2025 年 9 月 8 日至 9 月 14 日周报

何瑞杰
中山大学, 大湾区大学

1. 文献阅读

1.1. Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song, Stefano Ermon | NeurIPS 2019 | https://arxiv.org/abs/1907.05600

1.1.1. Score matching

1.1.1.1. 生成模型和 Score matching 动机

生成模型的目的是获取所需要生成范畴中的对象（如图片）的隐藏分布。生成的过程就是从该分布中采样。假设有从一个未知分布 $p_{\text{data}}(\boldsymbol{x})$ 中采样得到的数据集 $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^n$。我们要尝试估计该分布。一个自然的假设是

$$p_{\text{data}}(\boldsymbol{x}) = \frac{\exp(-f_\theta(\boldsymbol{x}))}{Z(\theta)}$$

其中 $f_\theta : \mathbb{R}^D \to \mathbb{R}$ 是某个函数，$Z(\theta)$ 是归一化因子。若不加其他考虑，直接处理 $p_{\text{data}}(\boldsymbol{x})$ 将会不可避免地遇到计算 $Z(\theta)$ 的困难。因此 Score matching 的一个核心思路是转而去估计分布的 Score function，其“几何直观”的意义是指向概率密度增加的方向：

$$s_\theta(\boldsymbol{x}) := \nabla_{\boldsymbol{x}} \log p_\theta(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}) - \underbrace{\nabla_{\boldsymbol{x}} \log Z(\theta)}_{=0} = -\nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}).$$

1.1.1.2. 以方便计算为目的的 Score matching 目标函数变换

自然地，我们有 Score matching 的原始目标函数：

$$J(\theta) := \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\|s_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\|].$$

但由于我们不知道原始数据的分布，因此我们无法求得 $\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})$，可以做下面的变换

$$\begin{aligned}
&\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\|] \\
=& \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \cancel{\mathbb{E}_{p_{\text{data}}}[\|\nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\|]} - \mathbb{E}_{p_{\text{data}}}[\langle s_\theta(\boldsymbol{x}), \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\rangle] \\
=& \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \int p(\boldsymbol{x})\langle s_\theta(\boldsymbol{x}), \nabla_{\boldsymbol{x}} \log p_{\text{data}}(\boldsymbol{x})\rangle \mathrm{d}x + \text{constant} \\
=& \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \int \langle s_\theta(\boldsymbol{x}), \nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\rangle \mathrm{d}x + \text{constant} \\
=& \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \left(\cancel{\int_{\partial\mathbb{R}^D} s_\theta(\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x}) \mathrm{d}x} + \int_{\mathbb{R}^D} p_{\text{data}}(\boldsymbol{x}) \nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x})\right) + \text{constant} \\
=& \frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \int p(\boldsymbol{x})\ \mathrm{div}(s_\theta(\boldsymbol{x}))\mathrm{d}x + \text{constant} \\
=& \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\|s_\theta(\boldsymbol{x})\| + \mathrm{tr}(\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}))\right]
\end{aligned}$$

1.1.1.3. 降低目标函数计算成本

上式中的 $\mathrm{tr}(\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}))$ 计算成本还是太高。庆幸我们可以对原数据增加 Gaussian 噪声，从而将其经过一个条件分布 $q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x}) \sim N(0, \sigma^2 I)$ 得到加噪声后的数据 $\bar{\boldsymbol{x}}$。经计算后我们能得到更加实际的目标函数。我们可以从扰动后的数据向量 $\bar{\boldsymbol{x}} \sim q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})$ 对应的损失函数开始推导：

$$
\begin{aligned}
J(\theta) &= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}}) - \nabla_{\bar{\boldsymbol{x}}} \log p_\sigma(\bar{\boldsymbol{x}})\|_2^2\right] \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2\right] + \cancel{\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|\nabla_{\bar{\boldsymbol{x}}} \log p_{\mathrm{data}}(\bar{\boldsymbol{x}})\|_2^2\right]} - \mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}[\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} \log p_\sigma(\bar{\boldsymbol{x}})\rangle] \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2\right] - \int p_\sigma(\bar{\boldsymbol{x}})\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} \log p_\sigma(\bar{\boldsymbol{x}})\rangle \mathrm{d}\bar{\boldsymbol{x}} + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2\right] - \int \langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} p_\sigma(\bar{\boldsymbol{x}})\rangle \mathrm{d}\bar{\boldsymbol{x}} + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2\right] - \int \left\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} \int q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x}) p_{\mathrm{data}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \right\rangle \mathrm{d}\bar{\boldsymbol{x}} + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2\right] - \iint p_\sigma(\bar{\boldsymbol{x}}) p_{\mathrm{data}}(\boldsymbol{x}) \langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})\rangle \mathrm{d}\boldsymbol{x}\mathrm{d}\bar{\boldsymbol{x}} + \text{constant} \\
&= \mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}}), \boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[\frac{1}{2}\|s_\theta(\bar{\boldsymbol{x}})\|_2^2 - \langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})\rangle\right] + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\bar{\boldsymbol{x}}\sim p_\sigma(\bar{\boldsymbol{x}}), \boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[\|s_\theta(\bar{\boldsymbol{x}})\|_2^2 - 2\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})\rangle + \underbrace{\|\nabla_{\bar{\boldsymbol{x}}} q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})\|_2^2}_{\text{constant w.r.t. } \theta}\right] + \text{constant} \\
&= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}, }\left[\|s_\theta(\bar{\boldsymbol{x}}) - \nabla_{\bar{\boldsymbol{x}}} \log q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})\|_2^2\right]
\end{aligned}
$$

注意此时 $\nabla_{\bar{\boldsymbol{x}}} \log q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x})$ 还可以写为 $-\dfrac{1}{\sigma}\cdot\varepsilon$，其中 ε 是从 $N(0, I)$ 中采样得到的噪声，原目标函数就变成

$$
J(\theta) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}, }\left[\left\|s_\theta(\bar{\boldsymbol{x}}) + \frac{1}{\sigma}\cdot\varepsilon\right\|_2^2\right]
$$

换句话说，score function 在这个意义下正在预测加入到训练数据中的噪声。在实际操作中，我们需要权衡 ε 的取值，如果太小，该方法起不到明显效果；如果太大，加噪声后的分布和原分布的区别大，难以学习原分布的特征。

1.1.2. Langevin 动力学

假如我们已经训练好 $s_\theta(\boldsymbol{x})$，我们应该如何做“生成”这个动作呢？根据 score function 拟合概率的对数梯度 $\nabla_{\boldsymbol{x}} \log p_{\mathrm{data}}(\boldsymbol{x})$，它指向概率密度上升的方向。我们可以按照自然地想法做梯度上升，这样迭代就可以得到很有可能属于该分布的点：

$$
\boldsymbol{x}_{\{t+1\}} \leftarrow \boldsymbol{x}_t - \varepsilon s_\theta(\boldsymbol{x}_t)
$$

但这一方法总会使得若干步后的采样结果收敛于原始分布之密度函数的若干极大值点，与生成模型的多样性目标不符。因此为解决这一问题，我们可以用 Langevin Dynamics 来采样。其与原来方法的区别在于在每一步中加入噪声，最后迭代得到的结果将会服从原始分布 $p_{\mathrm{data}}(\boldsymbol{x})$：

$$
\boldsymbol{x}_{\{t+1\}} \leftarrow \boldsymbol{x}_t - \frac{\varepsilon}{2} s_\theta(\boldsymbol{x}_t) + \sqrt{\varepsilon} z_t
$$

其中 $z_t \sim N(0, I)$。

1.1.3. Score-based 生成模型的问题

1. 一是流形假设造成的问题。我们所处世界中的高维数据往往分布在一个低维流形上。但上文中提到的对数据分布密度函数在低维流形的环绕空间 $\mathbb{R}^D$ 中求梯度是没有意义的。
2. 二是低概率密度区域的估计问题。如果原分布是一个混合分布，且两个“峰”中间存在一个低概率密度的区域，模型学习时将难以获取该区域的信息，最后训练的结果在该区域的表现将会很差。

如果我们使用 Gauss 分布对原分布做扰动，则得到的新分布的支持集将会是整个环绕空间，而不是流形。另一方面，恰当强度的扰动也会使得低概率密度区域的概率密度增加，从而更容易采样到该区域中的点，为模型训练提供更多的信息。

1.1.4. 方法

上文中提到，对数据做扰动时，大强度的扰动会使得训练变得简单，但扰动后的分布与原分布相差很大；小强度的扰动使得扰动后分布近似原分布，但会有诸如低概率密度区域训练点不足的问题。

文中提出一个整合两者有点的方法，即不考虑单个扰动强度 σ，而是考虑一个序列 $\{\sigma_i\}_{i=1}^n$ 其中 σ_n 是一个足够小的数 (例如 0.01)，σ_1 是一个足够大的数 (例如 25)。我们训练一个条件 score function $s_\theta(\boldsymbol{x}, \sigma)$ 预测不同扰动强度下的噪声方向。此时目标函数变为

$$\ell(\theta, \sigma) := \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}), \bar{\boldsymbol{x}} \sim N(\boldsymbol{x}, \sigma^2 I)}\left[\left\| s_\theta(\bar{\boldsymbol{x}}) + \frac{\bar{\boldsymbol{x}} - \boldsymbol{x}}{\sigma^2} \right\|_2^2\right]$$

$$\ell(\theta, \{\sigma_i\}_{i=1}^n) := \frac{1}{n}\sum_{i=1}^n \lambda(\sigma_i)\ell(\theta, \sigma_i)$$

其中 $\lambda(\sigma_i)$ 是权重函数，常取为 $\lambda(\sigma_i) = \sigma_i^2$，以平衡不同扰动强度下 score function 的范数。在采样时，我们就做类似模拟退火的采样动作。首先选取最高的扰动强度 σ_1 然后在该噪声强度下迭代若干次，然后选取次高的扰动强度 σ_2 并在该噪声强度下迭代若干次，以此类推。这样就可以综合利用大扰动强度和小扰动强度的有点，从而更好的采样。在实际实验中，作者提出的方法在 CIFAR-10 数据集上取得了不错的效果。整个过程如下面的算法所示。

Algorithm 1: Annealed Langevin Dynamics

procedure ANNEALED LANGEVIN DYNAMICS($\{\sigma_i\}_{i=1}^n$, ε, T)
  ▷ Initialize the search range
  $\bar{\boldsymbol{x}}_0 \leftarrow \boldsymbol{v}$
  for $t \leftarrow 1, ..., L$ **do**
    ▷ Set step size α_i
    $\alpha_i \leftarrow \varepsilon \cdot \sigma_i^2 / \sigma_L^2$
    for $t \leftarrow 1, ..., T$ **do**
      Draw $\boldsymbol{z}_t \sim N(0, I)$
      $\bar{\boldsymbol{x}}_t \leftarrow \bar{\boldsymbol{x}}_{t-1} + \alpha_i/2 \cdot s_\theta(\bar{\boldsymbol{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\boldsymbol{z}_t$
    end
    $\bar{\boldsymbol{x}}_0 \leftarrow \bar{\boldsymbol{x}}_T$
  end
  return $\bar{\boldsymbol{x}}_T$
end

参考资料

1. https://www.youtube.com/watch?v=B4oHJpEJBAA

2. 项目进展

2.1. 使用神经网络学习生命游戏的演化动力学

2.1.1. 神经网络的预测规则验证

本周我根据杨武岳老师的建议开始分析训练好的模型。一个对于训练好模型的解释是其是否满足下面的生命游戏演化动力学。假如 c 代表某个细胞的状态，$\boldsymbol{n} = [n_1, n_2, ..., n_8]$ 为它八个邻居的存活状态，则其演化动力学可以写成

$$f(c, \boldsymbol{n}) = \begin{cases} 1 \text{ if } c = 0 \text{ and } \|\boldsymbol{n}\|_1 = 3 \\ 1 \text{ if } c = 1 \text{ and } \|\boldsymbol{n}\|_1 \in \{2, 3\} \\ 0 \text{ otherwise} \end{cases}$$

我获取训练好的简化序贯模型后，对全数据集中抽样检查了其中的 1400 组左右的系统状态变换的预测情况，包含大致 1.4×10^7 个细胞状态变换。统计方法为建立一个字典，其键形如 (x, y)，其中 $x, y \in \{0, 1\}$ 对应的值为一个列表，抽取数据中细胞状态为 x 被模型预测下一状态为 y 时，它周围的存活细胞数量。经过统计可得下图。

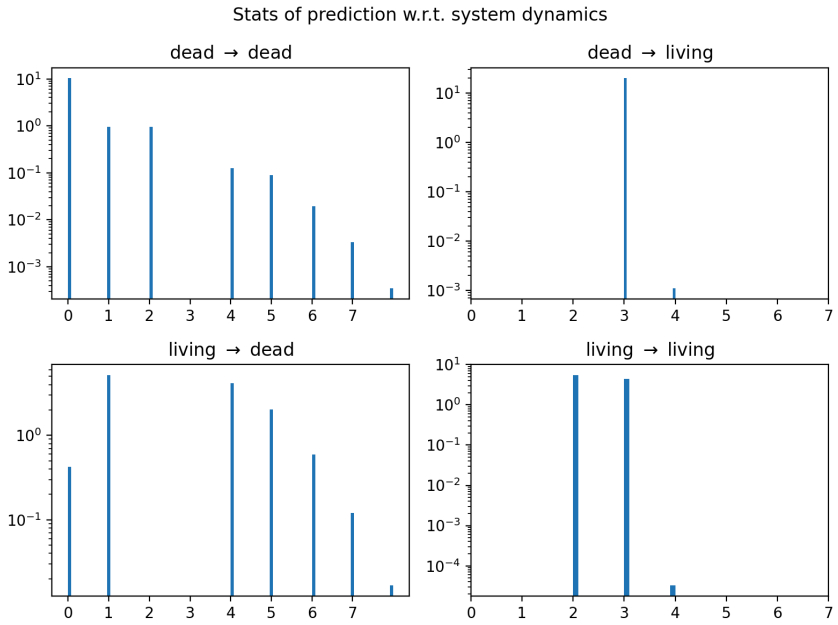

Figure 1: 神经网络对生命游戏状态预测的统计

可见除了极少数预测错误外，网络的预测规则和生命游戏的演化动力学基本契合。

2.1.2. 首层卷积核可视化和冗余卷积核的去除

其次，我对原来特征图通道数为 8 的串行 CNN 模型进行进一步的简化，先将通道数降为 4，发现训练完成后仍有大量卷积核参数矩阵的分量绝对值较低，于是进一步将通道数降低到 2。下面是降低后的模型代码和结构

```
class SimpleCNNTiny(nn.Module):
    __version__ = '0.1.0'
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(2, 1, 3, 1, padding=1, padding_mode="circular")
        self.bn1 = nn.BatchNorm2d(1)
        self.act1 = nn.ReLU()
        self.conv2 = nn.Conv2d(1, 1, 3, 1, padding=1, padding_mode="circular")
        self.bn2 = nn.BatchNorm2d(1)
```

```
        self.act2 = nn.ReLU()
        self.conv3 = nn.Conv2d(1, 2, 3, 1, padding=1, padding_mode="circular")
    def forward(self, x: Float[Array, "batch 2 w h"]
                ) -> Float[Array, "batch 2 w h"]:
        x = self.act1(self.bn1(self.conv1(x)))
        x = self.act2(self.bn2(self.conv2(x)))
        return self.conv3(x)
```

torchinfo 输出的模型结构分析如下

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
SimpleCNNTiny                            [1, 2, 100, 100]          --
├─Conv2d: 1-1                            [1, 1, 100, 100]          19
├─BatchNorm2d: 1-2                       [1, 1, 100, 100]          2
├─ReLU: 1-3                              [1, 1, 100, 100]          --
├─Conv2d: 1-4                            [1, 1, 100, 100]          10
├─BatchNorm2d: 1-5                       [1, 1, 100, 100]          2
├─ReLU: 1-6                              [1, 1, 100, 100]          --
├─Conv2d: 1-7                            [1, 2, 100, 100]          20
==========================================================================================
Total params: 53                         Trainable params: 53
Non-trainable params: 0                  Total mult-adds (M): 0.49
==========================================================================================
Input size (MB): 0.08                    Forward/backward pass size (MB): 0.48
Params size (MB): 0.00                   Estimated Total Size (MB): 0.56
==========================================================================================
```

我对进行 2 epoch 训练后的模型的首层卷积核参数进行可视化，发现对应于 x_t 的卷积核的激活程度明显小于对应于 x_{t+1} 的激活程度。因此我认为可以恢复原来给定 x_t 预测 x_{t+1} 的模式。

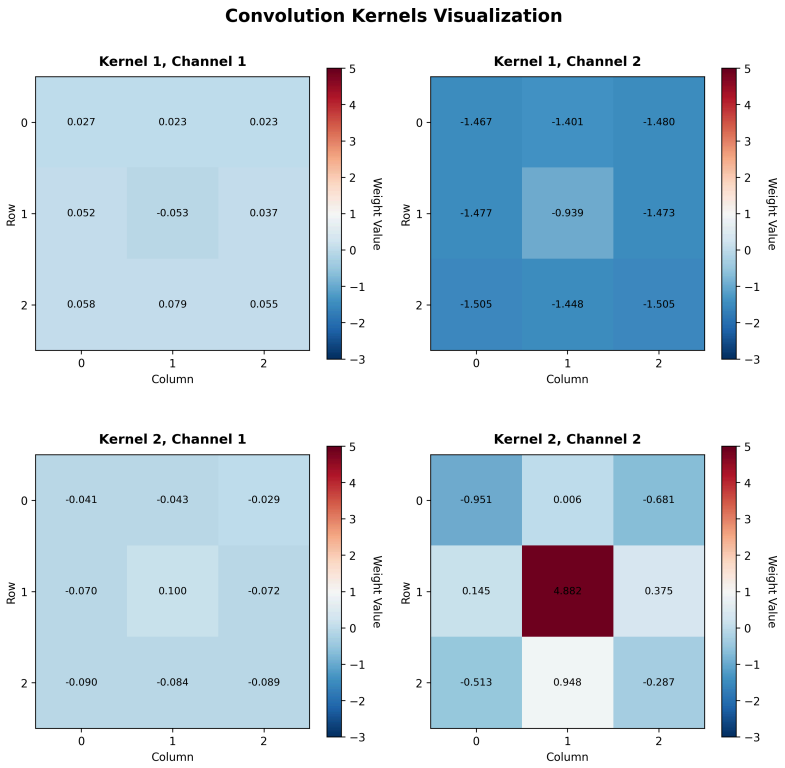

Figure 2: 对双通道特征图串行结构 CNN 的首层卷积核的可视化结果

3. 学习进度

3.1. Gray-Scott 系统

Gray-Scott 描述了包含两个反应的体系

$$\begin{aligned} U + 2V &\longrightarrow 3V \\ V &\longrightarrow P \end{aligned}$$

体系中的 U 和 V 的浓度 u 和 v 的变化可用下面的偏微分方程描述

$$\begin{aligned} \frac{\partial u}{\partial t} &= -uv^2 + f(1-u) + D_u \Delta u \\ \frac{\partial v}{\partial t} &= \quad uv^2 - (f+k)v + D_v \Delta v \end{aligned}$$

其中 D_u 和 D_v 是扩散系数，f 和 k 是对应物质的填补和消耗速率常数。上式中等号右边第一项表示化学反应速率，第二项表示体系中物质 U 的添补速率和 V 的消耗速率，第三项表示两种物质在体系中的扩散。

在实际的数值模拟中，大多数 (f,k) 的组合都会使得系统演化向无趣的平衡状态：如果组合位于相图阈值线的一边，系统加入的 U 物质将被立刻反应；如果位于另一边，系统中反应产生的 V 将被立刻耗尽。但当 (f,k) 接近阈值线时，系统将会演化出不同类型的斑图。关于具体稳定点和分岔的分析暂时没有看懂。

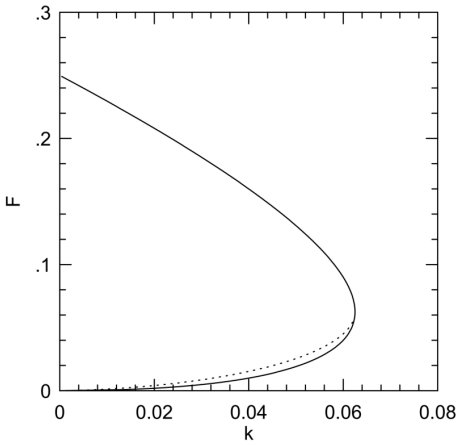

Figure 3: Gray-Scott 系统的相图

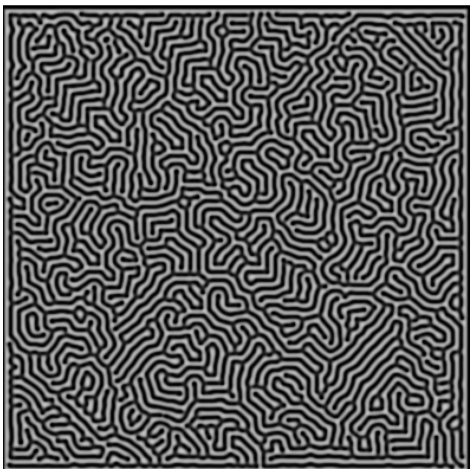

Figure 4: Gray-Scott 系统当 $F = 0.029; k = 0.057$ 时呈现的“迷宫”斑图

参考资料

1. https://itp.uni-frankfurt.de/~gros/StudentProjects/Projects_2020/projekt_schulz_kaefer/
2. https://groups.csail.mit.edu/mac/projects/amorphous/GrayScott/

3.2. 随机过程

第一次课首先进行了概率论部分的简单回顾，但我了解到了一些之前不曾关注或早已遗忘的知识。首先是累积分布的间断点至多可数。以及实值非负随机变量 X 的期望 $\mathbb{E}[X]$ 可以写成

$$\mathbb{E}[X] = \int x f_{X(x)} \mathrm{d}x = \int x \mathrm{d}F_X(x)$$

此时我们可以做一个变换，将“竖着”的 Riemann 和变成横着的：x 变成 $\mathrm{d}x$，F_X 变成 $F_X(+\infty) - F_X(x) = F_X^{\mathrm{c}}(x)$。这样我们就能得到 $\mathbb{E}(X) = \int F_X^{\mathrm{c}}(y)\mathrm{d}y$。

第一次课对应的讲义例题中提到了两个比较有趣的事。首先是分拆数，这一问题隐藏在球盒问题中。其设定为，有 n 个球，需要放入 M 个盒中，球和盒都无法区分，每个盒所装的球数量不限。该问题可以理解为，将正整数 n 做 $k=1,\dots,M$ 次分拆，然后把所有的情况加起来。而分拆数本身则是一个经典的递归/动态规划算法题。记将正整数 n 做 k 次分拆的所有情况总数为 $p_k(n)$（这里的分拆是正整数的无序分拆），定义 $p_0(0)=1$ 那么首先可以得到边界条件：

1. $p_0(n)=0, n>1$
2. $p_1(n)=1$（整数的平凡分拆）
3. $p_k(n)=0, k>n$ （不可能多分）
4. $p_k(k)=1$（每份恰好是 1）

接着为了得到递推式，可以做下面的讨论。我们关注最小的一份：

- 假如这一份是 1，那么我们可以把这一份扔掉，剩下的所有情况数量是 $p_{k-1}(n-1)$
- 假如最小的一份大于 1，那说明分出来的每一份都大于 1，我们把每一份都减一，分拆的份数不变，因此所有情况的数量是 $p_k(n-k)$。

总而言之，可以得分拆数的递归式 $p_k(n)=p_{k-1}(n-1)+p_k(n-k)$。

另一个提到的事是一列独立同分布随机变量的最大值、最小值和极差的分布。首先看最大值。假设一列随机变量 $\{X_i\}_{i=1}^n$ 的最大值小于等于 k，则蕴含对任意 i，都有 $X_i\leqslant k$。因此有

$$F_{\max}(k)=P(X_1\leqslant k,\dots,X_n\leqslant k)=\prod_{i=1}^n F_{X_i}(X_i\leqslant k)$$

最小值分布同理，只需在推导中善用 $F_X(k)=1-P(X\leqslant k)$ 这一恒等式，结果为

$$F_{\min}(k)=1-\prod_{i=1}^n\left[1-F_{X_i}(k)\right]$$

比较麻烦的是极差，但其想法很简单。先假设一列随机变量的最小值是 m，极差为 k，所以除了最小、最大取值的两个随机变量，剩余随机变量都位于 $[m,m+k]$ 中。因此我们有

$$\begin{aligned}
F_Z(k)&=\int_{-\infty}^{k}\int_{-\infty}^{\infty}2!\cdot\binom{n}{2}[F_X(m+t)-F_X(m)]^{n-2}f_X(m)f_X(m+t)\mathrm{d}m\mathrm{d}t\\
&=\int_{-\infty}^{k}\int_{-\infty}^{\infty}n(n-1)[F_X(m+t)-F_X(m)]^{n-2}f_X(m)f_X(m+t)\mathrm{d}m\mathrm{d}(F_X(m+t)-F_X(m))\\
&=\int_{-\infty}^{\infty}nf_X(m)\int_{-\infty}^{k}(n-1)[F_X(m+t)-F_X(m)]^{n-2}\mathrm{d}(F_X(m+t)-F_X(m))\mathrm{d}m\\
&=\int_{-\infty}^{\infty}nf_X(m)[F_X(m+t)-F_X(m)]^{n-1}\,|_{t=-\infty}^{t=k}\,\mathrm{d}m\\
&=\int_{-\infty}^{\infty}nf_X(m)[F_X(m+k)-F_X(m)]^{n-1}\mathrm{d}m.
\end{aligned}$$

3.3. 随机微分方程

进度推进至 Itô 积分的定义，它的思路比较长。

首先有一个 **Paley-Wiener-Zygmund** 随机积分定义，它需要要求被积函数 g 是一个确定的函数，无法满足 $\int_0^t \boldsymbol{B}(\boldsymbol{X},s)\mathrm{d}\boldsymbol{W}$ 这样的情形。因此我们考虑从 Riemann 和的角度逐步推广。

首先对于一维 Brown 运动 W 和区间 $[0,T]$ 上的一个划分 P，我们可以定义 $\int_0^T W\mathrm{d}W$ 的 Riemann 和估计

$$R = R(P,\lambda) = \sum_{k=0}^{m-1} W(\tau_k)[W(t_{k+1}) - W(t_k)]$$

接着我们就研究当 P 的细度趋于零时这个 Riemann 和是否收敛。我们首先证明了一维 Brown 运动在 $[a,b]$ 的二次变差在 $L^2(\Omega)$ 中趋于 $b-a$ —— 这也侧面说明其在任意该区间上的变差几乎必然无限 —— 然后我们可以用此结果证明上面的 Riemann 和估计在划分变细时有极限

$$\lim_{|P|\to 0} R(P,\lambda) = \frac{W(T)^2}{2} + \left(\lambda + \frac{1}{2}\right)T$$

看似自然的取法是令 $\lambda = \frac{1}{2}$ 这将得到 **Stranovich** 积分。而 Itô 积分的取法是 $\lambda = 0$，也就是划分小区间的中点取得是小区间的左端点，这将在后续的处理中带来便利。

接着我们开始研究可以作为被积的随机过程。我们考虑的是在 $[0,T]$ 上二次可积的循序可测随机过程空间 $\mathbb{L}^2(0,T)$。和 Lebesgue 积分的定义类似，我们先从“简单”的循序可测过程开始，也就是阶梯过程。类似阶梯函数，阶梯过程 $G \in \mathbb{L}^2(0,T)$ 是这样的随机过程：存在 $[0,T]$ 上的一个划分 $P = \{0 = t_0 < t_1 < \cdots < t_m = T\}$，使得对任意 $t \in [t_k, t_{k+1})$，都有 $G(t) \equiv G(t_k) = G_k$。其中 $G(t_k)$ 是 $\mathcal{F}(t_k)$-可测的。有了阶梯过程的定义，我们容易给出其随机积分的形式

$$\int_0^T G\mathrm{d}W := \sum_{k=0}^{m-1} G_k[W(t_{k+1}) - W(t_k)]$$

接着，任意 $G \in \mathbb{L}^2(0,T)$ 都可以被有界阶梯过程逼近，从而可以形成 Itô 积分的良好定义：存在一个极限为 G 的阶梯过程序列 $G^{(n)}$，则 G 的随机 Itô 积分就定义为

$$\int_0^T G\mathrm{d}W := \lim_{n\to\infty} \int_0^T G^{(n)}\mathrm{d}W.$$

4. 问题解决记录

4.1. Typst 相关

我正学习使用 Typst 排版周报和回报 Slides，其语法相比 LaTeX 更加简洁，编译速度也快得多，使用者若有 markdown 或 LaTeX 的基础，上手十分容易。

4.1.1. 缩放内积括号

在本次周报的编辑中遇到了内积括号无法放大的问题，最终在官方问答区找到答案。具体而言，需要加上 `lr()`，即 `lr(angle.l ... angle.r)`，例如

$$\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} \int q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\rangle \qquad \left\langle s_\theta(\bar{\boldsymbol{x}}), \nabla_{\bar{\boldsymbol{x}}} \int q_\sigma(\bar{\boldsymbol{x}}|\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right\rangle$$

这是没有加上 `lr()` 的版本（左）和加上了 `lr()` 版本（右）的对比。

参考资料

1. https://forum.typst.app/t/how-to-use-latexs-langle-and-rangle-functions-in-typst/2974

4.2. 推导相关

4.2.1. 更一般的分部积分公式

在推导 Score matching 中的优化目标变换时，在

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}}[\|s_\theta(\boldsymbol{x})\|] + \int \langle s_\theta(\boldsymbol{x}), \nabla_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x})\rangle \mathrm{d}x = \mathbb{E}_{p_{\text{data}}}\left[\frac{1}{2}\|s_\theta(\boldsymbol{x})\| + \operatorname{tr}(\nabla_{\boldsymbol{x}} s_\theta(\boldsymbol{x}))\right] + \text{constant}$$

这一步遇到了问题。讲解视频中常常将 x 考虑为一维向量，从而规避了这里的推导。一维版本推导中使用到分部积分公式，我推测多维版本也使用了分部积分公式。我参考 https://www.jhanmath.com/?p=142 中的内容将其中的缺失步骤补齐，并填补先前没有学好的散度相关知识。

对于函数 $f:\mathbb{R}^3 \longrightarrow \mathbb{R}$，其对应的微分算子可以写成 $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]$，这与 f 的梯度相容，因为 $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]$。散度是该微分算子和向量场 $\boldsymbol{V} = [\boldsymbol{V}_x, \boldsymbol{V}_y, \boldsymbol{V}_z]$ 的内积：

$$\operatorname{div}(\boldsymbol{V}) = \langle \nabla, \boldsymbol{V}\rangle = \frac{\partial \boldsymbol{V}_x}{\partial x} + \frac{\partial \boldsymbol{V}_y}{\partial y} + \frac{\partial \boldsymbol{V}_z}{\partial z} = \operatorname{tr}(\nabla \boldsymbol{V}).$$

对于标量值函数 u，可以证明有下面的性质

$$\begin{aligned}
\langle \nabla, u\boldsymbol{V}\rangle &= \frac{\partial u\boldsymbol{V}_x}{\partial x} + \frac{\partial u\boldsymbol{V}_y}{\partial y} + \frac{\partial u\boldsymbol{V}_z}{\partial z} \\
&= \left[\frac{\partial u}{\partial x}\boldsymbol{V}_x + u\frac{\partial \boldsymbol{V}_x}{\partial x}\right] + \left[\frac{\partial u}{\partial y}\boldsymbol{V}_y + u\frac{\partial \boldsymbol{V}_y}{\partial y}\right] + \left[\frac{\partial u}{\partial z}\boldsymbol{V}_z + u\frac{\partial \boldsymbol{V}_z}{\partial z}\right] \\
&= \left[\frac{\partial u}{\partial x}\boldsymbol{V}_x + \frac{\partial u}{\partial y}\boldsymbol{V}_y + \frac{\partial u}{\partial z}\boldsymbol{V}_z\right] + u\left[\frac{\partial \boldsymbol{V}_x}{\partial x} + \frac{\partial \boldsymbol{V}_y}{\partial y} + \frac{\partial \boldsymbol{V}_z}{\partial z}\right] = \langle \nabla u, \boldsymbol{V}\rangle + u\langle \nabla, \boldsymbol{V}\rangle
\end{aligned}$$

通过该运算法则，就有向量场的分部积分公式。Ω 是 $\mathbb{R}^n$ 中的一个有界开集，其边界 $\Gamma = \partial\Omega$ 分段光滑，有

$$\int_\Omega \langle \nabla, u\boldsymbol{V}\rangle \mathrm{d}\Omega = \int_\Gamma u\langle \boldsymbol{V}, \hat{\boldsymbol{n}}\rangle \mathrm{d}\Gamma = \int_\Omega \langle \nabla u, \boldsymbol{V}\rangle \mathrm{d}\Omega + \int_\Omega u\langle \nabla, \boldsymbol{V}\rangle \mathrm{d}\Omega.$$

在 Score matching 的推导中，由于积分区域是全空间，并默认概率密度函数在无穷远处的极限为零，应用分部积分公式即可。

参考资料

1. https://www.jhanmath.com/?p=142