Теоретический анализ решения уравнения Ван дер Поля с помощью Physics-Informed Neural Networks

Аннотация

В настоящей работе представлен теоретический анализ решения нелинейного дифференциального уравнения Ван дер Поля с использованием метода *Physics-Informed Neural Networks* (PINN). Основной акцент сделан на математической постановке задачи, выводе градиентов, архитектурных особенностях и параметризации. Предложен двухэтапный подход: предобучение на численном решении и последующее дообучение с минимизацией невязки уравнения. Приведены конкретные гиперпараметры и результаты экспериментов, выполненных на CPU.

1 Постановка задачи

Рассматривается уравнение Ван дер Поля второго порядка:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0, \tag{1}$$

где $\mu = 5$ — параметр нелинейного затухания. Начальные условия: $x(0) = 2$, $\dot{x}(0) = 0$.

Для удобства перейдём к системе первого порядка, введя переменную $v = \dot{x}$:

$$\begin{cases} \dot{x} = v, \\ \dot{v} = \mu(1 - x^2)v - x. \end{cases} \tag{2}$$

Цель: найти нейросетевую аппроксимацию $\hat{x}(t), \hat{v}(t)$, удовлетворяющую (2) и начальным условиям.

2 Метод PINN

Нейронная сеть $\mathcal{N}_\theta(t) = [\hat{x}(t), \hat{v}(t)]^\top$ обучается минимизировать функцию потерь:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{ODE}}(\theta) + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}(\theta) + \alpha\mathcal{L}_{\text{reg}}(\theta), \tag{3}$$

где:

- $\mathcal{L}_{\text{ODE}} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\frac{d\hat{x}}{dt}(t_i) - \hat{v}(t_i)\right)^2 + \left(\frac{d\hat{v}}{dt}(t_i) - (\mu(1 - \hat{x}(t_i)^2)\hat{v}(t_i) - \hat{x}(t_i))\right)^2\right]$,
- $\mathcal{L}_{\text{IC}} = (\hat{x}(0) - 2)^2 + (\hat{v}(0) - 0)^2$,

- $\mathcal{L}_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}(t_i)^2 + \hat{v}(t_i)^2)$ — L2-регуляризация выходов.

Коэффициенты: $\lambda_{\text{IC}} = 100$ на первых 100 эпохах, затем $\lambda_{\text{IC}} = 10$; $\alpha = 0.001$.

3 Вычисление градиентов

Производные $\frac{d\hat{x}}{dt}, \frac{d\hat{v}}{dt}$ вычисляются автоматически с помощью `torch.autograd.grad`. Рассмотрим детали для одной точки t.

Пусть $\mathcal{N}_\theta(t) = [x_\theta(t), v_\theta(t)]^\top$. Тогда:

$$\frac{dx_\theta}{dt}(t) = \nabla_t x_\theta(t) = \sum_{j=1}^{d} \frac{\partial x_\theta}{\partial z_j} \cdot \frac{\partial z_j}{\partial t}, \tag{4}$$

где z_j — промежуточные активации сети. В PyTorch это реализуется как:

```
dx_dt = torch.autograd.grad(x, t, grad_outputs=torch.ones_like(x), create_graph=True)
```

Аналогично для dv/dt. Градиенты по параметрам θ вычисляются с помощью обратного распространения ошибки (backpropagation):

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathcal{L}_{\text{ODE}} + \lambda_{\text{IC}} \nabla_\theta \mathcal{L}_{\text{IC}} + \alpha \nabla_\theta \mathcal{L}_{\text{reg}}. \tag{5}$$

Для $\mathcal{L}_{\text{ODE}}$:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\text{ODE}} = \frac{2}{N}\sum_{i=1}^{N} \Bigg[&\left(\frac{d\hat{x}}{dt}(t_i) - \hat{v}(t_i)\right) \cdot \nabla_\theta \left(\frac{d\hat{x}}{dt}(t_i) - \hat{v}(t_i)\right) \\ &+ \left(\frac{d\hat{v}}{dt}(t_i) - f(\hat{x}(t_i), \hat{v}(t_i))\right) \cdot \nabla_\theta \left(\frac{d\hat{v}}{dt}(t_i) - f(\hat{x}(t_i), \hat{v}(t_i))\right) \Bigg],\end{aligned} \tag{6}$$

где $f(x, v) = \mu(1 - x^2)v - x$.

Заметим, что $\nabla_\theta \frac{d\hat{x}}{dt} = \frac{d}{dt}\nabla_\theta \hat{x}$ — это следствие коммутативности дифференцирования по параметрам и времени при условии, что сеть дифференцируема.

4 Архитектуры нейросетей

Рассмотрены две архитектуры:

4.1 SimpleMLP

$$\mathcal{N}_\theta(t) = W_3 \cdot \tanh(W_2 \cdot \tanh(W_1 \cdot t + b_1) + b_2) + b_3, \tag{7}$$

где: - $W_1 \in \mathbb{R}^{h\times 1}$, $W_2 \in \mathbb{R}^{h\times h}$, $W_3 \in \mathbb{R}^{2\times h}$, - $b_1, b_2 \in \mathbb{R}^h$, $b_3 \in \mathbb{R}^2$, - $h = 64$ — размер скрытого слоя.

Инициализация весов:

- $W_1, W_2 \sim \mathcal{U}(-0.1, 0.1)$,
- $W_3 \sim \mathcal{U}(-0.01, 0.01)$,
- $b_1 = b_2 = b_3 = 0$.

Выходы ограничиваются: $\hat{x} = 3 \cdot \tanh(x_{\text{raw}})$, $\hat{v} = 3 \cdot \tanh(v_{\text{raw}})$.

4.2 ResidualMLP

$$\mathcal{N}_\theta(t) = W_{\text{out}} \cdot \tanh(W_3 \cdot \tanh(W_2 \cdot \tanh(W_1 \cdot t + b_1) + b_2) + \text{residual} + b_3) + b_{\text{out}}, \quad (8)$$

где $\text{residual} = W_1 \cdot t + b_1$ — резидуальная связь.
Инициализация:

- $W_1, W_2, W_3 \sim \mathcal{U}(-0.05, 0.05)$,
- $W_{\text{out}} \sim \mathcal{U}(-0.005, 0.005)$,
- смещения равны нулю.

5 Гиперпараметры обучения

Все эксперименты проводились на CPU. Параметры приведены в Таблице 1.

Таблица 1: Гиперпараметры обучения

Параметр	Значение	Комментарий
Устройство	CPU	Не используется GPU
Оптимизатор	Adam	Стандартный
Learning rate (предобучение)	10^{-3}	Для MSE
Learning rate (PINN)	5×10^{-5}	Малый шаг для стабильности
Эпохи (предобучение)	800	
Эпохи (PINN)	1500	
Количество точек	500	На интервале $[0, 30]$
λ_{IC}	100 (первые 100 эпох), 10 (далее)	Вес начальных условий
α	0.001	Коэффициент регуляризации
Градиентный клиппинг	0.1	По значению

6 Результаты экспериментов

6.1 Предобучение

На этапе предобучения модель обучается минимизировать среднеквадратичную ошибку (MSE) относительно численного решения:

$$\mathcal{L}_{\text{supervised}} = \frac{1}{N} \sum_{i=1}^{N} \left((\hat{x}(t_i) - x_i)^2 + (\hat{v}(t_i) - v_i)^2 \right). \quad (9)$$

Результаты:

- **Simple MLP**: финальный MSE = 0.796092.
- **Residual MLP**: финальный MSE = 0.746465.

Residual MLP показывает лучшее качество предобучения за счёт более эффективного градиентного потока.

6.2 Дообучение PINN

На этом этапе применяется полная потеря (3). Финальные значения потерь:

- **Simple MLP**: loss = 0.889746 (ODE = 0.878, IC = 0.001).
- **Residual MLP**: loss = 2.151798 (ODE = 2.147, IC = 0.0005).

Хотя Residual MLP имела меньший MSE на этапе предобучения, она показала большую потерю на этапе PINN, что может указывать на переобучение или неустойчивость при адаптации к физическим ограничениям.

6.3 Ошибки предсказаний

После обучения вычислялись L2-ошибки относительно численного решения:

Таблица 2: Ошибки предсказаний (L2-норма)

Модель	$\|x - x_{\text{num}}\|_2$	$\|v - v_{\text{num}}\|_2$	Общая ошибка
Simple MLP	1.3894	1.5367	2.0717
Residual MLP	1.5488	1.4964	2.1536

Обе модели демонстрируют сопоставимую точность, но Simple MLP имеет немного меньшую общую ошибку.

7 Заключение

В работе проведён подробный теоретический анализ решения уравнения Ван дер Поля с помощью PINN. Были рассмотрены:

- Математическая постановка задачи и преобразование к системе первого порядка.
- Вывод градиентов через автоматическое дифференцирование.
- Архитектуры SimpleMLP и ResidualMLP с детальной инициализацией.
- Гиперпараметры обучения и стратегия двухэтапного обучения.
- Количественные результаты, подтверждающие работоспособность подхода на CPU.

Результаты показывают, что даже простые архитектуры могут давать приемлемое решение жёсткой нелинейной системы при правильной настройке потерь и стабилизации. Для дальнейшего улучшения можно рассмотреть использование SIREN, Fourier Features или адаптивной выборки точек.

Благодарности

Автор благодарит сообщество открытых научных вычислений за разработку библиотек PyTorch и SciPy, которые позволили реализовать и протестировать описанные методы.