Построение и исследование нейронной сети с модифицированной функцией потерь и периодической функцией активации для решения дифференциальных уравнений в частных производных

Диков Александр Евгеньевич
Б20-221

Научный руководитель:
Д. П. Макаревич, ООО "РЦР"

Содержание

Введение

- Дифференциальные уравнения в частных производных описывают множество физических явлений и зачастую они решаются традиционными численными методы. Однако, в ситуациях, когда нужно определить значение какой-либо величины (например, в экспериментальных установках) они оказываются неэффективными по времени.
- В рамках моего исследования я сосредоточился на применении машинного обучения для решения физических задач. Мной было рассмотрено несколько способов по улучшению качества предсказаний модели, среди которых использование модификации функции потерь и периодической функции активации.

Основные принципы

Пусть есть задача:

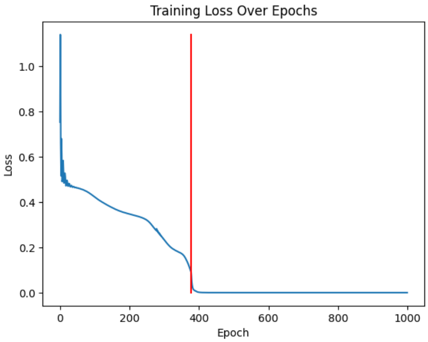
$$\begin{cases} \dfrac{\partial u}{\partial t} = \alpha^2 \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < L, \quad t > 0, \\ u(x,0) = u_0(x), \\ u(0,t) = u_L(t), \quad u(L,t) = u_R(t) \end{cases} \tag{1}$$

Введем компоненты функции потерь:

$$\begin{aligned} \mathcal{L} &= \omega_1 \mathcal{L}_{\mathrm{PDE}} + \omega_2 \mathcal{L}_{\mathrm{IC}} + \omega_3 \mathcal{L}_{\mathrm{BC}}, \\ \mathcal{L}_{\mathrm{PDE}} &= \frac{1}{N_{\mathrm{PDE}}} \sum_{i=1}^{N_{\mathrm{PDE}}} \left| \frac{\partial u}{\partial t} - \alpha^2 \frac{\partial^2 u}{\partial x^2} \right|^2, \\ \mathcal{L}_{\mathrm{IC}} &= \frac{1}{N_{\mathrm{IC}}} \sum_{k=1}^{N_{\mathrm{IC}}} \left| u(x_k, 0) - u_0(x_k) \right|^2, \\ \mathcal{L}_{\mathrm{BC}} &= \frac{1}{N_{\mathrm{BC}}} \sum_{j=1}^{N_{\mathrm{BC}}} \left| u(x_j, t_j) - u_b(x_j, t_j) \right|^2 \end{aligned} \tag{2}$$

Тестирование оптимизаторов

Было проведено исследование оптимизаторов. Было выбрано два наиболее распространённых при решении задачи - ADAM и L-BFGS. Как было описано в статье[a] L-BFGS свойственно находить минимум с большей точностью за меньшее число итераций, но при этом высок шанс попадания в локальный минимум - поэтому можно применять их совместно.



Гибридный оптимизатор

[a] S. Berrone, C. Canuto, M. Pintore, and N. Sukumar. Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks. Heliyon, 9(8):e18820, August 2023.

Тестирование оптимизаторов

На примере уравнения осциллятора результат попадания в локальный минимум выглядит так:

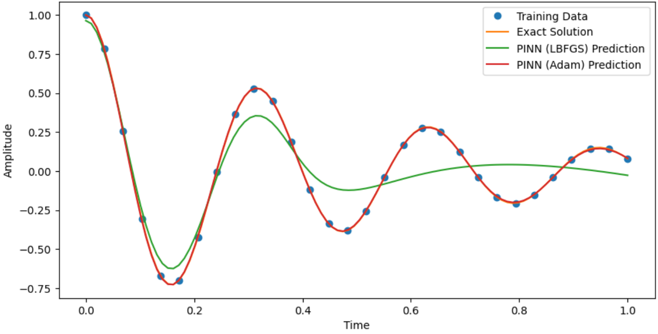
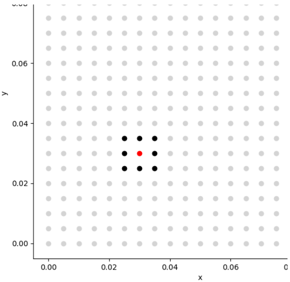
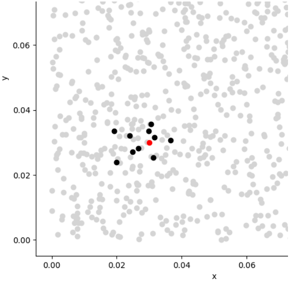


Рис. 1: Случай попадания оптимизатора в локальный минимум

Адаптивное добавление точек

Непосредственно подход добавления точек в выборку был предложен в статье[b], там же был описан принципиальный алгоритм. Предлагалось добавлять одну дополнительную точку коллокации к той, для которой функция потерь принимает максимальное значение. Однако ошибка будет уменьшаться быстрее, если добавлять не одну точку, а несколько, в ϵ окрестности этой точки с учетом выбранной конфигурации.





[b]Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis.Deepxde: A deep learning library for solving differential equations. SIAM Review, 63(1):208–228, 2021.

Влияние регуляризации

Для получения более гладких решений можно использовать регуляризацию весов модели. Была реализована возможность выбора $\mathcal{L}_1(Lasso)$, $\mathcal{L}_2(Ridge)$ и $L_3(ElasticNet)$ определяемых по следующим формулам:

$$\mathcal{L}_1(\theta) = \sum_{i=1}^{n} |\theta_i|, \tag{3}$$

$$\mathcal{L}_2(\theta) = \sum_{i=1}^{n} \theta_i^2, \tag{4}$$

$$\mathcal{L}_3(\theta) = r \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2} \sum_{i=1}^{n} \theta_i^2 \tag{5}$$

Влияние регуляризации

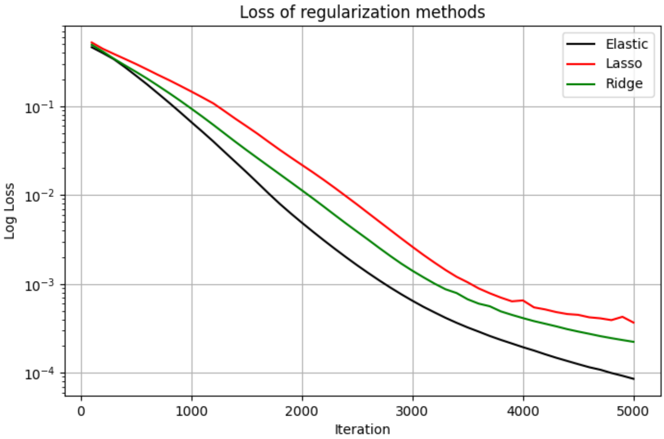


Рис. 2: Сравнение регуляризаций

Поскольку общая функция потерь состоит из трех слагаемых (без учета регуляризационного члена) можно каждому слагаемому присвоить вес, который будет отражать "важность"минимизации ошибки в точках определенной категории. Этот вес можно варьировать в процессе обучения для достижения более равномерной точности.

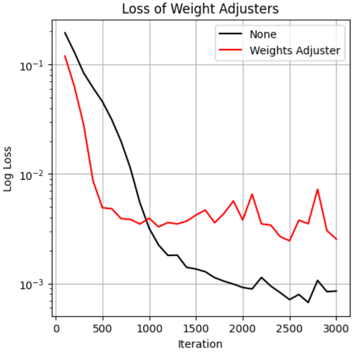


Рис. 3: Влияние множителей функции потерь на сходимость

Использование планировщиков

Одним из самых главных гиперпараметров сети является шаг скорости обучения.

$$\theta_i = \theta_{i-1} - \alpha \cdot \nabla_\theta J(\theta) \qquad (6)$$

Где θ_{i-1} — текущее значение параметров модели, α — скорость обучения, $J(\theta)$ — функция потерь.
Существуют планировщики (*scheduler*) - объекты, которые меняет размер шага при определенных условиях. Было рассмотрено два планировщика:

1. StepLR: Уменьшает скорость обучения на заданный коэффициент каждые N эпох.
2. ReduceLROnPlateau: Уменьшает скорость обучения, когда метрика ошибки перестает улучшаться.
3. ExponentialLR: Уменьшает скорость обучения на каждой эпохе, умножая её на заданный коэффициент.

Использование планировщиков

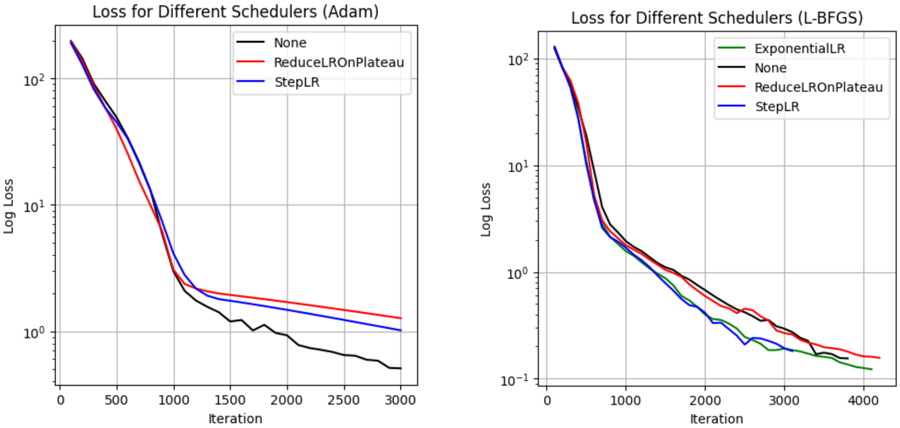


Рис. 4: Влияние планировщика на сходимость

Принципиальная схема

Ниже можно видеть принципиальное устройство полученной сети и вспомогательных модулей, реализующий все упомянутые выше доработки.

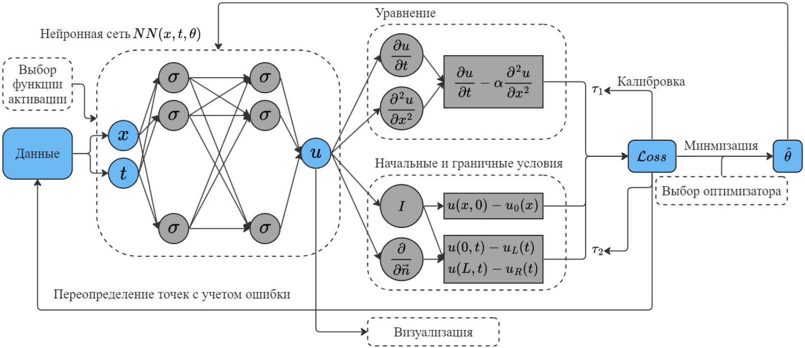


Рис. 5: Принципиальное устройство сети

Инициализация

Входной слой: $\mathcal{N}^0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$,
Скрытый слой: $\mathcal{N}^\ell(\mathbf{x}) = \sigma\left(\boldsymbol{W}^\ell \mathcal{N}^{\ell-1}(\mathrm{x}) + \boldsymbol{b}^\ell\right) \in \mathbb{R}^{N_\ell}$ для $1 \leq \ell \leq L-1$,
Выходной слой: $\mathcal{N}^L(\mathbf{x}) = \boldsymbol{W}^L \mathcal{N}^{L-1}(\mathbf{x}) + \boldsymbol{b}^L \in \mathbb{R}^{d_{\text{out}}}$

Algorithm 1 PINN для решения дифференциальных уравнений

1: Построить нейронную сеть $\hat{u}(\mathbf{x}; \boldsymbol{\theta})$ с параметрами $\boldsymbol{\theta}$.
2: Задать два набора обучающих данных $\mathcal{T}_{PDE}$, $\mathcal{T}_{IC}$ и $\mathcal{T}_{BC}$ для уравнения, начальных и граничных условий.
3: Задать функцию потерь, суммируя взвешенные L^2 нормы остатков уравнения, начальных и граничных условий.
4: Обучить нейронную сеть для нахождения лучших параметров θ^* путем минимизации функции потерь $\mathcal{L}(\boldsymbol{\theta}; \mathcal{T})$.

Дальнейшие планы

- Необходимо реализовать граничные условия второго и третьего рода. Существует несколько статей на эту тему, поскольку это не тривиальная задача.
- Исследовать распределение ошибки.
- Изучение возможности использования *fine – tuning* для дообучения на похожих задачах, что можно было экономить существенно время.
- Провести сравнение с методом конечного элемента. В одной из статей предлагалось использовать для этого FEniCS с открытым исходным кодом.

Выводы

- PINN предоставляет мощный инструмент для численного решения дифференциальных уравнений в частных производных, а благодаря гибкости нейронных сетей позволяет адаптировать метод для различных физических задач.
- Метод является бессеточным, что позволяет в перспективе работать со сложными геометриями или пользоваться быстротой работы обученной нейронной сети.
- Обучение может потребовать значительного времени и вычислительных ресурсов.
- Дальнейшие исследования могут сосредоточиться на оптимальных архитектурах сетей для различных задач, в том числе многомерных или с более сложным уравнениям.

Источники

[1] S. Berrone, C. Canuto, M. Pintore, and N. Sukumar.
Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks.
Heliyon, 9(8):e18820, August 2023.

[2] Stefano Markidis.
The old and the new: Can physics-informed deep-learning replace traditional linear solvers?, 2021.

[3] Yeonjong Shin.
On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes.
Communications in Computational Physics, 28(5):2042–2074, June 2020.

Источники

[4] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis.
Deepxde: A deep learning library for solving differential equations.
SIAM Review, 63(1):208–228, 2021.

[5] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein.
Implicit neural representations with periodic activation functions, 2020.

[6] M. Raissi, P. Perdikaris, and G.E. Karniadakis.
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational Physics, 378:686–707, 2019.

Спасибо за внимание