Nested Neural SINDy Approach

Discovering Equations and Numerical Methods

Clément Flint, Louis Fostier, Reyhaneh Hashemi

CEMRACS 2023

Supervisors: Victor Michel-Dansac, Emmanuel Franck, Camilla Fiorini, Wassim Tenachi

August, 2023

Regression Analysis in Interpretable Machine Learning

Definition

The relationship between a set of independent variables X (inputs) and a dependent variable Y (output):

$$y = f^*(x, \theta^*) + \epsilon \quad \mathbb{R}^d \to \mathbb{R}$$

Main Objective (in most regression problems)

Achieving **a good fit** between inputs and output, regardless of the form the mapping function assumes!

But! At times, we desire to represent f^* with a mathematical expression, ideally a "simple" one to enhance interpretability.

Approaches to Function Discovery

At times, f^* is straightforward to determine due to its pre-specified model structure: e.g., linear, polynomial, etc. $\cdots$

However, sometimes the solution space is either unknown in advance or vast, leading to the need for "symbolic regression".

Symbolic Regression (SR)

Main idea:

Inferring the best-fitting model (both structure and parameters) from a dataset in terms of both "accuracy" and "simplicity"

Symbolic Regression Methods

- Eureqa - Genetic Programming method family
- AIFeynman - Divide and Conquer method family
- Bayesian Symbolic Regression (BSR) - Markov Chain Monte Carlo method family
- **Sparse Identification of Nonlinear Dynamics (SINDy)**

SINDy Method

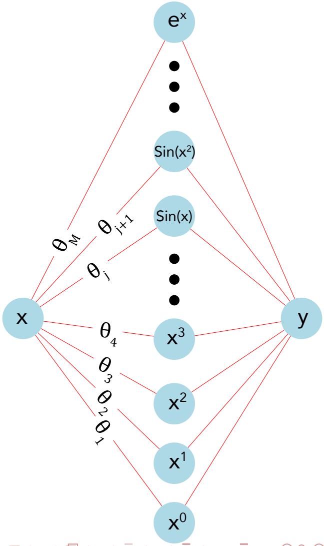
$$f_\theta(x) = \sum_{j=1}^{M} \theta_j f_j(x) \quad \forall j, f_j \in \mathcal{F}$$

$\mathcal{F}$: space of expressions, also called "function dictionary".

$$\mathcal{F} = \{x^0, x^1, x^2, x^3, \ldots, \\ \sin(x), \sin(x^2), \ldots, e^x\}$$

Central Assumption: Only a select few important f_i that govern the problem, implying:

$f(x)$ is sparse in the space of expressions
$\Rightarrow$ for most j: $\theta_j = 0$



SINDy Model Optimization

Regression Component

$$\min_{\theta} \sum_{i=1}^{N} \|y_i - f_\theta(x_i)\|_2^2$$

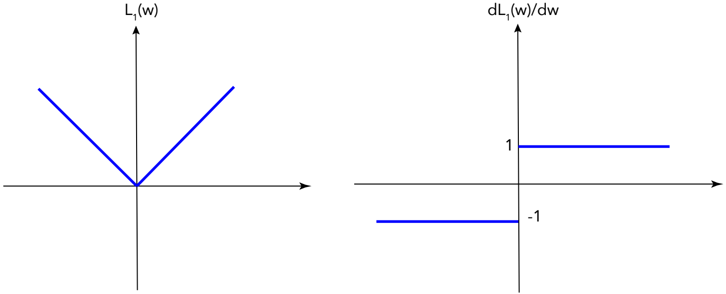
Representational Sparsity Component

Lasso L^1 norm penalty (on the representation):

$$\lambda \sum_{j=1}^{M} |\theta_j| \qquad \lambda \in [0, \infty]$$

Sparsity mechanism by L_1

$$L_1(w) = \frac{\mathrm{d}(L_1(w))}{\mathrm{d}w} = \mathrm{sign}(w) = \left(\frac{w_1}{|w_1|}, \frac{w_2}{|w_2|}, \frac{w_3}{|w_3|}, \cdots, \frac{w_M}{|w_M|}\right)$$



Weight Update Rule:

$$w_1 \leftarrow w_1 - \alpha \frac{\mathrm{d}L_1(w)}{dw}$$

Advantages of the SINDy Approach

- Straightforward implementation
- Compact solutions due to sparsity loss
- Simultaneous optimization of representation and form via gradient descent
- Strong generalization ability

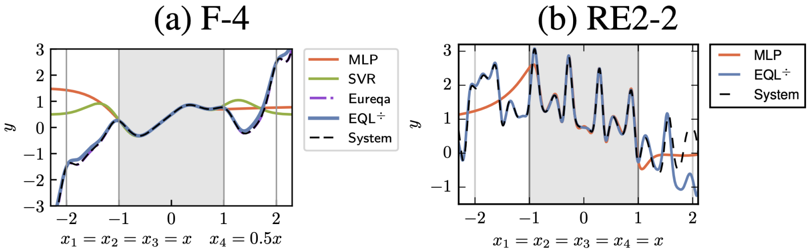


Figure 1: Generalization ability in the SINDy method Sahoo, Lampert, and Martius 2018

Limitations

Known cases where the SINDy approach is expected to struggle:

- When the unknown function involves function composition.
- When the unknown function is a product of multiple functions.

Example:

damped harmonic oscillator:

$$\exp^{-\alpha t}(A\cos(\omega t + \phi) + B\sin(\omega t + \phi))$$

Introduction to Nested SINDy

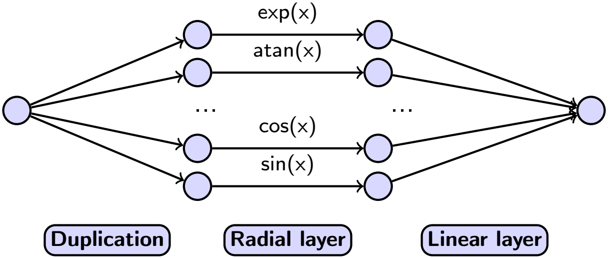


Figure 2: Base SINDy structure. Different layers can be added before and after the radial layer.

- SINDy models can be extended by increasing the number of layers.
- We can add various layers *before* and *after* the radial layer.

Justifications for Nested SINDy

The use of a nested structure has two main goals:

- **Expressivity**: Nested structures can capture complex function compositions.
- **Flexibility**: The number of blocks can be varied to adapt to different complexities.

However, the nested structure introduces new challenges:

- **Computational complexity**: The number of parameters increases with the number of blocks.
- **Learning**: The model is more likely to encounter local minima, as the presence of nonlinear functions across multiple layers complicates the optimization landscape.

We will present two models that extend SINDy to achieve greater representative potential.

The PR Block

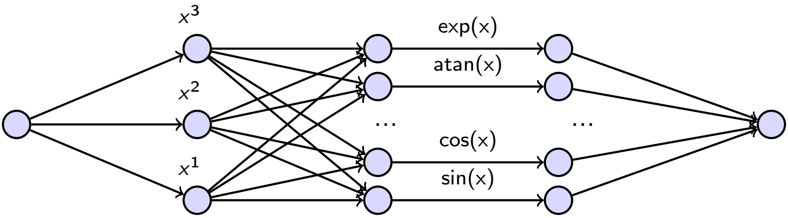


Figure 3: Structure of the PR model.

First model: the PR model (Polynomial - Linear - Radial - Linear layers):

- A polynomial layer, that creates multiple monomials from the input.
- A linear layer that combines them into a polynomial for each nonlinear function.

Structure of the PR Block

- The PR block is designed to extend the capabilities of the SINDy model.
- It is represented (in 1D) by:

$$f_\theta(x) = \sum_{j=1}^{l} c_j f_j \left(\sum_{i=1}^{d} \omega_{i,j} x^i + b_j \right) + B$$

- The trainable parameters are $\theta = \{c_j, \omega_{i,j}, b_j, B\}$.
- The f_j are functions from the dictionary $\mathcal{F}$.

PR Block: Advantages and Challenges

- **Advantage: Expressivity** - The PR block can express a composition of polynomials and functions from the dictionary. This reduce the number of necessary functions in the dictionary.
- **Challenge: Local Minima** - Classical problem in symbolic regression. Introducing a new nonlinear layer increases the likelihood of encountering local minima.
- **Solutions**:
  - Adjust the Lasso coefficient during the simulation.
  - Add a Brownian motion on the weight gradients in the learning process, at each epoch.
  - Reduce the model's parameter count (dictionary size, pruning, etc.).
  - Run several learning processes

PR Block: Example

Objective: Learn the function $f(x) = \cos(x^2)$ on the domain $[0, 3]$

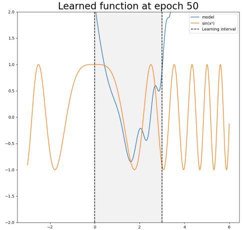
- **SINDy Limitations**: Cannot learn $\cos(x^2)$ unless it is explicitly in the dictionary.
- **PR's Strength**: Capable of learning compositions, such as x^2 with $\cos(x)$.

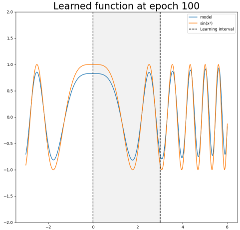
Learnt function (after training):

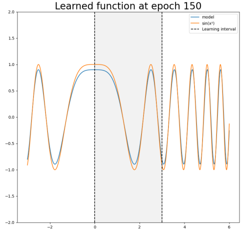
$$-0.985\cos(1.0x^2 + 3.13)$$

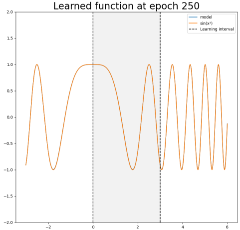
Achieves close approximation to $\cos(x^2)$. Only **3** out of **29** parameters are non-zero.
Sparsity: ✓

Evolution of the Learnt Model over Epochs









The PRP Block

An advanced model: the PRP model (Polynomial - Linear - Radial - Linear - Polynomial - Linear layers).

The PRP block is an extension of the PR model where:

- After the radial layer, we introduce another polynomial layer.
- This additional polynomial layer combines the results of the radial layer into different polynomial combinations.

The PRP model offers significantly improved expressivity.

Advantages and Challenges of the PRP Block

- **Advantage: Enhanced Expressivity** - The PRP block can represent more complex functions. In particular, a product of dictionary functions can now be expressed.
- **Challenge: Finding True Formula** - While the expressivity is high, the true formula is rarely identified.
- **Performance**: Even when the true formula is not identified, the PRP model typically demonstrate low Mean Squared Error (MSE).

Objective: Learn the function $f(X) = 2\sin(x_0)\cos(x_1)$ on the domain $[-2, 2]^2$

- **Impossible with SINDy** - With a 2-dimensional input, SINDy cannot express the required x_0 and x_1 combination.
- **Product of Dictionary Functions** - PRP can theoretically express a product of two functions.

Learnt Function:

$$-0.77\left(1 - 0.612\sin\left(0.032x_0^2 - 0.162x_0x_1 + 0.998x_0 + 0.035x_1^2 - x_1 - 0.14\right)\right)^2$$
$$+ 2.01\left(\cos\left(0.039x_0x_1 - 0.499x_0 - 0.497x_1 + 0.809\right)\right)^2$$

Sparsity: **14** non-zero coefficients from the original **32** parameters.

Not the exact function, but a close approximation over the domain.

PRP Block: Example

Objective: Learn the function $f(x) = 2\sin(x_0)\cos(x_1)$

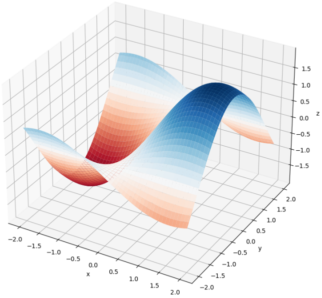


Figure 4: True Function

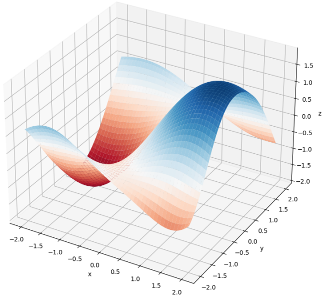


Figure 5: Model Prediction

Conclusion on the PRP Block

The PRP block:

- offers a promising approach for capturing complex function compositions
- balances between accurate function discovery and error minimization: it rarely captures the true formula
- can lead to sparser representations than classical regression approaches

Future work could focus on strategies to control the balance between sparsity and accuracy.

Conclusion on Nested SINDy

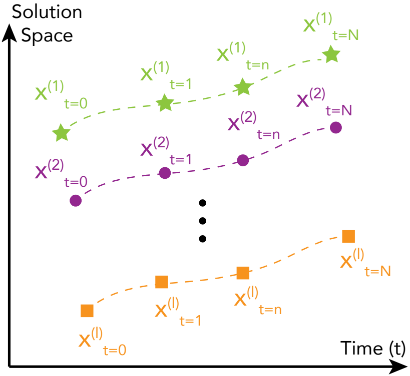
- **Strengths**:
  - Excels at capturing polynomial behaviors.
  - Effectively captures functions of the dictionary.
- **Challenges**:
  - More complex learning landscape.
  - Less guarantee of retrieving the true formula.
- In practice, adding one composition with polynomials does not prevent a proper learning (while greatly improving the expressivity).

In the next section, we will explore the application of Nested SINDy in ODE discovery.

Neural NestedSINDy for modeling dynamical systems

Introduced by Lee, Trask, and Stinis 2021 for Neural SINDy

Data:



Aim: Discover the unknown ODE $x'(t) = f(x(t))$ (i.e. recover f), from observed trajectories of this system.

How: By approaching the observed dynamics by the following dynamical system:

$$x'_\Theta(t) = f_\Theta(x(t))$$

where f_Θ is a NestedSINDy NN, parameterized by Θ.

Neural NestedSINDy

Loss function:

$$L(\Theta) = \frac{1}{n_{traj}} \sum_{l=1}^{n_{traj}} \sum_{i=1}^{N_t} \left\| x_{\Theta}^{(l)}(t_i) - x_{data}^{(l)}(t_i) \right\|_2 + \lambda_{lasso} \left\| \Theta \right\|_1$$

where $\left(x_{\Theta}^{(l)}(t_i) \right)_{i \geq 1} = ODEsolver \left(x_{data}^{(l)}(t_0), f_{\theta} \right)$ (Euler, RK2,dopri5,...)

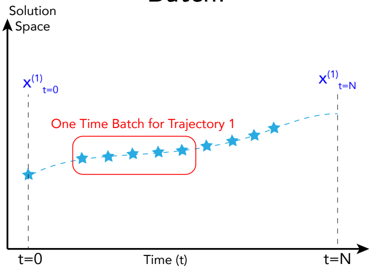
Neural SINDy - Training

Lee, Trask, and Stinis 2021

Algorithm 1: Neural SINDy training

```
Initialize Θ
for (i = 0; i < n_max; i = i + 1) do
    Sample n_batch trajectories randomly from D_train
    Sample initial points randomly from the sampled
      trajectories: x^r_{s(r)}, s(r) ∈ [0, ..., m − ℓ_batch − 1]
      for r = 1, ..., n_batch
    x̃(t_1),...,x̃(t_m)=ODESolve(x^r_{s(r)}, f_Θ, t_1,...,t_m),
      for r = 1, ..., n_batch
    Compute the loss L (Eq. (1))
    Update Θ via SGD
    Prune Θ based on the magnitude (Eq. (3))
end
```

Batch:



First experiments

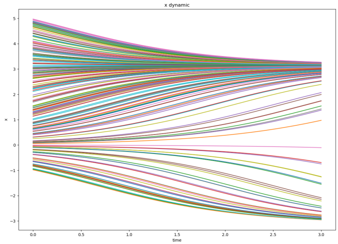
We performed some experiments, with "convenient" data, i.e. with dense, noise-free, data, for many trajectories, and with uniform time discretization along trajectories (however, the method is expected to be robust, as for the nSINDy presented in Lee, Trask, and Stinis 2021)

We tried to recover the dynamic of the following dynamic systems:

- An "easy" ODE (no composition) $x' = sin(x)$
- A "not so easy" ODE (composition) $x' = sin(x^2)$

$x' = sin(x)$

Data: 200 trajectories, 1000 sample points per trajectory



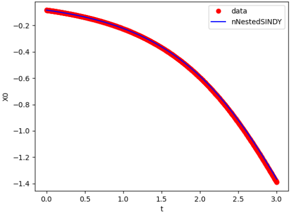
Neural network : PR

"Output space" :

$f_\Theta(x) = sin(P_1(x)) + P_2(x)$,

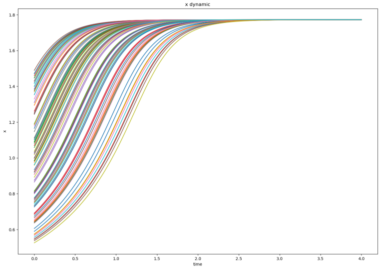
P_1 of degree 2, P_2 of degree 4

Results: Formula found without pruning : $-0.001x^2 + 0.005x + 0.991sin(1.000x - 0.002) + 0.004$ (see animation)



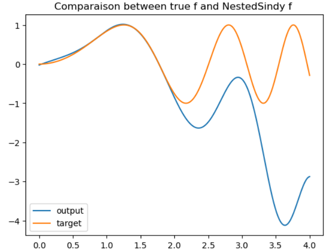
$x' = sin(x^2)$

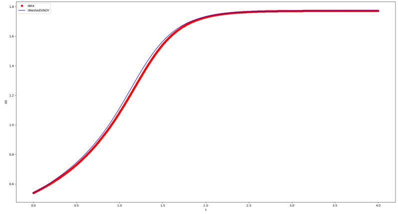
Data: 100 trajectories, 1000 sample points per trajectory



We do not recover $sin(x^2)$, but the dynamics is well approximated for initial data in $[0, \sqrt{\pi}]$. $f_\Theta(x) = -0.393x^2 + 0.291x + 1.19 sin(-0.983x^2 + 0.942x + 1.36) + 1.04 arctan(0.836x^2 + 0.304x - 0.935) - 0.406$

Results:





Conclusion

- Showcased the capabilities of Nested Neural SINDy for equation discovery.
- Future directions:
  - Refinement of hyperparameters such as learning rate, optimizer, and Lasso coefficient.
  - Exploration of SINDY-specific parameters:
    - Selection from a set of functions.
    - Decision-making on the choice of specific models.
    - Potential integration with predictive models for better parameter tuning.

Perspectives - numerical diffusion

Scalar conservation laws:

$$\partial_t u + \partial_x f(u) = 0$$

Finite volume scheme: $u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(F^n_{i+\frac{1}{2}} - F^n_{i-\frac{1}{2}}\right)$

with $F_{i+\frac{1}{2}} = \frac{f(u_i)+f(u_{i+1})}{2} - D(u_i^n, u_{i+1}^n)$, D numerical diffusion

Perspectives - numerical diffusion

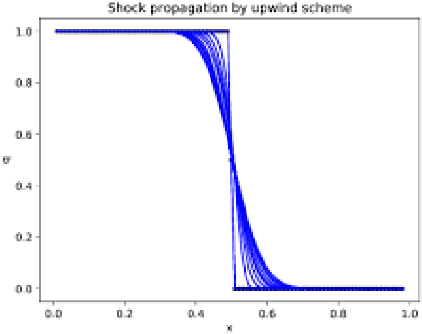
Scalar conservation laws:

$$\partial_t u + \partial_x f(u) = 0$$

Finite volume scheme: $u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(F_{i+\frac{1}{2}}^n - F_{i-\frac{1}{2}}^n\right)$
with $F_{i+\frac{1}{2}} = \frac{f(u_i)+f(u_{i+1})}{2} - D(u_i^n, u_{i+1}^n)$, D numerical diffusion
Example for the Upwind scheme with $f(u) = cu$:
$D(u_i^n, u_{i+1}^n) = \frac{c}{2}\left(u_{i+1}^n - u_i^n\right)$
The upwind scheme discretization is also the discretization of the following convection-diffusion equation:

$$\partial_t u + \partial_x(cu) - \frac{c\Delta_x}{2}\partial_{xx} u = 0$$

Perspectives - numerical diffusion



Goal: Find a interpretable diffusion term D that minimizes the error of the scheme, i.e. $\sum_{n=1}^{N} \| (u_i^n)_i - (u(t_n, x_i)_i \|$

Thank You

References I

Lee, Kookjin, Nathaniel Trask, and Panos Stinis (2021). "Structure-preserving Sparse Identification of Nonlinear Dynamics for Data-driven Modeling". en. In.

Sahoo, Subham, Christoph Lampert, and Georg Martius (2018). "Learning equations for extrapolation and control". In: *International Conference on Machine Learning*. PMLR, pp. 4442–4450.