ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2023, том 514, № 2, с. 109–117

УДК 004.81

О ЗАДАЧЕ УСТОЙЧИВОГО ПОИСКА ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

© 2023 г. А. А. Хватов[1,*], Р. В. Титов[1]

Представлено академиком РАН А.А. Шананиным
Поступило 31.08.2023 г.
После доработки 15.09.2023 г.
Принято к публикации 18.10.2023 г.

Поиск дифференциальных уравнений является подразделом машинного обучения, который используется для обучения компактных интерпретируемых моделей, особенно в физических приложениях. Обычно в подобных алгоритмах используется максимальное количество априорной информации – заранее заданная форма уравнения – для которой ищут лишь коэффициенты и возможно убирают незначимые слагаемые. В статье исследуются предпосылки и инструменты для более автономного поиска уравнений без участия эксперта, который определяет форму уравнения или физический процесс. Такая постановка больше соответствует принципам машинного обучения – модель получается из данных в отсутствие каких-либо предположений об их распределении. В области поиска уравнений это приводит к задаче устойчивого поиска уравнений – нам недостаточно найти какое-то уравнение, а необходимо оценить, насколько устойчива данная модель к изменениям параметров.

Ключевые слова: поиск дифференциальных уравнений, эволюционная оптимизация, многокритериальная оптимизация, физически-обоснованные нейронные сети

DOI: 10.31857/S2686954323601604, **EDN:** HRKADE

1. ВВЕДЕНИЕ

Начиная с SINDy [1] и PDE-FIND [2], поиск уравнений стал привлекательным инструментом машинного обучения для получения интерпретируемых и лаконичных моделей из физических данных. Было предложено множество расширений SINDy [3, 4]. Результатом работы каждого из описанных алгоритмов (кроме случая [4], который мы обсудим позже) является набор коэффициентов при слагаемых из относительно большого набора-библиотеки. Отсутствие произвола в итоговых коэффициентах справедливо для всех методов, основанных на регрессии. Это может быть разреженная регрессия [1], регрессия с помощью нейронной сети [5], символьная регрессия на графах с помощью генетического программирования [6], символьная регрессия на лесе (наборе графов слагаемых) с помощью более продвинутого генетического алгоритма [7].

Алгоритмы, основанные на SINDy [1], используют символьную (разреженную) регрессию с предопределенной библиотекой слагаемых. Каждое слагаемое является дифференциальным оператором, произведением дифференциальных операторов, примененных к входному полю данных или другой отдельно заданной нелинейностью. Ограничением и по сути недостатком этого подхода является предположение о том, что уравнение может быть выражено исключительно с использованием слагаемых, присутствующих в предопределенной библиотеке. Количество слагаемых в библиотеке может быть интерактивно сокращено для получения краткого выражения. Таким образом, требуется знать структуру нелинейности для каждого набора данных, что приводит нас к классической постановке обратной задачи, осложненной большим числом слагаемых в уравнении.

В случае, когда процесс, управляющий данными, неизвестен полностью или частично, постановка задачи, содержащая заранее определенный набор слагаемых или форму уравнения, становится непригодной. Например, нельзя восстановить волновое уравнение, ожидая уравнение параболического типа. В таких ситуациях используется вторая группа методов обнаружения уравнений, основанная на эволюционной оптимизации. Такие алгоритмы [6–9] менее требовательны к количеству априорных предположений и обеспечивают более широкое пространство по-

[1] *Национальный исследовательский университет ИТМО; Лаборатория композитного искусственного интеллекта, Санкт-Петербург, Россия*
***** *E-mail: alex_hvatov@itmo.ru*

иска, позволяя получать различные локальные оптимумы в ходе многократных запусков.

В контексте поиска уравнений часто акцент делается на проблеме восстановления известного уравнения, которое интегрируется численно для получения данных. Близость восстановленного уравнения к известному ответу является основной метрикой качества в литературе [1, 4, 6, 7, 10]. Для ответа на этот вопрос в общем случае нам необходимо оценить, насколько хорошо полученное уравнение описывает данные статистически, что включает учет неопределенности.

В статье [4] используются разнообразные стратегии бутстрэппинга, включая применение бутстрэпа к данным и предварительно определенной библиотеке слагаемых. В результате, даже с использованием методов градиентного спуска, можно получить разные уравнения от запуска к запуску. Основная идея заключается в том, что чем чаще мы получаем одно и то же уравнение, используя случайную часть доступной информации, тем выше вероятность того, что оно точно описывает процесс. Однако фиксированная форма $u_t = F(x, u_x, u_{xx}, \ldots)$ снижает вероятность нахождения кардинально новых уравнений, вместо этого приводя к изменениям в основном в коэффициентах.

В качестве альтернативы мы можем собрать все уравнения, полученные в результате эволюционной оптимизации, и оценить изменчивость коэффициентов каждого слагаемого и формы уравнения, т.е. провести анализ совместного распределения коэффициентов. Для построения сложных совместных распределений можно использовать байесовские сети, которые позволяют исследовать эти зависимости. Структурное обучение байесовских сетей выходит за рамки данной работы. Мы рекомендуем читателю обратиться к обзорной статье [11]. В регрессионном подходе байесовские сети также могут быть использованы для определения взаимовстречаемых слагаемых в уравнении, что поможет выявить устойчивые структуры.

Для понимания того, как неопределенность в коэффициентах или выбор устойчивой подструктуры на основе совместного распределения отражается на описании данных и решении, нам необходимо иметь возможность решать каждое уравнение, с которым мы сталкиваемся в процессе поиска. В классической постановке, как показано в [4], даже при бутстрэппинге данных и библиотеки мы получаем близкое уравнение, которое может отличаться только коэффициентами. Это упрощает решение уравнения и последующее сравнение с данными, поскольку мы можем настроить конкретный солвер для определенного типа уравнения и изменять только коэффициенты. Любые члены, не соответствующие заданному уравнению, при этом исключаются из рассмотрения вне зависимости от их значимости. К сожалению, такой подход не применим в общих случаях поиска, когда уравнение неизвестно.

При рассмотрении решателей дифференциальных уравнений, подходящих для случая, описанного выше, требуется довольно сильно ослабить требования на точность итогового решения в угоду возможности решать несколько классов уравнений без дополнительной настройки внутри одного запуска алгоритма. Алгоритмов, удовлетворяющих таким требованиям, немного. Решатель, основанный на деревьях решений и написанный на языке программирования Julia [12], а также решатели DeepXDE и TEDEouS, основанные на нейронных сетях и написанные на языке Python [13, 14].

Таким образом, для наиболее полного решения задачи устойчивого поиска дифференциального уравнения по данным требуется совместное использование трех различных алгоритмов – алгоритма поиска дифференциальных уравнений по данным (в нашем случае, EPDE [9]), алгоритма автоматизированного решения дифференциальных уравнений, основанного на нейронных сетях (TEDEouS [14]), и алгоритма определения структуры совместного распределения в виде байесовской сети (BAMT [15]). Использование данных алгоритмов в отличие от аналогичных решений, например [4], позволило отказаться от заранее заданной библиотеки слагаемых – единственным ограничением, в сущности, является максимальный порядок дифференциального уравнения, а так же не использовать заранее заданную форму уравнения. Реализация и численные эксперименты с использованием алгоритма устойчивого поиска описаны в последующем тексте.

2. ПОСТАНОВКА ЗАДАЧИ УСТОЙЧИВОГО ПОИСКА ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПО ДАННЫМ

В этом разделе представлен подход к учету неопределенности при поиске уравнений с помощью совместного использования трех алгоритмов – алгоритма поиска уравнений на основе эволюционной оптимизации, алгоритма определения структуры сложного совместного распределения в виде байесовской сети и алгоритма автоматизированного решения дифференциальных уравнений. Полная схема устойчивого поиска уравнений показана на рис. 1.

С точки зрения машинного обучения, задача поиска уравнения сводится к классическим проблемам символьной регрессии. Наша цель состоит в решении задачи оптимизации (1):

$$\theta^* = \arg\min_{\theta} \mathbb{E}[L(M(\theta), D)], \qquad (1)$$

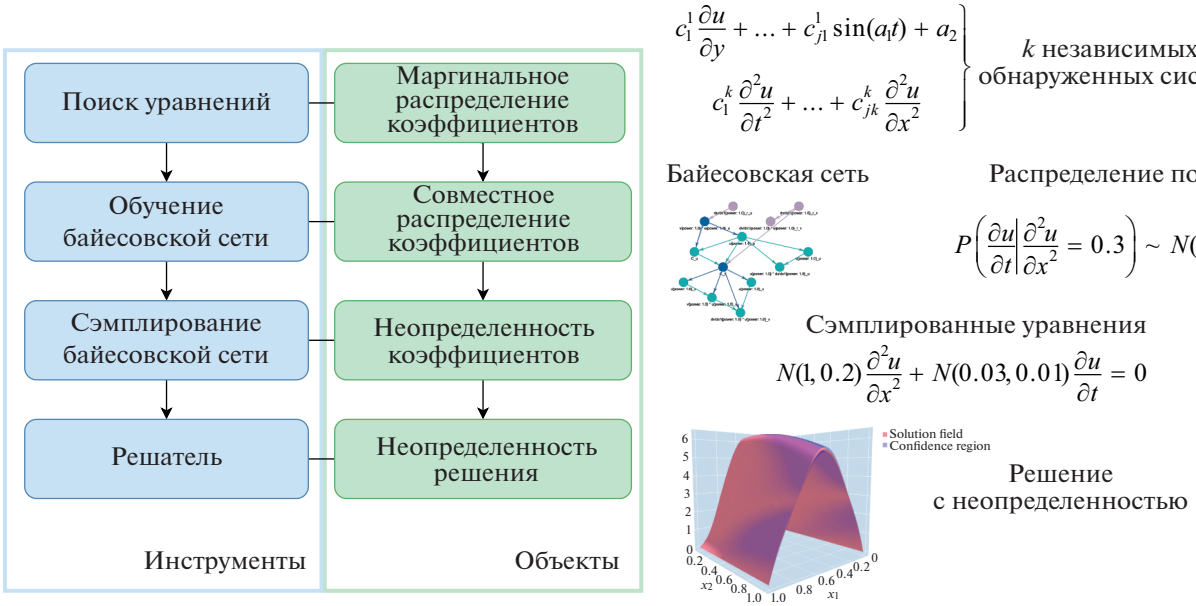

Рис. 1. Схема устойчивого поиска уравнений: слева – инструменты (типы алгоритмов), используемые для поиска уравнений, справа – ожидаемый вывод каждого из алгоритмов с примерами.

где $M(\theta)$ – параметризованная модель, данные D и $L(M(\theta), D)$ – функция потерь, определяющая соответствие между данными и моделью. При этом, ввиду постановки устойчивой задачи, оценивается математическое ожидание $\mathbb{E}[\cdot]$ функции потерь, так как набор параметров θ является случайной величиной.

В задаче поиска дифференциальных уравнений по данным сами данные D обычно представляют собой наблюдения на сетке $\bar{x}$ (пространственной, временной) или с использованием сетки других параметров. Исследование различных параметризаций может быть выгодно в некоторых случаях. Например, в случае различных классов уравнений. Поэтому число параметров $|\theta|$ может изменяться в ходе оптимизации. При поиске дифференциальных уравнений так же возможно учесть физические граничные условия в процессе оптимизации. Например, можно зафиксировать граничные условия и включить их в функцию потерь $L(M(\theta), D)$ в качестве параметра регуляризации.

Алгоритм поиска уравнения. Использование эволюционного алгоритма позволяет избежать ограничений, порождаемых заранее заданной библиотекой. В процессе эволюционной оптимизации слагаемые собираются из параметризованных семейств элементарных операций F, где каждое семейство содержит элементы $f_j(p_1^{(j)}, p_2^{(j)}, \ldots, \bar{x}) \in F$. Обычно рассматриваются только дифференциальные операторы $F_{diff} = \left\{\frac{\partial^{p_{n+1}} u}{\partial x_1^{p_1} \ldots \partial x_n^{p_n}}\right\}$. Кроме того, мы можем включить обратную координатную функцию $F_{inv} = \frac{1}{x_i}$ и другие параметризованные дискретные поля $F_{func} = v(p_1, \ldots, p_k)$. Входными данными алгоритма является выбранный набор элементарных операций $F = \bigcup_j F_j$, из которых создаются слагаемые.

Мы используем общепринятые процедуры эволюционной оптимизации независимо от выбранных семейств токенов для оптимизации. А именно, мы обучаем модель в форме

$$M(C, \mathbf{P}, \bar{x}) = \sum_{i=1}^{i \le L} c_i \cdot a_i(P_i, \bar{x}). \qquad (2)$$

В уравнении (2) $C = \{c_1, \ldots c_L\} \in R^L$ представляет собой набор констант (амплитуд слагаемых), $a_i(P_i, \bar{x}) = \prod_{j=1}^{j=N_{tokens}} f_j(p_1^{(i)}, p_2^{(i)}, \ldots, \bar{x})$ обозначает произведения токенов $f_j \in F$, выбранных из семейств токенов F, $P_i = \{p_1^{(i)}, p_2^{(i)}, \ldots\}$ представляет собой набор параметров для слагаемого a_i, и $\mathbf{P}$ представляет мультииндекс параметров. Количество параметров (и размер хромосомы) $|\mathbf{P}|$ может изменяться в процессе эволюционной оптимизации. В большинстве случаев рассматриваются только точки

наблюдений, т.е. $\bar{x} \in X$. Однако сетки данных и модели могут быть разделены.

Параметры модели **P** определяются с использованием эволюционной оптимизации. Процесс оптимизации включает операцию скрещивания, которая обменивает слагаемые между особями, и мутацию, которая может включать изменения токенов или параметров. Этот процесс аналогичен стандартной символьной регрессии. Для завершения формулировки задачи эволюционной оптимизации необходимо определить функцию приспособленности.

Мы включаем разреженную регрессию в шаг вычисления функции пригодности. Мы используем градиентный спуск для поиска набора коэффициентов C_{opt}, которые минимизируют расхождение. А именно, мы решаем задачу оптимизации, показанную в уравнении

$$c_{opt} = \arg\min_{c} \left\| a_{target} - \sum_{j \neq target} c_j a_j \right\|_2 + \lambda \|c\|_1, \qquad (3)$$
$$c = \{c_1, \ldots, c_{N_{terms}}\}.$$

В уравнении (3) случайное слагаемое выбирается в качестве объясняемой переменной, а остальные слагаемые в данной особи используются как признак. С помощью такой формулировки задачи мы отходим от фиксированной формы уравнения $u_t = F(u, u_x, u_{xx} \ldots)$ и допускаем произвольные уравнения. Кроме того, мы вводим l_1-регуляризацию с использованием l_1-нормы $\|\cdot\|_1$. Слагаемые с коэффициентами $c_i \in c$, $c_i < \varepsilon$, где ε – заданный порог, исключаются из модели как незначимые.

Автоматизированный инструмент для построения байесовской сети [15] используется для автоматического формирования совместного распределения. Впоследствии уравнения сэмплируются из байесовской сети, начиная с произвольной левой части с коэффициентом, равным единице. Каждое выбранное/сэмплированное уравнение затем решается с помощью нейросетевого метода [14] для получения неопределенностей в решении. Байесовская сеть служит в качестве устойчивой структуры мета-уравнения, из которой можно сэмплировать отдельные уравнения. Пример байесовской сети для дифференциального уравнения показан в Приложении А.

Описанный комбинированный подход позволяет автоматически оценивать неопределенности как в коэффициентах уравнений, так и в решении, используя совместные распределения вместо маргинальных распределений. В результате становится возможным статистически анализировать, насколько хорошо найденные уравнения описывают предоставленные данные. Вместо того чтобы полагаться на полноту библиотеки и фиксированную форму уравнений, мы фокусируемся на оценке совместного появления простых дифференциальных слагаемых, полученных в процессе стохастической оптимизации.

В то время как каждый шаг, включая дифференцирование, оптимизацию, дискретизацию распределения, оптимизацию байесовской сети и решение, вносит определенную ошибку, полученный доверительный интервал позволяет точно оценить качество выбранного метода поиска.

3. ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ

Для иллюстрации предлагаемого подхода рассмотрим различные задачи, включая поиск систем обыкновенных дифференциальных уравнений, а также поиск дифференциальных уравнений в частных производных.

Системы обыкновенных дифференциальных уравнений. Мы вновь обращаемся к набору данных Lynx-Hare [16], который ранее использовался в [4] для восстановления модели Лотки-Вольтерры. Мы используем его, поскольку он прост и, следовательно, нагляден.

Модель Лотки-Вольтерра в [4] для этих данных[1] была получена с использованием фиксированного набора слагаемых для каждого уравнения. А именно, была использована модель уравнения (4).

$$\begin{cases} \dot{u} = C_1 u + C_2 uv + C_3 v + C_4 u^2 + C_5 v^2 + \\ + C_6 + C_7 uv^2 + C_8 vu^2 + C_9 u^3 + C_{10} v^3 \\ \dot{v} = D_1 v + D_2 uv + D_3 u + D_4 u^2 + D_5 v^2 + \\ + D_6 + D_7 uv^2 + D_8 vu^2 + D_9 u^3 + D_{10} v^3. \end{cases} \qquad (4)$$

При оптимизации используются упрощенные модели, уравнение (4), полученные путем случайных исключений слагаемых. Такой процесс в статье называется бутстраппинг библиотеки. Неопределенность слагаемых была получена с помощью 1000 моделей. Полученное распределение предельных коэффициентов имеет вид уравнение (5).

$$\begin{cases} \dot{u} = (0.5274 \pm 0.0049)u + (-0.025 \pm \epsilon)vu \\ \dot{v} = (-0.9691 \pm 0.1779)v + (0.027 \pm \epsilon)vu + \\ + (-0.1193 \pm 0.1047)u + (0.1755 \pm 0.2248). \end{cases} \qquad (5)$$

В уравнении (5) $\epsilon = O(10^{-5})$ и коэффициенты представлены средними значениями вместе с двухсторонними доверительными интервалами, построенными с использованием 95-го перцентиля стандартного распределения.

Отметим, что в отличие от [4] мы не нормализуем данные по дисперсии. Таким образом, коэффициенты слагаемых uv должны быть умножены

[1] В открытом доступе в репозитории авторов: https://github.com/urban-fasel/EnsembleSINDy/blob/main/SINDY/main_runLotkaVolterra.m

на 16.565 и 21.414 для первого и второго уравнений соответственно, чтобы достичь соответствия с результатами статьи. Уравнения в оригинальной статье были решены с использованием метода Рунге–Кутты с помощью решателя `ode45` в Matlab с фиксированной формой уравнения (4).

В итоге, для обнаружения модели Лотки-Вольтерра из данных с использованием группы методов разреженной регрессии, необходимо знать форму уравнения, настроить решатель и, возможно, настроить метод бутстрэпа для получения различных результатов при каждом запуске. В качестве преимуществ мы получаем коэффициенты, близкие к теоретически полученным [17], которые показаны в уравнении (6) (остальные коэффициенты равны нулю).

$$\begin{cases} \dot{u} = 0.55u - 0.028uv \\ \dot{v} = -0.84v + 0.026uv. \end{cases} \tag{6}$$

Второе преимущество – скорость оптимизации. Методы, основанные на разреженной регрессии по предопределенной библиотеке, работают быстрее, чем любой эволюционный алгоритм, благодаря использованию градиентного спуска. Также современные настроенные решатели известны как быстрые инструменты.

Для оценки подхода, описываемого в статье, мы рассматриваем два случая обнаружения системы: (а) система, ограниченная уравнениями первого порядка, и (б), ограниченная уравнениями второго порядка. В отличие от заранее определенных библиотечных алгоритмов, каждое уравнение рассматривается независимо, что означает, что система состоит из нескольких взаимосвязанных уравнений, а не записывается в векторной форме.

Затем мы рассматриваем каждое слагаемое в уравнении как случайную переменную и используем все уравнения для определения совместного появления слагаемых. На основе этой информации байесовская сеть представляет совместные распределения коэффициентов слагаемых, как показано в Приложении А.

После анализа байесовской сети мы можем определить одну или несколько компонент графа. В нашем случае была обнаружена одна компонента. Затем мы выбираем произвольный начальный узел, например, $\dot{u}$ для первого уравнения и $\dot{v}$ для второго уравнения. После выбора начального узла мы сэмплируем 30 уравнений для оценки неопределенности коэффициентов и решаем их для определения неопределенности в области данных.

Для случая (а) мы выделяем две устойчивые системы, представленные в уравнении (7). Для наглядности все коэффициенты представлены в виде доверительных интервалов как для маргинальных распределений. Однако мы отмечаем, что каждый коэффициент – многомерная случайная величина, для которой учитывается совместное распределение.

$$\begin{cases} \dot{u} = (0.5598 \pm \epsilon)u + (-0.028 \pm \epsilon)uv + \\ + (0.0941 \pm 0.1157)\dot{v} + (0.0019 \pm 0.0001)\dot{u}v + \\ + (0.0023 \pm 0.0001)\dot{u}\dot{v} - 0.1073 \pm 0.008 \\ \dot{v} = (-0.8278 \pm \epsilon)v + (0.0256 \pm \epsilon)uv + \\ + (0.0037 \pm 0.0005)\dot{u} + (-0.0021 \pm 0.0001)\dot{u}\dot{v} + \\ + (0.0998 \pm 0.0045) \end{cases} \tag{7}$$

$$\begin{cases} \dot{u} = (18.1103 \pm \epsilon) + (-0.5132 \pm 0.0084)v + \\ + (0.0263 \pm 0.0017)\dot{v} + (0.0108 \pm 0.0001)u\dot{u} \\ \dot{v} = (0.8540 \pm 9.1659) + (0.0279 \pm \epsilon)v\dot{v}. \end{cases}$$

Первая система в уравнении (7) представляет собой систему Лотки-Вольтерра, которая является в основном правильной, с дополнительными шумовыми членами, определяющими дисперсию. Второе уравнение представляет собой процесс более высокого масштаба, а именно $\dot{u}, \dot{v} = C_1, C_2$, т.е. локально постоянное увеличение или уменьшение скорости роста популяции.

На рис. 2 показаны решения выбранных уравнений, полученные с использованием решателя на основе нейронных сетей.

В случае (б) размерность пространства поиска возрастает с учетом слагаемых второго порядка, но мы по-прежнему в основном получаем уравнения первого порядка. Итоговые уравнения для случая (б) показаны в (8).

$$\begin{cases} \dot{u} = (0.5426 \pm \epsilon)u + (-0.0277 \pm \epsilon)uv + \\ + (-0.1194 \pm 0.0009)\dot{v} + (-0.1461 \pm 0.0087) \\ \dot{v} = (-0.8535 \pm \epsilon)v + (0.0260 \pm \epsilon)uv + \\ + (0.1676 \pm 0.1113)u + (1.1087 \pm 0.3570) \end{cases}$$

$$\begin{cases} \dot{u} = (17.7157 \pm 0.0008) + (-0.0406 \pm 0.0005)\dot{v} + \\ + (-0.7592 \pm 0.0096)v \\ \dot{v} = (0.0131 \pm 0.6421) + (-0.8538 \pm \epsilon)v + \\ + (0.0260 \pm \epsilon)uv. \end{cases} \tag{8}$$

Решение сэмплированных уравнений для случая (б) показано на рис. 3.

По сравнению с уравнением (7), коэффициенты (8) менее похожи на теоретические коэффициенты (6), но более похожи на полученные с помощью E-SINDy (уравнение (5)). Однако мы все еще можем извлечь устойчивую подмодель и более точно подобрать их при необходимости.

Мы собрали процентную ошибку для всех моделей по сравнению с результатами, полученными в работе [17], в табл. 1. Однако, поскольку это несинтетический случай, у нас нет “правильного ответа”, и метрики ошибки не дают никакой информации.

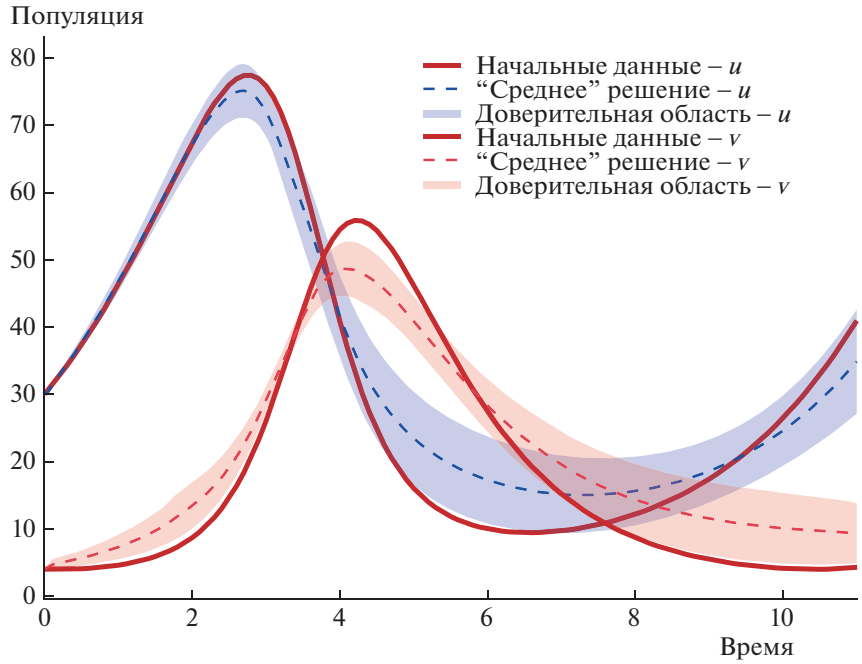

Рис. 2. Результаты решения сэмплированных уравнений с использованием системы на основе байесовской сети для случая (а) – ограничение на производные первого порядка. Красные линии – данные, пунктирная линия – “среднее” решение, соответствующие интервалы – максимум и минимум интегрированной функции на заданном шаге времени.

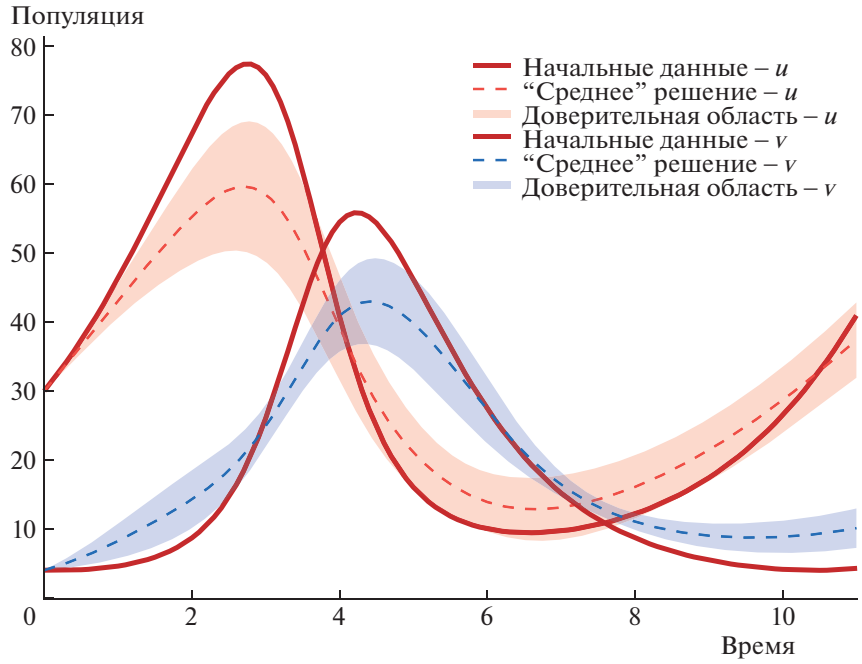

Рис. 3. Результаты решения сэмплированных уравнений с использованием системы на основе байесовской сети для случая (б) – ограничение на производные второго порядка. Красные линии – данные, пунктирная линия – “среднее” решение, соответствующие интервалы – максимум и минимум интегрированной функции на заданном шаге времени.

Метод, предложенный в данной работе, позволяет обнаруживать уравнения, не полагаясь на фиксированную структуру или заранее заданную форму (ср. [4] и (4)). В случае модели Лотки-Вольтерра входными данными являются значения u и v – популяций рыси и зайца соответственно, а также их численные производные до определенного порядка. Порядок и степень членов являются единственными ограничениями для алгоритма, которые могут быть легко расши-

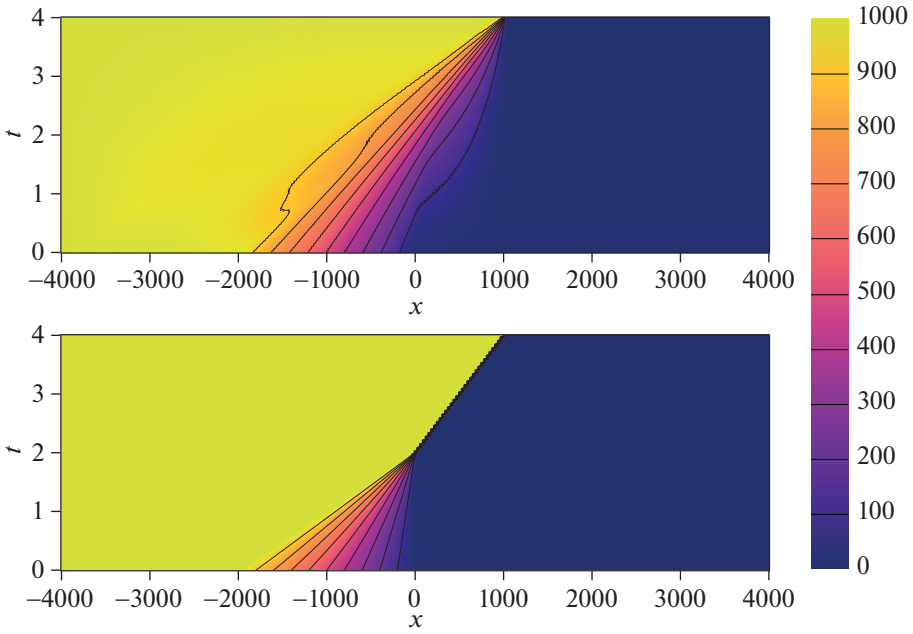

Рис. 4. Результаты решения сэмплированных уравнений с использованием системы на основе байесовской сети. Верхний рисунок – “среднее” решение, нижний рисунок – исходные данные.

рены, как показано при переходе от случая (а) к случаю (б).

Представленный эксперимент представляет собой значительное расширение классического подхода с заранее определенными библиотеками. С помощью этого подхода мы приобретаем возможность обнаруживать уравнения, независимо от экспертного мнения. Следует отметить, что время, затраченное экспертом, значительно сокращается в пользу времени выполнения алгоритма. Для классического алгоритма [4] мы говорим о секундах, тогда как для предложенного подхода в сложных случаях речь идет о десятках минут.

Уравнения в частных производных. Рассмотрим процесс поиска уравнения в частных производных на примере невязкого уравнения Бюргерса, описанного также в работе [4]. Это уравнение представляет собой нелинейное однородное уравнение первого порядка:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0. \tag{9}$$

Мы берем в качестве точных данных аналитическое решение, формирующее ударную волну:

$$u(x,t) = \begin{cases} A, & t \geq \max\left\{\frac{1}{A}x + \frac{1}{\alpha}, \frac{2}{A}x + \frac{1}{\alpha}\right\} \\ \frac{\alpha x}{1-\alpha t}, & A\left(t - \frac{1}{\alpha}\right) < x \leq 0 \\ 0, & \text{в остальных случаях} \end{cases},$$

где $\alpha = 0.5$, $A = 1000$. Данные без шума вычисляются в точках $(x_i, t_i) = (-4000 + i\Delta x, j\Delta t)$ с $\Delta x = 31.25$ и $\Delta t = 0.0157$ для $1 \geq i, j \geq 256$. Характеристики решения, используемого для определения невязкого уравнения Бюргерса, заключаются в том, что ударная волна формируется в момент времени $t = 2$ и движется вдоль линии $x = 500(t-2)$.

Применяя предложенный алгоритм, было успешно получено уравнение, описывающее динамику рассматриваемой системы:

$$(-1 \pm \epsilon)\frac{\partial u}{\partial t} + (-0.96925 \pm 0.05319)u\frac{\partial u}{\partial x} + (-0.00159 \pm 0.00087) = 0. \tag{10}$$

Отметим, что предельные коэффициенты перед членами уравнения (10) были оценены аналогичным образом. Кроме того, на рис. 4 представлены результаты решения сэмплированных уравнений на основе байесовской сети. Заметно, что применение алгоритма и в этом случае позволяет достичь хорошего соответствия между “средним” решением и исходными данными.

Таблица 1. Ошибки коэффициентов относительно уравнения (6). Взята средняя ошибка, поскольку в обоих уравнениях присутствует слагаемое uv

Слагаемые	u	v	mean(uv)	mean(error)
E-SINDy	4%	13%	7%	8%
Случай (а)	2%	1%	1%	1%
Случай (б)	1%	2%	1%	1%

4. ЗАКЛЮЧЕНИЕ

В данной статье был исследован потенциал улучшений, которые могут быть внесены в процесс поиска дифференциальных уравнений по данным, с целью превратить его в более эффективный инструмент машинного обучения. Хотя существующие методы, обсуждаемые в статье, успешно выполняют свои задачи. Процесс поиска уравнений может быть улучшен в следующих направлениях:

• Предопределенная библиотека слагаемых может быть заменена на алгоритм эволюционной оптимизации, который создает слагаемые из элементарных операций.

• Оценка неопределенности должна учитывать совместные распределения, чтобы обеспечить более полное понимание взаимосвязей между слагаемыми, в частности их взаимовстречаемость.

• Решатели уравнений могут быть использованы для исследования влияния неопределенности коэффициентов и выбора той или иной устойчивой подмодели на способность модели воспроизводить данные, сокращая разрыв между задачей поиска уравнений и ее практическими применениями.

Устойчивая постановка превращает задачу поиска уравнений в мощный инструмент, который применим не только к известным процессам и уравнениям, но также подходит для частично известных процессов (например, известное уравнение Навье–Стокса и неизвестное уравнение состояния) или даже полностью “черный ящик”. Это соответствует принципам машинного обучения, позволяя извлекать новые физические законы из данных и давая возможность компьютерам творчески предлагать новые уравнения и открывать действительно новые законы природы.

Огромным недостатком является общее увеличение времени работы. Это своего рода цена автоматизации. Все еще требуется много работы, чтобы предложенный подход стал полностью автономным. Например, требуется автоматическая работа с фронтом Парето, выбор гиперпараметров решателя и дифференцирование данных для процесса поиска.

А. ПРИМЕР БАЙЕСОВСКОЙ СЕТИ

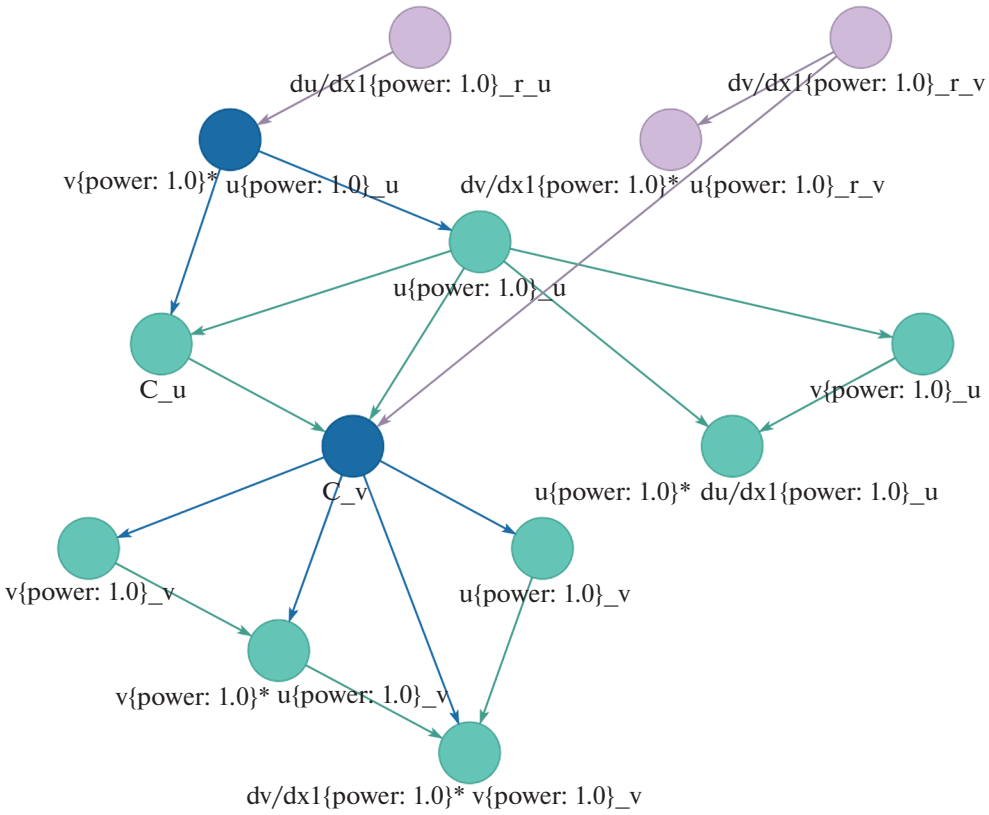

Рис. 5. Пример байесовской сети для случая (б). Индекс _u определяет первое уравнение, индекс _v определяет второе.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Исследование выполнено за счет гранта Министерства науки и высшего образования, соглашение FSER-2021-0012.

ДАННЫЕ И ПРОГРАММНЫЙ КОД

Код и данные для воспроизведения экспериментов доступен в репозитории https://github.com/ITMO-NSS-team/DAN_Mathematics2023_paper в режиме открытого доступа.

СПИСОК ЛИТЕРАТУРЫ

1. *Brunton S.L., Proctor J.L., Kutz J.N.* Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the national academy of sciences. 2016. V. 113. P. 3932–3937.
2. *Rudy S.H., Brunton S.L., Proctor J.L., Kutz J.N.* Data-driven discovery of partial differential equations. Science advances. 2017. V. 3. P. e1602614.
3. *Messenger D.A., Bortz D.M.* Weak SINDy for partial differential equations. Journal of Computational Physics. 2021. V. 443. P. 110525.
4. *Fasel U., Kutz J.N., Brunton B.W., Brunton S.L.* Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. Proceedings of the Royal Society A. 2022. V. 478. P. 20210904.
5. *Long Z., Lu Y.* Dong B. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. Journal of Computational Physics. 2019. V. 399. P. 108925.
6. *Atkinson S., Subber W., Wang L., Khan G., Hawi P., Ghanem R.* Data-driven discovery of free-form governing differential equations. arXiv preprint arXiv:1910.05117 (2019).
7. *Chen Y., Luo Y., Liu Q., Xu H., Zhang D.* Symbolic genetic algorithm for discovering open-form partial differential equations (SGA-PDE). Physical Review Research. 2022. V. 4. P. 023174.
8. *Xu H., Chang H., Zhang D.* DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm. Journal of Computational Physics. 2020. V. 418. P. 109584.
9. *Maslyaev M., Hvatov A., Kalyuzhnaya A.V.* Partial differential equations discovery with EPDE framework: Application for real and synthetic data. Journal of Computational Science. 2021. V. 53. P. 101345.
10. *Chen R.T., Rubanova Y., Bettencourt J., Duvenaud D.K.* Neural ordinary differential equations. Advances in neural information processing systems. 2018. V. 31.
11. *Kitson N.K., Constantinou A.C., Guo Z., Liu Y., Chobtham K.* A survey of Bayesian Network structure learning. Artificial Intelligence Review. 2023. P. 1–94.
12. *Rackauckas C., Ma Y., Martensen J., Warner C., Zubov K., Supekar R., Skinner D., Ramadhan A.* Universal Differential Equations for Scientific Machine Learning. arXiv preprint arXiv:2001.04385. 2020.
13. *Lu L., Meng X., Mao Z., Karniadakis G.E.* DeepXDE: A deep learning library for solving differential equations. SIAM Review. 2021. V. 63. P. 208–228.
14. *Hvatov A.* Automated differential equation solver based on the parametric approximation optimization. Mathematics. 2023. V. 11. P. 1787.
15. *Deeva I., Bubnova A., Kalyuzhnaya A.V.* Advanced approach for distributions parameters learning in Bayesian networks with Gaussian mixture models and discriminative models. Mathematics. 2023. V. 11. P. 343.
16. *Langton H.* The Conservation of the Wild Life of Canada, 1921.
17. *Howard P.* Modeling basics. Lecture Notes for Math. 2009. V. 442.

TOWARDS DISCOVERY OF THE DIFFERENTIAL EQUATIONS

A. Hvatov[a] and R. Titov[a]

[a] *NSS Lab, ITMO University, Saint-Petersburg, Russian Federation*

Presented by Academician of the RAS A.A. Shananin

Differential equation discovery, a machine learning subfield, is used to develop interpretable models, particularly in nature-related applications. By expertly incorporating the general parametric form of the equation of motion and appropriate differential terms, algorithms can autonomously uncover equations from data. This paper explores the prerequisites and tools for independent equation discovery without expert input, eliminating the need for equation form assumptions. We focus on addressing the challenge of assessing the adequacy of discovered equations when the correct equation is unknown, with the aim of providing insights for reliable equation discovery without prior knowledge of the equation form.

Keywords: differential equation discovery, evolutionary optimization, multi-objective optimization, physics-informed neural network