反应扩散方程

MG21210021 李庆春

2023 年 5 月 4 日

1 反应扩散方程简述

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 \qquad (x,t) \in (0,1] \times (0,1] \tag{1}$$

其中，扩散系数 D=0.01，反应速率 k=0.01，边值为零函数；
考虑初值为方程参数的问题，即初值不确定，记为 $u_0(x)$；
我们采用 DeepONet 去拟合这个算子，记 G_θ 为神经网络算子，G 为真解算子，则对于任意给的初值函数 $u_0^{(i)}$：

$$G_\theta(u_0^{(i)})(x,t) \approx G(u_0^{(i)})(x,t) = u^{(i)}(x,t) \tag{2}$$

$$\frac{\partial G_\theta(u_0^{(i)})(x,t)}{\partial t} \approx D\frac{\partial^2 G_\theta(u_0^{(i)})(x,t)}{\partial x^2} + k[G_\theta(u_0^{(i)})(x,t)]^2 \tag{3}$$

所以我们可以构造两类损失，第一类是神经网络模型的损失，即对于带有 label 的数据（也就是本例中满足初边值条件的点），它们的 label 值和网络预测值之间的差异，我们称之为算子损失，定义如下：

$$\mathcal{L}_{Operator}(\theta) = \frac{1}{NP}\sum_{i=1}^{N}\sum_{j=1}^{P}\left|G_\theta(u_0^{(i)})(x_{u,j}^{(i)}, t_{u,j}^{(i)}) - u^{(i)}(x_{u,j}^{(i)}, t_{u,j}^{(i)})\right|^2 \tag{4}$$

第二类是物理模型损失，在 PDE 的定义域内随机取点，带入方程，计算两边的差异：

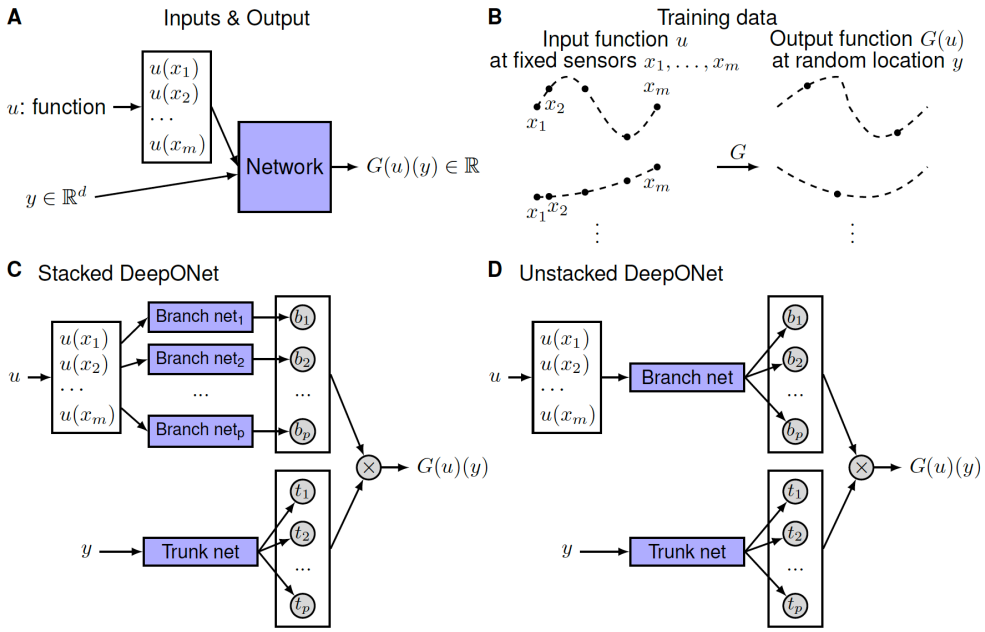
$$\mathcal{L}_{Physics}(\theta) = \frac{1}{NQ}\sum_{i=1}^{N}\sum_{j=1}^{Q}\left|\frac{\partial G_\theta(u_0^{(i)})(x_{r,j}^{(i)}, t_{r,j}^{(i)})}{\partial t} - D\frac{\partial^2 G_\theta(u_0^{(i)})(x_{r,j}^{(i)}, t_{r,j}^{(i)})}{\partial x^2} - k[G_\theta(u_0^{(i)})(x_{r,j}^{(i)}, t_{r,j}^{(i)})]^2\right|^2 \tag{5}$$

最终，损失函数为：

$$\mathcal{L}(\theta) = \mathcal{L}_{Operator}(\theta) + \mathcal{L}_{Physics}(\theta) \tag{6}$$

2 神经网络结构

网络结构采用 Unstacked DeepONet，这样的好处是代码清晰简洁点。分支网络输入初值函数的离散格式，主干网络输入 PDE 定义域内的点。将二者的输出做内积，得到网络的预测值；



(a) DeepONet

图 1: DeepONet

3 训练算法

1. 随机生成 N 个初值函数（本例中 N=5000）：

$$\{u_0^{(i)}(\mathbf{x})\}_{i=1}^{N} = [u_0^{(1)}(\mathbf{x}), u_0^{(2)}(\mathbf{x}), ..., u_0^{(N)}(\mathbf{x})] \tag{7}$$

2. 对于每个初值函数，在其定义域内等距选取 m 个点（本例中 m=100），取这些点对应的函数值作为分支网络的输入：

$$\begin{bmatrix} u_0^{(1)}(x_1) & u_0^{(1)}(x_2) & ... & u_0^{(1)}(x_m) \\ u_0^{(2)}(x_1) & u_0^{(2)}(x_2) & ... & u_0^{(2)}(x_m) \\ \vdots & \vdots & \ddots & \vdots \\ u_0^{(N)}(x_1) & u_0^{(N)}(x_2) & ... & u_0^{(N)}(x_m) \end{bmatrix} \tag{8}$$

3. 将其送入分支网络，经过前向传播：

$$b_k \begin{bmatrix} u_0^{(1)}(x_1) & u_0^{(1)}(x_2) & ... & u_0^{(1)}(x_m) \\ u_0^{(2)}(x_1) & u_0^{(2)}(x_2) & ... & u_0^{(2)}(x_m) \\ \vdots & \vdots & \ddots & \vdots \\ u_0^{(N)}(x_1) & u_0^{(N)}(x_2) & ... & u_0^{(N)}(x_m) \end{bmatrix} \tag{9}$$

4. 从初边值选择 P 个点（本例中 P=100）

$$\mathbf{y}_u = y_{u1}, y_{u2}, ..., y_{uP} \tag{10}$$

5. 将其送入主干网络，经过前向传播：

$$t_k(y_{u1}, y_{u2}, ..., y_{uP}) \tag{11}$$

6. 将二者输出做点积，得到近似的算子

$$G_\theta(u)(\mathbf{y}) = \sum_{k=1}^{q} \underbrace{b_k\left(u(x_1), u(x_2), ..., u(x_m)\right)}_{Branch} . \underbrace{t_k(\mathbf{y})}_{Trunk} \tag{12}$$

7. 根据公式 4，计算算子损失；

8. 从定义域中随机选 Q 个点

$$\mathbf{y}_r = y_{r1}, y_{r2}, ..., y_{rQ} \tag{13}$$

9. 根据公式 5，计算物理模型损失；

10. 计算总损失

$$\mathcal{L}(\theta) = \mathcal{L}_{operator}(\theta) + \mathcal{L}_{Physics}(\theta) \tag{14}$$

11. 反向传播，利用梯度下降更新神经网络参数，使损失最小化；

12. 重复上述过程直至 $G_\theta(u_0)(x,t) \approx G(u_0)(x,t)$

这里需要注意的是，初值函数集合要和训练点集合做卡氏积，即对于任何一个初值函数，它要和所有的训练点分别进行一次网络前馈，并做点积。

4 主程序解释

4.1 RBF

高斯径向基函数，是一种高斯核函数，用于计算两个时刻的高斯变量的协方差，用 RBF 得到的是一个半正定矩阵；

$$k(x, x') = \sigma^2 exp(-\frac{||x - x'||^2}{2l^2}) \tag{15}$$

4.2 gp_sample

高斯过程采样函数，对 x 的定义域 [0, 1] 做网格划分，得到一列点 $x_0, x_1, \cdots, x_n$，记为向量 $\mathbf{x}$，作为参数调用 RBF 函数，得到协方差矩阵，再利用 cholesky 分解定理得到一列符合正态分布的随机变量 $y_0, y_1, \cdots, y_n$，记为 $\mathbf{y}$；以 $\mathbf{x}$ 和 $\mathbf{y}$ 作插值函数，就生成了一个初值函数；方法中通过 for 循环生成初值函数列表，注意这里返回的就是 function 对象。

4.3 ADRICBCDataset

初边值条件的数据集，主要逻辑再 init 方法里面。

1. 初始化初值训练点
   本例中选取 100 个初值点进行训练，先对高斯过程采样生成的初值函数列表进行离散化，取离散化的函数值作为分支网的输入，注意这里每个函数要和所有的初边值点作用一下，所以要对函数进行复制，复制的次数等于初值点的个数，对 [0 ,1] 均匀采样 100 个点作为初值训练点，并调用初值函数得到函数值作为训练的标签值，这里同样需要注意的是，对于每个初值训练点，都要调用所有的初值函数。

2. 初始化边值训练点
   逻辑和上面的初值训练点差不多，区别就是本例采用零边值条件，所以对应的标签值都为 0;

4.4 ADRPhysicsDataset

物理信息的数据集，逻辑和初边值训练点差不多，区别在于是对 $[0,1]\times[0,1]$ 均匀采点;

4.5 ADRNet

算子网络的具体实现

4.5.1 init

激活函数设置为双曲正切 (Tanh)，损失函数使用均方误差 (MSE)，主干网络和分支网络根据传入的参数构造多层感知机，就是多层全连接网络。

4.5.2 forward

将初值函数经过分支网络前向传播，训练点 (x,t) 经过主干网络前向传播，对两个网络的输出做内积，得到预测值。

4.5.3 loss_icbc

计算初边值训练损失，将初边值训练点经过网络前馈，计算得到的预测试和标签值的均方误差。

4.5.4 loss_physics

计算物理模型损失，将物理训练点经过网络前馈，对得到的预测值计算各阶导数，按照公式 5计算损失。

4.5.5 loss

将初边值损失和物理损失进行加总，这里相当于设置二者的权重各位 0.5。

连续复利是指在无限短的时间间隔内按照一定利率复利的过程。和年复利不同，年复利是在固定的时间间隔内进行复利的。

考虑一个本金为 P，年利率为 r 的债券，按照连续复利，每时每刻的利率是 $r_c = e^r - 1$，其中 e 是自然对数的底数。假设持有该债券 t 年，则连续复利后的本金总额是：

$P_c = P \cdot e^{rt}$

推导过程如下：

根据连续复利的定义，我们可以将一年分成 n 个等分，每个等分的时间间隔为 $\frac{1}{n}$ 年，每个等分内的复利率为：

$r_n = \frac{r}{n}$

则该债券在一年内的复利因子为：

$(1 + \frac{r}{n})^n$

当 n 趋近于无穷大时，上式趋近于：

e^r

所以一年后，本金为 P 的债券按照连续复利的复利因子为：

e^r

持有 t 年后，本金为 P 的债券按照连续复利的复利因子为：

$(e^r)^t = e^{rt}$

因此，连续复利后的本金总额为：

$P_c = P \cdot e^{rt}$

这是连续复利的基本公式。