx \nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}(t)) + \nabla_{\boldsymbol{x}}\log p_t(\hat{\boldsymbol{y}}(t)|\boldsymbol{x}(t))\\
&\approx \boldsymbol{s}_{\boldsymbol{\theta}^*}(\boldsymbol{x}(t),t) + \nabla_{\boldsymbol{x}}\log p_t(\hat{\boldsymbol{y}}(t)|\boldsymbol{x}(t))
\end{aligned} \tag{8}$$

其中 $\hat{\boldsymbol{y}}(t)$ 是从 $p(\boldsymbol{y}(t)|\boldsymbol{y})$ 中采样得到的。得到梯度的形式后就可以将它放进条件逆向 SDE 中求解了。

3. 学习进度

3.1. 随机微分方程

本周结束了 Ito 随机微分那一章。其中一度卡在了章末的注记法部分。以乘积公式为例，

接着我最终进入随机微分方程一章，并学习了若干种典型形式的随机微分方程，例如股市模型、Brown 桥、随机谐振子、Langevin 方程以及其更一般地形式，即 Ornstein–Uhlenbeck 过程。

4. 下周计划

论文阅读

1. 生成模型
   - 薛定谔桥（精读）
   - Stable Neural Stochastic Differential Equations in Analyzing Irregular Time Series Data（精读）
   - DDIM（泛读）

项目进度

1. 使用神经网络学习生命游戏的演化动力学
   - 实现数据供应代理模块和等变神经网络模块
2. 耦合约瑟夫森结
   - 将 MATLAB 模拟代码全部迁移至 Python
   - 考虑简单的 Neural SDE 方法解带参 OU 过程的参数

理论学习

1. 随机过程课程
   - 复习 Poisson 过程
2. 随机微分方程
   - 第五章

参考文献

[1] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," *CoRR*, 2020, [Online]. Available: https://arxiv.org/abs/2011.13456