Министерство науки и высшего образования российской федерации федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)

ОТЧЁТ

по дисциплине «Проектная практика»
на тему «Решение начально-краевой задачи для уравнения Кортевега-де Вриза (КдВ) с использованием PINN»

Группа Б23-215

Студенты Аллабергенов М., Денисов А.

Руководители работы
к.ф.-м.н., доценты Карачурин Р.Н, Ладыгин С.А

Москва 2024

Аннотация

Отчёт о выполнении задачи:

Реализовать PINN метод решения начально-краевой задачи для уравнения Кортевега-де Вриза (КдВ). Исследовать влияние количества нейронов и скрытых слоёв на точность и эффективнось решения.

Уравнение КдВ:

$$\frac{\partial u}{\partial t} + 6 * u * \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, x \in [-L, L]. \quad (0.1)$$

Начальное и граничные условия:

$$\begin{cases} u(x,0) = f(x),\ x\epsilon[-L,L], \\ u(-L,t) = u(L,T) = 0, t > 0, \\ \frac{\partial u}{\partial x}(-L,t) = 0, t > 0. \end{cases} \quad (0.2)$$

Содержание

Введение

Исследование посвящено решению начально-краевой задачи для уравнения Кортевега-де Вриза с помощью PINN (Physics-Informed Neural Networks). Дифференциальные уравнения - очень важная часть математики, это целый математический аппарат (Дифференциальные уравнения в частных производных) широко применяемый при разработке моделей в разных областях науки и техники.

Однако, явное решение этих уравнений в аналитическом виде оказывается возможным только в частных простых случаях. Поэтому для анализа математических моделей, основанных на дифференциальных уравнениях, используются приближенные численные методы. В данном исследовании для решения поставленной задачи используется метод PINN.

Physics Informed Neural Network (PINN), также называемые теоретически обученными нейронными сетями (TTN), представляющие собой тип апроксиматоров универсальных функций, которые могут встраивать знания о любых физических законах, управляющие данными в процессе обучения, и могут быть описаны дифференциальными уравнениями в частных производных.

1. Физическое описание

Уравне́ние Кортеве́га — де Фри́за (уравнение КдФ (КдВ); также встречается написание де Вриза, де Вриса, де Фриса, Де Фриса; англ. Korteweg–de Vries equation) — нелинейное уравнение в частных производных третьего порядка, играющее важную роль в теории нелинейных волн, в основном гидродинамического происхождения. Впервые было получено Жозефом Буссинеском в 1877 году, но подробный анализ был проведён уже Дидериком Кортевегом и Густавом де Врисом в 1895 году.

Для уравнения Кортевега — де Вриза найдено большое количество точных решений, представляющих собой стационарные нелинейные волны. В том числе данное уравнение имеет решения солитонного типа следующего вида:

$$u(x,t) = \frac{2*k^2}{\cosh^2[k*(x-4*k^2*t-x_0)]}, \quad (1.1)$$

где k — свободный параметр, определяющий высоту и ширину солитона, а также его скорость; x_0 — также произвольная константа, зависящая от выбора начала отсчёта оси x. Особое значение солитонам придаёт тот факт, что любое начальное возмущение, экспоненциально спадающее на бесконечности, с течением времени эволюционирует в конечный набор солитонов, разнесённых в пространстве. Точный поиск этих решений может быть проведён регулярным образом при помощи метода обратной задачи рассеяния.

Периодические решения уравнения Кортевега — де Вриза имеют вид кноидальных волн, описываемых эллиптическими интегралами:

$$x - c*t - x_0 = \int (2*E + c*u^2 - 2*u^3)^{-\frac{1}{2}} du, \quad (1.2)$$

где c, E — параметры волны, определяющие её амплитуду и период.

2. Обзор метода

Идея состоит в решении начально-краевой задачи, представляя апроксимированное численное решение с помощью нейронной сети и обучая полученную сеть, чтобы удовлетворить условиям, требуемые дифференциальным уравнением, а также начальным и граничными условиями. Это решение включает нейронную сеть в качестве основного элемента аппроксимации, параметры которой (веса и смещения) подстраиваются для минимизации функции ошибки (Loss). Для обучения используются методы оптимизации, которые требуют вычисления градиента ошибки по параметрам сети.

Разберем метод на нашем уравнении КдВ:

$$\frac{\partial u}{\partial t} + 6 * u * \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0, x \in [-L, L].$$

Возьмем, например, $L = 10$ и $t \in [0, 10]$. Вид граничных условий такой:

$$\begin{cases} u(x,0) = \dfrac{2}{\cosh^2 x}, \quad x\epsilon[-10,10], \\ u(-10,t) = u(10,T) = 0, t \in (0,10], \\ \dfrac{\partial u}{\partial x}(-10,t) = 0, t \in (0,10]. \end{cases} \tag{2.1}$$

Так как известно, что (2.1) является решением (0.1) при $x\epsilon R$, то в качестве начального условия будем использовать:

$$u(x,0) = \frac{2}{\cosh^2 x}, \text{из (2.1)при } t = 0, k = 1, x_0 = 0.$$

Для решения данной задачи решение аппроксимируется нейронной сетью к истинному:

$$NN(x,t) \approx u(x,t).$$

Если $NN(x,t)$ является истинным решением, то должно выполняться равенство

$$\frac{\partial NN(x,t)}{\partial t} + 6 * NN(x,t) * \frac{\partial NN(x,t)}{\partial x} + \frac{\partial^3 NN(x,t)}{\partial x^3} = 0, x \in [-10,10], t \in (0,10]. \tag{2.2}$$

Таким образом, мы превращаем это условие в нашу функцию потерь:

$$Loss_{eq} = \frac{1}{n}\sum_{\substack{i=1 \\ x_i\in[-10,10] \\ t_i\in(0,10]}}^{n} \left(\frac{\partial NN(x_i,t_i)}{\partial t} + 6 * NN(x_i,t_i) * \frac{\partial NN(x_i,t_i)}{\partial x} + \frac{\partial^3 NN(x_i,t_i)}{\partial x^3}\right)^2, \tag{2.3}$$

где $Loss_{eq}$ - среднеквадратичная функция потерь самого уравнения

Однако, нужно еще учесть начальные и граничные условия. Они также учитываются в среднеквадратичной форме:

$$Loss_{ic} = \frac{1}{n}\sum_{\substack{i=1 \\ x_i\in[-10,10]}}^{n} \left(NN(x_i,0) - \frac{2}{\cosh^2 x}\right)^2, \text{(потери из начального условия)} \tag{2.4}$$

(2.5)

(2.6)

$Loss_{bc} = \frac{1}{n}\sum^{n}_{\substack{i=1 \\ t_i \in (0,10]}} (NN(-10, t_i))^2 + (NN(10, t_i))^2$, (потери из граничных условий)

$$Loss_{dbc} = \frac{1}{n}\sum^{n}_{\substack{i=1 \\ t_i \in (0,10]}} (\frac{\partial NN(-10, t_i)}{\partial x})^2,$$

(потери из граничных условий после дифференцирования)

Таким образом конечная функция потерь определяется как сумма:

$$Loss = Loss_{eq} + Loss_{ic} + Loss_{bc} + Loss_{dbc} \quad (2.7)$$

Таким образом, при минимизации функции потерь получаемые численные значения из нейронной сети стремятся к реальным. Минимизация осуществляется при помощи оптимизатора.

3. Реализация

Описание модели нейронной сети

Решение основывается на библиотеке Python TensorFlow и Keras - это открытые программные библиотеки для машинного обучения, решения задач, построения и тренировки нейронных сетей с целью автоматического нахождения и классификации образов.

В качестве оптимизатора в данной работе применяется оптимизатор Adam.

Архитектура модели

Для изучения эффективности и точности решения были использованы следующие модели:

<table>
<tr><td rowspan="2">Количество слоев</td><td colspan="3">Количество нейронов в слое</td></tr>
<tr><td>Входной слой</td><td>Скрытые слои</td><td>Выходной слой</td></tr>
<tr><td>3</td><td rowspan="12">2</td><td rowspan="6">50</td><td rowspan="12">1</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td></tr>
<tr><td rowspan="6">4</td><td>8</td></tr>
<tr><td>16</td></tr>
<tr><td>32</td></tr>
<tr><td>64</td></tr>
<tr><td>96</td></tr>
<tr><td>128</td></tr>
</table>

Табл. 3.1: Описание структуры моделей

Функции активации

В модели используется встроенная функция активации ‘tanh’.

4. Представление результатов

После обучения нейронной сети были получены следующие результаты:

Количество слоев	Количество нейронов в скрытых слоях	Конечное значение $\mathcal{L}oss$
3	50	0.06087878
4		0.027840497
5		0.014964926
6		0.0061783395
7		0.0062547196
8		0.006010604
4	8	0.02473097
	16	0.021574993
	32	0.015606209
	64	0.00707663
	96	0.0075323
	128	0.010853056

Табл. 4.1: Представление результатов конечного Loss моделей

Также для сравнения были построены анимированные графики полученных значений и истинной функции: anim_pred.mp4 (график ожидаемой функции (1.1)), anim_true.mp4 (график полученной функции). Было проведено дополнительное исследование, в ходе которого были построены графики extra_ic.mp4 (график полученной функции с дополнительным условием постоянства высоты волны) и without_bc.mp4 (график полученной функции без граничного условия отвечающего за $Loss_{bc}$).

5. Вывод

Анализируя полученные значения функции Loss, можно сделать выводы:

1) Увеличение количества слоев увеличивает точность и эффективность решения, т.к. значение Loss убывает. Однако, начиная уже с 7-ого слоя, прирост эффективности значительно замедляется, поэтому для данного набора нейронов наиболее эффективно иметь шесть слоев в модели

2) Увеличение количества нейронов увеличивает точность и эффективность решения, т.к. значение Loss убывает. Однако, эффективность начинает падать, начиная с 96 нейронов в скрытых слоях. Как видно из таблицы (4.1), начиная с 96 нейронов функция потерь начинает возрастать. Отсюда следует, что эффективное количество в модели с 4-мя слоями – 64 нейронов.

Анализируя построенные графики, можно увидеть, что они ведут себя практически одинаково. Однако, в графике полученной функции волна начинает быстро терять высоту приближаясь к правой граничной точке по «*x*», когда в графике ожидаемой функции волна визуально не теряет высоту. Из дополнительного исследования получен вывод, что потеря высоты обуславливается либо отсутствием дополнительного начального условия, либо наличием граничных условий, либо слишком большим влиянием граничных условий на процесс обучения нейронной сети. Одним из решений данной проблемы можно взять уменьшение значимости граничных условий в процессе обучения, т.е. изменение коэффициента перед $Loss_{bc}$. Например:

$$Loss = Loss_{eq} + Loss_{ic} + 0.01 * Loss_{bc} + Loss_{dbc}$$

Список литературы и интернет-ресурсов

[1] https://www.tensorflow.org/api_docs — TensorFlow документация

[2] https://keras.io/api/ — Keras документация

[3] Кудряшов Методы нелинейной математич 2008 – учебник с информацией об уравнении КдВ

[4] Physics-informed neural networks - Wikipedia – информация о методе PINN из Википедии

[5] Adam — PyTorch 2.5 documentation – документация об оптимизаторе Adam (PyTorch)

[6] GunBladeMan/KortevegaDeVriz: PINN method for solution this equation – ссылка на репозиторий с кодом, графиками и результатами