基于 MindSpore 的 pfnn-toolkit 实现报告

引言

PFNN (Penalty-free neural network)方法是一种基于神经网络的微分方程求解方法，适用于求解复杂区域上的二阶微分方程。该方法克服了已有类似方法在处理问题光滑性约束和边界约束上的缺陷，具有更高的精度，效率和稳定性。

我们利用华为自研的 MindSpore 框架实现了利用 PFNN 方法求解多种偏微分方程的工具包 pfnn-toolkit，并利用数据并行方法提供了 PFNN 方法的并行版本。

方法描述

PFNN采用神经网络逼近微分方程的解。不同于大多数只采用单个网络构造解空间的神经网络方法，PFNN采用两个网络分别逼近本质边界和区域其它部分上的真解。为消除两个网络之间的影响，一个由样条函数所构造的length factor函数被引入以分隔两个网络。为进一步降低问题对于解的光滑性需求，PFNN利用Ritz变分原理将问题转化为弱形式，消除损失函数中的高阶微分算子，从而降低最小化损失函数的困难，有利于提高方法的精度

论文：H. Sheng, C. Yang, PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, Journal of Computational Physics 428 (2021) 110085.

软件环境

我们的开发主要依赖于 Mindspore 华为自研全场景深度学习框架。

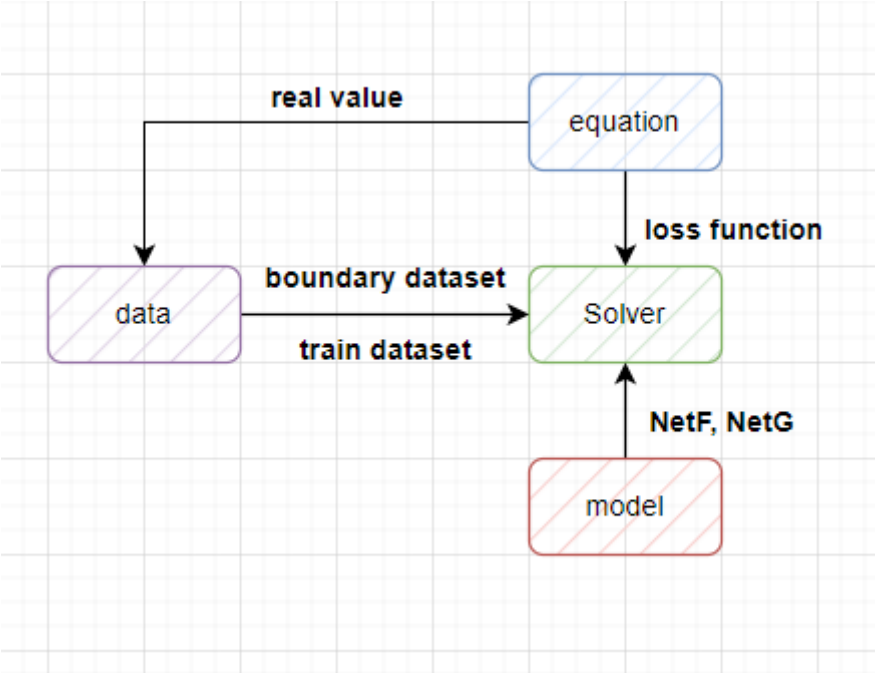
Mindspore 旨在实现易开发、高效执行、全场景覆盖三大目标。其中，易开发表现为API友好、调试难度低；高效执行包括计算效率、数据预处理效率和分布式训练效率；全场景则指框架同时支持云、边缘以及端侧场景。

我们基于 MindSpore 框架开发的 pfnn-toolkit 主要利用了该框架如下特点：

1. 利用 MindSpore 框架的静态图模式，在运行前首先生成神经网络的图结构，帮助编译器通过使用图优化等技术来获得更好的执行性能。
2. 神经网络求解 PDE 要求我们对输出进行微分操作，我们利用 MindSpore 框架的函数式可微分编程架构对网络的输出求一阶导数。
3. 使用MindSpore提供的高阶API---Model，进行模型的训练、评估和推理， 为不同问题的求解提供了统一的模型训练、推理和导出等接口。
4. 利用 Mindspore 提供的分布式并行训练，通过数据并行方式加速模型的训练

程序描述

整个程序共分为四个模块，data 模块负责生成数据集，equation 模块负责保存方程相关信息，model 模块负责提供模型，solver 模块负责综合各个模块实现问题的求解



数据集(data)

data 模块负责生成训练所需数据并生成对应的训练集，测试集

- 训练集：分为内部集和边界集，分别在计算区域内部和边界上采样得到。
  - 内部集：在计算区域内部采样，并计算控制方程右端项在这些点上的值作为标签。
  - 边界集：在Dirichlet边界和Neumann边界上分别采样，并计算边界方程右端项在这些点上的值作为标签。
- 测试集：在整个计算区域上采样，并计算真解在这些点上的值作为标签

方程(equation)

equation 模块负责记录方程信息，用于计算标签以及定义损失函数

- real value: 为训练集和测试集计算标签
- loss function: 为训练神经网络提供变分形式的损失函数

其中，loss function 利用MindSpore的自动微分功能对网络的输出值进行微分操作

模型(model)

Model 模块提供负责内部区域的 F 网络和负责边界区域的 G 网络

其中，我们利用 MindSpore 的静态图模式来获得更好的性能

求解器(Solver)

Solver 模块负责整个问题的求解

接受来自 data 模块的数据，利用 equation 部分提供的损失函数对 model 模块提供模型进行训练，从而实现问题的求解

其中，我们利用 MindSpore 提供的高阶 API---Model 来统一实现对于不同问题的求解

算例展示

在 demo 文件中，我们展示利用 pfnn-toolkit 对两类偏微分方程，扩散方程(Diffusion equation) 和刘维尔方程(Liouville-Bratu equation) 的求解算例。

Diffusion equation

$$-\nabla \cdot \boldsymbol{A} \nabla u + c = 0,\ x \in \Omega$$

我们令 $\boldsymbol{A} = \begin{bmatrix} (x_1 + x_2)^2 + 1 & -x_1^2 + x_2^2 \\ -x_1^2 + x_2^2 & (x_1 - x_2)^2 + 1 \end{bmatrix}$, $\Omega = [0,\ 1]^2$

主要参数设置如下：

- 内部采样数目：40000
- 边界采样数目：40000
- Epoch: 6000
- 学习率: 0.01

最终相对误差为 3.797e-03

Liouville-Bratu equation

$$-\nabla \cdot (\nabla u) - \lambda \exp(u) + c = 0$$

我们令 $\lambda = 0.3,\ \Omega = [0,\ 1]^2$

主要参数设置如下：

- 内部采样数目：40000
- 边界采样数目：40000
- Epoch: 6000
- 学习率: 0.01

最终相对误差为 3.086e-03

分布式训练

利用神经网络求解 PDE 网络的训练时间占据求解时间的绝大部分，我们利用 Mindspore 提供的数据并行接口实现网络的分布式训练。

按照如上的参数设置，分别在 1, 2, 4 张 Nvidia RTX-2080 Ti 上进行训练，两个网络训练时间分别如下所示：

Diffusion equation

Device num	1	2	4
Net G trianTotal time(s)	16.98	12.72	8.23
Net F trianTotal time(s)	173.50	125.64	84.37

Liouville-Bratu equation

Device num	1	2	4
Net G trianTotal time(s)	13.87	8.97	7.11
Net F trianTotal time(s)	167.15	110.94	72.48