Robust equation discovery as a machine learning method

Alexander Hvatov, Mikhail Maslyaev, Roman Titov, Damir Aminev, Julia Schvartsberg,
Nikita Demyanchuk, Elizaveta Ivanchik, Ilya Markov
mailto:alex_hvatov@itmo.ru

DLCP 22.06.2023

Модели на данных

Классические модели:

«+»: интерпретируемость, широкая применимость, воспроизводимость и т.д., всё идеально, но…

«-»: трудно получить

Нейронные сети:

«+»: воспроизводят довольно сложные явления

«-»: (почти не) интерпретируемы, зависят от данных, воспроизводят только частные закономерности

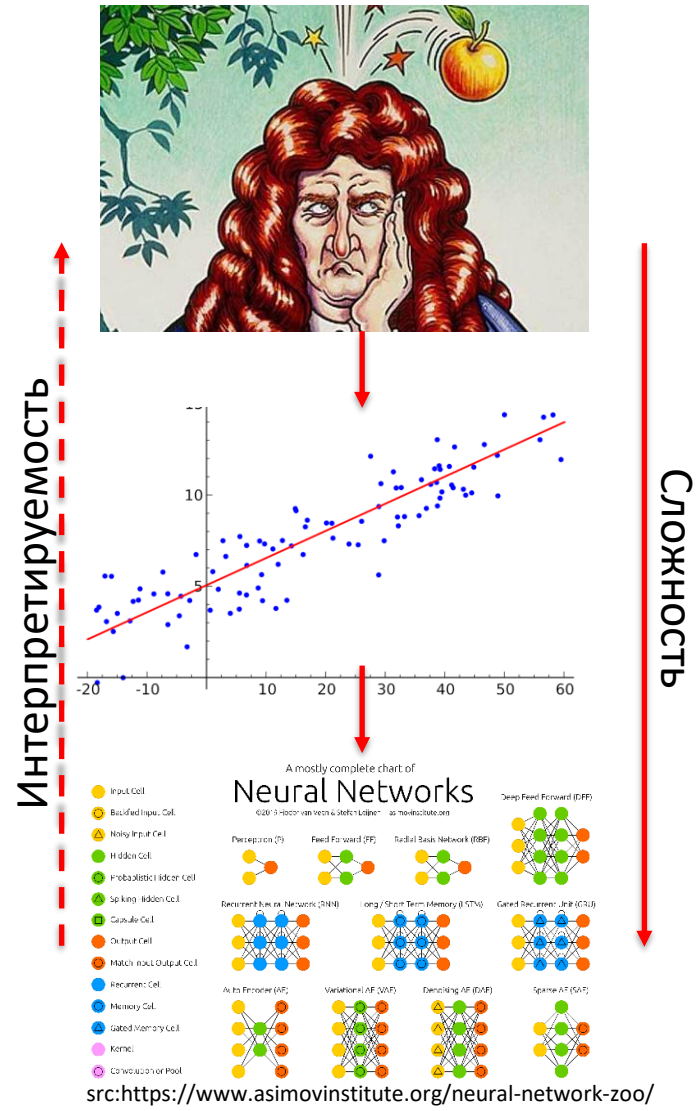
Регрессия:

«+»: простота получения, (частичная) интерпретируемость

«-»: относительная простота модели (воспроизводит узкий круг явлений)

К сожалению, мат. модель для многих задач **не доступна** или слишком общая или интегрируется очень долго. Поэтому приходится выбирать модели на данных.

В моделях на данных для физических процессов приходится балансировать между **сложностью** (уровнем воспроизводимого явления) и **интерпретируемостью** (может ли эксперт сказать, что делает именно этот параметр в данном случае)



src:https://www.asimovinstitute.org/neural-network-zoo/

Модели на данных

Модели, которые воспроизводят **физику** явления могут иметь различную форму, в зависимости от типа и масштаба описываемого явления. Для каждого типа существуют рабочие алгоритмы, позволяющие по данным наблюдений получить выражение с помощью методов МО

- Алгебраические (иногда с расширением на специальные функции) уравнения и системы

Например, известно выведение закона всемирного тяготения по данным передвижения планет. В целом, можно описать любое неизменяющееся явление или связь между величинами [1].

С точки зрения МО достаточно обычной регрессии, зачастую даже линейной.

- Обыкновенные дифференциальные уравнения (ОДУ)

Очевидно, можно учитывать зависимость от одной переменной. На этом же уровне можно рассматривать например, модель в виде функции зависящей, например, от времени.

С точки зрения МО так же достаточно регрессии, в данном случае разреженной [2].

- ДУ в частных производных (ДУЧП)

Один из наиболее общих способов описать процесс. Можно использовать для получения различных зависимостей [3].

$$F = G\frac{m_1 m_2}{r^2}$$

$$f(t) = A\sin\omega t + \cdots \qquad \ddot{f} + f = 0$$

$$\begin{cases} F(u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_2^2}, \ldots, \frac{\partial^2 u}{\partial t^2}, \ldots) = 0; \\ G(u) = 0,\ u \in \Gamma(\Omega) \times [0, T]; \end{cases}$$

[1] Schmidt M., Lipson H. Distilling free-form natural laws from experimental data // Science. – 2009. – Т. 324. – №. 5923. – С. 81-85.

[2] 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. – Т. 113. – №. 15. – С. 3932-3937.

[3] Rudy S. H. et al. Data-driven discovery of partial differential equations //Science Advances. – 2017. – Т. 3. – №. 4. – С. e1602614.

ДУ и разреженная регрессия (1/2)

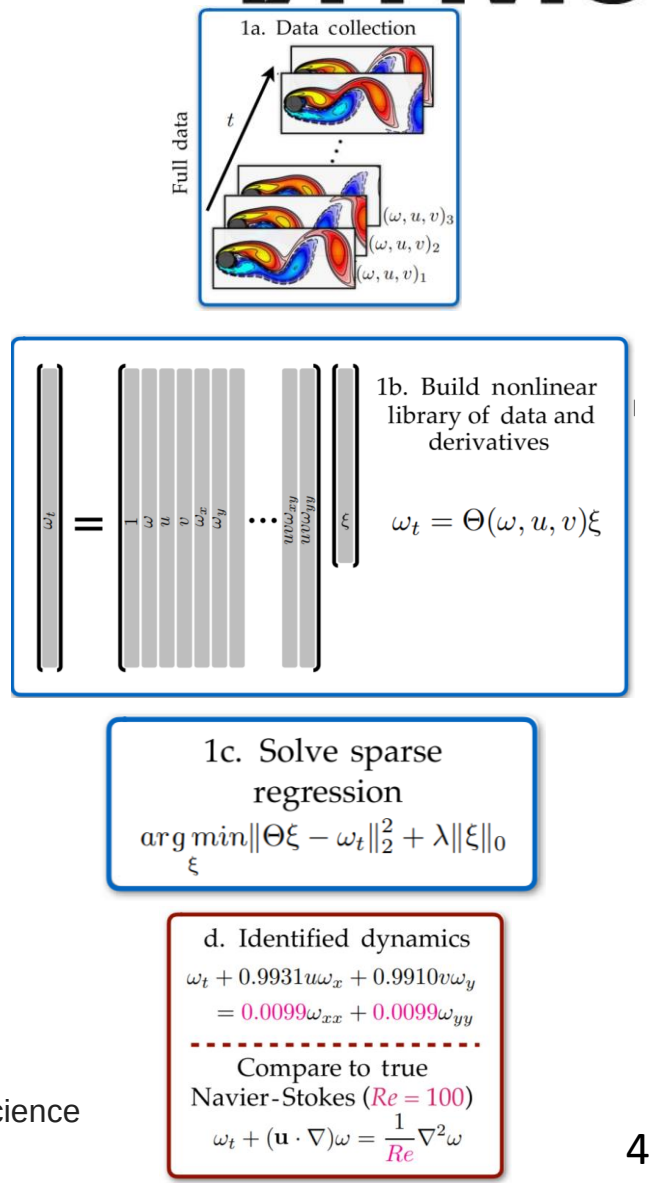
Более требовательный к подготовке путь – разреженная регрессия по заданному набору слагаемых. В качестве примера берём PDEfind

Цель – получить с одной стороны самое «простое» выражение, а с другой наиболее близкое к данным из заданного набора слагаемых

«+»: решается довольно общая задача

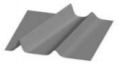
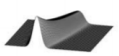
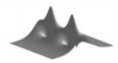
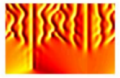
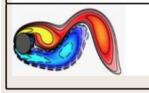
«-»: зависит от заранее заданного набора слагаемых, так же требует фильтр или устойчивую к шуму схему дифференцирования

Здесь мы ограничены заранее заданным набором данных. Если набор достаточно широк, поиск оптимального выражения занимает довольно длительное время.



Rudy S. H. et al. Data-driven discovery of partial differential equations //Science Advances. – 2017. – Т. 3. – №. 4. – С. e1602614.

ДУ и разреженная регрессия (2/2)

PDE	Form	Error (no noise, noise)	Discretization
 KdV	$u_t + 6uu_x + u_{xxx} = 0$	$1 \pm 0.2\%, 7 \pm 5\%$	$x \in [-30, 30], n = 512,\ t \in [0, 20], m = 201$
 Burgers	$u_t + uu_x - \epsilon u_{xx} = 0$	$0.15 \pm 0.06\%, 0.8 \pm 0.6\%$	$x \in [-8, 8], n = 256,\ t \in [0, 10], m = 101$
 Schrödinger	$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0$	$0.25 \pm 0.01\%, 10 \pm 7\%$	$x \in [-7.5, 7.5], n = 512,\ t \in [0, 10], m = 401$
 NLS	$iu_t + \frac{1}{2}u_{xx} + \lvert u\rvert^2 u = 0$	$0.05 \pm 0.01\%, 3 \pm 1\%$	$x \in [-5, 5], n = 512,\ t \in [0, \pi], m = 501$
 KS	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$	$1.3 \pm 1.3\%, 52 \pm 1.4\%$	$x \in [0, 100], n = 1024,\ t \in [0, 100], m = 251$
u  v Reaction Diffusion	$u_t = 0.1\nabla^2 u + \lambda(\mathrm{A})u - \omega(\mathrm{A})v$ $v_t = 0.1\nabla^2 v + \omega(\mathrm{A})u + \lambda(\mathrm{A})v$ $\mathrm{A}^2 = u^2 + v^2, \omega = -\beta \mathrm{A}^2, \lambda = 1 - \mathrm{A}^2$	$0.02 \pm 0.01\%,\ 3.8 \pm 2.4\%$	$x, y \in [-10, 10], n = 256,\ t \in [0, 10], m = 201$ subsample 1.14%
 Navier-Stokes	$\omega_t + (\mathbf{u} \cdot \nabla)\omega = \frac{1}{Re}\nabla^2 \omega$	$1 \pm 0.2\%\ ,\ 7 \pm 6\%$	$x \in [0, 9], n_x = 449,\ y \in [0, 4], n_y = 199,$ $t \in [0, 30], m = 151,$ subsample 2.22%

Алгоритм выбирает наиболее представительный набор слагаемых с помощью разреженной регрессии.

Основные выводы:

- Если есть необходимые слагаемые, то трудно придумать что-то более быстрое
- Достаточно стабильный алгоритм, что хорошо для воспроизведения заранее заданных уравнений
- Можно регулировать «подробность» модели с помощью параметра разреженности

Rudy S. H. et al. Data-driven discovery of partial differential equations //Science Advances. – 2017. – Т. 3. – №. 4. – С. e1602614.

ДУ и нейронные сети (1/2)

Самый «простой» путь – обучить нейронную сеть. В качестве примера берём PDEnet 2.0

Цель – предсказывать с определённым горизонтом

«+»: частная задача под конкретные данные – решается быстро и качественно

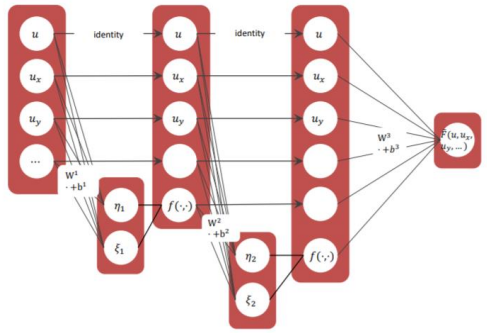
«-»: ограничения в типе уравнения 2D-пространство+время, только «параболический» тип (первая производная по времени приравнивается полиному по степеням ∇), новые данные = новая задача

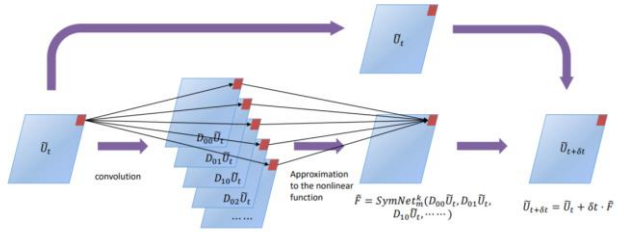
Таким образом временная зависимость сильно ограничена, хотя и круг процессов весьма широкий.

Нейронные сети сильно зависят от качества данных поэтому требуется фильтрация (в PDEnet это делается с помощью другой нейронной сети)

Чем сложнее уравнение, тем больше требуется данных.

Можно ли рассматривать «гиперболические» уравнения? С виду просто, но на деле всё не так – дело в граничных условиях.





$$U_t(t,x,y) = F(U, U_x, U_y, U_{xx}, U_{xy}, U_{yy}, \ldots)$$

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. – Т. 399. – С. 108925.

ДУ и нейронные сети (2/2)

Correct PDE	$u_t = -uu_x - vu_y + 0.05(u_{xx} + u_{yy})$ $v_t = -uv_x - vv_y + 0.05(v_{xx} + v_{yy})$
Frozen-PDE-Net 2.0	$u_t = -0.906uu_x - 0.901vu_y + 0.033u_{xx} + 0.037u_{yy}$ $v_t = -0.907vv_y - 0.902uv_x + 0.039v_{xx} + 0.032v_{yy}$
PDE-Net 2.0	$u_t = -0.979uu_x - 0.973u_yv + 0.052u_{xx} + 0.051u_{yy}$ $v_t = -0.973uv_x - 0.977vv_y + 0.053v_{xx} + 0.051v_{yy}$

Correct PDE	$u_t = -uu_x - vu_y + 0.1\Delta u + (v-u)(u^2+v^2) + u$ $v_t = -uv_x - vv_y + 0.1\Delta v - (v+u)(u^2+v^2) + v$
Frozen-PDE-Net 2.0	$u_t = -0.86uu_x - 0.90vu_y + 0.09u_{xx} + 0.09u_{yy}$ $+1.02u^2v - 1.02u^3 - 1.01uv^2 + 1.01u + 0.99v^3$ $v_t = -0.87uv_x - 0.85vv_y + 0.09v_{xx} + 0.09v_{yy}$ $+1.04u^2v - 1.02uv^2 - 1.01v^3 + 0.99v - 0.99u^3$
PDE-Net 2.0	$u_t = -0.98vu_y - 0.93uu_x + 0.10u_{xx} + 0.10u_{yy}$ $-1.05uv^2 + 0.99v^3 - 0.98u^3 + 0.98u + 0.97u^2v$ $v_t = -0.99uv_x - 0.96vv_y + 0.10v_{yy} + 0.10v_{xx}$ $-1.04u^2v - 1.02v^2 - 1.02uv^2 + 1.01v - 1.00u^3$

Нейронная сеть что-то предсказывает, с переменным успехом.

Основные выводы:

- Не все задачи можно решить нейронной сетью (только оптимизация предсказания)
- Трудно интерпретировать это с точки зрения итогового ДУ (что будет, если закрыть верхнюю строчку, какое уравнение выбрать?)
- Решение справедливо только для конкретного набора данных – другие данные влекут за собой обучение фильтра и «обучение» предсказательной модели заново

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. – Т. 399. – С. 108925.

ДУ и генетические алгоритмы (1/2)

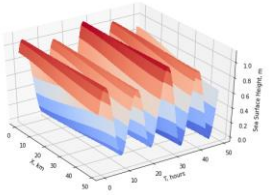
Наиболее требовательный к настройке – генетический алгоритм. В работу включается эволюционная оптимизация.

Цель – предсказывать наиболее подходящую комбинацию из заданных дифференциальных операторов

«+»: наиболее широкий набор итоговых моделей

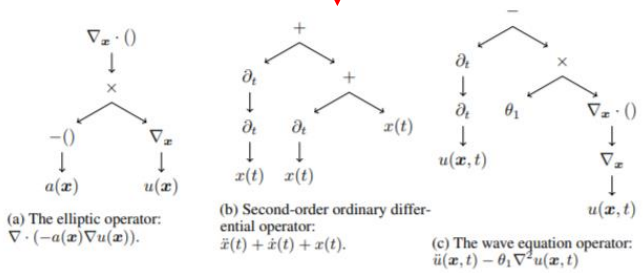
«-»: модель разрастается с каждым поколением, нужно ограничивать рост, а, следовательно и сложность итоговой модели

Таким образом мы можем получить сложную модель, однако со сложностью растёт число «неподходящих» элементов. То есть, «значимая» часть иногда заменяется сложной комбинацией, которая чуть хуже.



Data (measurements)

Models



Genetic programming

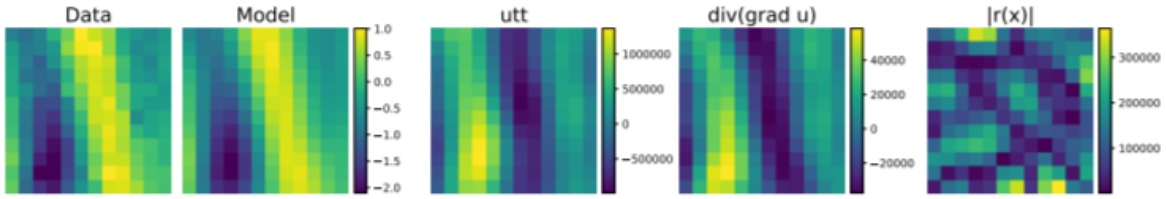
Resulting equation

Atkinson S. et al. Data-driven discovery of free-form governing differential equations //arXiv preprint arXiv:1910.05117. – 2019.

ДУ и генетические алгоритмы (1/2)

Differential equation	Differential operator	Dependent variables	$f \stackrel{?}{=} 0$
ODE	$\ddot{u} + \dot{u} + u$	$u(t) : [0, 10] \to \mathbb{R}$	Yes
Heterogeneous elliptic PDE	$\frac{d}{dx}(-a(x)\frac{du}{dx})$	$a : [0, 1] \to \mathbb{R}^+, u(x) : [0, 1] \to \mathbb{R}$	No
Nonlinear elliptic PDE	$\frac{d}{dx}(-a(u)\frac{du}{dx})$	$u(x) : [0, 1] \to \mathbb{R}$	No
2D Heterogeneous elliptic PDE	$\nabla \cdot (-a(x)\nabla u)$	$a : [0, 1]^2 \to \mathbb{R}^+, u(x) : [0, 1]^2 \to \mathbb{R}$	Yes

$\mathcal{L}[u]$	$q(\boldsymbol{\theta})$ (mean, std)	ELBO
$\theta_1 \nabla^2 u + \theta_2 + \theta_3 u$	$(21.7, 0.14), (120, 2300), (-69, 2900)$	$\mathbf{-6.17445 \times 10^4}$
$\theta_1 \nabla^2 u + \theta_2 u + \theta_3 u^2$	$(21.7, 0.14), (-69, 3400), (-3.78, 4.70)$	-6.17450×10^4
$\theta_1 \nabla^2 u + \theta_2 + \theta_3 u^2$	$(21.7, 0.14), (110, 2300), (83, 2400)$	-6.17452×10^4
$\theta_1 \nabla^2 u + \theta_2 u^2 + \theta_3 u^3$	$(21.7, 0.14), (84, 2200), (-45, 1400)$	-6.17455×10^4
$\theta_1 \nabla^2 u + \theta_2 u + \theta_3 u_t$	$(21.7, 0.14), (-69, 3400), (-3.78, 4.70)$	-6.17509×10^4
$\theta_1 \nabla^2 u$	$(21.7, 0.14)$	-6.17626×10^4



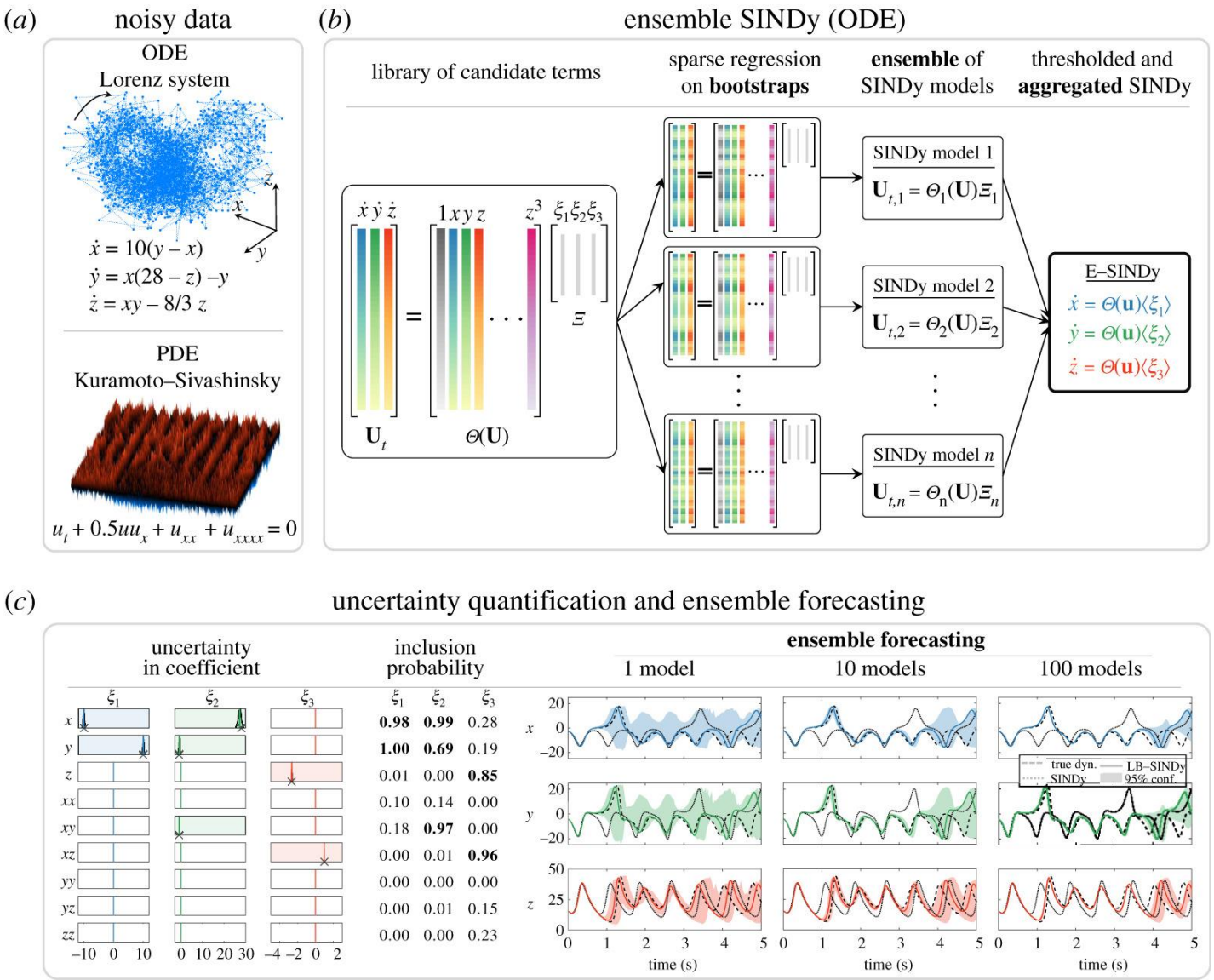
Качество символьной регрессии довольно высокое, но требуется «очистка» модели и дополнительные методы для обеспечения устойчивости модели.

Основные выводы:

- Можно подобрать произвольную модель, если задать подходящее пространство поиска
- Поиск неустойчив
- Требуется регуляризация модели

Atkinson S. et al. Data-driven discovery of free-form governing differential equations //arXiv preprint arXiv:1910.05117. – 2019.

Концепция E-SINDy (1/2)



Если подразумевается, что уравнение неизвестно, нас не устраивает один «ответ» - нужна какая-то «неопределённость».

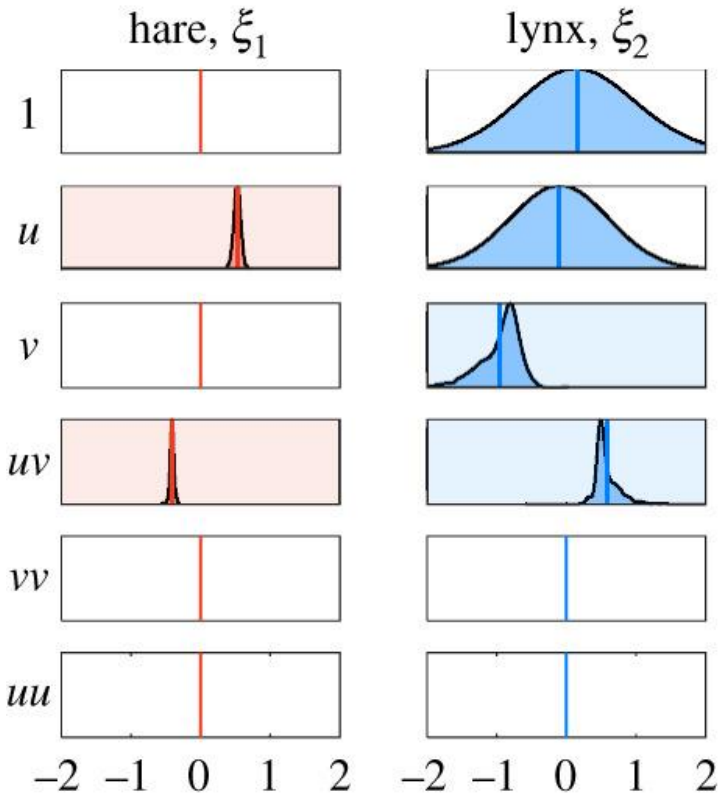
«+»:Имитируя генетику, можно каждый раз создавать под-библиотеки из большой – есть шанс что ответы будут разные по слагаемым и точно будут по коэффициентам

С данными тоже самое – можно выбирать часть

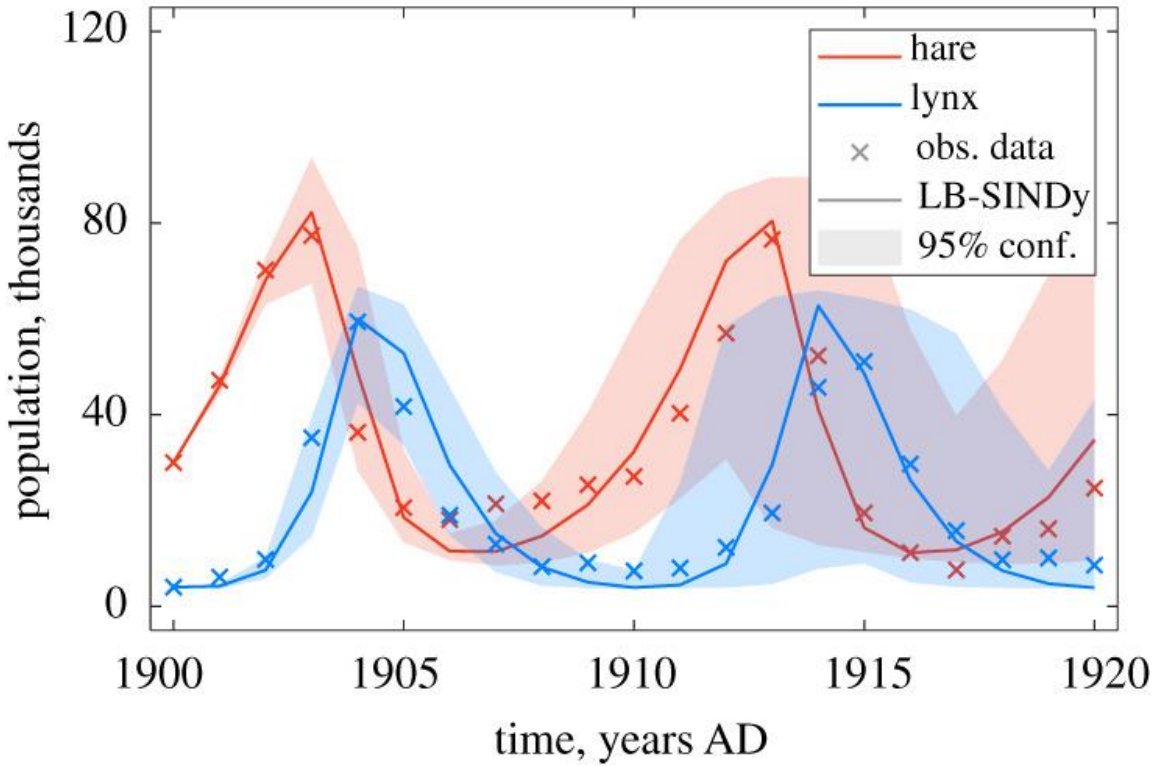
«-»: Старый-добрый SINDy

Концепция E-SINDy (2/2)

(a) uncertainty in coefficient (b) inclusion probability (c) model reconstruction



ξ_1	ξ_2
0.12	0.38
0.90	0.33
0.22	**0.76**
0.79	**0.76**
0.00	0.50
0.00	0.50

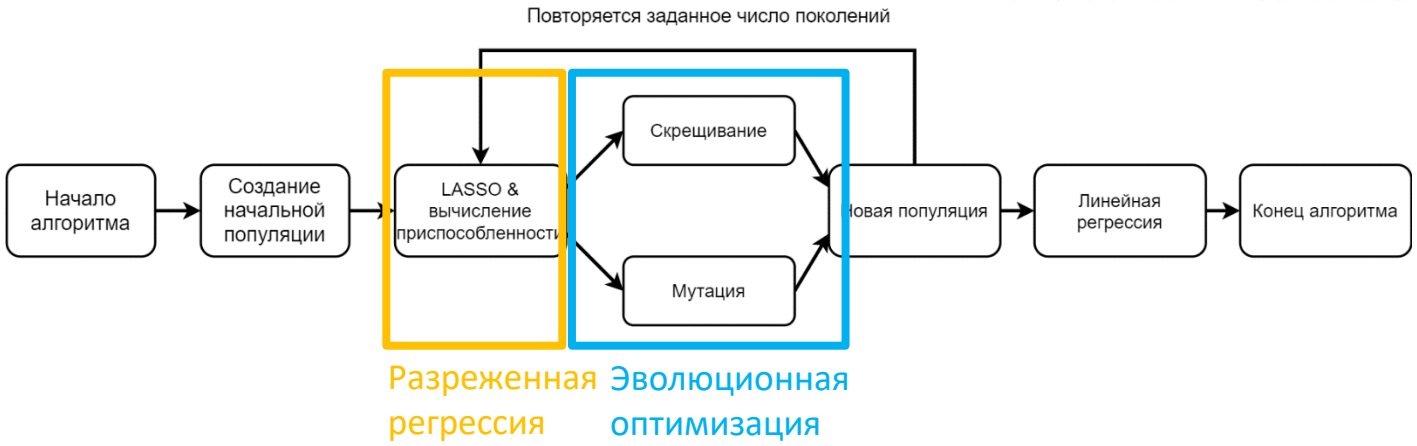


Забегая вперёд: когда мы знаем что решать, мы можем это решить, чтобы показать, как это выглядит на данных.

Что можно предложить?

Можно взять всё лучшее из существующих решений - способность эволюционных алгоритмов генерировать **разнообразные модели** и способность разреженной регрессии «отфильтровывать» **изменчивость**

Мы решили сделать алгоритм, в котором эволюционируют сами операторы, а модель представлена в виде суммы произведений



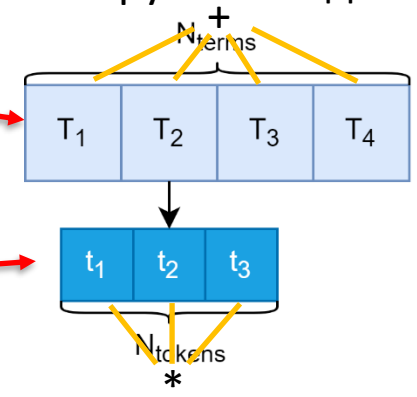
Каждое слагаемое – произведение заранее заданных «кирпичиков» $C = \{c_m = T_1^{(J_1)} \cdot \dots \cdot T_k^{(J_k)} | T_l^{(J_l)} \in T_l, m \in \overline{1, M}\}$

Итоговое уравнение имеет вид суммы произведений с вещественными коэффициентами $F = \sum_{i=1}^{i=M} \alpha_i c_i = 0, \alpha_i \in \mathbb{R}$

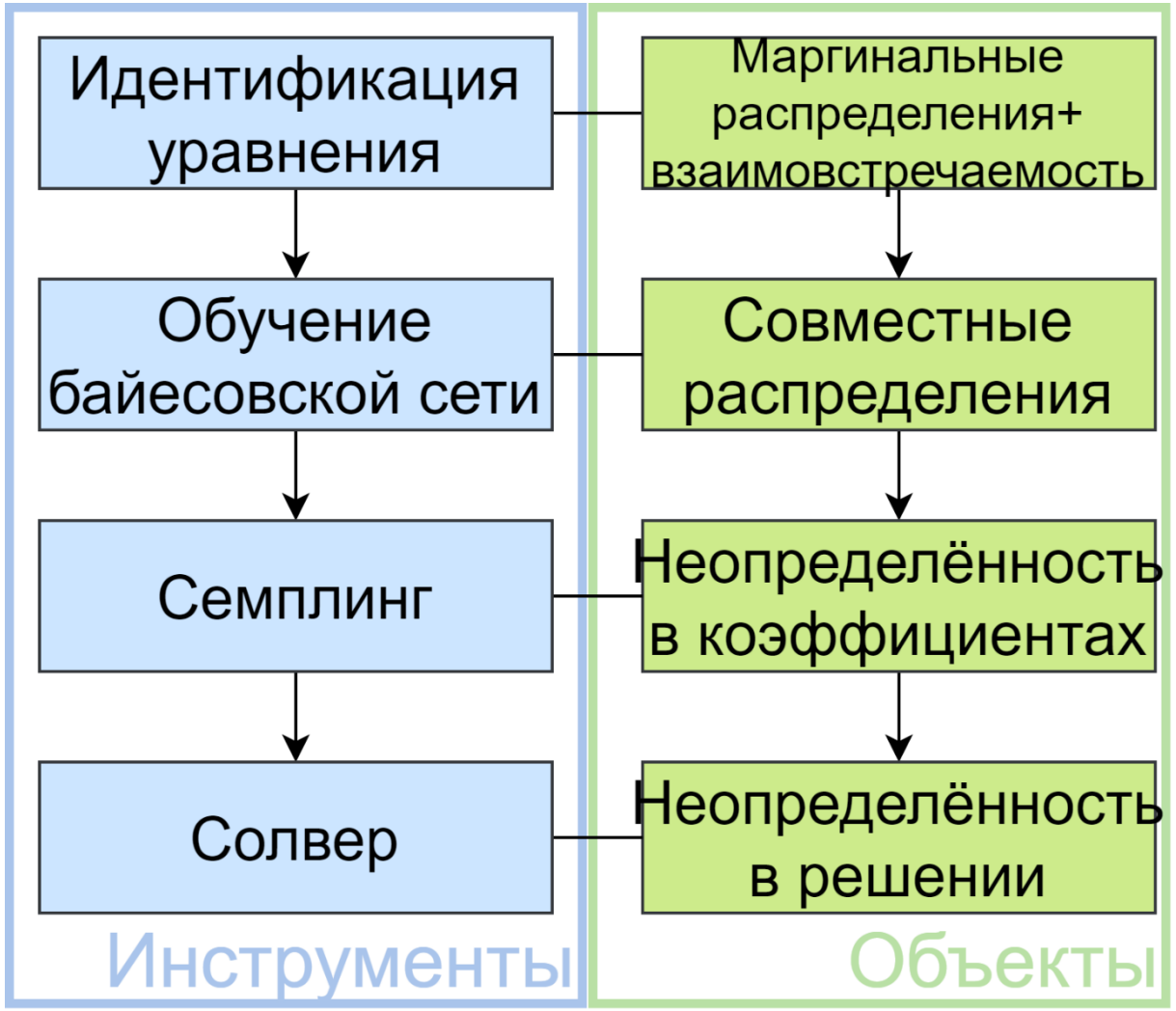
Множество блоков $T = \bigcup_{i=1}^{i=n} T_i$ задаётся заранее.

Например, это может быть набор производных по одной переменной: $T_i = \left\{\frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2} \dots, \right\}$, при этом J_i в $T_i^{(J_i)}$ - мультииндекс, который в данном случае означает порядок производной и степень в которую возводится выражение $T_i^{(2,5)} = \left(\frac{\partial^2 u}{\partial t^2}\right)^5$

В терминах эволюционного алгоритма: хромосома – это само уравнение; ген – это слагаемое, выраженное произведением блоков.



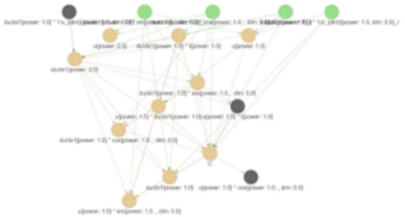
Post E-SINDy (1/3)



Общая схема подхода

$$\left.\begin{array}{l} c_1^1 \frac{\partial u}{\partial t} + \cdots + c_{j1}^1 \sin a_1 t + a_2 \\ \cdots \\ c_1^k \frac{\partial^2 u}{\partial t^2} + \cdots + c_{jk}^k \frac{\partial^2 u}{\partial x^2} \end{array}\right\}$$

k независимо полученных уравнений



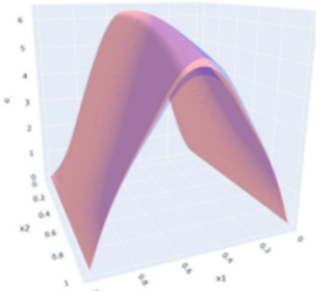
Байесовская сеть

$$P\left(\frac{\partial u}{\partial t}\middle|\frac{\partial^2 u}{\partial x^2} = 0.3\right) \sim N(1, 0.2)$$

Распределения в узлах

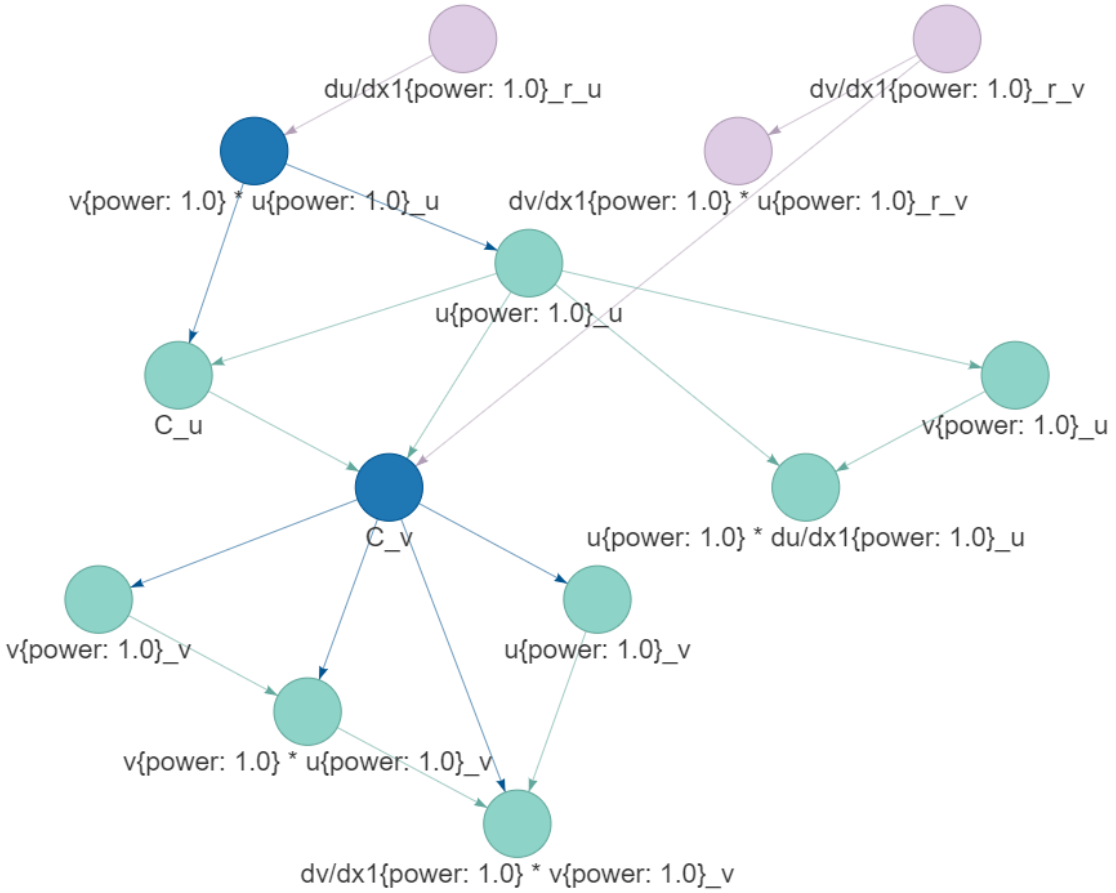
$$N(1, 0.2)\frac{\partial^2 u}{\partial x^2} + N(0.03, 0.01)\frac{\partial u}{\partial t} = 0$$

Семплированные уравнения



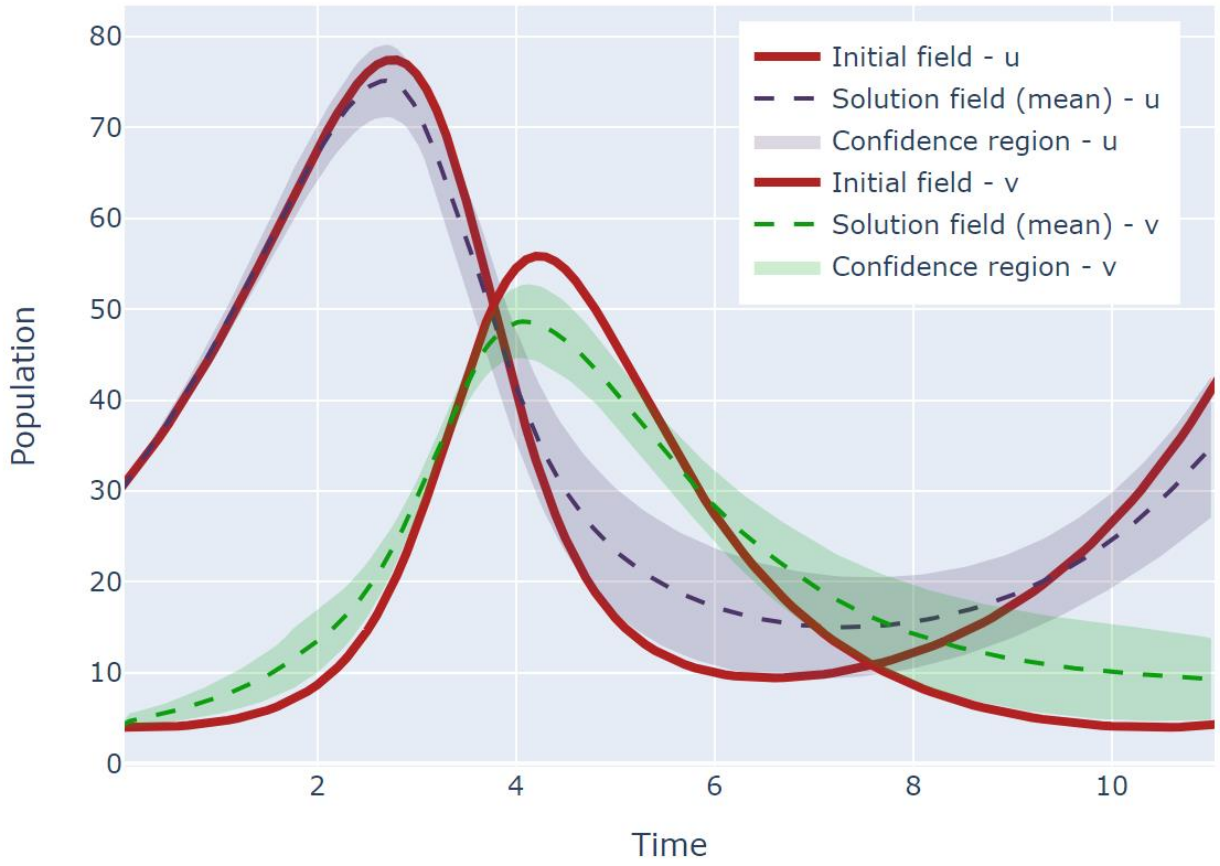
Неопределённость в решении

Post E-SINDy (2/3)



- Байесовская сеть – модель многомерного условного распределения
- Каждый узел – признак/событие
- Связь – условная зависимость
- В каждом узле по данным моделируется многомерное условное распределение с учётом их условной зависимости
- В нашем случае узлы - слагаемые дифференциального уравнения, связи – взаимовстречаемость
- Семплирование производится с произвольной «правой» части с коэффициентом 1.

Post E-SINDy (3/3)



Решения систем, семплированных с помощью байесовской сети

- Алгоритм EPDE знает только значения u, v и $\dot{u}, \dot{v}$ и способен получить систему Лотка-Вольтерра в виде

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

- Ничего другого алгоритм не знает(!)

E-SINDy использует заготовку и получает

$$\begin{cases} \dot{u} = C_1 u + C_2 uv + C_3 v + C_4 u^2 + C_5 v^2 + C_6 + \\ +C_7 uv^2 + C_8 vu^2 + C_9 u^3 + C_{10} v^3 \\ \dot{v} = D_1 v + D_2 uv + D_3 u + D_4 u^2 + D_5 v^2 + D_6 + \\ +D_7 uv^2 + D_8 vu^2 + D_9 u^3 + D_{10} v^3 \end{cases} \quad \begin{cases} \dot{u} = (0.5274 \pm 0.0049)u + (-0.025 \pm \epsilon)vu \\ \dot{v} = (-0.9691 \pm 0.1779)v + (0.027 \pm \epsilon)vu + \\ +(-0.1193 \pm 0.1047)u + (0.1755 \pm 0.2248) \end{cases}$$

Итоги

Поиск уравнений:

- 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. – С. 101345.

Решение:

- Hvatov A. Automated differential equation solver based on the parametric approximation optimization //Mathematics. – 2023. – Т. 11. – №. 8. – С. 1787.

Git:

- https://github.com/ITMO-NSS-team/torch_DE_solver/
- https://github.com/ITMO-NSS-team/EPDE

Сайт лаборатории:

- https://itmo-nss-team.github.io

ИТМО

Спасибо за внимание!

Robust equation discovery as a machine learning method

Alexander Hvatov, Mikhail Maslyaev, Roman Titov, Damir Aminev, Julia Schvartsberg,
Nikita Demyanchuk, Elizaveta Ivanchik, Ilya Markov
mailto:alex_hvatov@itmo.ru

DLCP 22.06.2023