Lec02 Flow and Diffusion Models

本节中，我们将描述如何通过模拟一个适当构造的微分方程来获得所需的转换。例如，流匹配和扩散模型分别涉及模拟常微分方程（ODE）和随机微分方程（SDE）。因此，本节的目标是定义和构建这些生成模型。具体来说，我们首先**定义 ODE 和 SDE**，并讨论它们的模拟。其次，我们描述如何**使用深度神经网络对 ODE/SDE 进行参数化**。从中推导出流模型和扩散模型以及从这些模型中采样的基本算法的定义。在后面的章节中，我们将探讨如何训练这些模型。

2.1 流模型

我们首先定义**常微分方程 (Ordinary Differential Equations ODEs)**。常微分方程的解由**轨迹 (Trajectory)**定义，即以下形式的函数：

$$X : [0, 1] \to R^d,\ t \mapsto X_t$$

- 该函数将时间$t \in [0, 1]$映射到空间R中的某个位置。

每个常微分方程都由一个**矢量场 (Vector Field)** u 定义，即以下形式的函数：

$$u : R^d \times [0, 1] \to R^d,\ (x, t) \mapsto u_t(x)$$

- 即对于每个事件t和位置x，我们都得到一个向量$u_t(x) \in R^d$，它指定了空间中的速度。

常微分方程对轨迹施加了一个条件：我们希望轨迹X沿着向量场u_t的“线条“进行，并在$t = 0$时刻从点x_0开始。我们可以将此类轨迹形式化为方程的解：

$$\frac{d}{dt} X_t = u_t(X_t)$$

$$X_0 = x_0$$

- 上面一个式子要求X_t的导数由u_t给出的方向指定。下面一个式子要求我们在$t = 0$的时候从点x_0开始。

现在我们可能会问：如果我们在$t = 0$时刻从$X_0 = x_0$出发，那么在时间t的时候我们在哪里（即X_t）？ 这个问题的由一个称为**流 (Flow)**的函数来回答，它是ODE的解：

$$\psi : R^d \times [0, 1] \mapsto R^d,\ (x_0, t) \mapsto \psi_t(x_0)$$

$$\frac{d}{dt} \psi_t(x_0) = u_t(\psi_t(x_0))$$

$$\psi_0(x_0) = x_0$$

对于初始条件$X_0 = x_0$，我们通过$X_t = \psi_t(X_0)$来恢复常微分方程的轨迹。因此，向量场、常微分方程和流直观上是同一对象的三种描述方法：**向量场定义了常微分方程，其解是轨迹，流是针对各种初始条件的轨迹的集合。**与其他的每个方程一样，我们应该问自己关于常微分方程（ODE）的问题：是否存在解，如果是的话，它是否唯一？

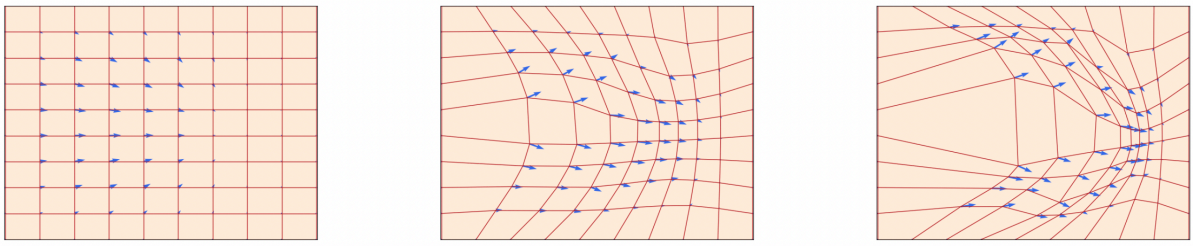


Figure 1: A flow $\psi_t : \mathbb{R}^d \to \mathbb{R}^d$ (red square grid) is defined by a velocity field $u_t : \mathbb{R}^d \to \mathbb{R}^d$ (visualized with blue arrows) that prescribes its instantaneous movements at all locations (here, $d = 2$). We show three different times t. As one can see, a flow is a diffeomorphism that "warps" space. Figure from [15].

基本回答是"Yes!"，我们只需要对u_t施加弱假设：

定理：流的存在性和唯一性

如果$u : R^d \times [0,1] \to R^d$是连续可微且导数有界，则该ODE有唯一解，该解由流ψ_t给出。在此情况下，ψ_t对与任意t都是微分同胚，即ψ_t是连续可微的且ψ_t^{-1}同样连续可微。

注意，对于流的存在性和唯一性所需的假设在机器学习中几乎总是得到满足，因为我们使用神经网络来参数化$u_t(x)$，并且它们总是有有界的导数。因此在我们感兴趣的情况下，流永远存在，并且是常微分方程的唯一解。

举例：线性ODE

简单的向量场$u_t(x) = -\theta x$ $(\theta > 0)$，则流的解析式为$\psi_t(x_0) = \exp(-\theta t)x_0$

证明：

1. *初始条件：$\psi_{t=0}(x_0) = \exp(0)x_0 = x_0$*
2. *ODE：$\frac{d}{dt}\psi_t(x_0) = -\theta \exp(-\theta t)x_0 = -\theta\psi_t(x_0) = u_t(\psi_t(x_0))$*

模拟常微分方程：一般来说，如果u_t不是像线性函数那么简单，就无法显式地计算流ψ_t。在这种情况下，人们使用数值方法(Numerical Methods)来模拟常微分方程。这是一个经典且经过深入研究的话题，存在许多强大的方法。其中最简单、最直观的方法是**欧拉法 (Euler Method)**。在欧拉法中，我们初始化$X_0 = x_0$，并且通过以下方式更新：

$$X_{t+h} = X_t + hu_t(X_t) \quad (t = 0, h, 2h, 3h, \ldots, 1-h)$$

- 其中$h = n^{-1} > 0$，$n \in N$是一个步长超参数。

我们也可以尝试更复杂的**Heun's Method**，其通过以下更新法则定义：

$$X'_{t+h} = X_t + hu_t(X_t)$$

$$X_{t+h} = X_t + \frac{h}{2}(u_t(X_t) + u_{t+h}(X'_{t+h}))$$

- 直观地说，Heun's Method首先猜测下一步X'_{t+h}可能是什么，但通过更新猜测来纠正最初的方向。

流模型：我们现在可以通过一个常微分方程构建一个生成模型。记住，我们的目标是把一个简单分布p_{init}转换成一个复杂的分布p_{data}。因此，模拟ODE是这种转换的自然选择，**流模型**由以下ODE描述：

$$X_0 \sim p_{init}$$

$$\frac{d}{dt}X_t = u_t^\theta(X_t)$$

- 其中向量场u_t^θ是一个带有参数θ的神经网络。

目前，我们将u_t^θ视为一个通用的神经网络。稍后，我们将会讨论特定神经网络架构的选择。我们的目标是使轨迹的终点X_1具有分布p_{data}，即：

$$X_1 \sim p_{data} \iff \psi_1^\theta(X_0) \sim p_{data}$$

- 其中ψ_1^θ描述了由u_t^θ诱导的流。

请注意：虽然被称为流模型，但是**神经网络参数化的是向量场**，而不是流。为了计算流，我们需要模拟ODE。在算法1中，我们总结了从流模型中采样的方法：

Algorithm 1 Sampling from a Flow Model with Euler method

Require: Neural network vector field u_t^θ, number of steps n
1: Set $t = 0$
2: Set step size $h = \frac{1}{n}$
3: Draw a sample $X_0 \sim p_{\text{init}}$
4: **for** $i = 1, \dots, n-1$ **do**
5: $\quad X_{t+h} = X_t + h u_t^\theta(X_t)$
6: $\quad$ Update $t \leftarrow t + h$
7: **end for**
8: **return** X_1

2.2 Diffusion Models

随机微分方程 (SDEs)扩展了**常微分方程 (ODEs)**的确定性轨迹，具有**随机轨迹**，随机轨迹通常被称为**随机过程 (Stochastic)** $(X_t)_{0 \le t \le 1}$

其由下式给出：

$$\forall t \in [0,1], X_t \text{ is a random variable}$$

$$X : [0,1] \to R^d,\ t \mapsto X_t \text{ is a random trajectory for every draw of X}$$

尤其是在模拟相同的随机过程两次时，我们可能会得到不同的结果，因为动力学被设计成随机的。

布朗运动：通过布朗运动构建SDEs——这是一个源于物理扩散过程研究的基本随机过程。可以把布朗运动想象成一个连续的随机游走。

- 定义：布朗运动$W = (W_t)_{0 \le t \le 1}$是一个随机过程，满足$W_0 = 0$，轨迹$t \mapsto W_t$连续，并且满足以下两个条件：
  1. **Normal Increments:** 增量是一个**方差随着时间线性增长的高斯分布**

$$\forall s \in [0,t), W_t - W_s \sim \mathcal{N}(0, (t-s) I_d)$$

     - 其中I_d是单位矩阵。
  2. **Independent Increments:**

$$0 \le t_0 < t_1 < \cdots < t_n = 1$$

$$W_{t_1} - W_{t_0}, \dots, W_{t_n} - W_{t_{n-1}} \text{ are independent random variables}$$

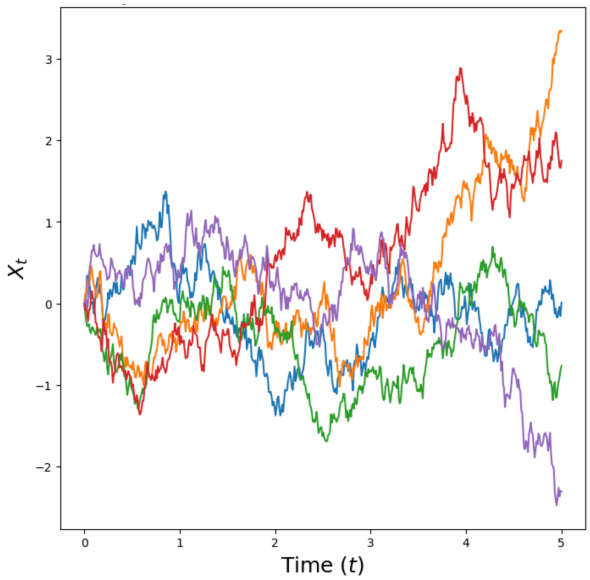
布朗运动也被称为**Wiener Process**，这就是为什么我们用$W_.$来表达。

我们可以用以下方式大致模拟一个布朗运动：

$$h > 0, W_0 = 0, \epsilon_t \sim \mathcal{N}(0, I_d) \quad (t = 0, h, 2h, \dots, 1-h)$$

$$W_{t+h} = W_t + \sqrt{h}\epsilon_t$$

下图展示了利用上式模拟的维度为 1 时的布朗运动的轨迹：



布朗运动对于随机过程的研究就像高斯分布对于概率分布的研究一样核心。从金融到统计物理再到流行病学，布朗运动的研究在机器学习之外有着广泛的应用。例如，在金融领域，布朗运动被用来模拟复杂金融工具的价格。同样，作为一个数学构造，布朗运动也非常迷人：例如，虽然布朗运动的路径是连续的，但它们是无限长的。

从常微分方程到随机微分方程。随机微分方程（SDE）的想法是通过添加由布朗运动驱动的随机动力学来扩展常微分方程的确定性动力学。因为一切都是随机的，我们可能不再能像以前那样求导。因此，我们需要找到一个**不使用导数的ODEs的等效公式**，我们可以如下重写ODEs的轨迹$(X_t)_{0\le t\le 1}$：

$$\frac{d}{dt}X_t = u_t(X_t)$$

$$\Leftrightarrow \frac{1}{h}(X_{t+h} - X_t) = u_t(X_t) + R_t(h)$$

$$\Leftrightarrow X_{t+h} = X_t + hu_t(X_t) + hR_t(h)$$

- 其中R_t表示一个对于小量h可以忽略的函数，即$\lim_{h\to 0} R_t(h) = 0$。

上述推导只是简单地重申了我们已知的内容：常微分方程的轨迹$(X_t)_{0\le t\le 1}$在每一个时间步长都沿着方向$u_t(X_t)$迈出一个小步。现在我们可以修改最后一个方程，使其变为随机过程：随机微分方程的轨迹$(X_t)_{0\le t\le 1}$在每一个时间步长都沿着方向$u_t(X_t)$迈出一个小步，并加上来自布朗运动的某些贡献：

$$X_{t+h} = X_t + \underbrace{hu_t(X_t)}_{\text{deterministic}} + \sigma_t \underbrace{(W_{t+h} - W_t)}_{\text{stochastic}} + \underbrace{hR_t(h)}_{\text{error term}}$$

- 其中$\sigma_t \ge 0$是**扩散系数**，$R_t(h)$是随机误差项，其服从$\lim_{h\to 0} \sqrt{\mathbf{E}[\|R_t(h)\|^2]} \to 0$

上式描述了一个**随机微分方程 (SDE)**。通常用以下符号表示：

$$dX_t = u_t(X_t)dt + \sigma_t dW_t$$

$$X_0 = x_0$$

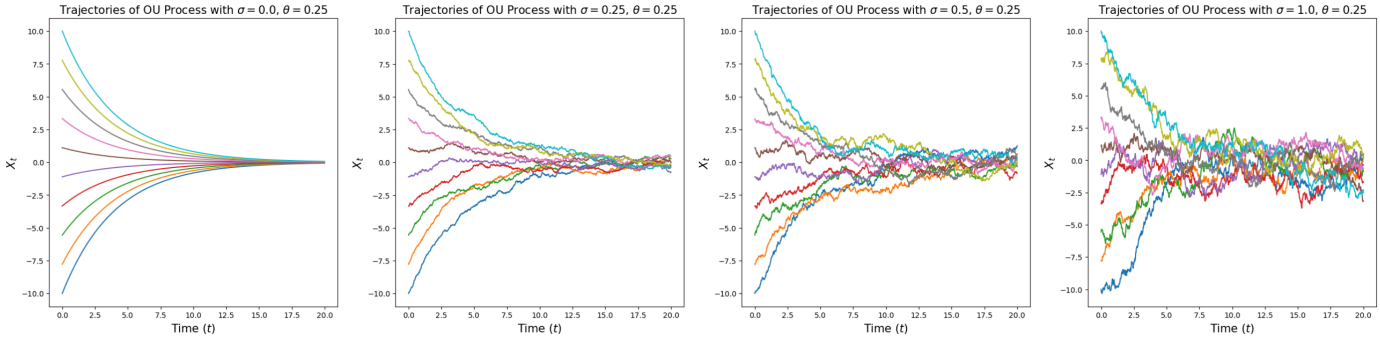
然而，始终要记住上面的dX是方程(14)的纯非正式表示，并且方程(14)中的h在此处用dt表示。

不幸的是，随机微分方程不再有流映射ϕ_t。这是因为X_t的值不再完全由$X_0 \sim p_{init}$决定，因为其演化本身是随机的。尽管如此，与常微分方程一样，我们有：

定理：SDE的解的存在性和唯一性

如果$u : R^d \times [0,1] \to R^d$是连续可微且导数有界的，且$\sigma_t$是连续的，那么式$(15)$中的SDE有解，解为满足方程$(14)$的唯一随机过程$(X_t)_{0\le t\le 1}$

实际上每个ODE只不过是SDE的一种特殊情况，即$\sigma_t = 0$，因此当我们讨论SDE的时候，我们将ODE视为一种特殊情况。



Ornstein-Uhlenbeck (OU) Process

考虑一个由常微分线性方程$u_t(x) = -\theta x, \theta > 0$和一个常数扩散系数$\sigma_t = \sigma \ge 0$构成的SDE：

$$dX_t = -\theta X_t dt + \sigma dW_t$$

上述SDE的一个解$(X_t)_{0\le t\le 1}$被称为Ornstein-Uhlenbeck (OU) Process，其可视化过程如上图所示。

模拟随机微分方程：如果你觉得SDEs的概念太抽象了，不用担心。一个更直观的思考SDEs的方法是思考如何模拟SDE。最简单的方法被称为**Euler-Maruyama Method**，它对于SDEs的作用就像欧拉方法对于ODEs的作用。利用Euler-Maruyama方法，我们初始化$X_0 = x_0$并且通过以下公式递推更新：

$$X_{t+h} = X_t + hu_t(X_t) + \sqrt{h}\sigma_t \epsilon_t, \;\; \epsilon_t \sim \mathcal{N}(0, I_d)$$

- 其中$h = n^{-1} > 0, \; n \in N$是步长超参数。

换句话说，用Euler-Maruyama Method来模拟SDEs时，我们在$u_t(X_t)$方向上迈出一小步，并且添加一点按比例$\sqrt{h}\sigma_t$缩放的高斯噪声。这是主要的模拟方法

扩散模型：现在，我们可以通过SDE构建生成模型。我们的目标是将简单的分布p_{init}转化为复杂的分布p_{data}。与ODE一样，对于这种转化，用$X_0 \sim p_{init}$随机初始化的SDE模拟是一种自然的选择。为了参数化这个SDE，我们可以简单的参数化它的中心成分，也就是把向量场u_t利用神经网络参数化为u_t^θ

因此扩散模型由下式得出：

$$dX_t = u_t^\theta dt + \sigma_t dW_t$$

$$X_0 \sim p_{init}$$

在算法 2 中，我们描述了用Euler-Maruyama方法从扩散模型中采样的过程。

Algorithm 2 Sampling from a Diffusion Model (Euler-Maruyama method)

Require: Neural network u_t^θ, number of steps n, diffusion coefficient σ_t

Set $t = 0$
Set step size $h = \frac{1}{n}$
Draw a sample $X_0 \sim p_{\text{init}}$
for $i = 1, \ldots, n-1$ **do**
    Draw a sample $\epsilon \sim \mathcal{N}(0, I_d)$
    $X_{t+h} = X_t + h u_t^\theta(X_t) + \sigma_t \sqrt{h}\epsilon$
    Update $t \leftarrow t + h$
end for
return X_1

Summary

在本文中，扩散模型由神经网络u_t^θ组成，参数θ表示矢量场，扩散系数σ_t固定：

$$\text{Neural Network} : u^\theta : R^d \times [0,1] \to R^d,\ (x,t) \mapsto u_t^\theta(x) \text{ with parameters } \theta$$

$$\text{Fixed} : \sigma_t : [0,1] \to [0,\infty],\ t \mapsto \sigma_t$$

从我们的SDE模型中获取样本（即生成对象），步骤如下：

$$\text{Initialization} : X_0 \sim p_{init}$$

$$\text{Simulation} : dX_t = u_t^\theta dt + \sigma_t dW_t$$

$$\text{Goal} : X_1 \sim p_{data}$$

一个$\sigma_t = 0$的扩散模型就是流模型。