不确定性下的物理信息深度学习全能库

科学计算项目高级编程

学生: Giulia Mescolini, Luca Sosta
导师: Andrea Manzoni, Stefano Pagani

2023 年 2 月 27 日

简介

项目概览

- 基于深度学习的技术在科学计算中的效率可以被利用，当我们想要考虑物理信息和不确定性量化 (UQ) 时，有趣的挑战就出现了。
- 我们选择在一个新库的背景下从头开始实现贝叶斯物理信息神经网络 (B-PINNs) 方法，该库旨在灵活并倾向于新的扩展。
- 如今，有很多深度学习的工具和库，但在我们的特定领域，它们大多是面向单一应用的，创建时没有考虑到被他人重复使用。

库的目标

- 在这个项目中，我们专注于 B-PINNs 库的基础开发，优先考虑其实现和潜力，而不是早期的结果。
- 首先，我们希望制定尽可能简单的工作流程，以保持添加其他功能的灵活性。
- 这也通过模块化结构得到了强调，其中主流程的每一步都是独立的，并针对多种方法和设置（例如不同的物理问题或优化器的选择）进行了结构化。
- 这些属性可以通过使用面向对象范式来保证。

Python 中的面向对象编程

如今，基于深度学习的技术备受瞩目，大多数实现都依赖于 Python 语言，因为它开发速度快，并且提供了丰富的高级库（如 TensorFlow）。

有时 Python 被认为被高估了，或者比编译语言弱，但它在处理面向对象编程时仍然提供了各种各样的功能和 *ad hoc* 方法，例如：

- `@property`: 属性 getter 和 setter 的高级版本
- `@dataclass`: 用于数据处理的类的轻量级版本
- `ABC`: 用于继承中虚拟性的抽象基类模块

方法和库基础

贝叶斯神经网络 (BNNs)

BNN 重构给定数据下的输出后验分布 $p(u|D)$。

通过关系 $u(x) = \mathcal{NN}_\theta(x)$，我们可以通过考虑 NN 参数的后验分布 $p(\theta|D)$ 来获得它。

贝叶斯定理使我们能够通过先验和似然来检索它，得益于：

$$p(\theta|D) \propto p(\theta)p(D|\theta)$$

[神经网络结构示意图]

优化器

我们没有 $p(\theta|D)$ 的计算上可行的表达式，因此我们需要算法来实现两个目的：

- 改进后验的近似
- 从重构的分布中获取样本

优化器负责这些任务，它们在涉及 epoch 循环的迭代过程中执行这些任务，就像在经典 NN 理论中一样。

物理信息神经网络 (PINNs)

物理信息的引入伴随着 PINNs，它通过在损失中增加一项代表方程残差的项，训练网络产生满足 PDE 的输出。

PINNs 的训练数据集主要分为两类：

- **拟合点 (Fitting points)**，我们在此处强加测量值
- **配点 (Collocation points)**，我们在此处强加方程的满足

通过将这些方面整合到已经描述的 BNN 框架中，我们最终获得了 B-PINNs。

核心特性

在这种背景下，我们选择构建一个模块化管道，由各种块组成，这些块有不同的版本以处理各种应用和方法。

在库内，有必要组织三个基本块之间的通信：

- 数据和批次组织
- 物理信息神经网络
- 不确定性量化

为此，类属性和属性的组织起着至关重要的作用。

主要流程

我们库的主要工作流程可以概括为 7 点：

1. 配置参数的处理
2. 数据生成
3. 数据集的预处理
4. 模型和优化器初始化
5. 训练阶段
6. 预测和性能评估
7. 结果的存储和绘图

辅助实现

参数处理部分管理三个信息源：`.json` 配置文件和用于数据生成的 `@dataclass`。这些被合并并且可以被命令行参数覆盖，并存储在一个单一的 `Param` 对象中。

Storage 类将图表、日志和值保存到文件夹中。因此，它们准备好由 `Plotter` 重新生成，`Plotter` 工作时只询问文件夹路径。

Equation 类管理测试用例的所有物理信息，并使用自动微分模块计算 PDE 的残差。它还将物理域转换为计算域。

核心实现

数据生成

在这个项目中，我们考虑了一个完整的测试用例工作流程，它直接从合成数据的生成开始。

处理 PINNs 时，我们需要不同类型的数据集，这需要不同的采样方法：

- **Uniform**: 在网格上生成，用于计算误差和绘图
- **Random**: 域内点的随机均匀采样
- **Sobol**: 低差异的点的伪随机采样

我们还实现了子域或多个子域中的数据生成，以模拟某些区域缺乏测量的情况。

数据集创建

预处理阶段首先询问生成数据的子集，以避免对类似测试用例进行无用的重新生成。此外，它使用 `Equation` 提供的方法转换数据集；在我们的测试用例中，我们应用了数据范围的标准化。

处理 PINNs 时，我们需要在不修改原始数据的情况下将数据集拆分为多个部分，这是通过 `@property` 装饰器实现的，它提供了所有的私有化功能。

最后，为了提高训练期间的效率，数据集的所有组件都被分批处理，允许在每个 epoch 中使用少量数据样本。这是通过类的特殊方法 `__next__()` 执行的。

Theta 类

TensorFlow 将 NN 参数存储在 Tensors 列表中，但对于算法中使用的逐元素或基于范数的操作来说，这既不切实际又多余。

我们实现了 `Theta` 类，在其中重写了代数运算以避免代码块的重复。

这个类提高了可读性，并且在处理微分保留时使我们更不容易出错。

Operation	Method	Sign
Opposite	`__neg__`	
Sum	`__add__`, `__radd__`	+
Subtraction	`__sub__`, `__rsub__`	-
Multiplication	`__mul__`, `__rmul__`	*
Division	`__truediv__`	/
Power	`__pow__`	**

CoreNN 和 PhysNN

我们实现了 B-PINNs 作为在 TensorFlow 全连接模型上通过继承构建的超结构。正如在物理信息背景中典型的那样，网络非常小。

CoreNN 类是第一个构建块，代表基本的 NN：它存储和管理模型及其参数，并执行前向步骤。

PhysNN 继承自它，并包含 `Equation`、PDE 参数以及从真实域到计算域转换所需的信息。

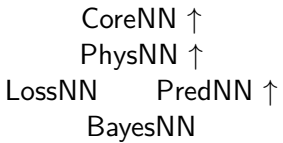
BayesNN

`BayesNN` 是第三个超结构，它是可执行脚本中实例化的类。

它负责两个不相交的任务：

- 损失计算
- 预测，同时也提供 UQ

上述功能已被分离到两个不同的类中，`BayesNN` 在多重继承的背景下从中继承。



LossNN

LossNN 类通过分离其组件来计算损失和指标。

对于其计算，它依赖于已实现的静态方法，特别是对于物理损失，它与父类 PhysNN 交互以检索方程残差。

它还计算损失相对于 NN 参数的梯度，这是所有训练算法所需的量。

PredNN

这个类的功能在优化器的动作之后被调用，优化器填充了 NN 参数样本的列表。

给定参数集，它重构最终输出的样本集，这不仅通过前向传递获得，还通过转换回物理域获得。

它还负责误差和不确定性量化，自动在不同方向上进行评估。

算法 (Algorithm)

优化器继承自抽象基类 `Algorithm`，它确定了训练管道的蓝图。
它接收并将数据集存储在一个属性中，并使用其方法对 `BayesNN` 的实例起作用。

两个抽象方法使得可以将算法彼此区分开来：

- `sample_theta()`，在每次迭代中生成一个 NN 参数样本
- `select_thetas()`，它在训练结束时确定最终的样本集

Trainer 和 Adam

`Trainer` 类是主脚本和算法之间的接口，并扮演训练管理器的角色。在这个意义上，它还包括包含确定性预训练的可能性。

Adam 优化器是最流行、快速和有效的优化器之一。我们将其安装在贝叶斯算法的相同结构上，只需重写：

- `sample_theta()` 作为经典的 Adam 训练步骤
- `select_thetas()` 作为仅选择列表最终样本的选择器

哈密顿蒙特卡洛 (HMC)

HMC 方法结合了 MCMC 和哈密顿动力学。

- `sample_theta()` 负责蛙跳 (leap-frog) 和接受-拒绝 (acceptance-rejection) 步骤
- `select_thetas()` 通过引入老化 (burn-in) 和步幅 (stride) 来选择最终的样本集

这种方法需要合理的计算时间，但它涉及许多参数，因此调整起来具有挑战性。

变分推断 (Variational Inference)

该算法用多元高斯分布近似后验，其参数 μ 和 ρ 在确定性训练中学习。然后，可以从学习的分布中提取任意数量的样本。

- `sample_theta()` 在 epoch 循环中执行 μ 和 ρ 的更新
- `select_thetas()` 作为仅选择列表最终样本的选择器

Stein 变分梯度下降 (SVGD)

该方法具有复杂的底层理论，涉及从初始近似分布开始最小化 KL 散度。
其实施提出了挑战，例如并行管理和交换 N 个网络之间的损失信息的任务。

- `sample_theta()` 训练网络集合
- `select_thetas()` 返回所有最终参数

结果展示

阻尼谐振子

这个测试用例涉及一个 ODE，仅在初始时间有可用测量值。

$$\frac{d^2x}{dt^2} + 2\delta\frac{dx}{dt} + \omega^2 x = 0 \quad \delta = 2, \omega = 8$$

[图表：NN 插值 vs PINN 重构]

NN 执行良好的插值仅在第一个区域。

由于物理定律，PINN 可以重构整个演变过程。

HMC 泊松问题

我们考虑泊松问题 $-\Delta u = f$，解为 $u(x) = \sin^3(6x)$。

对 f 的了解仅限于稀疏和嘈杂的测量。

此外，通过这种方法，我们还可以利用关于 u 的信息，无论是在边界上还是在域内。

[图表: 泊松问题结果]

结论

结论

- 该项目的主要目的是优先考虑代码的新实现和可扩展性，而不是获得深入的结果。
- 在这个演示中，我们只展示了一些主要特性和突出它们所需的少量结果。
- 拟议的库版本成功实现了最初定义的灵活性和模块化目的。

以此为基础的进一步发展

建议的发展方向包括：

- 其他方程和算法的实现
- 第三方数据接口和相关的绘图工具
- 参数估计问题设置
- 调优过程的自动化

主要参考文献

1. B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data, Liu Yang, Xuhui Meng, George Em Karniadakis, Mar 2020.
2. Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems, Kevin Linka, Amelie Schäfer, Xuhui Meng, Zongren Zou, George Em Karniadakis, and Ellen Kuhl, May 2022.
3. Bayesian Physics-Informed Neural Networks for Inverse Uncertainty Quantification problems in Cardiac Electrophysiology, Master Thesis at Politecnico di Milano by Daniele Ceccarelli.

谢谢您的关注!