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

Наиболее актуальные статьи по решению дифференциальных уравнений с помощью нейронных сетей можно разделить на несколько категорий.

- К первой можно отнести те статьи, в которых описаны основные принципы PINN.
- Ко второй категории отнесем статьи, описывающие технологии, такие как метод автоматического дифференцирования.
- Третья категория статей описывает различные приемы и идеи, по ускорению сходимости или улучшению качества решения. Обычно они касаются изменения функции активации, общей архитектуры или функции потерь.
- В четвертой категории находятся статьи в которых вводятся необходимые теоремы, освещающие вопросы сходимости и доказывается непротиворечивость PINN.

Physics-informed neural networks

В этой статье впервые был представлен термин Physics-informed neural networks (PINN)*, введены многие принципы и решения, которые впоследствии были использованы и другими авторами.
Пусть есть задача:

$$u_t + \mathcal{N}[u] = 0, x \in \omega, t \in [0, T], \tag{1}$$

Введем f как левую часть уравнения:

$$f := u_t + \mathcal{N}[u] \tag{2}$$

Введем компоненты функции потерь:

$$\begin{aligned} MSE &= MSE_u + MSE_f, \\ MSE_u &= \frac{1}{N_u} \sum_{i=1}^{N_u} \left| u(t_u^i, x_u^i) - u^i \right|^2, \\ MSE_f &= \frac{1}{N_f} \sum_{i=1}^{N_f} \left| f(t_f^i, x_f^i) \right|^2. \end{aligned} \tag{3}$$

Physics-informed neural networks

Авторами было проведено несколько исследований на различных моделях и различных уравнениях. Во всех случая метод проявил себя хорошо. Отдельного внимания заслуживает таблица сравнения архитектур - подобное исследование, для более современной архитектуры и уравнения теплопроводности я приводил ранее.

Table A.2
Burgers' equation: Relative $\mathbb{L}_2$ error between the predicted and the exact solution $u(t, x)$ for different number of hidden layers and different number of neurons per layer. Here, the total number of training and collocation points is fixed to $N_u = 100$ and $N_f = 10{,}000$, respectively.

Layers \ Neurons	10	20	40
2	7.4e−02	5.3e−02	1.0e−01
4	3.0e−03	9.4e−04	6.4e−04
6	9.6e−03	1.3e−03	6.1e−04
8	2.5e−03	9.6e−04	5.6e−04

Рис. 1: Зависимость L2 меры от архитектуры

Table A.1
Burgers' equation: Relative $\mathbb{L}_2$ error between the predicted and the exact solution $u(t, x)$ for different number of initial and boundary training data N_u, and different number of collocation points N_f. Here, the network architecture is fixed to 9 layers with 20 neurons per hidden layer.

N_u \ N_f	2000	4000	6000	7000	8000	10000
20	2.9e−01	4.4e−01	8.9e−01	1.2e+00	9.9e−02	4.2e−02
40	6.5e−02	1.1e−02	5.0e−01	9.6e−03	4.6e−01	7.5e−02
60	3.6e−01	1.2e−02	1.7e−01	5.9e−03	1.9e−03	8.2e−03
80	5.5e−03	1.0e−03	3.2e−03	7.8e−03	4.9e−02	4.5e−03
100	6.6e−02	2.7e−01	7.2e−03	6.8e−04	2.2e−03	6.7e−04
200	1.5e−01	2.3e−03	8.2e−04	8.9e−04	6.1e−04	4.9e−04

Рис. 2: Зависимость L2 меры от числа точек

*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.

Automatic Differentiation in Machine Learning

В статье "Automatic Differentiation in Machine Learning: a Survey"* рассматривается техника автоматического дифференцирования (AD). AD относится к семейству методов, которые вычисляют производные путем накопления значений во время выполнения кода для создания числовых оценок производных, а не с помощью конкретных выражений.
В то время как численное дифференцирование может быть неустойчиво, использовать подход символьного дифференцирования для некоторой сложной функции f(x) будет вызывать возникновение вложенных дубликатов любых вычислений, такой подход может легко привести к появлению экспоненциально больших выражений.

Automatic Differentiation in Machine Learning

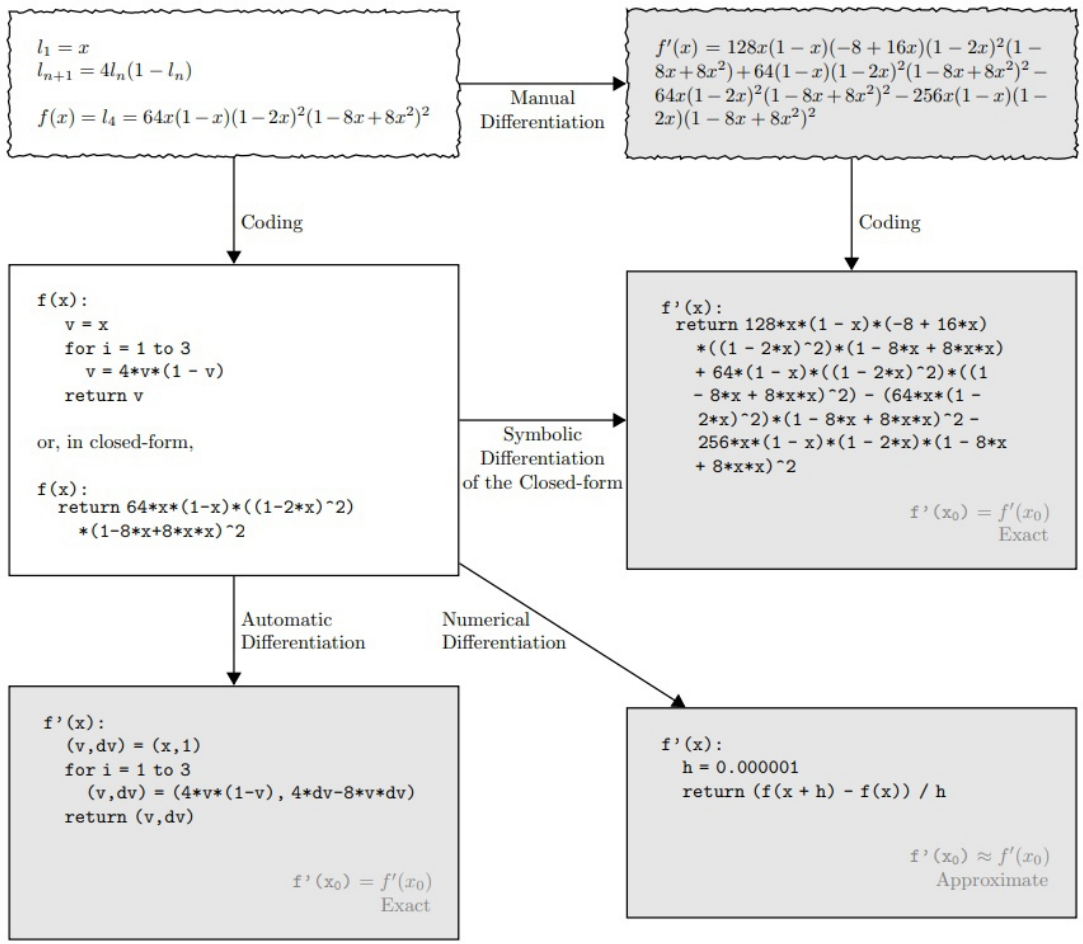


Рис. 3: Отличие автоматического дифференцирования от численного и символьного

В основе AD лежит следующий подход: применять символьное дифференцирование на уровне элементарных операций и сохранять промежуточные численные результаты.

Forward pass	Backward pass
$x_1 = 2$ $x_2 = 1$	$\frac{\partial y}{\partial y} = 1$
$v = -2x_1 + 3x_2 + 0.5 = -0.5$ $h = \tanh v \approx -0.462$	$\frac{\partial y}{\partial h} = \frac{\partial (2h-1)}{\partial h} = 2$ $\frac{\partial y}{\partial v} = \frac{\partial y}{\partial h}\frac{\partial h}{\partial v} = \frac{\partial y}{\partial h}\operatorname{sech}^2(v) \approx 1.573$
$y = 2h - 1 = -1.924$	$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial v}\frac{\partial v}{\partial x_1} = \frac{\partial y}{\partial v} \times (-2) \approx -3.146$ $\frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial v}\frac{\partial v}{\partial x_2} = \frac{\partial y}{\partial v} \times 3 \approx 4.719$

Таблица 1: Пример вычислений

*Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey, 2018.

DeepXDE: A Deep Learning Library for Solving Differential Equations

В этой статье* метод был описан еще более подробно и предложено сразу множество идей по улучшению:

- Добавление настраиваемых весов к потерям с разных точек.
- Использование *callback* функции для подбора параметров архитектуры сети.
- Точки для вычисления остатков можно выбирать разными способами:
  - Один раз в начале (случайно или сеткой) и не менять.
  - На каждой итерации обучения разные точки.
  - Улучшать их расположение в зависимости от величины функции потерь.

Algorithm 1 Улучшение распределения остаточных точек для обучения (RAR)

1: Выбрать начальные точки и обучить нейронную сеть ограниченное количество итераций.
2: Оценить средний остаток ДУ $\mathcal{E}_r$ с использованием метода Монте-Карло. Это делается усреднением значений по набору случайно выбранных наборов точек $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{S}|}\}$:

$$\mathcal{E}_r \approx \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right)\right\|. \quad (4)$$

3: Если $\mathcal{E}_r < \mathcal{E}_0$, завершить процедуру. В противном случае добавить новые точки m с наибольшими остатками из $\mathcal{S}$ к точкам $\mathcal{T}$, повторно обучить сеть и вернуться к Шагу 2.

DeepXDE: A Deep Learning Library for Solving Differential Equations

Кроме того, в статье освещается вопрос ошибок, данная терминология будет использована и в дальнейших работах. Ошибки при решении связаны с различными факторами:

- Аппроксимация - разница между действительным решением и самой ближайшей из функций, которую можно выбрать из всего семейства функций, представляемых выбранной сетью.
- Оценка - связана с конечностью набора точек для обучения.
- Генерализация - связанная с выбором точек (их положением и количеством) в которых находятся остатки. Ошибки аппроксимации и оценки вместе дают ошибку генерализации.
- Оптимизация - связана с тем, что оптимизатор ищет лучшую функцию из семейства, но находит лишь ее аппроксимацию, с точностью зависящей от *learningrate* и числа шагов.

DeepXDE: A Deep Learning Library for Solving Differential Equations

Более сложная сеть будет обладать меньшей ошибкой аппроксимации и большей ошибкой генерализации - это явление известно и в математической статистике как bias-variance tradeoff. Над данный момент ошибки - это лишь теоретические оценки, вычислить их значения невозможно.

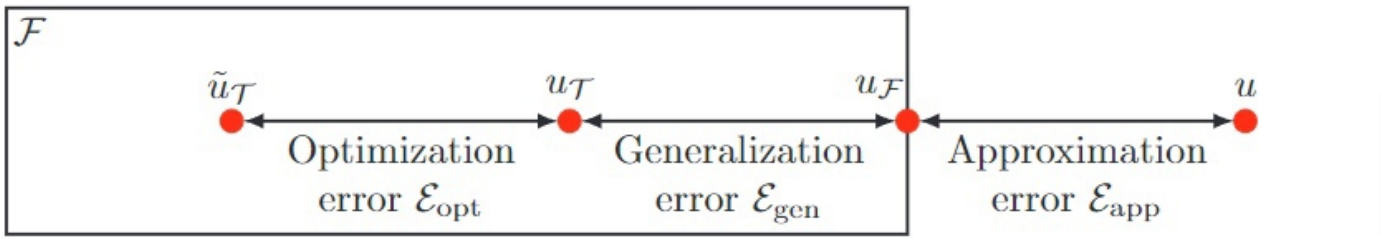


Рис. 4: Виды ошибок

*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.

Implicit Neural Representations with Periodic Activation Functions

Рассмотрим функции класса:

$$F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}}\Phi, \nabla^2_{\mathbf{x}}\Phi, \ldots\right) = 0, \quad \Phi : \mathbf{x} \mapsto \Phi(\mathbf{x}). \tag{5}$$

Будем искать решение некоторой общей задачи среди функций этого класса. Представляем это как задачу нахождения функции Φ, удовлетворяющей набору ограничений M состоящем из условий $\{C_m(\mathbf{a}(\mathbf{x}), \Phi(\mathbf{x}), \nabla\Phi(\mathbf{x}), \ldots)\}_{m=1}^{M}$, каждому из которых соответствует функция Φ и/или ее производные к величинам $\mathbf{a}(\mathbf{x})$:

$$\text{Ищем } \Phi(\mathbf{x}), \text{ при этом } C_m(\mathbf{a}(\mathbf{x}), \Phi(\mathbf{x}), \nabla\Phi(\mathbf{x}), \ldots) = 0, \forall \mathbf{x} \in \Omega_m, m = 1, \ldots, M \tag{6}$$

Implicit Neural Representations with Periodic Activation Functions

Эту оптимизационную задачу можно выразить с помощью функции потерь, которая штрафует отклонения от каждого из ограничений в некоторой области Ω_m :

$$\mathcal{L} = \int_{\Omega} \sum_{m=1}^{M} \mathbf{1}_{\Omega_m}(\mathbf{x}) \left\| C_m(\mathbf{a}(\mathbf{x}), \Phi(\mathbf{x}), \nabla\Phi(\mathbf{x}), \ldots) \right\| d\mathbf{x}, \tag{7}$$

Где $\mathbf{1}_{\Omega_m}(\mathbf{x})$ индикаторная функция, равна 1, если $\mathbf{x} \in \Omega_m$ и 0, когда $\mathbf{x} \notin \Omega_m$. На практике функция потерь реализуется путем выборки Ω. Набор данных $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{a}_i(\mathbf{x}))\}_i$ представляет собой набор кортежей координат $\mathbf{x}_i \in \Omega$ вместе с выборками величин $\mathbf{a}(\mathbf{x}_i)$, которые входят в ограничения. Таким образом, потери в уравнении применяются к координатам $\mathbf{x}_i$ выбранным из набора данных, что дает функцию потерь:

$$\tilde{\mathcal{L}} = \sum_{i \in \mathcal{D}} \sum_{m=1}^{M} \left\| C_m\left(a\left(\mathbf{x}_i\right), \Phi\left(\mathbf{x}_i\right), \nabla\Phi\left(\mathbf{x}_i\right), \ldots\right) \right\| \tag{8}$$

Implicit Neural Representations with Periodic Activation Functions

На практике набор данных $\mathcal{D}$ отбирается динамически во время обучения, аппроксимируя $\mathcal{L}$ лучше по мере роста количества выборок, как при интегрировании Монте-Карло.
Авторы статьи приводят SIREN* в качестве решения задачи.

$$\Phi(\mathbf{x}) = \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto \phi_i\left(\mathbf{x}_i\right) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right). \quad (9)$$

Здесь $\phi_i : \mathbb{R}^{M_i} \mapsto \mathbb{R}^{N_i}$ где i^{th} это уровень сети. Оно состоит из аффинного преобразования, определяемого весовой матрицей $\mathbf{W}_i \in \mathbb{R}^{N_i \times M_i}$ и смещениями $\mathbf{b}_i \in \mathbb{R}^{N_i}$, приложенными к входным данным $\mathbf{x}_i \in \mathbb{R}^{M_i}$, за которыми следует синусоидальная нелинейность, применяемая к каждому компоненту результирующего вектора.

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

Adaptive activation functions accelerate convergence in deep and physics-informed neural networks

В статье* рассматриваются параметрические функции активации, названные адаптивными:

- Функции активации активирует конкретный нейрон.
- Без функции активации веса и смещение выполняют простое линейное преобразование, что соответствует случаю линейной регрессионной модели.
- Нелинейная функция активации позволяет модели решать более сложные задачи.
- Функции активации делают алгоритм обратного распространения ошибки возможным.
- Необходимо выбирать функцию активации, менее подверженной проблеме затухающего и взрывающегося градиента.

Adaptive activation functions accelerate convergence in deep and physics-informed neural networks

$$\text{Sigmoid:} \quad \frac{1}{1+e^{-ax}}, \quad \text{Hyperbolic tangent:} \quad \frac{e^{ax}-e^{-ax}}{e^{ax}+e^{-ax}}, \tag{10}$$
$$\text{ReLU:} \quad \max(0, ax), \quad \text{Leaky ReLU:} \quad \max(0, ax) - v\max(0, -ax).$$

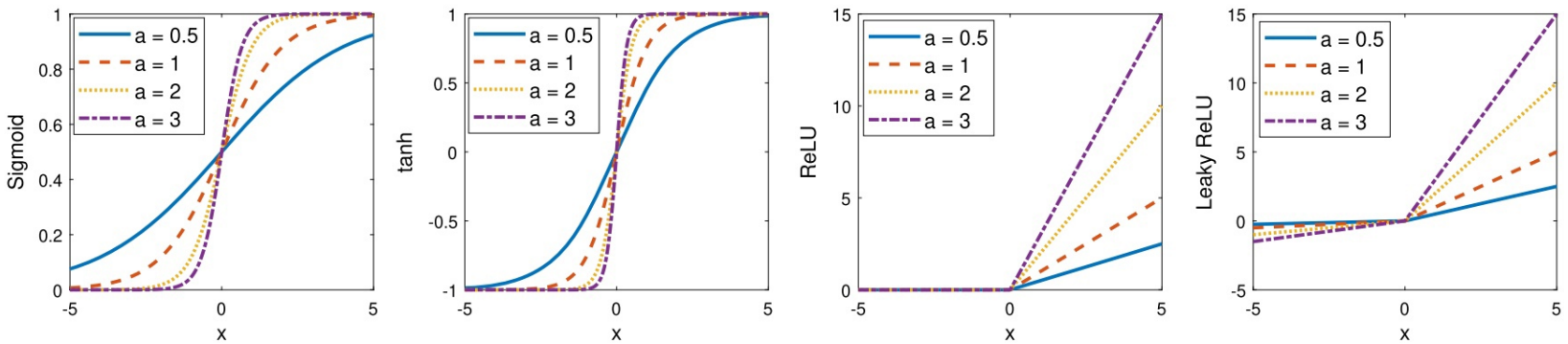
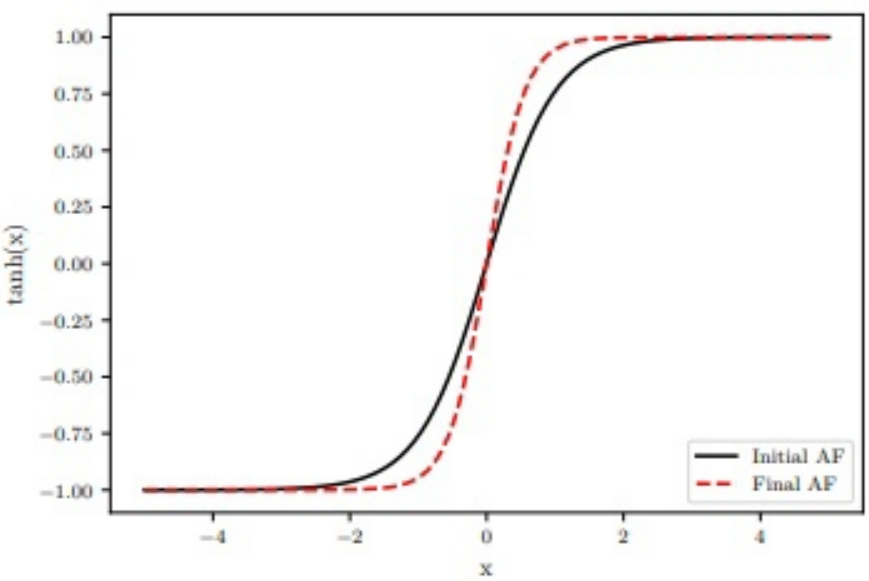


Рис. 5: Графики адаптивных функций активации

Adaptive activation functions accelerate convergence in deep and physics-informed neural networks

Пример изменения функции в процессе обучения в случае функций активации *tanh* и *sin* соответственно.



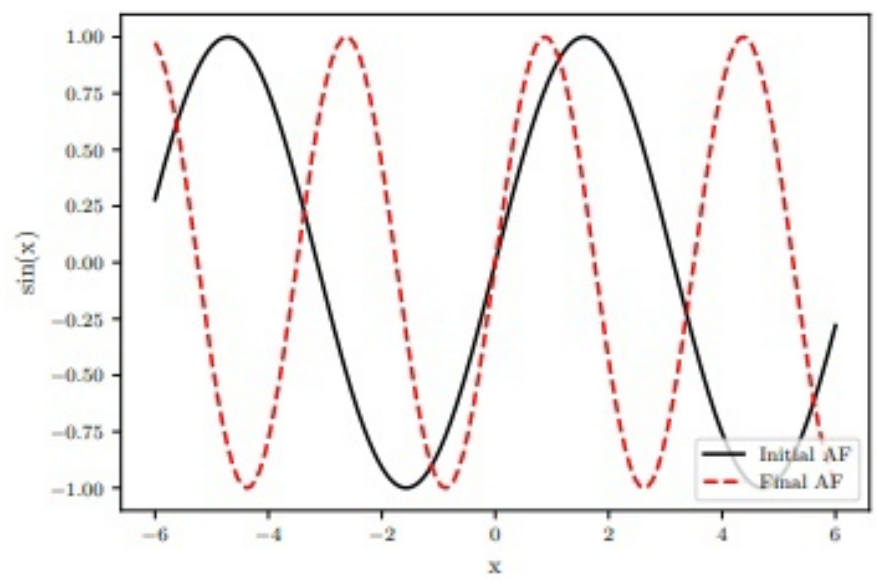


Рис. 6: Начальная и конечная функция

Adaptive activation functions accelerate convergence in deep and physics-informed neural networks

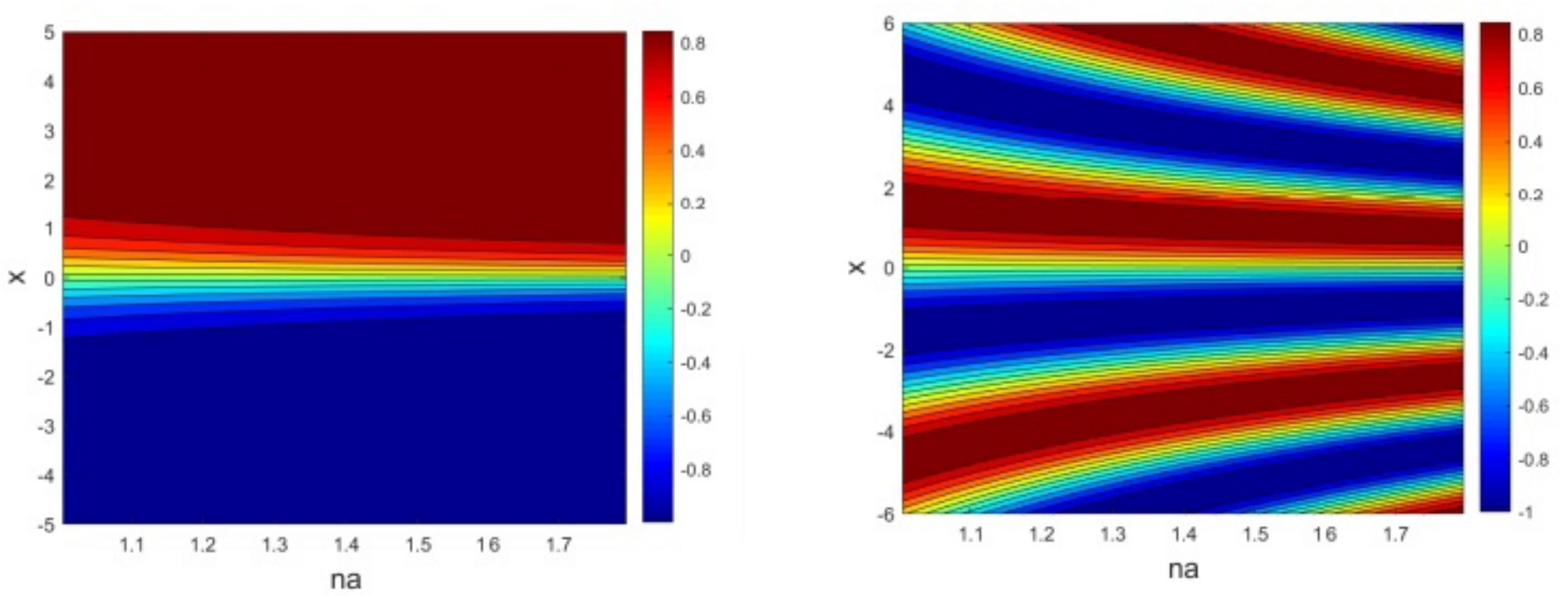


Рис. 7: Плоскость активации

Adaptive activation functions accelerate convergence in deep and physics-informed neural networks

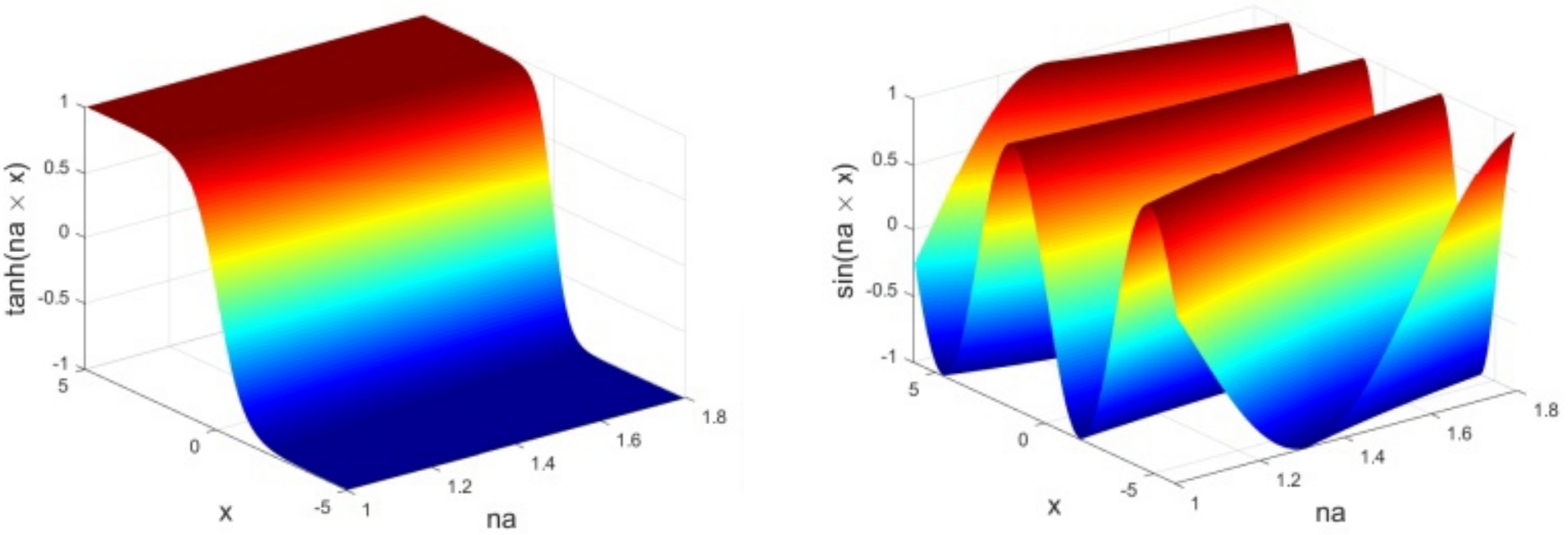


Рис. 8: Поверхность активации

*Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics, 404:109136, March 2020.

Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks

В статье* рассматривается два подхода к локально-адаптивным функциям активации:

- Послойные - L-LAAF.
  $\sigma\left(na^k \mathcal{L}_k\left(z^{k-1}\right)\right), \quad k = 1, 2, \cdots, D-1$
- Понейронные - N-LAAF.
  $\sigma\left(na_i^k \left(\mathcal{L}_k\left(z^{k-1}\right)\right)_i\right), \quad k = 1, 2, \cdots, D-1, \quad i = 1, 2, \cdots, N_k$

Чтобы получить ускорение в сходимости метода, в функцию потерь добавляется член восстановления наклона на основе наклона функции активации:

$$\mathcal{S}(a) = \begin{cases} \dfrac{1}{\frac{1}{D-1}\sum_{k=1}^{D-1} exp(a^k)} & \text{для L-LAAF} \\ \dfrac{1}{\frac{1}{D-1}\sum_{k=1}^{D-1} exp(\frac{\sum_{i=1}^{N_k} a_i^k}{N_k})} & \text{для N-LAAF} \end{cases}$$

Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks

Ряд математических следствий из статьи:

- Доказано, что алгоритмы градиентного спуска не стягиваются к локальным минимумам при практических условиях для адаптивных функций.
- Доказано, что динамика градиента предложенного метода недостижима базовыми методами с любыми (адаптивными) темпами обучения.
- Продемонстрировано, что адаптивные методы ускоряют сходимость, неявно умножая матрицы условий на градиент базового метода без явного вычисления матрицы условий.
- Показано, что различные адаптивные функции активации порождают различные неявные матрицы условий, а предложенные методы с восстановлением наклона ускоряют процесс обучения.

Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks

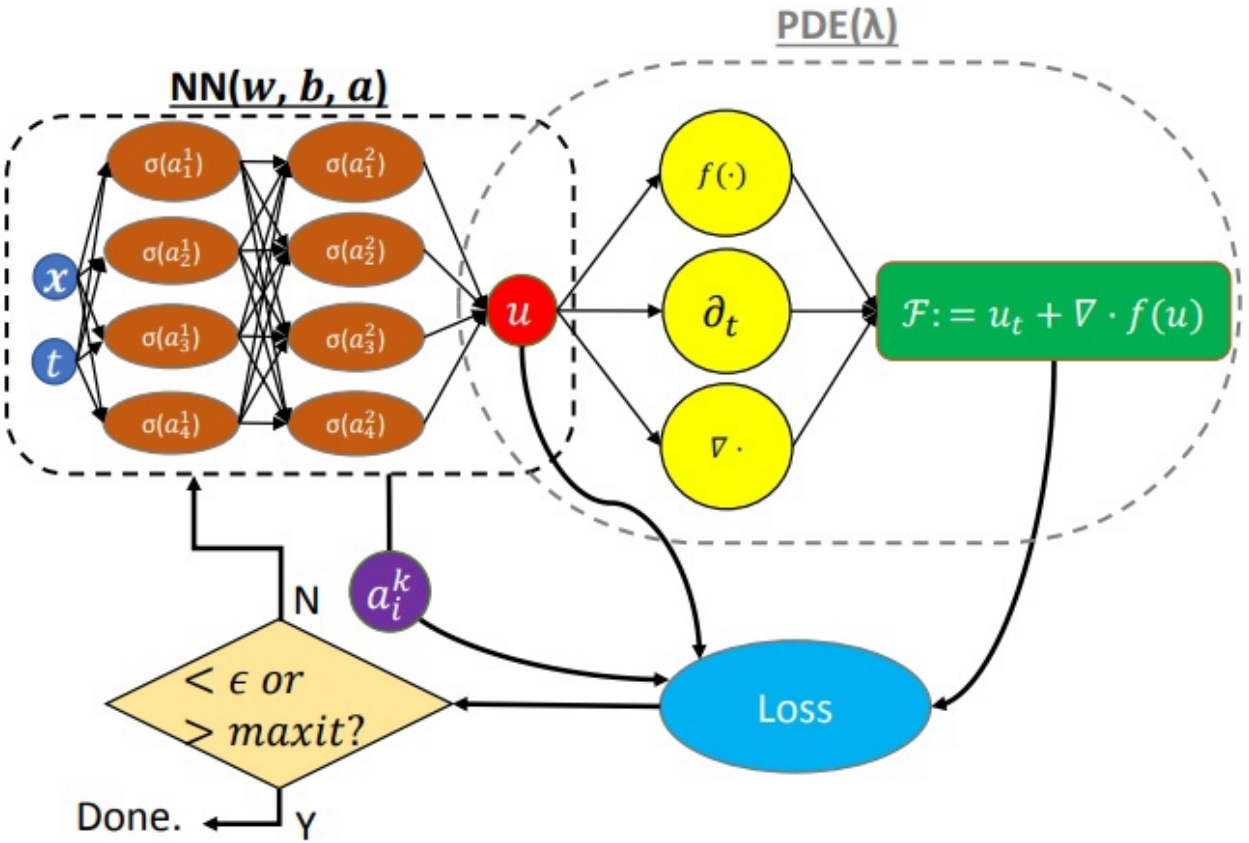


Рис. 9: Схема сети с LAAF для уравнения Бюргерса

Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks

Algorithm 2 LAAF-PINN

1: Обучающие данные: $u_{\hat{\Theta}}$ для $\left\{\mathbf{x}_u^i\right\}_{i=1}^{N_u}$, Остатки: $\mathcal{F}_{\hat{\Theta}}$ для $\left\{\mathbf{x}_f^i\right\}_{i=1}^{N_f}$

2: Создайте нейронную сеть $u_{\hat{\Theta}}$ с параметрами $\hat{\Theta}$.

3: Создайте остаточную нейронную есть $\mathcal{F}_{\hat{\Theta}}$ путем подстановки $u_{\hat{\Theta}}$.

4: Введите функцию потерь:

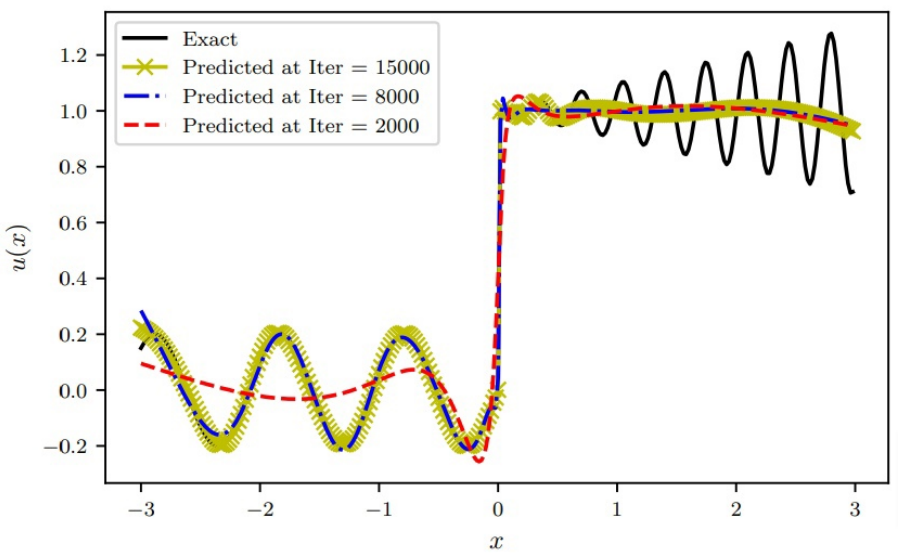
$$\tilde{\mathcal{L}}(\hat{\Theta}) = \frac{\omega_{\mathcal{F}}}{N_f}\sum_{i=1}^{N_f}\left|\mathcal{F}_{\hat{\Theta}}\left(\mathbf{x}_f^i\right)\right|^2 + \frac{\omega_u}{N_u}\sum_{i=1}^{N_u}\left|u^i - u_{\hat{\Theta}}\left(\mathbf{x}_u^i\right)\right|^2 + \omega_a \mathcal{S}(a),$$

5: Найдите лучшие параметры $\hat{\Theta}^*$ для минимизации функции потерь $\tilde{\mathcal{L}}(\hat{\Theta})$:

$$\hat{\Theta}^* = \arg\min_{\hat{\Theta}\in\hat{\mathcal{V}}} \tilde{\mathcal{L}}(\hat{\Theta})$$

Locally adaptive activation functions with slope recovery term for deep and physics-informed neural networks

В статье демонстрируется, что использование адаптивных функций активации значительно улучшает осциллирующие решения.



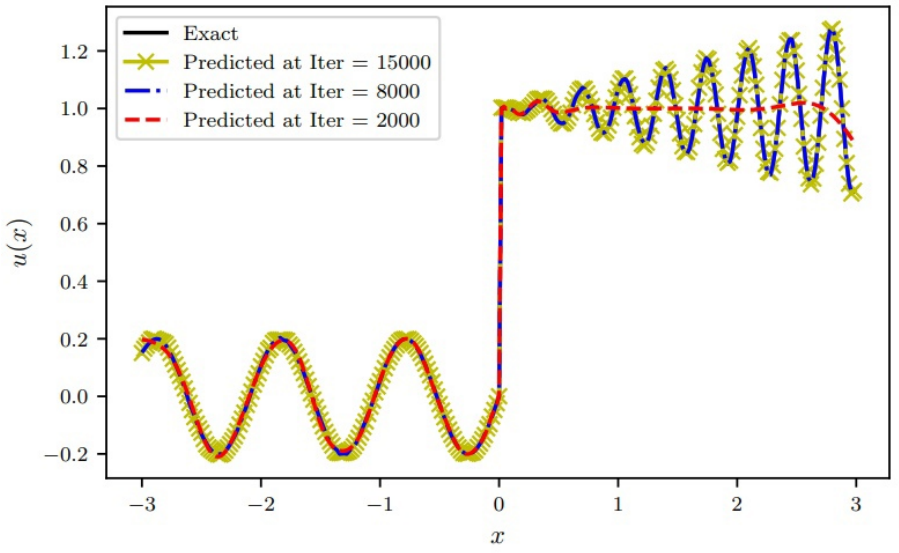


Рис. 10: Результаты сравнения для уравнения Пуассона

*Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476(2239):20200334, July 2020.

The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?

Еще одна важная идея была изложена в обзорной статье*, в которой PINN рассматривался как новый перспективный метод. Идея заключается в использовании сразу двух оптимизаторов - ADAM + L-BFGS-B. Авторы провели анализ эволюции ошибки обучения, из которого стало понятно, что ошибка обучения оптимизатора ADAM стабилизируется примерно в диапазоне $5 \cdot 10^{-3} - 10^{-2}$ после 2000 эпох и далее не наблюдается какого-либо улучшения. Оптимизатор L-BFGS-B дает ошибку от $5 \cdot 10^{-3} - 10^{-2}$ до $2,79 \cdot 10^{-5}$ и отвечает за значительное уменьшение ошибки обучения. Однако важно, что L-BFGS-B не используется в начале обучения, поскольку может быстро сходиться к локальному минимуму в задаче оптимизации.

The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?

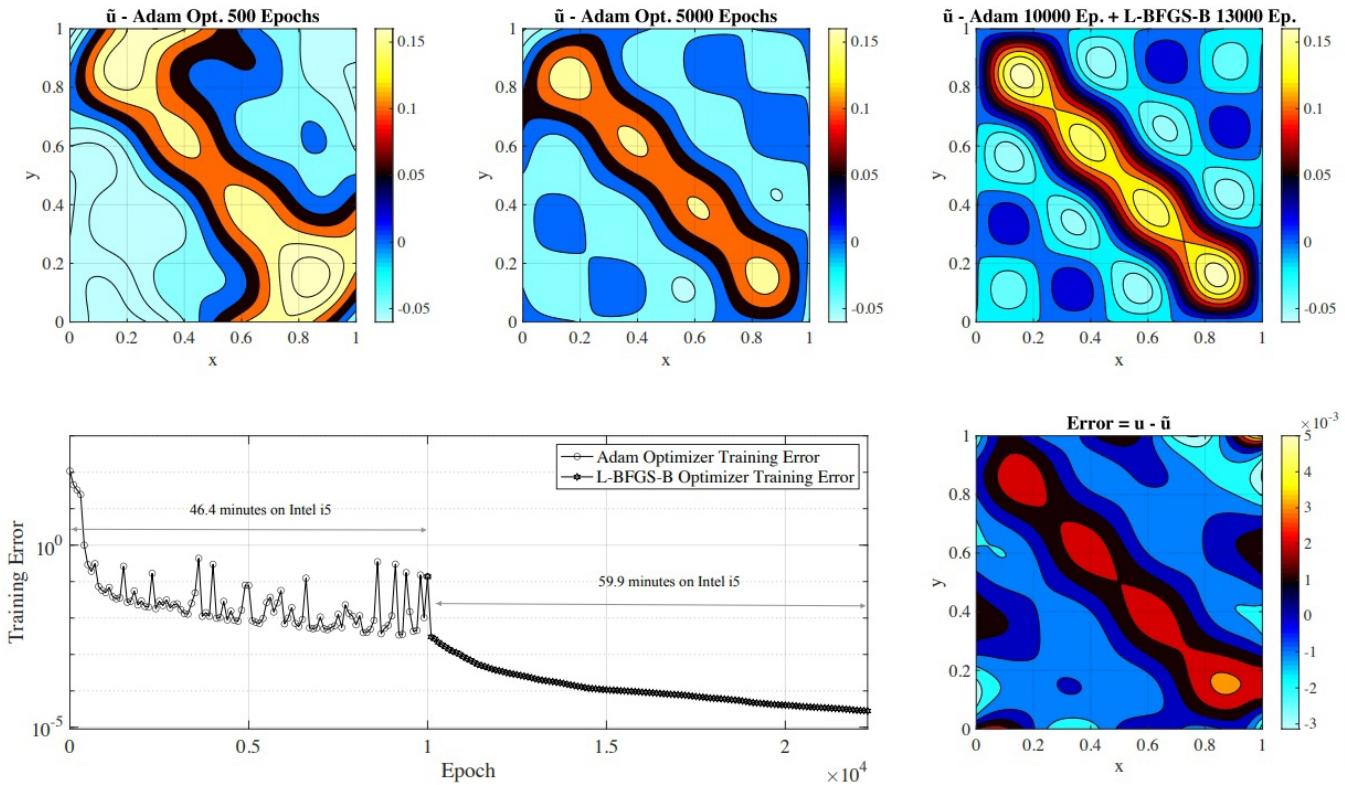


Рис. 11: Сравнение различных оптимизаторов на уравнении Пуассона

The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers?

Статья фокусируется на анализе положения PINN среди других методов, возможной замене или ускорения традиционных методов решения. Однако, на текущий момент, PINNs не могут полностью заменить традиционные методы. Гибридные стратегии, объединяющие PINNs с традиционными подходами, представляют собой наиболее перспективную область. Использование GPUs существенно улучшает производительность, но отсутствует GPU-реализация оптимизатора L-BFGS-B.

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

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

В основе рассуждений об аппроксимации решений нейронными сетями лежит теорема Пинкуса.

Theorem 1

Пусть $\mathrm{m}^i \in \mathbb{Z}_+^d, i = 1, \ldots, s$, и есть набор $m = \max_{i=1,\ldots,s} \left| \mathrm{m}^i \right|$. Предположим, что $\sigma \in C^m(\mathbb{R})$ и σ не является полиномом. Тогда пространство нейронных сетей с одним скрытым слоем

$$\mathcal{M}(\sigma) := \operatorname{span}\left\{\sigma(\mathbf{w} \cdot \mathbf{x} + b) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\right\}$$

есть полносвязная сеть, т. е. $C^{\mathbf{m}^1,\ldots,\mathbf{m}^s}\left(\mathbb{R}^d\right) := \cap_{i=1}^s C^{\mathbf{m}^i}\left(\mathbb{R}^d\right)$, т.е. для любого $f \in C^{\mathbf{m}^1,\ldots,\mathbf{m}^s}\left(\mathbb{R}^d\right)$, любого компакта K $\mathbb{R}^d$ и любого $\varepsilon > 0$ существует $g \in \mathcal{M}(\sigma)$, удовлетворяющий условию $\max_{\mathbf{x}\in K} \left|D^{\mathbf{k}} f(\mathrm{x}) - D^{\mathbf{k}} g(\mathbf{x})\right| < \varepsilon$ для всех $\mathrm{k} \in \mathbb{Z}_+^d$, для которых $\mathrm{k} \leq \mathrm{m}^i$ для некоторого i.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Эта теорема показывает, что нейронные сети прямого сигнала с достаточным количеством нейронов могут равномерно и непрерывно аппроксимировать любую функцию и ее частные производные. Однако на практике нейронные сети имеют ограниченный размер. Пусть $\mathcal{F}$ обозначает семейство всех функций, которые могут быть представлены выбранной нами архитектурой нейронной сети. Решение u вряд ли принадлежит семейству $\mathcal{F}$, и мы определим $u_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} \|f - u\|$ как лучшая функция в $\mathcal{F}$, близкая к u.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Пусть U — ограниченная область в $\mathbb{R}^d$, принадлежащая как минимум классу $C^{0,1}$, и Γ — замкнутое подмножество ∂U. Пусть μ_r и μ_b — распределения вероятностей, определенные на U и Γ соответственно. Пусть ρ_r — плотность вероятности μ_r относительно d-мерной меры Лебега на U. Пусть ρ_b — плотность вероятности μ_b относительно $(d-1)$-мерной меры Хаусдорфа на Γ.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Lemma 2

- Пусть ρ_r и ρ_b в $\bar{U}$ и Γ соответственно. Кроме того, $\inf_U \rho_r > 0$ и $\inf_\Gamma \rho_b > 0$.
- Для $\epsilon > 0$ существуют разбиения U и Γ $\left\{U_j^\epsilon\right\}_{j=1}^{K_r}$ и $\left\{\Gamma_j^\epsilon\right\}_{j=1}^{K_b}$, зависящие от ϵ такие, что для каждого j существуют кубы $H_\epsilon\left(\mathbf{z}_{j,r}\right)$ и $H_\epsilon\left(\mathbf{z}_{j,b}\right)$ с длиной стороны ϵ с центром в $\mathbf{z}_{j,r} \in U_j^\epsilon$ и $\mathbf{z}_{j,b} \in \Gamma_j^\epsilon$ соответственно, удовлетворяющие $U_j^\epsilon \subset H_\epsilon\left(\mathbf{z}_{j,r}\right)$ и $\Gamma_j^\epsilon \subset H_\epsilon\left(\mathbf{z}_{j,b}\right)$.
- $c_r, c_b > 0$: $\forall \epsilon > 0$, полученные выше разбиения удовлетворяют $c_r \epsilon^d \leq \mu_r\left(U_j^\epsilon\right)$ и $c_b \epsilon^{d-1} \leq \mu_b\left(\Gamma_j^\epsilon\right)$ для всех j. $C_r, C_b > 0$: $\forall \mathbf{x}_r \in U$ и $\forall \mathbf{x}_b \in \Gamma, \mu_r\left(B_\epsilon\left(\mathbf{x}_r\right) \cap U\right) \leq C_r \epsilon^d$ и $\mu_b\left(B_\epsilon\left(\mathbf{x}_b\right) \cap \Gamma\right) \leq C_b \epsilon^{d-1}$, $\mathrm{B}_\epsilon(\mathbf{x})$ — замкнутый шар радиуса ϵ с центром в $\mathbf{x}$. Здесь C_r, c_r зависят только от (U, μ_r), а C_b, c_b зависят только от (Γ, μ_b).
- Когда $d = 1$, мы предполагаем, что все граничные точки доступны.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Theorem 3

Пусть m_r и m_b — количество выборок из μ_r и μ_b соответственно. Пусть для некоторого $0 < \alpha \leq 1$ $h, f, g, R_r(h), R_b(h)$ удовлетворяют условиям

$$[\mathcal{L}[h]]^2_{a;U} \leq R_r(h) < \infty, \quad [\mathcal{B}[h]]^2_{a;\Gamma} \leq R_b(h) < \infty, \quad [f]_{a;U'}[g]_{a;} < \infty.$$

Пусть $\lambda = (\lambda_r, \lambda_b)$ — фиксированный вектор. Пусть $\hat{\lambda}^R_m = \left(\hat{\lambda}^R_{r,m}, \hat{\lambda}^R_{b,m}\right)$ — вектор, элементы которого необходимо определить.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Theorem 3

- Для $d \geq 2$ с вероятностью не менее $\left(1 - \sqrt{m_r}\left(1 - 1/\sqrt{m_r}\right)^{m_r}\right)\left(1 - \sqrt{m_b}\left(1 - 1/\sqrt{m_b}\right)^{m_b}\right)$, имеем

$$\mathrm{Loss}^{PINN}(h;\lambda) \leq C_m \cdot \mathrm{Loss}_m\left(h;\lambda,\hat{\lambda}^R_m\right) + C'\left(m_r^{-\hat{A}} + m_b^{-\frac{\alpha}{d-1}}\right),$$

где $\kappa_r = \frac{C_r}{c_r}, \kappa_b = \frac{c_b}{c_b}, C_m = 3\max\left\{\kappa_r\sqrt{d}^d m_r^{\frac{1}{2}}, \kappa_b\sqrt{d}^{d-1} m_b^{\frac{1}{2}}\right\}, C'$ — универсальная константа, зависящая только на $\lambda, d, c_r, c_b, \alpha, f, g$ и

$$\hat{\lambda}^R_{r,m} = \frac{3\lambda_r\sqrt{d}^{2\alpha} c_r^{-\frac{2\alpha}{d}}}{C_m} \cdot m_r^{-\frac{\alpha}{d}}, \quad \hat{\lambda}^R_{b,m} = \frac{3\lambda_b\sqrt{d}^2\ c_b^{-\frac{2\pi}{d-1}}}{C_m} \cdot m_b^{-\frac{\alpha}{d-1}}.$$

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Theorem 3

- Для $d = 1$, с вероятностью не менее $1 - \sqrt{m_r}\left(1 - 1/\sqrt{m_r}\right)^{m_r}$, имеем

$$\mathrm{Loss}^{PINN}(h;\lambda) \leq C_m \cdot \mathrm{Loss}_m\left(h;\lambda,\hat{\lambda}_m^R\right) + C' m_r^{-\alpha},$$

где $C_m = 3\kappa_r m_r^{\frac{1}{2}}, \hat{\lambda}_{r,m}^R = \frac{\lambda_r c_r^{-2\pi}}{\kappa_r} \cdot m_r^{-\alpha-\frac{1}{2}}, \hat{\lambda}_{b,m}^R = 0$ и C' — универсальная константа, зависящая только от $\lambda_r, c_r, \alpha, f$.

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Lemma 4

Пусть k — старший порядок производной, указанной в УЧП. Для некоторого $0 < \alpha \leq 1$ пусть $f \in C^{0,\alpha}(U)$ и $g \in C^{0,\alpha}(\Gamma)$.

- Пусть для каждого $\boldsymbol{m}$ $\mathcal{H}_m$ — класс нейронных сетей в $C^{k,\alpha}(U) \cap C^{0,\alpha}(\,barU)$ такой, что для любых $h \in \mathcal{H}_m$, $\mathcal{L}[h] \in C^{0,\alpha}(U)$ и $\mathcal{B}[h] \in C^{0,\alpha}(\Gamma)$.
- Для каждого $m\mathcal{H}_m$ содержит сеть u_m^*, удовлетворяющую $\mathsf{Loss}_m^{\mathrm{PINN}}\left(u_m^*; \lambda\right) = 0$.
- И $\sup_m \left[\mathcal{L}\left[u_m^*\right]\right]_{a;U} < \infty, \quad \sup_m \left[\mathcal{B}\left[u_m^*\right]\right]_{\alpha;} < \infty$

On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs

Theorem 5

Пусть m_r и m_b — количество выборок из μ_r и μ_b соответственно, а $m_r = O\left(m_b^{\frac{d}{d-1}}\right)$.

Пусть $\lambda_{m_r}^R$ — вектор, удовлетворяющий (3.2). Пусть $h_{m_r} \in \mathcal{H}_{m_r}$ — минимизатор регуляризованных потерь Гёльдера $\mathsf{Loss}_{m_r}\left(\cdot; \lambda, \lambda_{m_r}^R\right)$ (3.3). Тогда имеет место следующее:

- С вероятностью не менее $\left(1 - \sqrt{m_r}\left(1 - c_r/\sqrt{m_r}\right)^{m_r}\right)\left(1 - \sqrt{m_b}\left(1 - c_b/\sqrt{m_b}\right)^{m_b}\right)$ по выборкам,
$$\mathsf{Loss}^{PINN}\left(h_{m_r}; \lambda\right) = O\left(m_r^{-\frac{g}{d}}\right).$$
- С вероятностью 1 по выборкам,
$$\lim_{m_r \to \infty} \mathcal{L}\left[h_{m_r}\right] = f \text{ в } C^0(U), \quad \lim_{m_r \to \infty} \mathcal{B}\left[h_{m_r}\right] = g \text{ в } C^0(\Gamma).$$

Источники

[1] 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.

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

[3] 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.

[4] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind.
Automatic differentiation in machine learning: a survey, 2018.

Источники

[5] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis.
Adaptive activation functions accelerate convergence in deep and physics-informed neural networks.
Journal of Computational Physics, 404:109136, March 2020.

[6] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis.
Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks.
Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 476(2239):20200334, July 2020.

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

[8] 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

[9] Siddhartha Mishra and Roberto Molinaro.
Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes, 2023.

Спасибо за внимание