2025 年 10 月 27 日至 11 月 2 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

本周将 CNN-small 的层数从三层降低至两层，在 8 epoch 内可以成功收敛，其权重待分析。

1.1.1. 下一步

目前将目标从提取显式的规则改为从已知部分规则和黑箱规则代理产生的少量数据，结合带有等变性的神经网络训练作为动力学仿真器。规则发现的具体流程是，从一些已知规则出发，首先根据一条有偏的轨道从一个规则空间中选取一系列的假设规则，在神经网络学习这条轨道后，再通过神经网络作为演化模拟器的演化统计结果，和原来的统计数据对比，以确定某些候选规则存在或不存在，从而逐步确定系统演化的真正动力学。

2. 文献阅读

2.1. Denoising Diffusion Probabilistic Models [1]

Jonathan Ho, Ajay Jain and Pieter Abbeel | https://arxiv.org/abs/2006.11239

本周把 DDPM 的剩余部分补完。

2.1.1. 实验

2.1.1.1. 预测目标和目标函数匹配程度

实验发现，预测 $\tilde{\boldsymbol{\mu}}$ 需要匹配未化简形式的加权期望损失；而预测噪声 $\boldsymbol{\varepsilon}$ 需要匹配简化的损失函数。这两个搭配效果几乎一样好。

2.1.1.2. 去噪过程的协方差矩阵

如果将去噪过程的协方差矩阵变成科学系的对角形式，训练过程会变得不稳定，采样得到的图像质量也会下降。

2.1.2. DDPM 的信息论视角

2.1.2.1. 编码效率和失真度

模型在 CIFAR 上训练得到的对应于每一数据维度的信息比特数相差 0.03，这说明模型没有在训练集上过拟合。此外，再回到目标函数，已知有这样的分解：

$$\mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}[q(\boldsymbol{x}_T|\boldsymbol{x}_0)|p(\boldsymbol{x}_T)]}_{L_T}+\sum_{t=2}^{T}\underbrace{D_{\mathrm{KL}}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)]}_{L_{t-1}}\underbrace{-\log p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)}_{L_0}\right] \tag{1}$$

注意 $L_1+\cdots+L_{T-1}$ 中的每一项都是 KL 散度，分别描述神经网络参数化的 $p_{\boldsymbol{\theta}}$ 和分布 q 的区别，它是使用 $p_{\boldsymbol{\theta}}$ 描述 q 需要的额外编码开销（见附录），因此可将其看做生成过程中的**编码效率 (rate)**，这体现在相对于该体系中认为的最优逆向过程 q 而言所需的额外编码量；而最后一项 L_0 是一个负对数似然，它将 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_0$ 对齐比较，我们可将其看做从 $\boldsymbol{x}_T$ 经过降噪过程一路走来的结果相比于 $\boldsymbol{x}_0$ 的**失真度 (distortion)**。

2.1.2.2. 分步有损压缩的信道模型

我们可以将 DDPM 的去噪过程看作是发送端向接受端发送信息。接收端在接收前只知道 p，发送端可以同步逐步使用 $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$ 编码服从分布 $q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1},\boldsymbol{x}_0)$ 的数据 $\boldsymbol{x}_t$，然后接收端用 p_θ 对其进行解码。如果发送端仅发送 x_T，接收端仅依靠此信息和 p_θ 估计 $\hat{\boldsymbol{x}}_0$，这样会产生较高的失真度。

Algorithm 1: Sending x_0

procedure SENDING($\boldsymbol{x}_0$)
  Sendin $\boldsymbol{x}_T\sim q(\boldsymbol{x}_T|\boldsymbol{x}_0)$ using $p(\boldsymbol{x}_T)$
  for $t=T-1,\ldots,2,1$ **do**
    Send $\boldsymbol{x}_t\sim q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1},\boldsymbol{x}_0)$ using $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$
  end
  Send $\boldsymbol{x}_0$ using $p_\theta(\boldsymbol{x}_0|\boldsymbol{x}_1)$
end

Algorithm 2: Recieving

procedure RECIEVING()
  Receive $\boldsymbol{x}_T$ using $p(\boldsymbol{x}_T)$
  for $t=T-1,\ldots,1,0$ **do**
    Receive $\boldsymbol{x}_t$ using $p_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$
  end
  return $\boldsymbol{x}_0$
end

每个时刻 t，接收端都会计算估计得到的 $\hat{\boldsymbol{x}}_0$ 和真实值的均方损失作为失真率，而记录自传输开始至该时刻接收器获得的所有每维度比特数为码率：即 $H(\boldsymbol{x}_t)+D_{\mathrm{KL}}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)]$，画出的码率-失真率曲线可以看出，大量的信息被分配至肉眼难以看见的细节中：

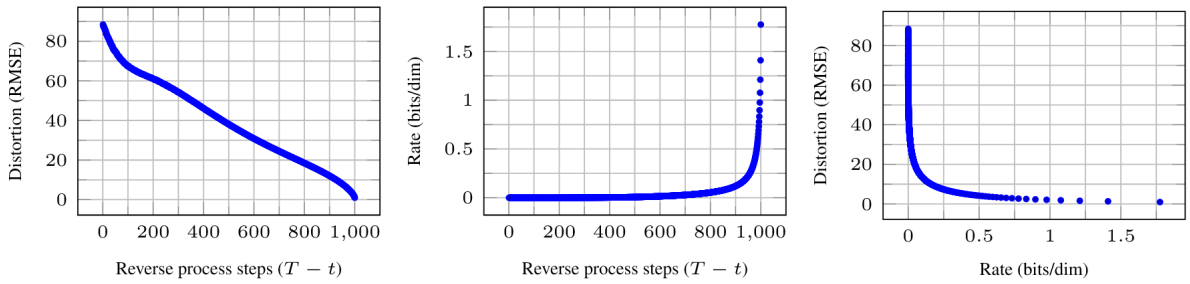

Figure 1: 信道模型中随迭代次数接收器对 $\boldsymbol{x}_0$ 预测的失真率和码率的关系

2.1.2.3. 分步生成

作者还用各部得到的 $\boldsymbol{x}_t$ 直接预测 $\boldsymbol{x}_0$，得到的结果由下图所示。可见随着 t 的减小，预测得到的 $\hat{\boldsymbol{x}}_0$ 逐步先显现总体特征，再逐步丰富局部细节。

Figure 2: CIFAR10 数据集上随 t 减小从 $\boldsymbol{x}_t$ 预测得到的 $\hat{\boldsymbol{x}}_0$ 结果，从左至右 t 逐渐降低

如果使模型都从某个共用的 $\boldsymbol{x}_t$ 出发执行降噪过程，越大的 t 出发降噪得到的结果差异越大；越小的 t 出发得到的降噪结果越相似。

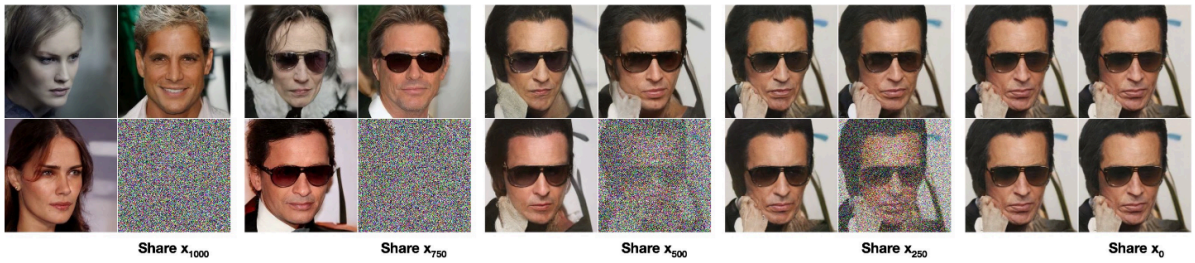

Figure 3: CelebA-HQ 数据集从同一个 $\boldsymbol{x}_t$（每个子图的右下角）出发降噪得到的三个结果

2.1.3. DDPM 和自回归生成模型的比较

最后讨论 DDPM 的降噪过程和序贯生成的自回归生成模型（例如 ChatGPT 的模式）的相似性和区别。我们考虑这样的场景（暂时抛开加噪过程是加 Gauss 过程这件事）：生成图像的尺寸是 $N \times N$，所谓的“加噪过程”和“降噪过程”的步数为 N^2，$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$ 是一个离散的 dirac-delta 点质量，它做的事情是将图像从左至右，从上至下的第 t 个像素变成空白；相应地，$q(\boldsymbol{x}_{t-1})$ 做的事情就是将第 t 个像素恢复成原来的颜色。这样经过一整条“加噪过程”后，图像就变成了完全空白；再经过理想的“降噪过程”后图像又被逐像素涂色成了原来的样子。

这是我们假设神经网络建模的 p_θ，足够强大——它可以学习到上面假设中 $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ 的分布，那么这俨然成了自回归模型。然而实践过程中，**DDPM 选择的使用 Gauss 噪声加噪、去噪的过程，可以看作是更为广义的自回归生成模型，它按照图像中的某种空间语义 token 按照从整体至局部的顺序逐渐生成图像**。相比于添加更强归纳偏置的一般自回归生成模型——它们都假设后生成的区域依赖于先生成的区域，而这在图像生成上没有道理！——DDPM 拥有更高的灵活性和更少的归纳偏置，这也解释了其优异的生成能力。

2.1.4. 隐空间插值

最后，作者将不同图像加噪过程中某个 t（例如 $t = 500$）得到的 $\boldsymbol{x}_t$ 和 $\hat{\boldsymbol{x}}_t$ 做凸组合，再做去噪过程，得到了原图 $\boldsymbol{x}_0$ 和 $\hat{\boldsymbol{x}}_0$ 之间的顺滑渐变。

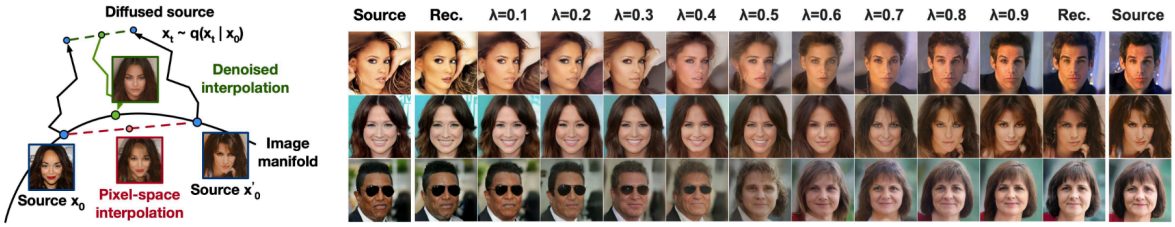

Figure 4: CelebA-HQ 数据集中的插值实验

2.2. Mean Flows for One-step Generative Modeling [2]

Zhengyang Geng et al. | https://doi.org/10.48550/arXiv.2505.13447

2.2.1. 流匹配

2.2.1.1. 流轨迹

流轨迹是指从先验分布 p_{prior} 连续地到数据分布 p_{data} 的“流动”轨迹：对于 $x \sim p_{\text{data}(x)}$ 和与之配对的某个 $\varepsilon \sim p_{\text{prior}(\varepsilon)}$，它们之间随时间 t 相互演变的轨迹可以是

$$z_t = a_t x + b_t \varepsilon \tag{2}$$

其中 a_t 和 b_t 都是预先定义好的确定函数。常用的流轨迹是 $z_t = (1-t)x + t\varepsilon$，当时间 t 从 0 逐渐增加至 1 时，数据点从 x 逐渐流动至 ε，反之亦然。

对给定的流轨迹和 p_{data} 中的数据点 x，定义时刻 t 时的条件流场 v_t 为 z_t 对时间的导数：

$$v_t = a_t' x + b_t' \varepsilon \tag{3}$$

在上节的常用例子 $z_t = (1-t)x + t\varepsilon$ 中，流速为 $v_t = \varepsilon - x$。

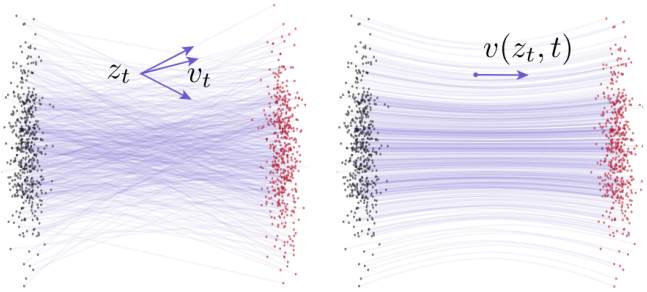

Figure 5: 条件流场（左）和边缘流场（右）

2.2.2. 流匹配

条件流速依赖于 x，如果将 x 平均掉，就得到边缘流场

$$v(z_t, t) := \mathbb{E}_{p_t(v_t \mid z_t)}[v_t] \tag{4}$$

它考虑的是指轨迹在 t 时刻交汇于 v_t 的不同情况速度的加权平均。考虑边缘流场的意义在于其普适性：由于其不依赖于 x 的选取，我们可以直接从先验分布中采样一个 ε，然后求解下面的初值问题：

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} z_t &= v(z_t, t), \qquad t \in [0, 1] \\ z_1 &= \varepsilon \end{aligned} \tag{5}$$

显然，要得到 z_r，只需要做积分：

$$z_r = z_1 - \int_r^1 v(z_\tau, \tau)\mathrm{d}\tau \tag{6}$$

然而边缘流场我们不知道，因此就造一个神经网络来学它：最小化下面的损失

$$J_{\text{FM}}(\theta) = \mathbb{E}_{t, p_t(z_t)} \| v_\theta(z_t, t) - v(z_t, t) \|^2. \tag{7}$$

不过由于 $v(z_t,t)$ 不依赖 x，我们在不知道 p_{data} 的时候根本没法算它，所以实际情况下我们借用条件流场采样到的值，然后对起点 x、终点 ε 和时间 t 做平均，得到下面条件流场版本的边缘流场模型的目标：

$$J_{\text{CFM}}(\theta)=\mathbb{E}_{t,x,\varepsilon}\|v_\theta(z_t,t)-v(z_t\mid x)\|^2 \tag{8}$$

二者已被证明是等价的。类似分数匹配 (score matching)，这里的网络尝试拟合 $v(z_t,t)$，因此谓之流匹配。有了神经网络的流模型后，采样就变成了求上面那个 ODE 的数值解。

2.2.3. 平均流

本文的主要贡献——流模型使用了一个更一般化的平均流。顾名思义，它是从 τ 时刻起在 t 时刻汇聚于 z_t 之边缘（平均化了的）轨迹上流速的平均：

$$u(z_t,r,t):=\frac{1}{t-r}\int_r^t v(z_\tau,\tau)\mathrm{d}\tau \tag{9}$$

若固定 t，令 $r\to t$，可得 $u(z_t,r,t)\to v(z_t,t)$，因此它和原来的边缘流场的定义也是相容的。除此之外，由于位移可以写为平均速度乘时间，平均流还有另一层次的一致性：跨一大步 $[r,t]$ 相当于跨两小步：$[r,s],[s,t]$。根据平均速度定义不难验证

$$(t-r)u(z_t,r,t)=(s-r)u(z_s,r,s)+(t-s)u(z_t,s,t). \tag{10}$$

为什么要引入平均流？上文中，我们规定 0 时刻对应数据分布中的元素 x，1 时刻对应先验分布中的元素 ε，如果令 $r=0$，$t=1$，就得到

$$u(\varepsilon,0,1)=1\cdot u(z_1,0,1)=\int_0^1 v(z_\tau,\tau)\mathrm{d}\tau=z_1-z_0=\varepsilon-z_0 \tag{11}$$

因此可以进行**一步生成 (single Number of Function Evaluations, single-NFE)**：

$$x=u(\varepsilon,0,1). \tag{12}$$

2.2.4. 平均流的一些性质

如果假设 r 是一个与 t 不相关的独立常数，将时间 r 到 t 的总位移对 t 求导，得到

$$\frac{\mathrm{d}}{\mathrm{d}t}(t-r)u(z_t,r,t)=\frac{\mathrm{d}}{\mathrm{d}t}\int_r^t v(z_\tau,\tau)\mathrm{d}\tau\Longrightarrow u(z_t,r,t)=v(z_t,t)-(t-r)\frac{\mathrm{d}}{\mathrm{d}t}u(z_r,r,t). \tag{13}$$

对于最后一项，有

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u&=\frac{\mathrm{d}z_t}{\mathrm{d}t}\partial_z u+\frac{\mathrm{d}r}{\mathrm{d}t}\partial_r u+\frac{\mathrm{d}t}{\mathrm{d}t}\partial_t u\\&=v(z_t,t)\partial_z u+\partial_t u\end{aligned} \tag{14}$$

可以看出这是 u 的 Jacobian 和向量 $[v,0,1]^\top$ 的乘积，它可以通过 `torch.func.jvp` 方便实现。有了这两个等式，联立后可以得到

$$u(z_t,r,t)=v(z_t,t)-(t-r)[v(z_t,t)\partial_z u+\partial_t u]. \tag{15}$$

2.2.5. 训练

我们发现直接拿着上面的式子当匹配的目标还是不行，因为等式右边还存在着 u。因此我们可以考虑自助法 **(bootstrapping)**，即在等式右边使用 u_θ 作为自助法的目标：

$$
\begin{aligned}
J(\theta) &= \mathbb{E}\left[\left\| u_{\theta(z_t,r,t)} - \text{StopGrad}\left(u_{\text{target}}\right)\right\|_2^2\right] \\
\text{in which } u_{\text{target}} &= v(z_t,t) - (t-r)[v(z_t,t)\partial_z u_\theta + \partial_t u_\theta]
\end{aligned}
\tag{16}
$$

为了以免让优化过程变得复杂，我们需要停止目标项中的梯度传播，即使用 StopGrad$(\cdot)$ 操作。特别地，如果 $r \equiv t$，上面的平均流匹配就退化成了一般形式的流匹配。

2.2.6. 带引导的平均流

有趣的是，对于条件生成，本文对其的处理和前文中的流匹配极其相似。其核心方法是无分类器引导 **(Classifier-Free Guidance, CFG)**。给定类别 c，定义一个实况场 **(ground-truth field)** v^{cfg} 为

$$
\begin{aligned}
v^{\text{cfg}}(z_t,t \mid c) &:= \omega v(z_t,t \mid c) + (1-\omega)v(z_t,t) \\
\text{in which } v(z_t,t \mid c) &:= \mathbb{E}_{p_t(v_t \mid z_t,c)}[v_t]; \\
v(z_t,t) &:= \mathbb{E}_c[v(z_t,t \mid c)]
\end{aligned}
\tag{17}
$$

它是类条件速度场和类边缘速度场的线性组合。如果引入 v^{cfg} 对应的平均流 u^{cfg}，根据上面的结果，依然有

$$
u(z_t,r,t) = v(z_t,t) - (t-r)\frac{\mathrm{d}}{\mathrm{d}t}u(z_r,r,t) \tag{18}
$$

此时稍加一些推导，v^{cfg} 可重写为

$$
v^{\text{cfg}}(z_t,t \mid c) := \omega v(z_t,t \mid c) + (1-\omega)u^{\text{cfg}}(z_t,t,t). \tag{19}
$$

这样情境下训练的目标和先前的目标大同小异：

$$
\begin{aligned}
J(\theta) &= \mathbb{E}\left\| u_\theta^{\text{cfg}}(z_t,r,t \mid c) - \text{StopGrad}\left(u_{\text{target}}\right)\right\|^2 \\
\text{in which } u_{\text{target}} &= \tilde{v} - (t-r)\left[\tilde{v}\partial_z u_\theta^{\text{cfg}} + \partial_t u_\theta^{\text{cfg}}\right] \\
\tilde{v} &= \omega v_t + (1-\omega)u_\theta^{\text{cfg}(z_t,t,t)}
\end{aligned}
\tag{20}
$$

可见当 $\omega = 1$ 时，上述目标退化为先前的无类别生成的目标。在训练带有类别引导的目标时，我们在数据中以一定概率丢弃类别标签，以同时训练类条件和无类条件的版本。其生成流程也和先前一样简单：如果不指定类别 c，那么生成的结果是无类别标签的，如果需要生成某一类的结果，只需在参数中添加所需的类别 c。

2.2.7. 实验

消融实验发现

1. 采用平均流而不是流匹配一般的配置的训练结果更好
2. `jvp` 模块中参与乘法的向量必须要是正确的
3. 对于平均流的函数形式，$u(z_t,t,t-r)$ 相比于 $u(z_t,t,r)$ 更好，但先前的推导需要修正
4. t，r 自特定参数的对数正态分布中采样效果更好

% of $r \neq t$	FID, 1-NFE
0% (= FM)	328.91
25%	**61.06**
50%	63.14
100%	67.32

(a) **Ratio of sampling** $r \neq t$. The 0% entry reduces to the standard Flow Matching baseline.

jvp tangent	FID, 1-NFE
$(v, 0, 1)$	**61.06**
$(v, 0, 0)$	268.06
$(v, 1, 0)$	329.22
$(v, 1, 1)$	137.96

(b) **JVP computation**. The correct `jvp` tangent is $(v, 0, 1)$ for Jacobian $(\partial_z u, \partial_r u, \partial_t u)$.

pos. embed	FID, 1-NFE
(t, r)	61.75
$(t, t-r)$	**61.06**
$(t, r, t-r)$	63.98
$t-r$ only	63.13

(c) **Positional embedding.** The network is conditioned on the embeddings applied to the specified variables.

t, r sampler	FID, 1-NFE
uniform(0, 1)	65.90
lognorm(–0.2, 1.0)	63.83
lognorm(–0.2, 1.2)	64.72
lognorm(–0.4, 1.0)	**61.06**
lognorm(–0.4, 1.2)	61.79

(d) **Time samplers.** t and r are sampled from the specific sampler.

p	FID, 1-NFE
0.0	79.75
0.5	63.98
1.0	**61.06**
1.5	66.57
2.0	69.19

(e) **Loss metrics.** p=0 is squared L2 loss. p=0.5 is Pseudo-Huber loss.

ω	FID, 1-NFE
1.0 (w/o cfg)	61.06
1.5	33.33
2.0	20.15
3.0	**15.53**
5.0	20.75

(f) **CFG scale:**. Our method supports 1-NFE CFG sampling.

Figure 6: Mean Flows 模型在 *ImageNet* 256 × 256 上训练后一步生成的消融实验

得益于其一步生成的快速和低成本，Mean Flow 模型可以以更低的计算代价达到个更好的效果：

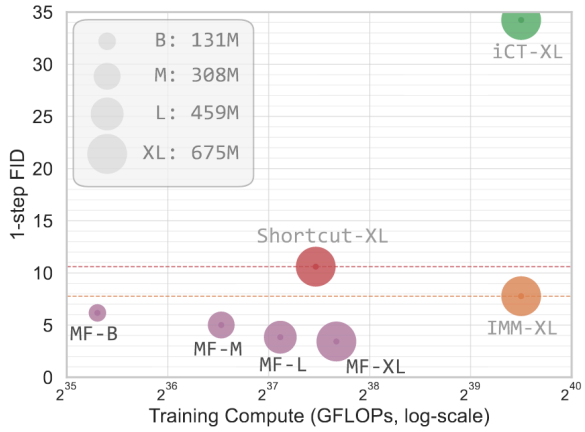

Figure 7: 不同体量的 *Mean Flows* 模型在 *ImageNet* 256 × 256 上与其他一步生成模型的比较

3. 学习进度

3.1. 随机过程

本周开始系统学习 Markov 链。

3.2. 随机微分方程

3.2.1. 随机微分方程的数值模拟

考虑一个自守的 SDE

$$\mathrm{d}X = F(X)\mathrm{d}t + G(X)\mathrm{d}\boldsymbol{W} \tag{21}$$

下面给出模拟它的三种方法。

3.2.1.1. Euler-Maruyama 法

Euler-Maruyama 法主要思想是 Brown 运动 $W(\cdot)$ 的增量独立性，即 $W(t+\Delta t) - W(t) \sim \mathcal{N}(0,1)$，因此有

$$X(t+\Delta t) \approx X(t) + F(X)\Delta t + \sqrt{\Delta t}\cdot G(X)Z \tag{22}$$

其中 $Z \sim \mathcal{N}(0,1)$

3.2.1.2. Milstein 法

Milstein 法相比于 Euler-Maruyama 法更细，来源于对 $\displaystyle\int_t^{t+\Delta t} G(X)\mathrm{d}W$ 更精确的估计。其形式为

$$\begin{aligned} X(t+\Delta t) \approx X(t) + F(X)\Delta t + \sqrt{\Delta t}\cdot G(X)Z \\ +\frac{1}{2}G(X)\frac{\partial}{\partial X}G(X)[Z^2 t - (\Delta t)^2] \end{aligned} \tag{23}$$

其推导我还需要再看一下。

3.2.1.3. 随机 Runge-Kutta 法

3.3. 信息论

3.3.1. 熵

给定一个分布 $p(x)$，它的熵定义为

$$H(p) \coloneqq \mathbb{E}_p[\log p(x)] = \int p(x)\log p(x)\mathrm{d}x \tag{24}$$

它在信息论中的意义分布 p 下编码 x 所需的最小平均比特数

3.3.2. 交叉熵

类似地，给定分布 $p(x)$ 和 $q(x)$，其互信息 $H(p,q)$ 定义为

$$H(p,q) \coloneqq \mathbb{E}_p[\log q(x)] = \int p(x)\log q(x)\mathrm{d}x \tag{25}$$

它表示使用基于分布 q 的最短编码方案编码服从分布 p 的数据的平均比特数

3.3.3. KL 散度

KL 散度一般用于描述分布之间的差异，给定分布 $p(x)$ 和 $q(x)$，有

$$D_{\mathrm{KL}}[p \parallel q] = \mathbb{E}_p\left[\log \frac{p(x)}{q(x)}\right] = H(p,q) - H(p) \tag{26}$$

它表示使用基于分布 q 的最短编码方案编码服从分布 p 的数据，相比于使用 p 的最短编码方式造成的平均额外的比特数开销。

4. 下周计划

论文阅读

1. 生成模型
   - Score Matching 和 SDE（完结）
   - 薛定谔桥（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
   - 阅读等变 CNN 的代码
   - 尝试开始做单轨道训练样本的生成、训练和演化动力学统计
2. 耦合约瑟夫森结
   - 了解约瑟夫森结的基本知识
   - 浏览 matlab 数值模拟代码

理论学习

1. 随机过程课程
   - 学习完毕 Markov 过程
2. 随机微分方程
   - 完成第四章 随机积分
   - 第五章 随机微分方程 开头

参考文献

[1] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” *CoRR*, 2020, [Online]. Available: https://arxiv.org/abs/2006.11239

[2] Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He, “Mean Flows for One-step Generative Modeling,” *CoRR*, 2025, doi: 10.48550/ARXIV.2505.13447.