Model Discovery of Partial Differential Equations

Gert-Jan Both

Model Discovery of Partial Differential Equations
Découverte de modèles d'équations aux dérivées partielles

Soutenue par
Gert-Jan Both
Le 10 decembre 2021

École doctorale n°474
Frontières de l'innovation en recherche et éducation

Spécialité
Machine Learning

Contents

Chapter 1

Introduction

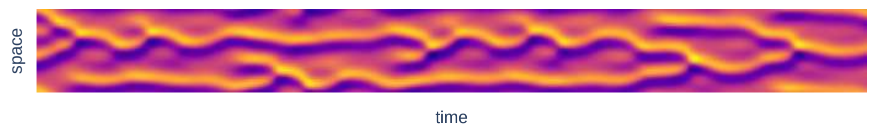

Figure 1.1: One-dimensional solution of the Kuramoto-Shivashinsky equation.

How much can a picture tell us? The image above shows an initial disturbance slowly breaking up into many small waves, which themselves merge and split as time progresses. Such complex behaviour is certainly beautiful to behold, but as scientists we wish take it a step further. Can we understand the underlying system and capture it in a (quantitative) model? This seems a daunting task considering figure 1.1, yet this process is at the heart of science. The classical approach to construct such models is known as *First-Principle modelling*. It starts, as the name implies, from basic principles and relies on simple abstractions and laws such as conservation of mass and energy to derive a model. FP modelling is an iterative process, tweaking and improving the model until it describes the observed data sufficiently, making it very time-consuming. Additionally, it requires expert knowledge; without any knowledge about the underlying system, it is next to impossible to come up with or improve the model. Its biggest drawback however is its ability to only model fairly simple systems. Figure 1.1 is a one-dimensional solution of the so-called Kuramoto-Shivashinsky equation,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} - u\frac{\partial u}{\partial x}. \tag{1.1}$$

While this is a fairly complex equation due to its fourth-order derivatives and non-linear terms, it does consist of only four terms. Faced with complex systems such as the airflow over a wing, the behaviour of crowd or the movement of impurities in a plasma, first-principle modelling becomes impossible; it would simply take too much time to derive a (effective) model describing all the interactions and processes. In these cases one resorts to *data-driven modelling.* In this paradigm we seek only to mathematically describe the data we observe. Instead of concerning itself with *why* the data looks like it does, it restricts itself to accurately modelling the data without knowledge of the underlying system. This does not make data-driven modelling more trivial than first-principle modelling: constructing a model which accurately describes the noisy, often high-dimensional data is no easy feat [1].

While data-driven modelling thus allows us to work with very complex systems, it lacks the key feature which makes first-principle modelling so powerful and popular: **interpretability**. This powerful property has various connotations across various fields, and we consider it here that a given dataset can be described by a fairly compact equation. Interpretable models in this form are implicitly tied to *trust* (can we prove mathematically a certain state can or cannot be reached?), *abstraction* (we can decompose complex behaviour as the interaction between subsystems), and *exploitation* (if we would suppress this interaction, we could use it), and thus forms a key property of science. The ideal approach would thus marry the interpretability of first-principle modelling to the ability of data-driven modelling to handle complex data. Simply put, we wish for an algorithm which, given figure 1.1, would be capable of autonomously discovering eq. (1.1) [2].

[1]A nice example of the time it takes for theories to develop is the anomalous perihelion procession of Mercury. It was already known in 1859 that Mercury's orbit could not be completely described by Newtonian mechanics, but it remained unexplained until Einstein's postulation of general relativity in 1915.

[2]A great example of the extrapolation power of models is Planck's law. To explain the blackbody emission spectrum, Planck derived a model which exchanged energy in discrete

1.1 Model discovery

The process of using an algorithm to autonomously discover a systems governing equation is known as *model discovery* [3]. Before I introduce the typical approaches to model discovery, I want to discuss what these models typically look like. In the context of model discovery we typically recognize three categories. Given some spatio-temporal data $u(x, t)$,

- **explicit models** give an explicit solution for the data we're measuring, $u(x, t) = f(x, t)$. Having an expression for the observable makes these among the most powerful, but in unfortunately these models can usually only be derived for a very restricted subset of systems.
- A much more common approach is to derive a model in terms of its temporal evolution, i.e. in the form of a differential equation $\partial u/\partial t = f(...)$. It is usually much simpler and more natural to specify a model in this way, as and such they admit a much larger class of problems than explicit models. We can recognize two distinct subclasses:
  - With **ordinary differential equations** [4] the propagation function f is a function of the state of the system, i.e. $\partial u/\partial t = f(1, u, u^2, ...)$,
  - With **partial differential equations** the propagation function also contains higher-order partial derivatives with respect to other (spatial) features, i.e. $\frac{\partial u}{\partial t} = f(1, u, u_x, ...)$. The distinction between ODEs and PDEs might seem minor, but, as I will discuss later on, the spatial derivatives make discovering PDEs a much more challenging problem.

These categories are ofcourse an oversimplification, but they correspond well to the different approaches and main problems encountered in model discovery. No matter the type of model, a first, naive approach to model discovery would be to simply fit *all* possible models and compare them. However, there are simply too many models to construct and test - such an approach would take prohibitively long, even on small datasets with powerful computers. Luckily, this brute-force approach is not necessary: all we need is a method to smartly search the space of all models.

amounts. While he initially thought this to be wrong, he inadvertently (and unwillingly) opened the door to quantum physics!

[3] This term is mostly used by natural scientists working on physical datasets. Several other terms are in use, depending on the field and the exact goal. In engineering, it is often referred to as system identification, while in machine learning it is often known as causal inference.

[4] In this setting also often referred to as dynamical systems.

1.1.1 Symbolic regression

Symbolic regression covers a large class of approaches to model discovery, which all, to a varying degree, are based on searching the space of all possible models. Before discussing various approaches to this search, we must first understand *how* to computationally construct a symbolic equation. Equations can be considered so-called expression trees. For example, the expression tree for the equation $(a + b) \cdot c + 7$ is given by figure 1.2.

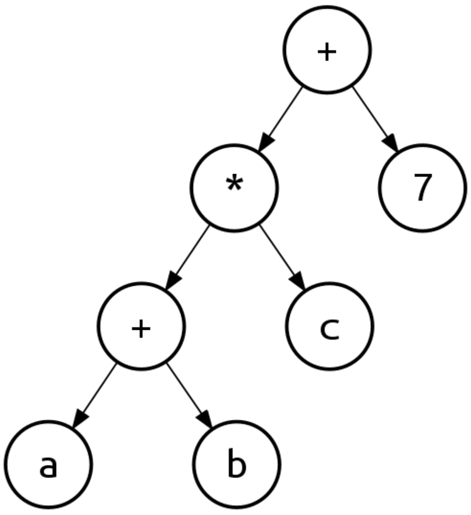

Figure 1.2: Example of binary expression tree. Figure taken from https://en.wikipedia.org/wiki/Binary_expression_tree.

The expression tree approach allows to computationally work with symbolic equations - for example, to represent the equation $(a + b) \cdot c + 7 \cdot d$, simply add '*' and 'd' nodes to the '7' node. In other words, expression trees also easily allow to change a given equation. This idea forms the core of genetic programming (GP), one of the most popular approaches to symbolic regression. Starting with a random population of these trees, GP randomly mutates the trees, conserving only the well-performing trees and repeat this until the convergence, analogous to how a population of genes evolves [Bongard and Lipson, 2007, Schmidt and Lipson, 2009, Maslyaev et al., 2019]. Alternatively, Petersen [2020] train a recurrent neural network to generate the trees corresponding to the unknown equations. A completely different approach to symbolic regression is taken by the team behind AI-Feynman [Udrescu and Tegmark, 2019, Udrescu et al., 2020]. They consider specifically equations originating from physics, and note that these equations often possess symmetry, separability or other various simplifying properties. By recursively applying these simplifications, the problem is split into a number of much easier to solve sub-problems, discovering a variety of seminal equations from physics. A final approach worth mentioning is that of the Bayesian model scientist [Guimerà et al., 2020]. While the entire model space might be (infinitely) large, in practice many models and equations look alike or share much of their expression. Having mined all the equations on Wikipedia, the Bayesian model scientist generates various hypotheses for the form of the equation by combining fitting with this knowledge of all previous equations.

Symbolic regression is an active area of research, with the field seemingly focusing on discovering complex, explicit equations from synthetic, low-noise datasets [Udrescu and Tegmark, 2019]. However, most real, physical systems are governed by (coupled) differential equations, and the observations are noisy and sparse. Symbolic regression struggles in these settings, and as I will discuss later, the difficulty lies not in finding the equation, but in accurately calculating the features (i.e. the derivatives). An alternative approach based on sparse regression is much more for these systems and settings.

1.1.2 Sparse regression

Model discovery can be greatly simplified if we consider only differential equations - this doesn't reduce the generality of our approach, as many physical systems are expressed as DEs. Considering again equation (1.1), note that it essentially is a linear combination of features, just like many other PDEs. This implies that many models can be written as [5],

$$\begin{aligned}\frac{\partial u}{\partial t} &= w_0 X_0 + ... + w_M X_M \qquad (1.2)\\ &= w X^T \qquad (1.3)\end{aligned}$$

where w_i is the coefficient corresponding to feature X_i and X the full feature matrix. Even within this restricted model space (i.e. the model is a linear combination of features), a combinatorial approach remains prohibitively expensive. However, by constructing a large set of possible features and combining these into a single matrix Θ, model discovery can be approached as a regression problem,

$$\tilde{w} = \min_w \left\| \frac{\partial u}{\partial t} - w\Theta \right\|^2 . \qquad (1.4)$$

[5] I'm neglecting a huge group of models here: those with spatially or temporally varying coefficients! I'll return to this issue in the next chapter.

where $\tilde{w}$ determines the model: $\tilde{w}_i = 0$ if feature i is not a part of the equation. More precisely, this has transformed model discovery into a *variable selection* problem; finding the correct model involves selecting the correct (i.e. non-zero) components of w. The number of candidate terms N is typically much larger than the number of terms making up the equation M, implying that most components of $\tilde{w}$ *should* be zero. To promote such solutions, a sparsity promoting penalty is added - hence this approach is known as **sparse regression**. It was pioneered by Brunton et al. [2016] and has since become the de-facto method of performing model discovery.

1.2 Contributions

The sparse regression approach is elegant, flexible and widely applicable, but struggles on noisy and sparse data. When working with differential equations, the features making up the equation ($\partial u / \partial t$ for both ODEs and PDEs and higher order spatial derivatives for PDEs) are not directly observed. Rather, the observed data $u(x, t)$ must first be numerically differentiated to calculate the features. Numerical differentiation fundamentally is build on approximations, and when faced with sparse and noisy data it produces highly inaccurate results [6]. This in turn fundamentally limits model discovery to low-noise, densely sampled datasets; if the features are corrupted, no approach will be able to select the right ones. As higher-order derivatives are more inaccurate, this issue specifically affects model discovery of PDEs, which often contains several higher-order spatial derivatives. If the calculation of the features is the limiting factor, this implies that model discovery should be approached as a distinct two-step process: calculating the features (step 1) is just as important, if not more, as selecting the right variables (step 2).

Motivated by this line of reasoning, we realized that if our goal was to apply model discovery on experimental data, improving the accuracy of the features would have much more impact than constructing more robust sparse regression algorithms. A common approach to improve the accuracy of the features is to approximate the data using some data-driven model, for example using polynomials or a fourier series, and perform sparse regression based on the features calculated from that approximation. My work is build on the idea that by integrating these two steps, model discovery can be made much more robust: the approximation makes it easier to discover the governing equation, while this equation in turn can be used to improve the approximation. In other words, I argue that data-driven and first-principle modelling are not opposites, but that they can mutually improve each other[7].

[6]I'll illustrate later how bad numerical differentiation really is.

[7]More generally put, data-driven modelling improves first-principle modelling by yield-

Specifically, my work focuses on 1) constructing accurate representations of (noisy) data using neural networks and 2) how to incorporate the knowledge of the governing equation back into these networks. Our overarching goal was to improve model discovery of partial differential equations on noisy and sparse datasets, opening the way for use on real, experimental datasets.

In Both et al. [2019] we show that such neural networks significantly improve the robustness of model discovery compared to unconstrained neural networks or classical approaches. Improving on this, in Both et al. [2021b] we present a modular framework, showing how these constrained networks can utilize any sparse regression algorithm and how to iteratively apply the sparse regression step, boosting performance further. In our latest work [Both and Kusters, 2021], we build upon the first two articles to construct a fully differentiable model discovery algorithm by combining multitask learning and Sparse Bayesian Learning. Besides these methodological papers, we show in Both et al. [2021a] that neural network based model discovery needs much less samples to find the underlying equation when they are randomly sampled, as opposed to on a grid. This marks a fundamental difference with classical approaches, which struggle with off-grid sampled data. In Both and Kusters [2019] we introduce Conditional Normalizing Flows. CNFs allow to learn a fully flexible, time-dependent probability distribution, for example to construct the density of single particle data. In Both and Kusters [2021] we build on this to discover a population model directly from single particle data. This series of articles was released under the name DeepMoD - Deep Learning based Model Discovery. Most of this work is collected in an open-source software package called DeePyMoD. Taken together, our work strongly establishes the argument for neural network based surrogates, constrained by the underlying physics, for model discovery on real, experimental data.[8]

1.3 Organization of this thesis

I've written my thesis using Distill for R Markdown, a format based on the excellent online journal Distill. Distills typesetting and typography is strongly based on the Tufte style and is optimized for digital viewing: I've included some interactive plots, and hover over footnotes to see their content [9] or over citations to see the full one - often with a link to an Arxiv version [Both and Kusters, 2021]. All this means that while you can read it on paper, you will miss out on these features, so I highly recommend using this

ing providing better features, while first-principle modelling improves data-driven modelling by infusing it with physical biases.

[8]I've worked on several other topics, but I have chosen to focus on my main projects.

[9]No more scrolling!

website [10].

This thesis itself is also a small experiment. Instead of writing a long static document which is likely to only be read by the committee, can I write something a little more modern which will be useful to others, perhaps those not familiar with the field? Instead of writing a comprehensive review examining all the papers in the field, can I write something which gives a broader overview, discusses key papers in-depth and introduce the key challenges? The website in front of you is the answer I came up with.

- I have split the background of my work into two pages: **Regression** shortly discusses the regression approach to model discovery and presents the key papers of the field. As I argue in this thesis, surrogates should be a key component of model discovery algorithms. In **Surrogates** I discuss various classical approaches, but will spend the bulk of the chapter on Physics Informed Neural Networks (PINNs).
- **DeepMoD** is the umbrella under which I've collected all of my work done during my PhD on model discovery from January 2019 until July 2021. I start by explaining the main idea behind our approach, and then summarize our papers and list their respective contributions.
- In **Discussion** I reflect on our work and put it in context. Finally, I consider the open questions and challenges for model discovery.

Each chapter article can be read on its own, but I suggest following the order in which I've introduced them above. I hope this thesis is able to convey all that I've learned and done the last few years - happy reading!

[10]Distill has an excellent article discussing the uses of interactive articles.

Chapter 2

Regression

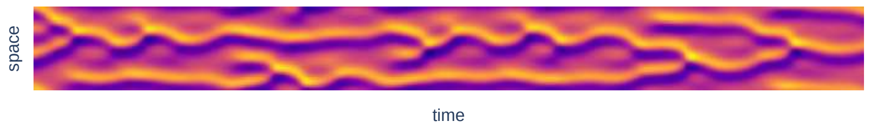

Figure 2.1: One-dimensional solution of the Kuramoto-Shivashinsky equation.

Consider again the Kuramoto-Shivashinsky equation,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} - u\frac{\partial u}{\partial x}. \tag{2.1}$$

It is a fourth order, non-linear partial differential equation giving rise to the beautifully rich behaviour of figure 2.1. This complexity - a fairly unusual fourth-order derivative and a non-linear term - also poses a challenge for model discovery. Is it really possible to discover an equation this complex autonomously from data? In this chapter I introduce how sparse regression makes this possible, using the Kuramoto-Shivashinksy equation as an example [1]. Before reviewing regression and various sparsity-promoting regularizations, I shortly consider *why* the sparse regression technique can be used. This chapter ends with an overview of the key extensions which have been developed.

2.1 Structure of differential equations

While the Kuramoto-Shivashinksy equation appears complex, its structure is not particularly unique. In fact, it shares many features with other models. Consider for example the Cahn-Hilliard equation, which describes phase separation of a fluid with concentration c, diffusion coefficient D and a length scale $\sqrt{\gamma}$:

$$\frac{\partial c}{\partial t} = D\nabla^2\left(c^3 - c - \gamma\nabla^2 c\right), \tag{2.2}$$

or the Korteweg-de Vries equation, which describes the speed u of waves in shallow water,

$$\frac{\partial u}{\partial t} = -\frac{\partial^3 u}{\partial x^3} + 6u\frac{\partial u}{\partial x}. \tag{2.3}$$

All three equations describe completely different systems, but share a similar structure: each consists of 1) a linear combination of 2) relatively simple features[2]. Indeed, considering the list of non-linear partial differential equations on Wikipedia, only very few equations have features composed of more than two terms, such as $u \cdot u_x \cdot u_{xxx}$. Property 2 implies that most models can be constructed from a small, finite set of elements consisting of [3],

1. **Polynomials** of the observable u up to order p: $1, u, u^2, ...$
2. **Derivatives** up to order q: $u_x, u_{xx}, u_{xxx}, ...$
3. **Combinations** of these two features: $u \cdot u_x, u^2 \cdot u_{xxx}, ...$

[1]For more information on the KS equation, see Wikipedia or Encyclopedia of Math.

[2]Although they all share a diffusive term - we can't seem to escape entropy.

[3]An interesting question is if this is a fundamental fact or a consequence of our approach to physics.

For most known equations, $p \leq 3$ and $q \leq 4$, typically yielding 20-30 candidate features. Even at this size, a combinatorial search is computationally too expensive, as it involves solving the PDE. However, by exploiting property 1, i.e. that the unknown equation is a linear combination of the candidate features, model discovery can be expressed as a regression problem [4].

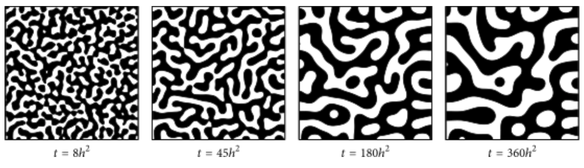

Figure 2.2: Solution of the Cahn-Hilliard equation. Image taken from https://www.hindawi.com/journals/mpe/2016/9532608/

2.2 Regression for model discovery

Expressing model discovery as a regression problem starts by combining all M candidate features into a single library matrix Θ,

$$\Theta = \begin{bmatrix} | & | & | & | & | \\ 1 & u & u_x & ... & u^2 u_{xx} \\ | & | & | & | & | \end{bmatrix} \tag{2.4}$$

i.e. each column of $\Theta \in \mathcal{R}^{N \times M}$ contains a single feature. Assuming the equation is a linear combination of these features, the unknown equation can be written as,

$$\frac{\partial u}{\partial t} = \Theta \xi \tag{2.5}$$

where $\frac{\partial u}{\partial t} \in \mathcal{R}^N$ is the time derivative and $\xi \in \mathcal{R}^M$ the coefficient vector describing the equation. If a component ξ_j is zero, the corresponding feature Θ_j is not a part of the equation, and vice versa. For example, if $\Theta = \begin{bmatrix} 1 & u & u_x & uu_x \end{bmatrix}$,

[4] I'm only considering non-coupled, one-dimensional cases here, but the argumentation extends to higher dimensions. The library size will be much bigger, and the combinatorial search will be even more expensive, but most features will still be a combination of two components.

$$\xi = \begin{bmatrix} 0.0 & 0.5 & 0.0 & -2.0 \end{bmatrix}^T \tag{2.6}$$

corresponds to the equation

$$\frac{\partial u}{\partial t} = \frac{1}{2}u - 2\frac{\partial^2 u}{\partial x^2}. \tag{2.7}$$

With eq. (2.5), instead of comparing all models which can be build with the candidate features, now we only need to find a single vector ξ by minimizing [5],

$$\hat{\xi} = \min_{\xi} \|u_t - \Theta\xi\|_2^2 \tag{2.8}$$

This least-squares problem is trivial to solve [6], but will, unfortunately, not yield the correct result in most cases [7]. Since the number of features M is likely to be much larger than the number of terms making up the unknown equation, solving eq. (2.8) will overfit the model: very little if any components of $\hat{\xi}$ will be zero. Luckily, equation (2.8) can be adapted to yield results $\hat{\xi}$ win which many components are *exactly 0*, a property known as **sparsity**. This is known as sparse regression and this approach to model discovery was pioneered by Brunton et al. [2016] for ODEs and extended to PDEs in Rudy et al. [2017] . Various ways to promote the sparsity of solutions to eq. (2.8) exist, and I discuss those in the next section.

2.3 Regularized regression

To promote sparsity of the resulting vector, we augment the objective function eq. (2.8) with a regularization function $R(\xi)$ on the unknown coefficients ξ [8],

[5]The way I like to think about this, is that with this approach we're essentially fitting *all* models at the same time.

[6]It has an analytical solution but for numerical stability one often uses QR decomposition or SVD. Addtionally, least squares is a so-called BLUE - which means that the estimate $\hat{\xi}$ is unbiased and has the lowest variance as long as the errors are uncorrelated.

[7]Technically this norm is only appropriate for Gaussian, white noise (i.e. zero-mean, uncorrelated). Extensions to other noise models are called generalized linear models, but in practice only white noise is used.

[8]The $1/N$ factor is not strict required, but has the benefit of making the regularization strength λ sample size-independent.

$$\hat{\xi} = \min_{\xi} \frac{1}{N} \|u_t - \Theta\xi\|_2^2 + \lambda R(\xi) \tag{2.9}$$

where λ sets the strength of the regularization. The choice of $R(\xi)$ affects which behaviour is penalized. Three common choices are,

- $R(\xi) = \|\xi\|_0$, also known as ℓ_0 regularization, penalizes the *number* of non-zero components, promoting sparsity in the final solution.
- $R(\xi) = \|\xi\|_1$, known as ℓ_1 regularization, penalizes the *magnitude* of ξ *linearly*, also promoting sparsity in the final solution.
- $R(\xi) = \|\xi\|_2^2$, also called ℓ_2 regularization, penalizes the *magnitude* of ξ *quadratically*, preventing large magnitude coefficiens in the final solution.

These penalties can also be combined: for example, the elastic net [Zou and Hastie, 2005] combines the benefits of both ℓ_1 and ℓ_2 regularization by defining

$$R(\xi) = \alpha \|\xi\|_1 + (1-\alpha) \|\xi\|_2^2 . \tag{2.10}$$

Since these regularizations are central to sparse regression and model discovery, I want to discuss each of these three regularizations in-depth.[9]

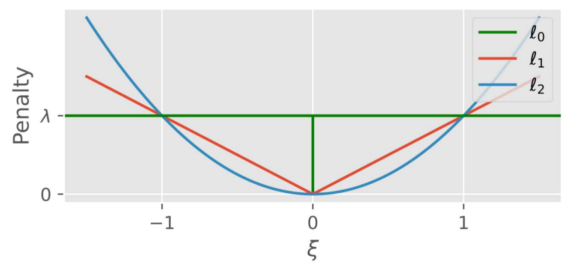

Figure 2.3: Penalty functions of ℓ_0, ℓ_1 and ℓ_2 regularization.

2.3.1 ℓ_0

ℓ_0 regularization penalizes the *number* of non-zero components through the penalty function,

[9]More generally, ξ can be penalized by any p-norm, but in practice only $p = 1, 2$ are used.

$$\|\xi\|_0 = \sum_j \xi_j^*, \quad \xi_j^* = \begin{cases} 1 & \text{if } \xi_j \neq 0 \\ 0 & \text{if } \xi_j = 0 \end{cases} \tag{2.11}$$

This makes it an obvious choice for promoting sparsity: if the amount of terms is penalized, solutions with fewer terms are favoured, and additionally, it does not affect the magnitude of the components, making it an unbiased estimator. Unfortunately, by penalizing the number of terms (also known as the *support*) the objective function becomes non-differentiable and non-convex. Simply put, this regularization still effectively involves solving a combinatorial problem. Maddu et al. [2019] show that an approximation known as Iterative Hard Thresholding can be an effective method in model discovery. A different avenue of research tackles these issues head-on by taking a probabilistic approach. For example, Louizos et al. [2018] consider the penalty as the sum of Bernouilli random variables, allowing gradient-based optimization. Nonetheless, these approaches are still computationally expensive.

2.3.2 ℓ_1 - Lasso

Given these issues, a common approach is to relax the problem and apply ℓ_1 regularization,

$$\|\xi\|_1 = \sum_j |\xi_j| \,. \tag{2.12}$$

Compared to an ℓ_0 regularized problem, this too yields sparse solutions (I will discuss *why* it does later on), but is convex and can be efficiently solved using proximal method such as FISTA [Beck and Teboulle, 2009], despite its non-differentiability at $x = 0$. Usually this approach is called Lasso (Least Absolute Shrinkage and Selection operator), as it the sparsest norm still yielding a convex problem [Tibshirani, 2011].

The Lasso is a popular approach when sparsity is required, but it does not come without its drawbacks. It yields a biased estimate of the coefficients, but a more significant issue is that it is not guaranteed to be *consistent*[10]. An estimator is consistent if the estimate approaches the true value with increasing sample size:

$$\hat{\xi} \rightarrow \xi_{\text{true}} \quad \text{as } N \rightarrow \infty \tag{Consistency}$$

[10]In literature this sometimes also referred to as the oracle property.

While this might seem like a trivial point, its implications are not: an inconsistent estimator will not be able to recover the truth, even with an infinite number of samples. The Lasso is consistent only if the Irrepresentable condition (IRC) is satisifed [Zhao and Yu]. In practice the IRC is often violated, implying the Lasso is unable to recover the ground truth. The violation is caused by correlation between variables, and the lasso can be made consistent by scaling the components in the penalty. For example, the adaptive Lasso [Zou, 2006] scales the penalty as

$$\sum_j w_j \left|\xi_j\right| \tag{Adaptive Lasso}$$

where $w_j = 1/\left|\xi_j^*\right|^\gamma$, with ξ_j^* the result of a different consistent estimator and γ a scaling factor.

2.3.3 ℓ_2 - Ridge

The third popular regularization is an ℓ_2 penalty,

$$\sum_j \left|\xi_j\right|^2, \tag{Ridge}$$

also known as ridge regression. Ridge regression can be analytically solved, thus having essentially the same computational cost as least squares [11], and can handle highly correlated variables. Unfortunately, ridge regression favours results where none of the coefficients are large, but not exactly zero either - in other words, it does not yield a sparse coefficient vector. To see why Lasso yields sparse results while ridge regression does not, consider figure 2.4. The ovals show the level sets of the loss due the fitting term (i.e. the first term of eq. (2.9)) and the loss due the regularization is represented by the circle (ridge) and square (lasso). The solution of the regularized problem is the intersection of these two losses. For the ℓ_1 regularization, the intersection happens at the corners, leading to sparse coefficients, whereas for the ℓ_2 regularization it happens in the middle, yielding small but non-zero coefficients. Model discovery often relies on heuristics and some approaches based on ridge regression show good performance, despite it not mathematically yielding sparse coefficients. For example, SINDy [Brunton et al., 2016], one of the most popular model discovery approaches, uses ridge regression.

[11]Simply add $\sqrt{\lambda}$ along the diagonal of the gram matrix, or augment the data and solve it like a standard least squares problem

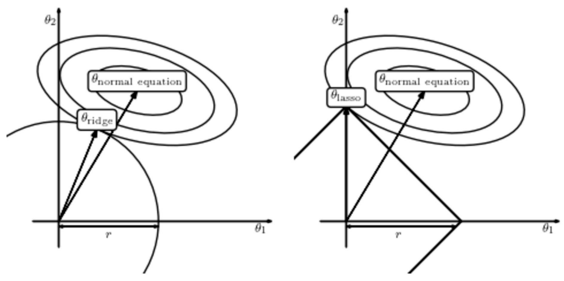

Figure 2.4: Comparison of Lasso and Ridge losses. Figure from https://www.astroml.org/book_figures/chapter8/fig_lasso_ridge.html

2.4 Heuristics

Besides these regularizations, various other heuristic are often applied to improve the sparsity of the obtained coefficients. I discuss two commonly used ones: thresholding and sequential regression.

2.4.1 Thresholding

It is quite common that we recover a sparse vector which is approximately right: the required terms stand out, but several other small but non-zero components exist, despite regularization. Performance can be strongly improved by **thresholding** this sparse vector. Since all components have different dimensions, the features are first normalised by their ℓ_2 norm [12],

$$\Theta_j^* = \frac{\Theta_j}{\left\|\Theta_j\right\|_2}, \tag{2.13}$$

allowing us to define normalized coefficients as,

$$\xi_j^* = \xi_j \cdot \left\|\Theta_j\right\|_2 . \tag{2.14}$$

Given some threshold value η, the threshold operation is then defined as

$$\xi_j = \begin{cases} \xi_j & \text{if } \xi_j^* \geq \eta \\ 0 & \text{if } \xi_j^* < \eta \end{cases} \tag{2.15}$$

[12]We can also normalize the target vector, i.e. the time derivative, to make everything dimensionless.

The value of η is usually preset (i.e. a hyperparameter), but it can also be learned [Rudy et al., 2017].

2.4.2 Sequential regression

A second trick is to **sequentially refine** the estimate by iteratively performing sparse regression on only the active features selected by the previous estimator,

$$\xi^{i+1} = \min_{\xi_i} \frac{1}{N} \left\| u_t - \Theta^i \xi^i \right\|_2^2 + \lambda R(\xi^i) \tag{2.16}$$

$$\Theta^{i+1} = \Theta^i [\xi^{i+1} \neq 0] \tag{2.17}$$

until ξ converges. While this does bias the results as coefficients can only be removed, not added, in practice it works well.

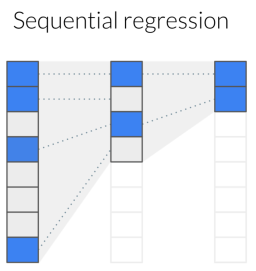

Figure 2.5: Illustration of sequential regression.

2.5 Extensions

Sparse regression forms a very flexible and general approach to model discovery and has been adapted, extended and improved in various ways since its introduction. In the next section I discuss the key papers which have either solved important open questions, or make significant progress towards it.

2.5.1 Stability selection

In the previous section we considered the effect of different regularizations on sparsity. An additional important factor is the strength of this regularization, denoted by the factor λ in eq. (2.9). Figure 2.6 shows the solution $\hat{\xi}$ as a function of the regularization strength λ. The resulting sparse vector is strongly dependent on λ; the lower the amount of regularization, the more features are selected.

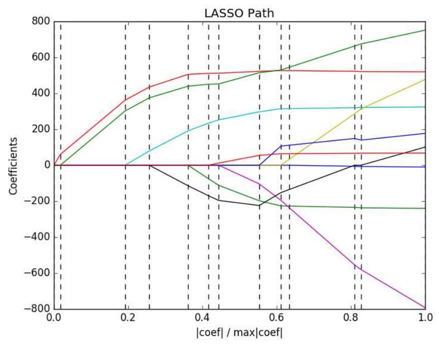

Figure 2.6: Lasso path. Figure from https://scikit-learn.org/0.18/auto_examples/linear_model/plot_lasso_lars.html

The default method to choose hyper-parameters such as λ is cross-validation (CV). CV consists of splitting the data into k folds[13], training the model on $k-1$ folds and testing model performance on the remaining fold. This process is repeated for all folds and averaging over the results yields a data efficient but computationally expensive approach to hyperparameter tuning. CV optimizes for *predictive* performance - how well does the model predict unseen data? -, but model discovery is interested in finding the underlying structure of the model. Optimizing for predictive performance might not be optimal; a correct model certainly generalizes well, but optimizing for it on noisy data will likely bias the estimator to include additional, spurious, terms. An alternative metric which optimizes for variable selection is stability selection [Meinshausen and Buehlmann, 2009].

The key idea is that while a single fit probably will not be able to discriminate between required and unnecessary terms we expect that among multiple fits on sub-sampled datasets the required terms will consistently be non-zero. Contrarily, the unnecessary terms are essentially modelling the error and noise, and will likely be different for each subsample. More specifically, stability selection bootstraps a dataset of size N into M subsets I_M of $N/2$ samples, and defines the selection probability Π_j^λ as the fraction of subsamples in which a term j is non-zero

$$\Pi_j^\lambda = \frac{1}{M}\sum_M 1(j \in S^\lambda(I_M)) \tag{2.18}$$

[13]subsets

where S_j^λ is the recovered support, i.e. whether a term is non-zero. The support can be determined using any sparse regression method: Meinshausen and Buehlmann [2009] use Lasso, but Maddu et al. [2019] use a relaxed ℓ_0 penalty. Calculating the selection probability for a range of regularization strengths Λ yields the stability path (figure 2.7).

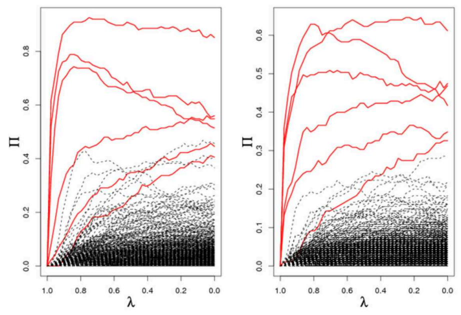

Figure 2.7: Stability paths. Figure from https://thuijskens.github.io/2018/07/25/stability-selection

Using the stability path, the final support is determined by selecting all components with a selection probability above a given threshold π_{thr},

$$S^{\text{stable}} = \{j : \max_{\lambda \in \Lambda} \Pi_j^\lambda > \pi_{\text{thr}}\} \tag{2.19}$$

As the active components are determined from a range of λ, stability selection does not require making a point estimate for the regularization strength as with CV. Stability selection has been shown to outperform cross validation in model discovery, especially at higher noise levels [Maddu et al., 2019, 2020].

2.5.2 Multi-experiment datasets

A dataset rarely consists of a single experiment - they are comprised of several experiments, each with different parameters, initial and boundary conditions. Despite these variations, they all share the same underlying dynamics. To apply model discovery on these datasets we need a mechanism to exchange information about the underlying equation among the experiments. Specifically, we need to introduce and apply the constraint that all experiments share the same support (i.e. which terms are non-zero).

Consider k experiments, all taken from the same system, $\{\Theta \in \mathcal{R}^{N\times M}, u_t \in \mathcal{R}^N\}_k$, with an unknown sparse coefficient vector $\xi \in \mathcal{R}^{M\times k}$. Applying sparse regression on each experiment separately is likely violate this constraint; while it will yield a sparse solution for each experiment, the support will not be the same (see figure 2.8). Instead of penalizing single coefficients, we must penalize groups - an idea known as group lasso [14] [Huang and Zhang, 2009]. Group lasso first calculates a norm over all members in each group (typically an ℓ_2 norm), and then applies an ℓ_1 norm over these group-norms (see figure 2.8),

$$R(\xi) = \sum_{j=1}^{k} \left| \sqrt{\sum_{i=1}^{M} |\xi_{ij}|^2} \right| \qquad \text{(Group sparsity)}$$

This approach drives groups of coefficients to zero, rather than single coefficients, and thus satisfies the constraint (see figure 2.8). Group sparsity has been successfully used for model discovery on multiple experiments [de Silva et al., 2019], or with space- or time-dependent coefficients (see next section), but can also be exploited to impose symmetry [Maddu et al., 2020].

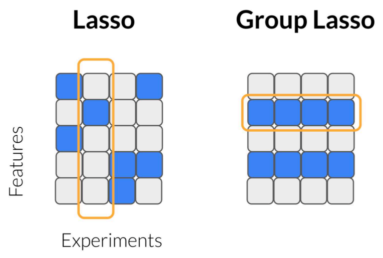

Figure 2.8: Lasso applies the penalty vertically, while group lasso applies the penalty horizontally.

2.5.3 Time/space dependent coefficients

So far we have implicitly assumed that all the governing equations have fixed coefficients. This places a very strong limit on the applicability of model discovery; usually coefficients are fields depending on space or time (or even both!), so that $\xi = \xi(x, t)$. As a first attempt at solving this, Rudy et al. [2018] learn the spatial or temporal dependence of ξ by considering each corresponding slice as a separate experiment with different coefficients, and applying the group sparsity approach we introduced in the previous section. A Bayesian version of this approach was studied by Chen and Lin [2021].

[14]Technically this is group sparsity, but since almost always lasso is used, these terms are used interchangeably.

It is important to note that in both these works $\xi(x, t)$ is not parametrized; it is not the functional form which is inferred, but simply its values at the locations of the samples. [15]

2.5.4 Information-theoretic approaches

The sparse regression approach tacitly assumes that the sparsest equation is also the correct one. While not an unreasonable assumption, it does require some extra nuance, as there often is not a single correct model. Many (effective) models are constructed as approximations of a certain order depending on the accuracy required. A first order approximation yields the sparsest equation, while the second order model could describe the model better. Another example is that a system can be locally modelled by a simpler, sparser equation. In both cases neither model is incorrect - they simply make a different trade-off. Information-theoretic approaches give a principled way to balance sparsity of the solution with accuracy.

Consider two models A and B, each with likelihood $\mathcal{L}_{A,B}$ and $k_{A,B}$ terms. Model B has a higher likelihood, $\mathcal{L}_B > \mathcal{L}_A$, indicating a better fit to the data, but also consists of more terms, $k_B > k_A$. The Bayesian Information Criterion (BIC) and the closely related Akaike Information Criterion (AIC) are two metrics to decide which balance these two objectives,

$$\mathrm{BIC} = k \ln n - 2 \ln \mathcal{L} \tag{2.20}$$

and the Akaike Information Criterion,

$$\mathrm{AIC} = 2k - 2 \ln \mathcal{L}. \tag{2.21}$$

If model B has a lower AIC or BIC than model A it is a better, even if $k_B > k_A$. In other words, these criteria give a principled way to decide the trade-off between how well the model fits fits the data and the complexity of the model. This approach has been used by Mangan et al. [2017] to decide between various discovered models, and a similar approach based on minimum description length was used by Udrescu and Tegmark [2019] .

As we discussed, it is likely for multiple models to be correct simultaneously, all with similar BIC or AIC. Each of these models is essentially the model best describing the data at a certain accuracy, and all of the models together

[15]A interesting approach not based on sparse regression is taken by Long et al. [2019] . They note that since derivatives can be approximated as a convolution, learnable filters can be used to discover the underlying equation, even with spatially varying sources.

describe a curve in model space known as the Pareto Frontier [16]. All models on this curve are correct, but of varying complexity, and we merely need to pick the model corresponding to the accuracy to which we wish to describe the data [Udrescu et al., 2020].

2.5.5 Bayesian approaches

The requirement that the resulting coefficient vector ξ be sparse is essentially a form of prior knowledge. A principled way to include such prior knowledge is given by Bayes' theorem, which states for some data X and parameters w

$$\overbrace{p(w \mid X)}^{\text{posterior}} = \frac{\overbrace{p(X \mid w)}^{\text{likelihood}} \ \overbrace{p(w)}^{\text{prior}}}{p(X)}. \tag{2.22}$$

The likelihood describes the probability of the data X for a given parameter w, while the prior describes the probability distribution of the parameter w. The posterior then is the distribution of w given the data and the prior. Contrary to maximum likelihood approaches which yield point estimates, Bayesian approaches yield distributions, allowing to quantify uncertainty. Additionally, the inclusion of the prior effectively regularizes the problem, making Bayesian regression typically much more robust than MLE approaches.

The sparsity of ξ can be encoded in the prior in various ways. A common approach known as Sparse Bayesian Learning [Tipping, 2001] takes a zero-centred Gaussian as prior,

$$p(\xi_j \mid \alpha_j) = \mathcal{N}(w_i \mid 0, \alpha_j^{-1}). \tag{2.23}$$

The SBL assumes all terms are zero, but the confidence of that decision (α_j) is inferred from data. As $\alpha_j \to \infty$, the prior becomes infinitely sharp at zero, and the component is pruned from the model. The Gaussian prior is not particularly sparse, but Yuan et al. [2019] nonetheless show good performance with model discovery. Sparsity can be more strongly promoted by using a Laplacian prior to yield a Bayesian Lasso [Helgøy and Li, 2020], or a so-called spike-and-slab-prior [Nayek et al., 2020].[17]

[16]Pareto optimality is a very wide applicable concept, as it essentially describes how to make the trade-off between multiple conflicting objectives.

[17]The field of Bayesian approaches to model selection is so rich it deserves a book on its own. I restrict myself here mainly to some approaches used in the model discovery of PDEs.

Chapter 3

Differentiation and surrogates

Surrogates [1] are a widely used approach to approximate a dataset or model by a different, data-driven model with certain desirable properties. For example, a computationally cheap surrogate can be used to replace a computationally expensive simulation, or an easily-differentiable surrogate can be used to approximate a dataset we wish to differentiate. I discuss various surrogates used in the model discovery community and make the case for neural networks as surrogates. A neural network-based surrogate can be strongly improved by constraining it to the underlying physics using an approach known as Physics Informed Neural Networks (PINNs), and I end the chapter with a short introduction to Normalizing Flows (NFs). Before all of this though, I want to discuss *why* surrogates are needed in model discovery.

3.1 The need for surrogates

Using the sparse regression approach to discover PDEs requires calculating various higher-order derivatives. For example, a diffusive term is given by a second order derivative, $\nabla^2 u$, while the Kuramoto-Shivashinksy equation requires a fourth order derivative. Consider the definition of the derivative,

$$\frac{du}{dx} = \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}. \tag{3.1}$$

[1]also known as digital twins [Rasheed et al., 2020]

Since data is always sampled at a finite spacing, in practice a derivative must always be approximated,

$$\frac{du}{dx} \approx \frac{u(x+h) - u(x)}{h}. \tag{3.2}$$

This equation forms the basis of *numerical differentiation.* In what follows we assume the data has been sampled on a grid with spacing Δx, allowing the use of a more natural notation in terms of grid index i. Equation (3.2) takes into account only the 'following' sample u_{i+1} at position $+h$ and is hence known as a forward-difference scheme. A more accurate estimate can be obtained by taking into account both the previous sample u_{i-1} and following sample u_{i+1}, yielding the popular second-order central difference scheme [2],

$$\frac{du_i}{dx} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}. \tag{3.3}$$

Applying eq. (3.3) recursively yields higher order derivatives,

$$\frac{d^2 u_i}{dx^2} \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \tag{3.4}$$

$$\frac{d^3 u_i}{dx^3} \approx \frac{-u_{i-2} + 2u_{i-1} - 2u_{i+1} + u_{i+2}}{2\Delta x^3} \tag{3.5}$$

Numerical differentiation through finite differences has various attractive properties: it's easy to implement and computationally cheap. However, these advantages come with large drawbacks:

- The approximation is only valid for small Δx. If the samples are spaced far apart, the resulting derivative will be inaccurate. Higher order schemes can alleviate this only slightly.
- Higher order derivatives are calculated by successively applying the approximation (see (3.5)). Consequently, small errors in lower orders get propagated and amplified to higher orders, making estimates of these higher-order derivatives highly inaccurate.
- At the edges of the dataset (e.g. u_0), the derivatives cannot be calculated. Doing so would require extrapolating, which is highly inaccurate, so typically the data at the edge is simply discarded. Since higher order derivatives utilize large stencils, this can amount to a significant part of the data when working with small datasets.

[2]Actually, we can exactly calculate the derivative, but the trade-off is that we need to approximate the data using some other model - a surrogate!

However, the dominant issue is numerical differentiation's inability to deal with noisy data. Even low noise levels of i.i.d. white noise (<10%) can corrupt the features to such an amount that model discovery becomes impossible. We illustrate these issues in figure 3.1, where we show with a toy function $f(x) = cos(x^2)$ the effect of data sparsity and 10% applied noise on the first and third derivative. Note that while the effect on the first derivative is still relatively limited, the third derivative is strongly corrupted. For the sparse data we can still qualitatively observe the behaviour, albeit with much lower peaks, but the noise has corrupted the data almost completely.

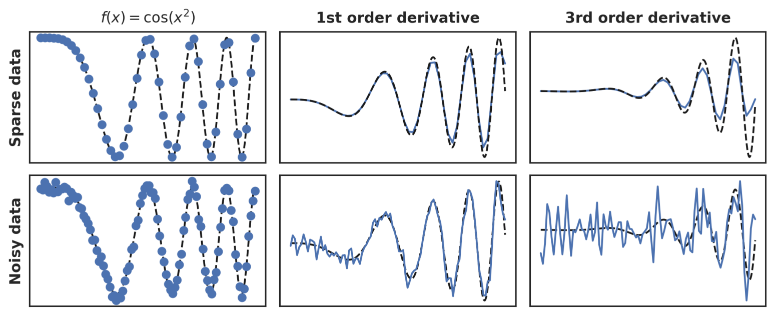

Figure 3.1: Numerical differentiation of noisy and sparse data. Blue are the samples, black dashed line the ground truth.

Denoising the to-be-differentiated data and regularizing the differentiation process increases accuracy [Van Breugel et al., 2020], but remains inadequate for very noisy data. Luckily, surrogates offer a way out. By fitting an easily-differentiable model such as a polynomial to the data, the problem of numerical differentiation is circumvented; the derivative of a polynomial can be trivially calculated symbolically. This does not mean that calculating the features is easy; the difficulty of determining the derivatives is now displaced to the task of accurately fitting the surrogate model the noisy data. In the next section I discuss the surrogate models which are often used in model discovery.

3.2 Surrogates: local versus global

Model discovery requires surrogates to be easily differentiable, but also to be flexible. The surrogate model must be able to accurately model the data: given some data $u(x)$, the surrogate $p(x)$ ought to have enough degrees of freedom to approximate u reasonably, i.e. $p(x) \approx u(x)$. A classical approach is to represent $p(x)$ as a series expansion [3],

[3]Note that the surrogate model doesn't need to be interpretable - we're merely trying to represent and interpolate the data in such a a way that we can easily take derivatives.

$$p(x) = \sum_n a_n h_n(x) \tag{3.6}$$

where a_n are picked such that $p(x) \approx u(x)$. The derivatives are now easily (and accurately!) calculated - it's simply the sum over the derivatives of the basis functions:

$$\frac{du}{dx} \approx \frac{dp(x)}{dx} = \sum_i a_i \frac{dh_i(x)}{dx} \tag{3.7}$$

The flexibility of the surrogate strongly depends on the choice of basis functions. A choice well-known to physicists is to use the Fourier basis,

$$h_n(x) = e^{2\pi inx}. \tag{3.8}$$

Using Fourier series has several attractive properties: they're computationally efficient, well established, and are particularly useful for calculating derivatives. The fourier transform of a pth order derivative is [4],

$$\mathcal{F}\left(\frac{d^p f}{dx^p}\right) = (ik)^p \hat{f}(k), \tag{3.9}$$

which allows to efficiently calculate all derivatives [5]. Additionally, the Fourier representation allows natural denoising of the data by applying a low-pass filter. This approach has been successfully applied to model discovery [Schaeffer, 2017], but is limited by its issues with sharp transitions (Gibbs ringing) and its requirement for periodic boundary condition.

A different choice of basis would be polynomials, but these do no have enough expressivity to model most data. However, data can locally be approximated polynomially, and **spline interpolation** exploits this idea by locally fitting a polynomial in a sliding window, and ensuring continuity at the edges [6]. Various types of splines exist, depending on 1) the type of interpolating polynomial used, 2) the order of the interpolating polynomials and 3) the continuity conditions at the edges. They are computationally cheap, have well known properties, and can be used to smooth data through with a smoothness parameter. These properties make it a popular approach to calculate derivatives, but when data is extremely noisy, sparse or high-dimensional results are inaccurate

[4]Transforming the entire PDE to Fourier space can indeed get rid off all derivatives, but unfortunately non-linear terms correspond to convolutions in fourier space, making everything much more complicated.

[5]This is also known as the spectral method to calculating derivatives, and is often used when solving PDEs [Schaeffer, 2017]

[6]I'm seeing this as basically Taylor expanding around every sample, and making sure everything is continuous.

3.3 Neural networks as surrogates

The most basic multilayer perceptron (MLP) neural networks cconsist of a basic matrix multiplication with an applied non-linearity,

$$z = f(xW^T + B) = h(x) \tag{3.10}$$

where $x \in \mathcal{R}^{N\times M}$ is the input, $W \in \mathcal{R}^{K\times M}$ the kernel or weight matrix, $b \in \mathcal{R}^K$ the bias and f a non-linearity such as tanh. These layers are composed to increase expressive power, yielding a deep neural network,

$$z = h_n \circ h_{n-1} \dots \circ h_0(x) = g_\theta(x) \tag{3.11}$$

in the rest of this thesis, we refer to a neural network of arbitrary depth and width as $g_\theta(x)$, where θ are the networks weights and biases.

Neural networks are excellent function interpolators, making them very suitable for surrogates: they scale well to higher dimensions, are computationally efficient and are very flexible [7]. Given a dataset u, x_i, we typically train neural networks by minimizing the mean squared error,

$$\mathcal{L} = \sum_{i=1}^{N} |u_i - g_\theta(x_i)|^2 \tag{3.12}$$

A neural networks flexibility is also its weakness; it is very sensitive to overfitting. To counter this, the dataset is split into a test and train set and train until the MSE on the test set starts increasing. While this works well in practice for many problems, the approach so far is completely *data-driven*; the only information used to train the network are the samples in the dataset. We often have more information available about a dataset - for example, its underlying (differential) equation [8]. How can we include such knowledge in a neural network?

3.3.1 Physics Informed Neural Networks

One of the most popular ways to include physical knowledge in neural networks is using so-called Physics Informed Neural Networks (PINNs) [Raissi et al., 2017a,b]. Consider a (partial) differential equation

[7]In fact, neural networks are known to be universal function approximators, meaning that a network with a single hidden layer of infinite width can approximate any continuous function.

[8]but not the solution!

$$\frac{\partial u}{\partial t} = f(...) \tag{3.13}$$

and a neural network $\hat{u} = g_\theta(x,t)$. PINNs train a neural network by minimizing

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_i^N \left| \frac{\partial \hat{u}_i}{\partial t} - f(\hat{u}_i, ...), \right|^2 . \tag{3.14}$$

As $\mathcal{L} \to 0$, the output of the network $\hat{u}$ satisfies the given differential equation [9], and the network learns the solution of a differential equation, without explicitly solving it.

PINNs can also be used to perform parameter inference. Given a dataset $\{u, t, x\}_i$ and a differential equation with unknown coefficients w, PINNs minimize,

$$\mathcal{L}(\theta, w) = \frac{1}{N}\sum_i^N |u_i - \hat{u}_i|^2 + \frac{\lambda}{N}\sum_i^N \left| \frac{\partial \hat{u}_i}{\partial t} - f(w, \hat{u}_i, ...), \right|^2 . \tag{3.15}$$

Note that we minimize both the neural network parameters θ and the unknown parameters w. Here the first term ensures the network learns the data, while the second term ensures the output satisfies the given equation. PINNs have become a very popular method for both solving differential equations and for parameter inference, for various reasons:

- PINNs are very straightforward to implement, requiring no specialized architecture or advanced numerical approaches. The parametrization of the data or solution as a neural network makes PINNs a *meshless* method. Not having to create meshes is a second simplifying property.
- Automatic differentiation can be used to calculate the derivatives, yielding machine-precision derivatives.
- When used for parameter inference, PINNs essentially act as physically-consistent denoisers, making them particularly robust and useful when working with noisy and sparse data.

Notwithstanding their popularity and ease of use, PINNs are not without flaws - specifically, they can unpredictably yield a low accuracy solution, or even fail to converge at all [Sitzmann et al., 2020, Wang et al., 2020b,a] .

[9]Similar terms can be added for initial and boundary conditions, but we omit these here for clarity.

3.3.2 Extensions & alternatives

PINNS have been extended and modified along several lines of research. To boost performance on complex datasets, it has proven fruitful to decompose the domain and apply a PINN inside each subdomain, while matching the boundaries [Hu et al., 2021, Shukla et al., 2021]. A different line of research instead focuses on appropriately setting the strength of the regularization term, adapting ideas from multitask learning λ [Maddu et al., 2021, McClenny and Braga-Neto, 2020]. A final avenue worth mentioning is the inclusion of uncertainty, either through Bayesian neural nets [Yang et al., 2020] or using dropout [Zhang et al.].

Various other approaches have been proposed to include physical knowledge in neural networks. Champion et al. [2019b] use a variation on a PINN with auto-encoders to discover an underlying equation. Other works highlight the connection between numerical differentiation and convolutions and include the equation through this [Long et al., 2017].

A very promising set of approaches are differentiable ODE and PDE solvers [Chen et al., 2018, Rackauckas et al., 2020], and are also often referred to as differentiable physics [Ramsundar et al., 2021]. Here, instead of modelling the solution u by a neural network, we model the time-derivative u_t and apply a solver. The upside of this approach is that the solution can be guaranteed to satisfy the physics, contrary to the 'soft-constraint' or regularization approach of PINNs, but it does reintroduce all the nuances involved in solving PDEs - discretisation, stability issues and possible stiffness. Calculating the gradient involves solving the adjoint problem, which itself can also be unstable.

Kernel-based methods have also been used to incorporate physical knowledge - specifically Gaussian Processes [Särkkä, 2019, Atkinson, 2020]. GPs have build-in uncertainty estimation and can be made to respect a given equation by using it as a kernel function. Unfortunately, this requires linearisation of the equation as GPs cannot accommodate non-linear kernels.[10]

A final approach worth mentioning explores including more abstract principles. Cranmer et al. [2020] present an architecture which parametrizes the Lagrangian of a system, while Greydanus et al. [2019] present a Hamiltonian-preserving architecture.

[10] In my opinion kernel methods such as GPs have been vastly underused. Kernel methods typically perform well in small datasets and can be made non-local through the use of neural networks, so-called Deep GPs.

3.3.3 Normalizing Flows

We discussed already that neural networks are universal function approximators and can thus theoretically approximate any function, including probability distributions. However, for a function $p(x)$ to be a proper distribution we require

$$\int_{-\infty}^{+\infty} p(x) = 1, \tag{3.16}$$

which cannot be calculated when using a normal neural network to model $p(x)$; the integral is intractable as it runs from $+\infty$ to $-\infty$. While the integral can be approximated, this property is never guaranteed and will always be an approximation. Other approaches to density estimation rely for example on a mixture of Gaussians, which guarantee the normalization but are neither flexible enough to model any arbitrary density, but also too computationally expensive. Normalizing Flows [Rezende and Mohamed, 2015] have become the prime method for modelling densities, being flexible, computationally cheap, and guaranteed to yield a proper probability distribution.

NFs are based on the change-of-variable theorem for probability distributions,

$$p(x) = \pi(z)\left|\frac{dz}{dx}\right|, \quad x = f(z). \tag{3.17}$$

Essentially, this allows us to express the *unknown* distribution $p(x)$ to be expressed in terms of a *known* distribution $\pi(z)$ and a transformation $x = f(z)$. NFs can thus learn any distribution by learning the transformation f. The only requirement is for f to be a bijective function - in other words, it needs to one-to-one and invertible. The expressive power of NFs depends on the transformation f, but constructing arbitrary invertible functions is extremely challenging [11]. Normalizing flows work around this by constructing a series of simple transformations [12],

$$x = f_K \circ f_{K-1} \circ ... f_0(z) \tag{3.18}$$

To train a normalizing flow, we typically minimize the negative log likelihood over the samples,

$$\mathcal{L} = -\sum_{i=1}^{N} \log p(x). \tag{3.19}$$

[11]Invertible neural networks are an active field - ICML has a recurring workshop on them.

[12]If every single layer is invertible, then so will the whole flow.

Various types of flows f have been proposed, from the relatively simple planar flows of Rezende and Mohamed [2015] to more complicated ones such as the Sylvester normalizing flows [Berg et al., 2018] or monotonic neural networks [Wehenkel and Louppe, 2019]. Recent work has focused on flows on different geometries and satisfying certain equivariances.

Chapter 4

DeepMoD

In the previous two chapters I have introduced two, largely separate ideas: using sparse regression to perform model discovery of differential equations, and using surrogates, specifically PINNs, to accurately interpolate data, consistent with a given equation. The central premise of my work is that these two ideas can be combined, yielding a robust model discovery algorithm. Integrating sparse regression in PINNs is easily achieved. Considering again the loss of a PINN with *known* equation $u_t = Xw$,

$$\mathcal{L} = \sum_i \|u_i - g_\theta(x_i, t_i)\|_2^2 + \sum_i \|(u_t)_i - X_i w\|_2^2 . \tag{4.1}$$

we simply replace this equation by the candidate library Θ and the sparse vector ξ,

$$\mathcal{L} = \sum_i \|u_i - g_\theta(x_i, t_i)\|_2^2 + \sum_i \|(u_t)_i - \Theta_i \xi\|_2^2 . \tag{4.2}$$

Simply put, this results in a PINN which discovers its constraining equation during training [1]. This approach has several benefits:

- The neural network is a very flexible surrogate, and we can use automatic differentiation to calculate machine-precision derivatives, circumventing the numerical differentiating problem.

[1]In this interpretation model discovery almost comes about as a side-effect; we simply train a PINN with many terms and it simply decides itself which to use.

- By constraining the network to solutions of the constraining equation, it acts as a physically-consistent denoiser and interpolator. In other words, the data is denoised and interpolated with knowledge of some underlying equation - in contrast to for example splines, which are purely data-driven.
- The integration of the two steps of model discovery - calculating features and sparse regression - into a single, differentiable model starts a 'virtuous cycle'. Accurate features improve the model discovery task, which in turn more strongly constrains the network, leading to a better estimate of the data, finally yielding more accurate features.

Equation (4.2) only describes the core idea, and in a series of three articles we explore and refine this idea under the name DeepMoD - Deep Learning for Model Discovery. Additionally, we worked on two tangential projects: one compares splines with DeepMoD and studies the effect of sampling strategies, while the second introduces Conditional Normalizing Flows (CNFs) to infer time-dependent probabilities and densities. In this chapter, I shortly discuss the contributions and methodology. followed by the results of each paper.

The result sections are adapted from the corresponding paper.

4.1 DeepMoD: Deep Learning for Model discovery in noisy data

***Gert-Jan Both**, Subham Choudhury, Pierre Sens, Remy Kusters - PDF*

This paper lays out and introduces the ideas I've sketched above, forming the basis for the rest of our work. We propose to train a neural network with the loss function

$$\mathcal{L}(\theta, \xi) = \underbrace{\sum_i \|u_i - \hat{u}_i\|_2^2}_{\mathcal{L}_{\text{fit}}} + \underbrace{\sum_i \|(\hat{u}_t)_i - \Theta_i \xi\|_2^2}_{\mathcal{L}_{\text{reg}}} + \underbrace{\lambda \sum_j |\xi_j|}_{\mathcal{L}_{\text{sparse}}} \tag{4.3}$$

where $\hat{u}_i = g_\theta(x_i, t_i)$ is the output of the neural network, and the features $\hat{u}_t$ and Θ are calculated from the output of the neural network. The first term in equation eq. (4.3) ensures the network learns the data, the second term constrains it to the given equation and the third term promotes the sparsity of ξ by applying an ℓ_1 penalty. Training the network involves both optimizing the neural network parameters θ and the sparse coefficient vector ξ. After convergence, the found ξ is normalized and thresholded to yield the final equation (see figure 4.1 for a schematic overview).

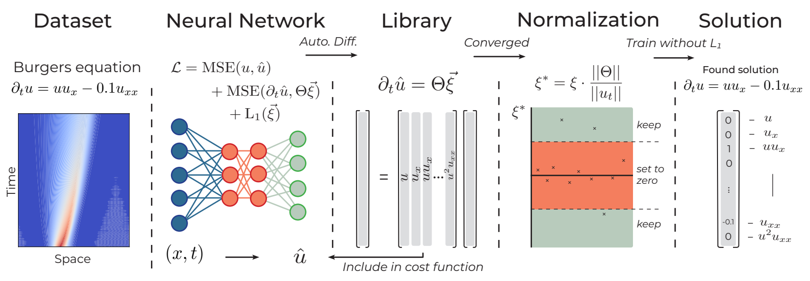
Dataset
Neural Network
Library
Normalization
Solution
Auto. Diff.
Converged
Train without L₁
Burgers equation
$\partial_t u = uu_x - 0.1u_{xx}$
Time
Space
$\mathcal{L} = \mathrm{MSE}(u, \hat{u}) + \mathrm{MSE}(\partial_t \hat{u}, \Theta\vec{\xi}) + \mathrm{L}_1(\vec{\xi})$
(x,t)
$\hat{u}$
$\partial_t \hat{u} = \Theta\vec{\xi}$
u u_x uu_x … u^2u_{xx}
Include in cost function
$\xi^* = \xi \cdot \frac{||\Theta||}{||u_t||}$
ξ^*
keep
set to zero
keep
Found solution
$\partial_t u = uu_x - 0.1u_{xx}$
0 – u
0 – u_x
1 – uu_x
0
-0.1 – u_{xx}
0 – u^2u_{xx}

Figure 4.1: Schematic representation of the first DeepMoD iteration.

4.1.1 Results

We test the performance of this approach, which we call DeepMoD, on a set of case studies: the Burgers' equation with and without shock, the 2D advection-diffusion equation and the Keller-Segel model for chemotaxis. These examples show the ability of DeepMoD to handle (1) non-linear equations, (2) solutions containing a shock wave, (3) coupled PDEs and finally (4) higher dimensional and experimental data.

4.1.1.1 Non-linear PDEs

The Burgers equation occurs in various areas of gas dynamics, fluid mechanics and applied mathematics and is evoked as a prime example to benchmark model discovery [Rudy et al., 2017, Long et al., 2017] and coefficient inference algorithms [Raissi et al., 2017a,b], as it contains a non-linear term as well as second order spatial derivative,

$$\partial_t u = -uu_x + \nu u_{xx}. \tag{4.4}$$

Here ν is the viscosity of the fluid and u its velocity field. We use the dataset produced by Rudy et al. [Rudy et al., 2017], where $\nu = 0.1$. The numerical simulations for the synthetic data were performed on a dense grid for numerical accuracy. DeepMoD requires significantly less datapoints than this grid and we hence construct a smaller dataset for DeepMoD by randomly sampling the results through space and time. From now on, we will refer to randomly sampling from this dense grid simply as sampling. Also note that this shows that our method does not require the data to be regularly spaced or stationary in time. For the data in Fig. 4.2 we add 10% white noise and sampled 2000 points for DeepMoD to be trained on.

We train the neural network using an Adam optimizer and plot the different contributions of the cost function as a function of the training epoch in Fig.

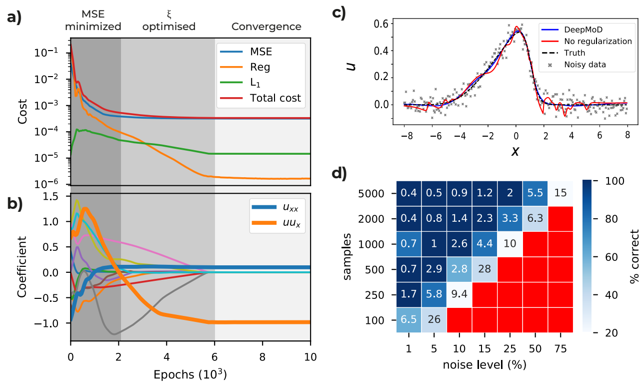

Figure 4.2: a) Cost functions and b) coefficient values as function of the number of epochs for the Burgers' dataset consisting of 2000 points and 10% white noise. Initially, the neural network optimizes the MSE and only after the MSE is converged the coefficient vector is optimized by the network. c) Velocity field u for $t = 5$ obtained after training with (no overfitting) and without (overfitting) the regression regularization $\mathcal{L}_{Reg}$. d) The values in the grid indicate the accuracy of the algorithm tested on the Burgers' equation, defined as the mean relative error over the coefficients, as function of the sample size of the data set and level of noise. The coloring represents the fraction of correct runs (Red indicates that in none of the five iterations the correct PDE is discovered).

4.2a and we show the value of each component of ξ as a function of the training epoch in Fig. 4.2b. Note that for this example, after approximately 2000 epochs, the MSE is converged, while at the same time we observe the components of ξ only start to converge after this point. We can thus identify three 'regimes': in the initial regime (0 - 2000 epochs), the MSE is trained. Since the output of the neural network is far from the real solution, so is Θ, and the regression task cannot converge (See first 2000 epochs in Fig. 4.2b). After the MSE has converged, $\hat{u}$ is sufficiently accurate to perform the regression task and ξ starts to converge. After this second regime (2000 - 6000 epochs), all components of the cost converged (>6000 epochs) and we can determine the solution. From this, we obtain a reconstructed solution (see Fig. 4.2b) and at the same time recover the underlying PDE, with coefficients as little as 1% error in the obtained coefficients. We show the impact of including the regression term in the cost function in Fig.4.2c, where the obtained solution of DeepMoD is compared with a neural network, solely trained on the MSE, to reconstruct the data. Including the regression term regularizes the network and prevents overfitting, despite the many terms in the library. We conclude that it is the inclusion of the regression in the neural network which makes DeepMoD robust to noisy data and prevents overfitting.

Next, we characterize the robustness of DeepMoD in Fig. 4.2d, where we run DeepMod for five times (differently sampled data set) for a range of sample sizes and noise levels. The color in Fig. 4.2d shows how many of the five runs return the correct equation and the value in the grid displays the mean error over all correct runs. Observe that at vanishing noise levels, we recover the Burgers equation with as little as 100 data-points, while for 5000 data points we recover the PDE with noise levels of up to 75%. Between the domain where we recover the correct equation for all five runs and the domain where we do not recover a single correct equation, we observe an intermediate domain where only a fraction of the runs return the correct equation, indicating the importance of sampling.

To benchmark DeepMoD, we can directly compare the performance of our algorithm with respect to two state-of-the-art methods, (i): PDE-Find by Rudy et al. [Rudy et al., 2017] and (ii) PDE-Stride by Maddu et al. [Maddu et al., 2019]. Considering an identical Burgers' data set with 10^5 data points, approach (i) recovers the correct equation for up to 1% Gaussian noise [Rudy et al., 2017] while method (ii) discovers the correct equation up to 5% noise [Maddu et al., 2019]. Compared to the results in Fig. 4.2d we note that even for two order of magnitude fewer samples points, 10^3 w.r.t. 10^5, DeepMoD recovers the correct equation up to noise levels $> 50\%$ Gaussian noise. DeepMoD allows up to two orders of magnitude higher noise-levels and smaller sample sizes with respect to state-of-the-art model discovery algorithms.

4.1.1.2 Shock wave solutions

If the viscosity is too low, the Burgers' equation develops a discontinuity called a shock (See Fig. 4.3). Shocks are numerically hard to handle due to divergences in the numerical derivatives. Since DeepMoD uses automatic differentiation we circumvent this issue. We adapt the data from Raissi et al. [Raissi et al., 2017b] which has $\nu = 0.01/\pi$, sampling 2000 points and adding 10% white noise (See Fig. 4.3). We recover ground truth solution of the Burgers' equation as well as the corresponding PDE,

$$\partial_t u = -0.99 u u_x + 0.0035 u_{xx}, \tag{4.5}$$

with a relative error of 5% on the coefficients. In Fig.4.3 we show the inferred solution for $t = 0.8$.

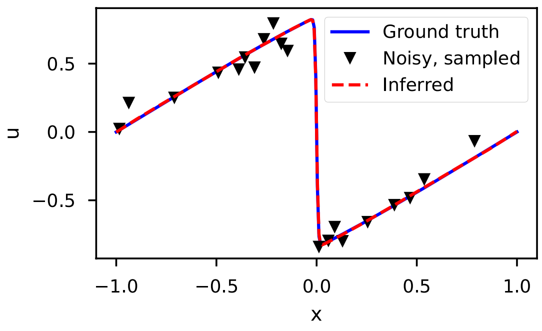

Figure 4.3: Ground truth, Noisy and Inferred data at $t = 0.8$ for the Burgers' equation with a shock wave (10% noise and 2000 sample points).

4.1.1.3 Coupled differential equations

Next, we apply DeepMod to a set of coupled PDE's in the form of the Keller-Segel (KS) equations, a classical model for chemotaxis [Painter, 2018]. Chemotactic attraction is one of the leading mechanisms that accounts for the morphogenesis and self-organization of biological systems. The KS model describes the evolution of the density of cells u and the secreted chemical w,

$$\begin{aligned} \partial_t u &= \nabla \cdot (D_u \nabla u - \chi u \nabla w) \\ \partial_t w &= D_w \Delta w - kw + hu. \end{aligned} \tag{4.6}$$

Here the first equation represents the drift-diffusion equation with a diffusion coefficient of the cells, D_u and a chemotactic sensitivity χ, which is a measure for the strength of their sensitivity to the gradient of the secreted chemical w. The second equation represents the reaction diffusion equation of the secreted chemical w, produced by the cells at a rate h and degraded with a rate k. For a 1D system, we sample 10000 points of u and w for parameter values of $D_u = 0.5$, $D_v = 0.5$, $\chi = 10.0$, $k = 0.05$ and $h = 0.1$ and add 5% white noise. We choose a library consisting of all spatial derivatives (including cross terms) as well as first order polynomial terms, totalling 36 terms. For these conditions we recover the correct set of PDEs,

$$\begin{aligned} \partial_t u &= 0.50u_{xx} - 9.99uw_{xx} - 10.02u_x w_x \\ \partial_t w &= 0.48w_{xx} - 0.049w + 0.098u, \end{aligned} \tag{4.7}$$

as well as the reconstructed fields for u and w (See Fig. 4.4). Note that even the coupled term, $u_x w_x$, which becomes vanishingly small over most of the domain, is correctly identified by the algorithm, even in the presence of considerable noise levels.

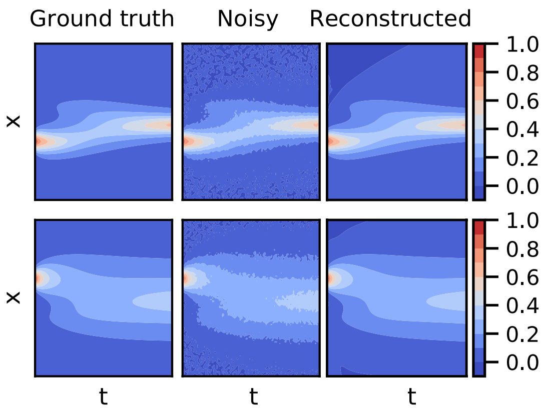

Figure 4.4: Ground truth, noisy and reconstructed solutions for the density of cells, u (top row) and the density of secreted chemicals w (bottom row) in the Keller Segel model for 5% white noise and 10000 samples.

4.1.1.4 Experimental data

To showcase the robustness of DeepMoD on high-dimensional and experimental input data, we consider a 2D advection diffusion process described by,

$$\partial_t u = -\nabla \cdot (-D\nabla u + \vec{v} \cdot u)\,, \tag{4.8}$$

where $\vec{v}$ is the velocity vector describing the advection and D is the diffusion coefficient. In the SI of the paper we apply DeepMod on a simulated data-set of eq. 4.8, with as initial condition, a 2D Gaussian with $D = 0.5$ and $\vec{v} = (0.25, 0.5)$. For as little as 5000 randomly sampled points we recover the correct form of the PDE as well as the vector $\vec{v}$ for noise levels up to $\approx 25\%$. In the absence of noise the correct equation is recovered with as little as 200 sample points through space and time. This number is surprisingly small considering this is an 2D equation.

Finally, we apply DeepMoD on a time-series of images from an electrophoresis experiment, tracking the advection-diffusion of a charged purple loading dye under the influence of a spatially uniform electric field. In Fig. 4.5a we show time-lapse images of the experimental setup where we measure the time-evolution of two initial localised purple dyes. Fig. 4.5b shows the resultant 2D density field for three separate time-frames (in arbitrary units), corresponding to the red square in Fig. 4.5a by substracting the reference image (no dye present). The dye displays a diffusive and advective motion with constant velocity v, which is related to the strength of the applied electric field.

We apply DeepMoD on 5000 sampled data-points sampled through space and time and consistently recover the advection term u_y as well as the two diffusive components (u_{xx} and u_{yy}). In Fig. 4.5c we show the scaled coefficients as function of the number of epochs. After thresholding the scaled coefficients ($|\xi| < 0.1$), we obtain for the unscaled coefficients, the resultant advection diffusion equation,

$$0.31u_y + 0.011u_{xx} + 0.009u_{yy} = 0. \tag{4.9}$$

Analysing the second diffusing dye (dashed box in Fig. 4.5) result in nearly identical values for the drift velocity, $v \approx 0.3$, and the diffusion coefficients, $D \approx 0.01$ indicating the robustness of the obtained value of D and v. In contrast to the artificial data presented in previous paragraphs, some higher-order non-linear terms, in particular uu_{yy} and uu_{xx} remain small, yet non-zero. This suggests that an automatic threshold strategy may not guarantee the desired sparse solution. Fixing a threshold of the scaled coefficients ($|\xi| < 0.1$ in this particular case) or other thresholding strategies such as coefficient cluster detection would be better suited for this task.

4.2 Sparsely constrained neural networks for model discovery of PDEs

Gert-Jan Both, *Gijs Vermarien, Remy Kusters - PDF*

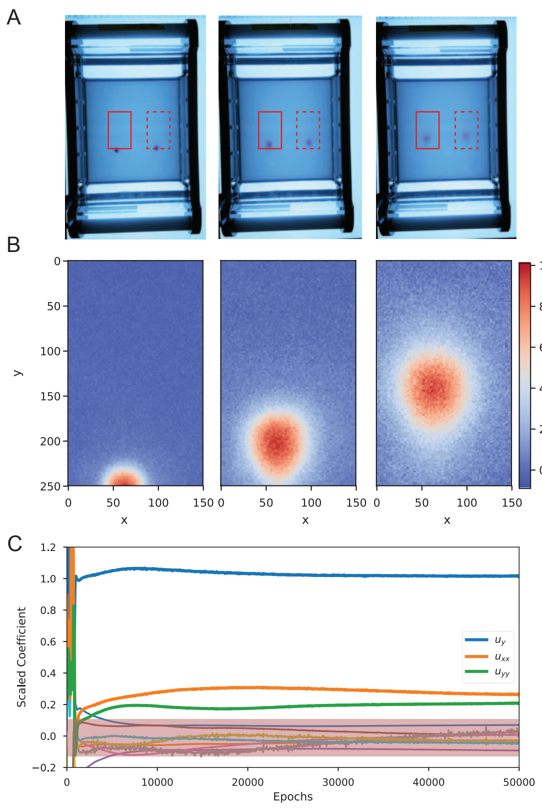

Figure 4.5: a) Time-series images of the electrophoresis essay. The two red boxes indicate the analysed region. b) Region indicated in the solid red box of (a) showing the density of the dye at three different time-frames (in pixels). c) Scaled coefficients values of all the candidate for a single run. The pink region indicates the terms with scaled coefficient $|\xi| < 0.1$.

This paper was conceived and written mostly during the first confinement in Paris - plenty of time to think and wonder!

The original DeepMoD outperformed other model discovery approaches, but left several questions unanswered. Specifically, we wondered:

- How can the optimal strength of ℓ_1 penalty, λ, be determined? Retraining the network would be prohibitively expensive, and while performance was not extremely sensitive to the choice of λ, it is a hyperparameter which much be correctly chosen.
- Lasso has several undesirable properties and perhaps suboptimal performance. Can other model discovery algorithms such as SINDy be integrated in this framework?
- We never remove terms from the equation; rather, they approach or become zero due to the ℓ_1 penalty. Can performance be improved further removing the zero components and iteratively refining the estimate, similar to the iterative regression technique?

Next to answering these questions, this paper also represented a deepening of our understanding of the DeepMoD framework. We realized the coefficient vector ξ essentially performs two tasks: 1) selecting which terms are active, i.e. the 'model discovery' task and 2) constraining the neural network. The main innovation in this paper is the introduction of a mask g defining the support of ξ, allowing us to separate these two tasks:

$$\mathcal{L}(\theta, \xi) = \sum_i \|u_i - \hat{u}_i\|_2^2 + \sum_i \|u_t - \Theta(g \cdot \xi)\|_2^2 . \tag{4.10}$$

This mask g is calculated non-differentiably, i.e. it effectively acts as an external 'forcing variable'. In ML applications, non-differentiability is typically considered a short-fall (see next section), but here it forms a bridge between neural-network based approaches and classical approaches. It is through the mask g that *any* sparse regression method can be utilized to perform variable selection on a neural-network based library. For example, lasso with cross validation can be used, but we also show an example with SINDy. We thus show that deep-learning based and classical methods can be synthesized, benefitting from both approaches. Together with this innovation we present a modular version of our framework, reprinted in figure 4.6.

In this framework, (I) A function approximator constructs a surrogate model of the data, (II) from which a library of possible terms and the time derivative is constructed using automatic differentiation. (III) A sparsity estimator constructs the sparsity mask g to select the active terms in the library using

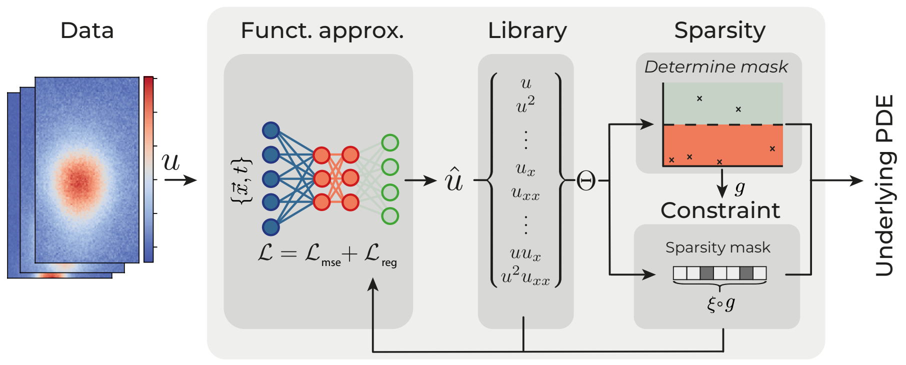

Figure 4.6: Schematic representation of the modular DeepMoD approach.

some sparse regression algorithm and (IV) a constraint constrains the function approximator to solutions allowed by the active terms obtained from the sparsity estimator.

We released this paper together with a pytorch-based framework, DeePyMoD. Overall the paper and the software package made our approach significantly more flexible: each module can be easily changed, independent of another.

4.2.1 Results

4.2.1.1 Constraint

The sparse coefficient vector ξ in eq. (4.10) is typically found by optimising it concurrently with the neural network parameters θ. Considering a network with parameter configuration θ^*, the problem of finding ξ can be rewritten as $\min_\xi |u_t(\theta^*) - \Theta(\theta^*)\xi|^2$. This can be analytically solved by least squares under mild assumptions; we calculate ξ by solving this problem every iteration, rather than optimizing it using gradient descent. In figure 4.7 we compare the two constraining strategies on a Burgers data-set, by training for 5000 epochs without updating the sparsity mask. Panel A) shows that the least-squares approach reaches a consistently lower loss. More strikingly, we show in panel B) that the mean absolute error in the coefficients is three orders of magnitude lower. We explain the difference as a consequence of the random initialisation of ξ: the network is initially constrained by incorrect coefficients, prolonging convergence. The random initialisation also causes the larger spread in results compared to the least squares method. The least squares method does not suffer from sensitivity to the initialisation and consistently converges.

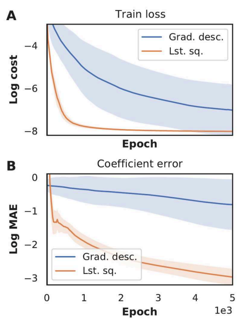

Figure 4.7: **A)** Loss and **B)** mean absolute error of the coefficients obtained with the gradient descent and the least squares constraint as a function of the number of epochs. Results have been averaged over twenty runs and shaded area denotes the standard deviation.

4.2.1.2 Sparsity estimator

Implementing the sparsity estimator separately from the neural network allows us to use any sparsity promoting algorithm. Here we show that a classical method for PDE model discovery, PDE-find [Rudy et al., 2017], can be used together with neural networks to perform model discovery in highly sparse and noisy data-sets. We compare it with the thresholded Lasso approach in figure 4.8 on a Burgers data-set with varying amounts of noise. The PDE-find estimator discovers the correct equation in the majority of cases, even with up to 60% - 80% noise, whereas the thresholded lasso mostly fails at 40%. We emphasise that the modular approach we propose here allows to combine classical and deep learning-based techniques. More advanced sparsity estimators such as SR3 [Champion et al., 2019c] can easily be included in this framework.

4.2.1.3 Function approximator

We show in figure 4.9 that a tanh-based NN fails to converge on a data-set of the Kuramoto-Shivashinksy (KS) equation (panel A and B). Consequently, the coefficient vectors are incorrect (Panel D). As our framework is agnostic to the underlying function approximator, we instead use a SIREN [Sitzmann et al., 2020], which is able to learn very sharp features in the underlying dynamics. In panel B we show that a SIREN is able to learn the complex dynamics of the KS equation and in panel C that it discovers the correct equation[2]. This example shows that the choice of function approximator can be a decisive factor in the success of neural network based model dis-

[2]In bold; uu_x: green, u_{xx}: blue and u_{xxxx}: orange

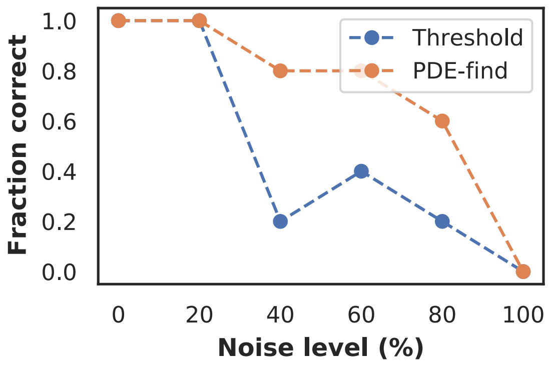

Figure 4.8: Fraction of correct discovered Burgers equations (averaged over 10 runs) as function of the noise level for the thresholded lasso and PDE-find sparsity estimator.

covery. Using our framework we can also explore using RNNs, Neural ODEs [Rackauckas et al., 2020] or Graph Neural Networks [Seo and Liu, 2019].

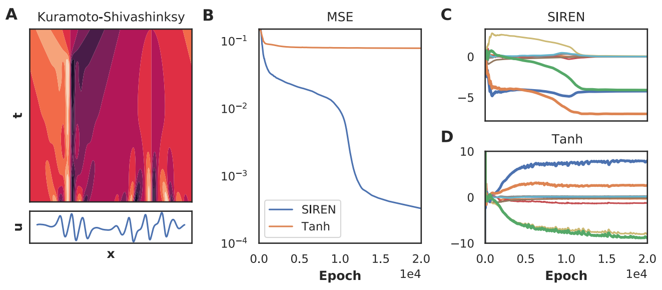

Figure 4.9: A) Solution of the KS equation. Lower panel shows the cross section at the last time point: $t = 44$. B) MSE as function of the number of epochs for both the tanh-based and SIREN NN. C) Coefficients as function of number of epochs for the SIREN and D) the tanh-based NN. The bold curves in panel C and D are the terms in the KS equation components; green: uu_x:, blue: u_{xx} and orange: u_{xxxx}. Only SIREN is able to discover the correct equation.

4.3 Fully differentiable model discovery

Gert-Jan Both, *Remy Kusters - PDF*

This is my favourite paper. I think it's an elegant approach solving many of the issues and questions of our previous papers, and as a bonus it also works in practice!

The modular approach in the second paper improved flexibility and performance, but left several things to be desired:

- It required the tuning of a fairly large set of hyperparameters: when to first update the mask, when to update it after, and which sparse regression approach to use (which itself introduces various hyperparameters).
- The non-differentiability of the mask allows to easily incorporate various sparse regression techniques, but also presents a weakness. If it is set incorrectly (for any reason), the training fails as the constraint is incompatible with the data. Additionally, due to this non-differentiability, model selection is effectively a side-effect of the training, and we hypothesize a fully differentiable method would increase performance and stability.

In this paper we create a fully-differentiable model discovery algorithm, which, considering eq. (4.10), requires making the mask g differentiable. Differentiable masking is challenging due to the binary nature of the problem, and instead *we relax the application of the mask to a regularisation problem.* Specifically, we propose to use Sparse Bayesian Learning [Tipping, 2001] to select the active features and act as constraint.

4.3.1 PINNs constrained by Sparse Bayesian Learning

Sparse Bayesian Learning (SBL) [Tipping, 2001] is a Bayesian approach to regression yielding sparse results. SBL defines a hierarchical model, starting with a Gaussian likelihood with noise precision $\beta \equiv \sigma^{-2}$, and a zero-mean Gaussian with precision α_j on each component ξ_j as prior,

$$p(\partial_t \hat{u};\ \Theta, \xi, \beta) = \prod_{i=1}^{N} \mathcal{N}(\partial_t \hat{u}_i;\ \Theta_i \xi, \beta^{-1}), \tag{4.11}$$

$$p(\xi;\ A) = \prod_{j=1}^{M} \mathcal{N}(\xi_j;\ 0, \alpha_j^{-1}), \tag{4.12}$$

with $\partial_t \hat{u} \in \mathcal{R}^N$, $\Theta \in \mathcal{R}^{N\times M}$, $\xi \in \mathcal{R}^M$, and we have defined $A \equiv \text{diag}(\alpha)$. The posterior distribution of ξ is a Gaussian with mean μ and covariance Σ,

$$\begin{aligned} \Sigma &= (\beta\Theta^T\Theta + A)^{-1} \\ \mu &= \beta\Sigma\Theta^T\partial_t\hat{u}. \end{aligned} \tag{4.13}$$

Many of the terms in A will go to infinity when optimised, and correspondingly the prior for term j becomes a delta peak. We are thus certain that

that specific term is inactive and can be pruned from the model. This makes SBL a very suitable choice for model discovery, as it gives a rigorous criterion for deciding whether a term is active or not. Additionally it defines hyper-priors over α and β,

$$\begin{aligned} p(\alpha) &= \prod_{j=1}^{M} \Gamma(\alpha_j;\ a, b) \\ p(\beta) &= \Gamma(\beta;\ c, d) \end{aligned} \tag{4.14}$$

The inference of A and β cannot be performed exactly, and SBL uses type-II maximum likelihood to find the most likely values of $\hat{A}$ and $\hat{\beta}$ by minimising the negative log marginal likelihood [3]

$$\mathcal{L}_{\text{SBL}}(A, \beta) = \frac{1}{2}\left[\beta \left\| u_t - \Theta\mu \right\|^2 + \mu^T A \mu - \log|\Sigma| - \log|A| - N \log\beta\right] - \sum_{j=1}^{M} (a \log\alpha_j - b\alpha_j) - c\log\beta + d\beta, \tag{4.16}$$

using an iterative method (see Tipping [2001]). The marginal likelihood also offers insight how the SBL provides differentiable masking. Considering only the first two terms of eq. (4.16),

$$\beta \left\| u_t - \Theta\mu \right\|^2 + \mu^T A \mu \tag{4.17}$$

we note that the SBL essentially applies a coefficient-specific ℓ_2 penalty to the posterior mean μ. If $A_j \to \infty$, the corresponding coefficient $\mu_j \to 0$, pruning the variable from the model. Effectively, the SBL replaces the discrete mask $m_j \in \{0, 1\}$ by a continuous regularisation $A_j \in (0, \infty]$, and we thus refer to our approach as *continuous relaxation*.

4.3.1.1 Model

To integrate SBL as a constraint in PINNs, we place a Gaussian likelihood on the output of the neural network,

$$\hat{u} : p(u;\ \hat{u}, \tau) = \prod_{i=1}^{N} \mathcal{N}(u_i;\ \hat{u}_i,\ \tau^{-1}), \tag{4.18}$$

and define a Gamma hyper prior on τ, $p(\tau) = \Gamma(\tau;\ e, f)$, yielding the loss function,

[3]Neglecting the hyper-prior, this loss function can also written more compactly as

$$\mathcal{L}_{\text{SBL}}(A, \beta) = \log|C| + \partial_t u^T C^{-1} \partial_t u, \quad C = \beta^{-1} I + \Theta A^{-1} \Theta^T, \tag{4.15}$$

but the format we use provides more insight how SBL provides differentiable masking.

$$\mathcal{L}_{\text{data}}(\theta, \tau) = \frac{1}{2}\left[\tau \left\|u - \hat{u}\right\|^2 - N\log\tau\right] - e\log\tau + f\tau. \tag{4.19}$$

Assuming the likelihoods factorise, i.e. $p(u, u_t;\ \hat{u},\ \Theta,\ \xi) = p(u;\ \hat{u}) \cdot p(u_t; \Theta,\ \xi)$, SBL can be integrated as a constraint in a PINN by simply adding the two losses given by eq. (4.16) and eq. (4.19),

$$\mathcal{L}_{\text{SBL-PINN}}(\theta, A, \tau, \beta) = \mathcal{L}_{\text{data}}(\theta, \tau) + \mathcal{L}_{\text{SBL}}(\theta, A, \beta) \tag{4.20}$$

Our approach does not rely on any specific property of the SBL, and thus generalises to other Bayesian regression approaches.

4.3.1.2 Training

The loss function for the SBL-constrained PINN contains three variables which can be exactly minimised, and denote these as $\hat{A}, \hat{\tau}$ and $\hat{\beta}$. With these values, we introduce $\tilde{\mathcal{L}}_{\text{SBL-PINN}}(\theta) \equiv \mathcal{L}_{\text{SBL-PINN}}(\theta, \hat{A}, \hat{\tau}, \hat{\beta})$ and note that we can further simplify this expression as the gradient of the loss with respect to these variables is zero. For example, $\nabla_\theta \mathcal{L}(\hat{A}) = \nabla_A \mathcal{L} \cdot \nabla_\theta A|_{A=\hat{A}} = 0$, as $\nabla_A \mathcal{L}|_{A=\hat{A}} = 0$. Thus, keeping only terms directly depending on the neural network parameters θ yields,

$$\begin{aligned}\tilde{\mathcal{L}}_{\text{SBL-PINN}}(\theta) &= \frac{\hat{\tau}}{2}\left\|u - \hat{u}\right\|^2 + \frac{\hat{\beta}}{2}\left\|u_t - \Theta\mu\right\|^2 + \mu^T \hat{A}\mu - \log|\Sigma| \\ &= \frac{N\hat{\tau}}{2}\underbrace{\left[\mathcal{L}_{\text{fit}}(\theta) + \frac{\hat{\beta}}{\hat{\tau}}\mathcal{L}_{\text{reg}}(\theta, \mu)\right]}_{=\mathcal{L}_{\text{PINN}}(\theta,\mu)} + \mu^T \hat{A}\mu - \log|\Sigma|\end{aligned} \tag{4.21}$$

where in the second line we have rewritten the loss function in terms of a classical PINN with relative regularisation strength $\lambda = \hat{\beta}/\hat{\tau}$ and coefficients $\xi = \mu$. Contrary to a PINN however, the regularisation strength is inferred from the data, and the coefficients μ are inherently sparse.

An additional consequence of $\nabla_\theta \mathcal{L}(\hat{A}, \hat{\beta}, \hat{\tau}) = 0$ is that our method does not require backpropagating through the solver. While such an operation could be efficiently performed using implicit differentiation [Bai et al., 2019], our method requires solving an iterative problem only in the forward pass. During the backwards pass the values obtained during the forward pass can be considered constant.

Considering eq. (4.21), we note the resemblance to multitask learning using uncertainty, introduced by Cipolla et al. [2018]. Given a set of objectives, the authors propose placing a Gaussian likelihood on each objective so that each task gets weighed by its uncertainty. The similarity implies that we are essentially reinterpreting PINNs as Bayesian or hierarchical multi-task models.

4.3.2 Physics Informed Normalizing Flows

Having redefined the PINN loss function in terms of likelihoods (i.e. eq. (4.21)) allows to introduce a PINN-type constraint to any architecture with a probabilistic loss function. In this section we introduce an approach with normalizing flows, called Physics Informed Normalizing Flows (PINFs). As most physical equations involve time, we first shortly discuss how to construct a time-dependent normalizing flow. We show in the experiments section how PINFs can be used to directly infer a density model from single particle observations.

4.3.2.1 Conditional Normalizing Flows

Normalizing flows construct arbitrary probability distributions by applying a series of K invertible transformations f to a known probability distribution $\pi(z)$,

$$\begin{aligned} z &= f_K \circ ... \circ f_0(x) \equiv g_\theta(x) \\ \log p(x) &= \log \pi(z) + \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k(z)}{\partial dz} \right|, \end{aligned} \tag{4.22}$$

and are trained by minimising the negative log likelihood, $\mathcal{L}_{\mathrm{NF}} = -\sum_{i=1}^{N} \log p(x)$. Most physical processes yield time-dependent densities, meaning that the spatial axis is a proper probability distribution with $\int p(x,t)dx = 1$. Contrarily, this is not valid along the temporal axis, as $\int p(x,t)dt = f(x)$. To construct PINFs, we first require a Conditional Normalizing Flow capable of modelling such time-dependent densities. Instead of following the method of Both and Kusters [2019], which modifies the Jacobian, we employ time-dependent hyper-network. This hyper-network h outputs the flow parameters θ, and is only dependent on time, i.e. $\theta = h(t)$, thus defining a time-dependent normalizing flow as $z = g_{h(t)}(x)$.

4.3.2.2 PINFs

Conditional normalizing flows yield a continuous spatio-temporal density, and the loss function of a PINF is defined as simply adding the SBL-loss to that of the normalizing flow, yielding

$$\tilde{\mathcal{L}}_{\mathrm{PINF}}(\theta) = \mathcal{L}_{\mathrm{NF}}(\theta) + \frac{N\hat{\beta}}{2}\mathcal{L}_{\mathrm{reg}}(\theta,\mu) + \mu^T\hat{A}\mu - \log|\Sigma|. \tag{4.23}$$

4.3.3 Results

We now show several experiments illustrating our approach. We start this section by discussing choice of hyperprior, followed by a benchmark on several datasets and finally a proof-of-concept with physics-informed normalizing flows.

4.3.3.1 Choosing prior

The loss function for the SBL constrained approach contains several hyperparameters, all defining the (hyper-) priors on respectively A, β and τ. We set uninformed priors on A and β, $a = b = e = f = 1e^{-6}$, but those on β, the precision of the constraint, must be chosen more carefully. Figure 4.10 illustrates the learning dynamics on a dataset of the Korteweg-de Vries equation [4] when the β hyperprior is uninformed, i.e. $c = d = 1e^{-6}$. Observe that the model fails to learn the data, while almost immediately optimising the constraint. We explain this behaviour as a consequence of our assumption that the likelihoods factorise, which implies the two tasks of learning the data and applying the constraint are independent. Since the constraint contains much more terms than required, it can fit a model with high precision to any output the neural network produces. The two tasks then are not independent but conditional: a high precision on the constraint is warranted only if the data is reasonably approximated by the neural network. To escape the local minimum observed in figure 4.10, we couple the two tasks by making the hyper-prior on β dependent on the performance of the fitting task.

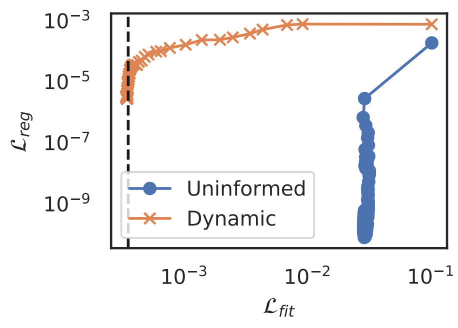

Figure 4.10: Regression loss as a function of fitting loss during training, comparing an uninformed prior with a dynamic prior.

Our starting point is the update equation for β (see Tipping [2001] for

[4]We choose to plot the losses of the original PINN loss $\mathcal{L}_{\text{data}}$ and $\mathcal{L}_{\text{reg}}$ because these are more easily interpreted than the likelihood-based losses we have introduced.

details),

$$\hat{\beta} = \frac{N - M + \sum_i \alpha_i \Sigma_{ii} + 2c}{N\mathcal{L}_{\text{reg}} + 2d} \tag{4.24}$$

We typically observe good convergence of normal PINNs with $\lambda = 1$, and following this implies $\hat{\beta} \approx \hat{\tau}$, and similarly $\mathcal{L}_{\text{reg}} \to 0$ as the model converges. Assuming $N \gg M + \sum_i \alpha_i \Sigma_{ii}$, we have

$$\hat{\tau} \approx \frac{N + 2c}{2d}, \tag{4.25}$$

which can be satisfied with $c = N/2, \; d = n/\hat{\tau}$. Figure 4.10 shows that with this dynamic prior the SBL constrained PINN does not get trapped in a local minimum and learns the underlying data. We hope to exploit multitask learning techniques to optimize this choice in future work.

4.3.3.2 Experiments

We present three experiments to benchmark our approach. We first study the learning dynamics in-depth on a solution of the Korteweg- de Vries equation, followed by a robustness study of the Burgers equation, and finally show the ability to discover the chaotic Kuramoto-Shivashinsky equation from highly noisy data. Reproducibility details can be found in the appendix of the paper.

4.3.3.2.1 Korteweg-de Vries The Korteweg-de Vries equation describes waves in shallow water and is given by $u_t = u_{xxx} - uu_x$. Figure 4.11a shows the dataset: 2000 samples with 20% noise from a two-soliton solution. We compare our approach with I) Sparse Bayesian Learning with features calculated with numerical differentiation, II) a model discovery algorithm with PINNs, but non-differentiable variable selection called DeepMoD [Both and Kusters, 2021] and III) PDE-find [Rudy et al., 2017], a popular model discovery method for PDEs based on SINDy [Brunton et al., 2016]. The first two benchmarks also act as an ablation study: method I uses the same regression algorithm but does not use a neural network to interpolate, while method II uses a neural network to interpolate but does not implement differentiable variable selection.

In figure 4.11b and c we show that the differentiable approach recovers the correct equation after approximately 3000 epochs. Contrarily, DeepMoD recovers the wrong equation. Performing the inference 10 times with different seeds shows that the fully-differentiable approach manages to recover the Kortweg-de Vries equation nine times, while DeepMoD recovers the correct equation only twice - worse, it recovers the same wrong equation the other 8 times. Neither PDE-find nor SBL with numerical differentiation is able to

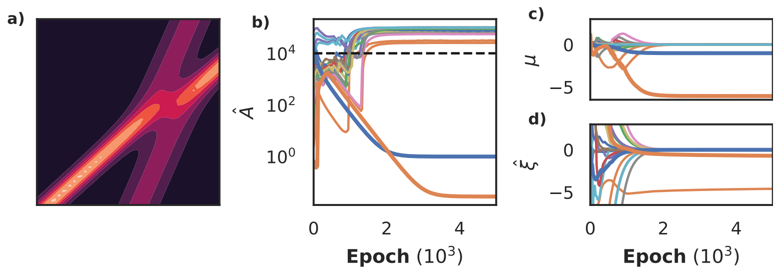

Figure 4.11: Comparison of a differentiable SBL-constrained model and an non-differentiable OLS-constrained model on a Korteweg-de Vries dataset (panel a) with a library consisting of 4th order derivatives and 3rd order polynomials, for a total of 20 candidate features. In panel b and c we respectively plot the inferred prior $\hat{A}$ and the posterior coefficients μ. In panel d we show the non-differentiable DeePyMod approach. In panels b and c we see that the correct equation (bold blue line: u_{xx}, bold orange line: uu_x) is discovered early on, while the non-differentiable model (panel d) selects the wrong terms.

discover the Korteweg-de Vries equation from this dataset, even at 0% noise due to the data sparsity.

4.3.3.2.2 Burgers We now explore how robust the SBL-constrained PINN is with respect to noise on a dataset of the Burgers equation, $u_t = \nu u_{xx} - uu_x$ (figure 4.12a). We add noise varying from 1% to 100% and compare the equation discovered by benchmark method II (DeepMoD, panel b) and our approach (panel c) - the bold orange and blue lines denote u_{xx} and uu_x respectively, and the black dashed line their true value. Observe that DeepMoD discovers small additional terms for >50% noise, which become significant when noise >80%. Contrarily, our fully differentiable approach discovers the same equation with nearly the same coefficients across the entire range of noise, with only very small additional terms ($\mathcal{O}(10^{-4})$). Neither PDE-find nor SBL with numerical differentiation is able to find the correct equation on this dataset at 10% noise or higher.

4.3.3.2.3 Kuramoto-Shivashinsky The Kuramoto-Shivashinksy equation describes flame propagation and is given by $u_t = -uu_x - u_{xx} - u_{xxxx}$. The fourth order derivative makes it challenging to learn with numerical differentiation-based methods, while its periodic and chaotic nature makes it challenging to learn with neural network based methods [Both et al., 2021b]. We show here that using the SBL-constrained approach we discover the KS-equation from only a small slice of the chaotic data (256 in space, 25 time steps), with 20% additive noise. We use a tanh-activated

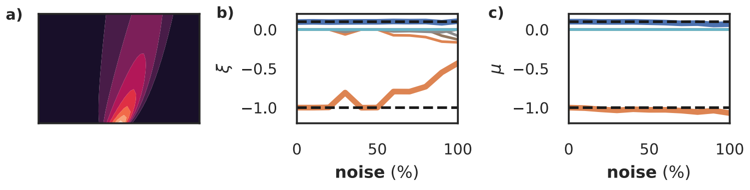

Figure 4.12: Exploration of robustness of SBL-constrained model for model discovery for the Burgers equation (panel a). We show the discovered equation over a range of noise for DeepMoD (panel b) and the approach presented in this paper (panel c). The bold orange and blue lines denotes $u_x x$ and uu_x, and black dashed line their true value.

network with 5 layers of 60 neurons each, and the library consists of derivatives up to 5th order and polynomials up to fourth order for a total of thirty terms. Additionally, we precondition the network by training without the constraint for 10k epochs.

Training this dataset to convergence takes significantly longer than previous examples, as the network struggles with the data's periodicity (panel b). After roughly 70k epochs, a clear separation between active and inactive terms is visible in panel c, but it takes another 30k epochs before all inactive terms are completely pruned from the model. Panels d and e show the corresponding posterior and the maximum likelihood estimate of the coefficients using the whole library. Remarkably, the MLE estimate recovers the correct coefficients for the active terms, while the inactive terms are all nearly zero. In other words, the accuracy of the approximation is so high, that least squares identifies the correct equation.

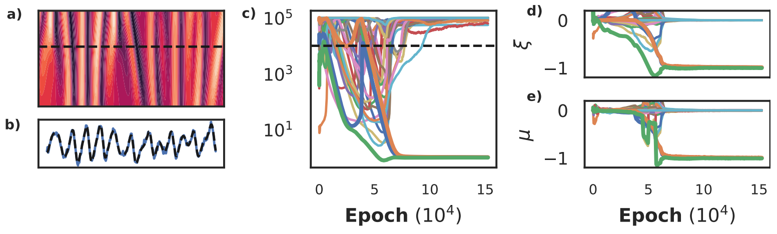

Figure 4.13: Recovering the Kuramoto-Shivashinsky equation. We show the chaotic data and a cross section in panels a and b. The periodicity makes this a challenging dataset to learn, requiring 200k iterations to fully converge before it can be recovered (panel c). Panels d and e show that the posterior and MLE of the coefficients yield nearly the same coefficients, indicating that the network was able construct an extremely accurate approximation of the data.

4.3.3.3 Model discovery with normalizing flows

Consider a set of particles whose movement is described by a micro-scale stochastic process. In the limit of many of such particles, such processes can often be described with a deterministic macro-scale density model, determining the evolution of the density of the particles over time. For example, a biased random walk can be mapped to an advection-diffusion equation. The macro-scale density models are typically more insightful than the corresponding microscopic model, but many (biological) experiments yield single-particle data, rather than densities. Discovering the underlying equation thus requires first reconstructing the density profile from the particles' locations. Classical approaches such as binning or kernel density estimation are either non-differentiable, non-continuous or computationally expensive. Normalizing Flows (NFs) have emerged in recent years as a flexible and powerful method of constructing probability distribution, which is similar to density estimation up to a multiplicative factor. In this section we use physics informed normalizing flows to learn a PDE describing the evolution of the density directly from unlabelled single particle data.

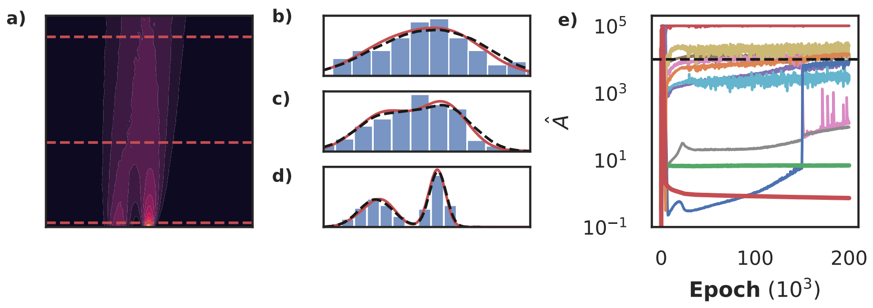

Figure 4.14: Using a tempo-spatial Normalizing Flow constrained by Sparse Bayesian Learning to discover the advection-diffusion equation directly from single particle data. Panel a shows the true density profile, and in panels b, c and d we show the density inferred by binning (blue bars), inferred by NF (red) and the ground truth (black, dashed) at $t = 0.1, 2.5, 4.5$. Note that although the estimate of the density is very good, we see in panel e that we recover two additional terms (bold blue line: u_x, bold orange line u_{xx}.

Since the conditional normalizing flow is used to construct the density, a precision denoting the noise level does not exist, and instead we set as prior for β $(a = N, b = N \cdot 10^{-5})$. We consider a flow consisting of ten planar transforms [Rezende and Mohamed, 2015] and a hyper-network of two layers with thirty neurons each. The dataset consists of 200 walkers on a biased random walk for 50 steps, corresponding to an advection-diffusion model, with an initial condition consisting of two Gaussians, leading to the density profile shown in figure 4.14a. The two smallest terms in panel e correspond to the advection (bold green line) and diffusion (bold red line) term, but not all terms are pruned. Panels b, c and compare the inferred density (red line) to the true density (dashed black line) and the result obtained by binning. In all three panels the constrained NF is able to infer a fairly accurate density from only 200 walkers. We hypothesise that the extra terms are mainly due to the small deviations, and that properly tuning the prior parameters and using a more expressive transformation would prune the remaining terms completely. Nonetheless, this shows that NF flows can be integrated in this fully differentiable model discovery framework.

4.4 Temporal Normalizing Flows

***Gert-Jan Both**, Remy Kusters. PDF*

This is another of my favourite papers. Although the execution leaves much to be desired, I believe the core idea is quite simple and elegant and can be of wide use.

In the biological sciences, single particle tracking (SPT) has become the method of choice to investigate many cellular processes. The obtained trajectories can be either analyzed as random walks, or from the perspective of walker densities. This density model is deterministic and typically more insightful than the stochastic random walk. However, in experiments the position of single particles, and from these positions a *time-dependent density* must be reconstructed. Classical methods such as binning or kernel density estimation are computationally expensive, non-differentiable [5], but above all unable to explicit model a time-dependent density. Rather, they construct snapshots of the density at given times. In this paper we introduce conditional normalizing flows (CNFs)[6], a flexible and differentiable method which models the entire continuous spatio-temporal density.

We construct a Conditional Normalizing Flow by constraining the Jacobian of the transformation. Consider a two dimensional normalizing flow in spatial coordinate x and time t:

$$p(x,t) = \pi(z,\tau)\left|\frac{dz}{dx}\frac{d\tau}{dt} - \frac{d\tau}{dx}\frac{dz}{dt}\right|, \quad [z,\tau] = f(x,t). \tag{4.26}$$

z and τ are thus some sort of latent coordinates. Since only the spatial axis represents a valid probability distribution, i.e. $\int p(x,t)dx = 1$, but

[5]meaning that the inferred density is a given and cannot be changed.

[6]When we wrote this paper, we only considered dependence on time - hence the name temporal normalizing flow. Calling them CNFs generalizes our approach and brings it in line with literature.

$\int p(x,t)dt = f(x)$, only x can be validly transformed to z. We enforce this by simply requiring $t = \tau$, i.e. the latent time τ is the real time t. The transformation then reduces to

$$p(x,t) = \pi(z,t)\left|\frac{dz(x,t)}{dx}\right|, \quad z = f(x,t) \tag{4.27}$$

i.e. the bijective transform depends on x and t, but the coordinate t is not transformed.

4.4.1 Results

We now demonstrate tNFs on three datasets:

- A **multi-scale toy problem** to show tNFs can accommodate different length scales in a single distribution;
- A dataset of **Brownian motion** to show how tNFs enhance density estimation for sparse datasets;
- A dataset of **chemotactic walkers** to show that tNFs can correctly estimate a multi-modal, non-Gaussian density.

4.4.1.1 Multi-scale density estimation

A key problem in density estimation is inferring an accurate distribution when vastly different length scales are present within a single dataset. Classical approaches such as binning and KDE require a single characteristic length scale, prohibiting an accurate estimate of a multi-scale distribution. We now show that normalizing flows, and by extension tNFs, are capable of accurately inferring such a distribution.

We build an artificial distribution consisting of three normal distributions with standard deviations σ of $0.01, 0.1$ and 1.0 (thus spanning three orders of magnitude) and respective weights $0.013, 0.13$ and 0.85. Figure 4.15 shows the inferred distribution from 5000 samples for the NF and the KDE with Scott's rule determining the lengthscale. Observe that, as expected, the KDE is unable to accommodate the different scales and that due the different weighting of each peak, the widest is dominating the lengthscale estimation. Contrarily, the NF provide an accurate density estimate for all lengthscales present in the problem, independent of their weights. We stress that tNF do not require any prior information of the different lengthscales present in the data-set.

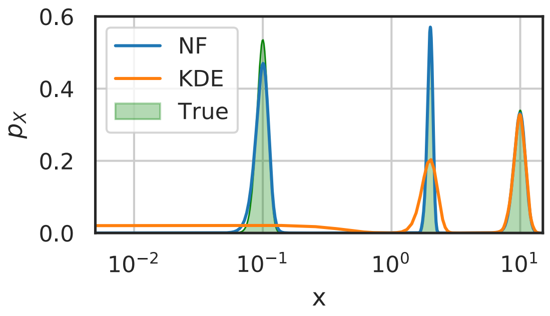

Figure 4.15: Comparison of density estimation with different scales. $N = 5000$ samples were taken from a density consisting of three Gaussians with widths 0.01, 0.1, and 1, at respective locations $0.1, 2$ and 10, and weights $0.013, 0.13$ and 0.85. The KDE used a Gaussian kernel with the kernel width set by Scott's rule.

4.4.1.2 Brownian motion

Brownian motion is the most basic and ubiquitous random walk and thus an ideal test case to assess the performance of tNFs, comparing them to time independent NFs and classical binning. We generate a single trajectory for a Brownian random walker by the recursive relation, $\vec{x}_{n+1} = \mathcal{N}(\vec{x}_n, \sqrt{2D\Delta t})$. Here n is the step number with x_0 the initial position, D the diffusive coefficient and Δt the time step. In the limit of an infinite number of walkers, the *walker density* c is described by the diffusion equation, $\partial_t c = D\nabla^2 c$.

Our dataset consists of $M = 500$ walkers with $D = 2.0$, with snapshots being taken every $\Delta t = 0.1$ for $N = 100$ frames. The initial positions were sampled from a Gaussian centered at $x = 1.5$ with width $\sigma = 0.5$; in this case, the diffusion equation can be solved exactly and the solution behaves as a spreading Gaussian in time. We show the estimated density at $t = 0.75$ and $t = 4.25$ in figure 4.16 (a) and (b) for the tNF, the time independent NF and binning. The tNF provides a significantly better density estimate than the time independent NF, illustrated by the difference in ℓ_2 error; $2.6 \cdot 10^{-5}$ for the tNF and $7.4 \cdot 10^{-5}$ for the NF, averaged over frame 15 and 85.

Normalizing flows are based on neural networks and hence prone to overfitting. We analyze the effect of overfitting in Appendix I of the paper and show that NFs overfit more strongly than tNFs and perform worse in terms of the ℓ_2 error. We mainly attribute this improvement to the temporal correlations in the dataset, which suppresses the natural frame-to-frame variations in the density estimate. Nonetheless, tNFs are not immune to overfitting and we speculate performance could be enhanced by applying techniques such as early stopping.

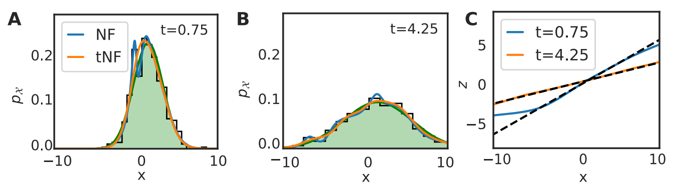

Figure 4.16: Results of inferring a Brownian walker density from $M = 500$ walkers with $D = 2$ and where a snapshot was taken every $\Delta t = 0.1$ for $N = 100$ frames. The initial position was sampled from a Gaussian placed at $x = 1.5$ with width $\sigma = 0.5$. Panel a and b compare the inferred density by binning, time independent normalizing flow (NF) and time dependent normalizing flow (tNF) at two different times. Panel c compares the learned mapping with the true mapping.

For the diffusion equation the true mapping can be trivially derived. We compare it to the learned mapping in figure 4.16c. It shows perfect agreement at $t = 4.25$, but deviates from the true curve for $x < -5$ at $t = 0.75$. As can be seen in figure 4.16a, no samples were present in this domain, explaining the deviance. Nonetheless, it implies that the network does not generalize well outside the sampling domain. We speculate that techniques such as batch normalization or a different architecture for the network (a recurrent network, for example) might further improve performance.

4.4.1.3 Chemotaxis

The Brownian motion presented in the previous section was a linear problem with a uni-modal, Gaussian solution. We now apply tNFs to so-called chemotactic walkers, a non-linear problem with a multi-modal solution. Bacteria and other micro-organisms sense gradients of chemicals throughout their environment and use this to guide their motion towards a food source. This effect is known as chemotaxis and is typically modelled by a random walker with a superimposed drift; $\vec{x}_{n+1} = \mathcal{N}(\vec{x}_n + \chi \nabla p(\vec{x}_n)dt, \sqrt{2Ddt})$, where p is the chemical density and χ is the *chemotactic sensitivity*, which controls the interaction between the chemical and the bacteria. In the infinite walker limit, the walker and chemical density are given by the Keller-Segel model: $\partial_t c = \nabla \cdot (D_c \nabla c - \chi c \nabla p)$ and $\partial_t p = D_p \nabla^2 p - Kp$. Here D_c and D_p are the diffusion coefficients of the bacteria and the chemical respectively and a decay set by K has been added to the chemical density.

Our dataset consisted of $M = 500$ walkers with $D = 0.5$ and we sampled the initial position from a Gaussian centred at $x = -2.5$. The food source was modelled by a Gaussian with diffusion coefficient $D = 0.25$, centred at $x = 2.5$; the walkers will thus drift towards food source over time. Figure 4.17 shows a comparison of the time independent NF, tNF and the binning

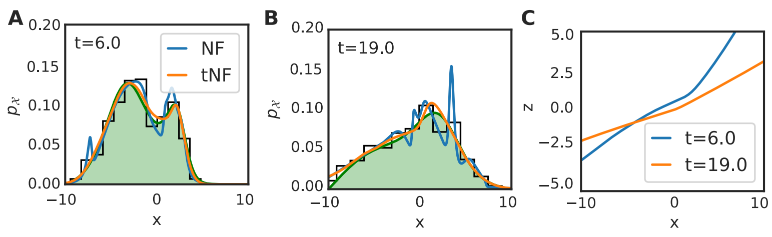

Figure 4.17: Results of inferring a chemotactic walker density from $M = 500$ walkers with $D = 0.5$ and chemotactic sensitivity $\chi = 10$, where a snapshot was taken every $\Delta t = 0.1$ for $N = 100$ frames. The initial position was sampled from a Gaussian placed at $x = -2.5$ with width $\sqrt{0.5}$, while the food source is a Gaussian located at $x = 2.5$ with width $\sqrt{0.5}$, diffusing with $D = 0.25$ and decaying at a rate of 0.05. Panel a and b compare the inferred density by binning, time independent normalizing flow (NF) and time dependent normalizing flow (tNF) at two different times. Panel c compares the learned mapping with the true mapping.

method. In figure 4.17(a) and (b) we find that the tNF leads to a significantly more accurate density estimation, illustrated by the difference in ℓ_2 error ($1.45 \cdot 10^{-4}$ for the NF versus $1.8 \cdot 10^{-5}$ for the tNF, averaged over $t = 6.0$ and 19.0). The tNF captures the multi-modal distribution at $t = 6.0$ excellently, without overfitting, contrarily to the time independent NF. The mapping, as shown in figure 4.17c, is non-linear, in contrast to the mapping obtained for the Brownian motion.

4.5 Model discovery in the sparse sampling regime

Gert-Jan Both, *Georges Tod, Remy Kusters. PDF*

This paper started with a curious observation: between our first and second paper we noticed a curious regression in performance. We eventually traced it back to a change in sampling strategy: when the data was randomly sampled instead of on a grid, DeepMoD was able to recover the underlying equation with much less data.

DeepMoD can handle much sparser data than other classical interpolation techniques. In this paper we compare our method in-depth to spline interpolation, how it fails as data becomes sparser and link it to the data's characteristic lengthscale.

4.5.1 Results

4.5.1.1 Synthetic data - Burgers equation

We consider a synthetic data set of the Burgers equation $u_t = \nu u_{xx} - u u_x$, with a delta peak initial condition $u(x, t = 0) = A\delta(x)$ and domain $t \in [0.1, 1.1], x \in [-3, 4]$. This problem can be solved analytically to yield a solution dependent on a dimensionless coordinate $z = x/\sqrt{4\nu t}$. We recognize the denominator as a *time-dependent* length scale: a Burgers data set sampled with spacing Δx thus has a time-dependent dimensionless spacing $\Delta z(t)$. We are interested in the smallest characteristic length scale, which for this data set is $l_c = \sqrt{4\nu t_0}$, where $t_0 = 0.1$ is the initial time of the data set. We set $A = 1$ and $\nu = 0.25$, giving $l_c = \sqrt{0.1} \approx 0.3$.

Splines do not scale effectively beyond a single dimension, making it hard to fairly compare across both the spatial and temporal dimensions. We thus study the effect of spacing only along the spatial axis and minimize the effect of discretization along the temporal axis by densely sampling 100 frames, i.e. $\Delta t = 0.01$. Along the spatial axis we vary the number of samples between 4 and 40, equivalent to $0.5 < \frac{\Delta x}{l_c} < 5$. We study the relative error ϵ as the sum of the relative errors for all the derivatives, normalized over every frame,

$$\epsilon = \sum_{i=1}^{3} \left\langle \frac{\left\|\partial_x^i u_j - \partial_x^i \hat{u}_j\right\|_2}{\left\|\partial_x^i u_j\right\|_2} \right\rangle_j \tag{4.28}$$

where i sums the derivatives and j runs over the frames. The derivatives are normalised every frame by the l_2 norm of the ground truth to ensure ϵ is independent of the magnitude of u. ϵ does not take into account the nature of noise (e.g. if it is correlated and non-gaussian), nor if the correct equation is discovered. However, taken together with a success metric (i.e if the right equation was discovered), it does serve as a useful proxy to the quality of the interpolation.

Figure 4.18b) shows ϵ as a function of the relative spacing $\Delta x/l_c$ and whether the correct equation was discovered. The error when using splines (yellow) increases with Δx and, as expected, we are unable to discover the correct equation for $\Delta x > 0.8 l_c$ (dots indicate the correct equation is discovered and triangles indicates it failed to do so). Considering the NN-based DeepMoD method, sampled on a grid (green), we observe that it is able to accurately interpolate and discover the correct equation up to $\Delta x \approx 1.2 l_c$. The reason for this is that NN-based interpolation constructs a surrogate of the data, informed by both the spatial and the temporal dynamics of the data set, while classical interpolation is intrinsically limited to a single time frame.

In figure 4.18c) we consider the same graph with 20% white noise on the data. Despite smoothing, the spline is unable to construct an accurate library and fails to discover the correct equation in every case. DeepMoD stands in stark contrast, discovering the correct equation with comparable relative error as in the 0% noise case.

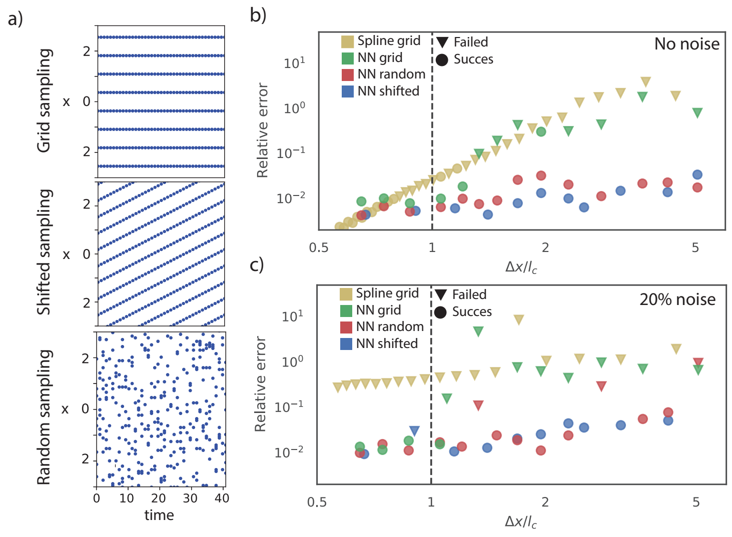

Figure 4.18: a) The three sampling strategies considered. b) and c) Error in the function library as function of the distance between the sensors Δx, normalized with $l_c = \sqrt{4\nu t_0}$, for b) noise-less data and c) 20% of additive noise. The yellow symbols correspond to a spline interpolation and the green, blue and red correspond to the NN-based model discovery with various sampling strategies. The circles indicate that model discovery was successful while the triangles indicate that the incorrect model was discovered. The horizontal dashed line indicates the smallest characteristic length-scale in the problem: $\Delta x/l_c = 1$.

4.5.1.1.1 Off-grid sampling Whereas higher-order splines are constrained to interpolating along a single dimension, DeepMoD uses a neural network to interpolate along both the spatial and temporal axis. This releases us from the constraint of on-grid sampling, and we exploit this by constructing an alternative sampling method. We observe that for a given number of samples n, DeepMoD is able to interpolate much more accurately if these samples are randomly drawn from the sampling domain. We show in figure 4.18b and c (Red) that the relative error ϵ in the sparse regime, can be as much as two orders of magnitude lower compared to the grid-sampled results at the same number of samples. We hypothesize that this is due to the spatio-temporal interpolation of the network. By interpolating along both axes, each sample effectively covers its surrounding area, and by randomly sampling we cover more of the spatial sampling domain. Contrarily, sampling on a grid leaves large areas uncovered; we are effectively sampling at a much lower resolution than when using random sampling.

To test whether or not this improvement is intrinsically linked to the randomness of sampling, we also construct an alternative sampling method called shifted-grid sampling. Given a sampling grid with sensor distance Δx, shifted-grid sampling translates this grid a distance Δa every frame, leading to an effective sample distance of $\Delta a \ll \Delta x$. This strategy, similarly as random sampling varies the sensor position over time, but does so in a deterministic and grid-based way. We show this sampling strategy in figure 4.18a, while panels b and c confirm our hypothesis; shifted grid sampling (Blue) performs similarly to random sampling. Shifted-grid sampling relies on a densely sampled temporal axis 'compensating' for the sparsely sampled spatial axes. This makes off-grid sampling beneficial when either the time or space axis, but not both, can be sampled with a high resolution. In the experimental section we show that if both axes are sparsely sampled, we do not see a strong improvement over grid sampling.

4.5.1.2 Experimental data - 2D Advection-Diffusion

In an electrophoresis experiment, a charged dye is pipetted in a gel over which a spatially uniform electric field is applied (see Figure 4.19a)). The dye passively diffuses in the gel and is advected by the applied electric field, giving rise to an advection-diffusion equation with advection in one direction: $u_t = D(u_{xx} + u_{yy}) + vu_y$. Both et al. [2019] showed that *DeepMoD* could discover the correct underlying equation from the full data-set (size 120 x 150 pixels and 25 frames). Here, we study how much we can sub-sample this data and still discover the advection-diffusion equation.

In figure 4.19 c) and d) we perform grid based as well as a random subsampling of the data. The neural network-based method discovers the advection-diffusion equation on as few as 6 x 8 spatial sampling points with 13 time-points, or with 20 x 25 pixels on only 3 time-points. The minimum number of required samples is similar for grid and random sampling, confirming that when both axes are poorly sampled, there is no benefit to sample randomly.

The smallest characteristic length scale in the problem is the width of the dye at $t = t_0$, which we estimate as $l_c \approx 10$ pixels. For the data presented in figure 4.19c) and 4.19d), at a resolution below 10×13 sensors classical approaches would inherently fail, even if no noise was present in the data set. This is indeed what we observe: using a finite difference-based library we

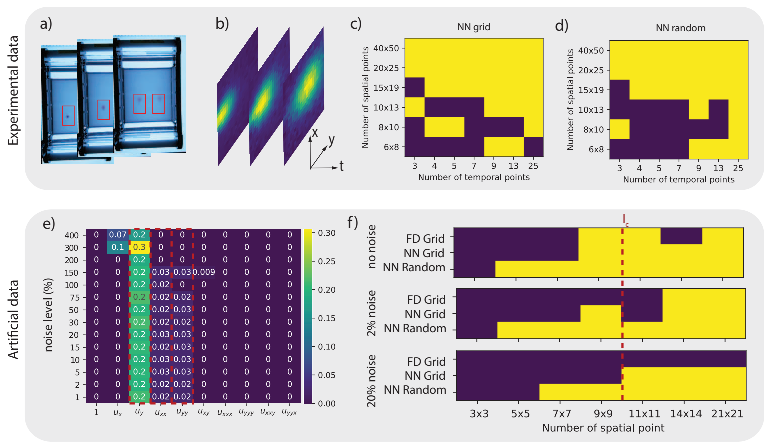

Figure 4.19: a) Experimental setup of the gel electrophoresis. b) Three time frames of the density with a spatial resolution of 20x25. c) and d) Success diagram for the experimental data indicating correct model discovery (Yellow indicates the correct AD equation $u_t = D(u_{xx} + u_{yy}) + vu_y$ is discovered) as function of the spatial and temporal resolution for c) grid sampling and d) random sampling. e) Obtained mask and coefficients ($D = 0.025$ and and $v = (0, 0.2)$) for the artificial data-set as function of the noise level (11x11 spatial resolution). Hereby, yellow indicates the terms selected by the algorithm and the red dashed box the terms that are expected in the PDE. f) Success diagrams for various levels of additive noise, comparing the result of DeepMoD with a grid and random sampling strategy and the classical LassoCV algorithm on a Finite Difference (FD)-based library (after SVD filtering of the different frames).

were unable to recover the advection-diffusion equation, even after denoising with SVD. However, we do observe that below this value we occasionally discover the correct PDE, despite the experimental noise present in the data. The use of a neural network and random sampling lead to non-deterministic behaviour: the neural network training dynamics depend on its initialization and two randomly sampled datasets of similar size might not lead to similar results. In practice this leads to a ‘soft’ decision boundary, where a fraction of a set of runs with different initialization and datasets fail.

4.5.1.2.1 Noiseless synthetic data set To further confirm our results from the previous section, we simulate the experiment by solving the 2D advection-diffusion with a Gaussian initial condition and experimentally determined parameters ($D = 0.025$ and and $v = (0, 0.2)$. Figure 4.19e) shows the selected terms and their magnitude as functions of the applied noise levels for a highly subsampled data-set (grid sampling, spatial resolution of 11x11 and temporal resolution 14). The correct AD equation is recovered up to noise levels of 100% (See figure 4.19e), confirming the noise robustness of the NN-based model discovery. In panel f) we compare the deep-learning based model discovery using grid and random sampling with classical methods for various noise levels and sensor spacing with a fixed temporal resolution of 81 frames (Data for the FD was pre-processed with a SVD filter, see SI for further details). We confirm that, similarly to the Burgers example of the previous section, the correct underlying PDE is discovered even below the smallest characteristic length-scale in the problem (indicated by a red dashed line in figure 4.19f).

This figure confirms our three main conclusions: 1) In the noiseless limit, classical approaches are only slightly less performing than NN-based model discovery for grid sampling. 2) Increasing the noise level dramatically impacts classical model discovery while barely impacting NN-based model discovery and 3) random sampling over space considerably improves performance, allowing model discovery with roughly 4-8 times fewer sample points for this particular data-set (depending on the noise level).

4.5.1.3 Experimental data - Cable equation

Applying a constant voltage to a RC-circuit with longitudinal resistance (see figure 4.20 a) result in time-dependent voltage increase throughout the circuit due to the charging of the capacitors. This rise is modeled by the cable equation, which is essentially a reaction-diffusion equation $ u_t = u_{xx}/(R_l C) + u/(R_m C)$ with C the capacitance, R_l the longitudinal resistance and R_m the parallel resistance of the circuit. The discrete nature of the experiment automatically gives $\Delta x = O(l_c)$. We consider an extreme case where we only have seven sensors throughout the circuit (i.e. spatial axis), but take 2500 samples along the time axis. Figure 4.20b shows the measured voltage at these seven elements. Initially, all the capacitors are uncharged and we observe a sharp voltage increase at the first element. As the capacitors charge, this further propagates through the circuit, charging the capacitors and resulting in the curves shown in the figure. We apply both a classical approach with the library generated with splines and DeepMoD to a varying amount of elements. Figure 4.20 c and d show that the DeepMoD discovers the cable equation with as few as seven elements, whereas classical methods are unable to find the cable equation at any number of elements.

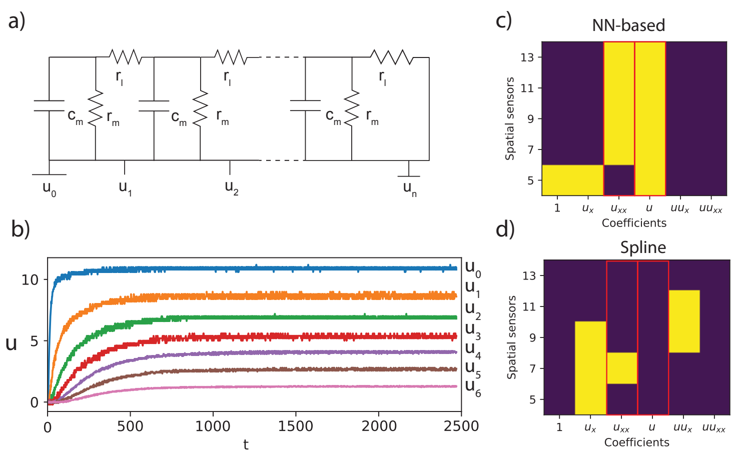

Figure 4.20: a) Schematic overview of the electronic setup to generate the cable equation. b) The voltage drop u as function of time for various positions along the circuit for a circuit with 7 elements. The mask obtained for c) NN-based and d) cross validated Lasso with spline based library model discovery (Yellow indicates the term was recovered by the algorithm). The red boxes indicate the two expected terms in the equation.

Chapter 5

Conclusion

Despite the massive progress in the last few years, model discover of (partial) differential equations is still a field in its infancy. At the start of this thesis in early 2019, both the sparse regression approach for PDEs and PINNs were novel; to our knowledge, no one had attempted to perform model discovery on noisy experimental data. Considering the experiments on synthetic data that would have been a futile attempt - they all showed model discovery required densely sampled datasets with very little noise (<10%, but more often <2% noise). With this we set out to construct methods able to handle noisy and sparse datasets originating from experiments. The line of work we have developed with DeepMoD makes large strides towards this goal. In each paper we show that our methods are able to handle >50% noise and work with an order of magnitude less data than classical methods on increasingly challenging (synthetic) datasets such as the Kuramoto-Shivashinsky equation. DeepMoD also recovered the underlying equation when applied to data generated by simple experiments, validating our approach.

A second, more implicit goal was to create *accessible* methods. Complex methods are less likely to be adapted than simple ones, and if the goal is to make model discovery a valuable addition to scientists' toolboxes, complexity should be strongly limited. Ideally, scientists without a background in numerical methods should be able to understand and apply our methods. Deep learning is an ideal vehicle for this, as many scientists these days have at least a basic understanding. Automatic differentiation, the key but complex mechanism at the core of DL, is often abstracted away to frameworks such as Pytorch, making it possible to construct powerful approaches without in-depth numerical knowledge. This leads to the peculiar situation where a neural network-based approach is regarded as more accesible than splines. Indeed, our work shows that an integrated combination of simple features (a basic MLP, a simple Lasso and a constraint) can easily and strongly outperform much more complex traditional approaches - a testament to the power of differentiable programming.

Our work also carries strong implications for future work and for model discovery as a field. First and foremost, it shows that the limiting factor is the accuracy of the features, implying that accurately modelling and denoising data is just as important as the sparse regression. Perhaps paradoxically then, significant progress can be made by focussing on data-driven modelling. In all cases, neural networks (especially PINNs) should feature prominently; the benefits of automatic differentiation and excellent inter- and extrapolation only become more pronounced in high-dimensional data. That is not to say that non-DL-based approaches should be neglected. A main thread in our work has been to show how ‘classical’ methods such as sparse and Bayesian regression can be integrated in DL approaches, and most progress can be made by synthesizing these approaches. The Bayesian regression and model selection literature is especially rich, and combining DL-based modelling with these methods should prove fruitful.

Taken together, our work strongly establishes the argument for physics-constrained, neural network-based surrogates for model discovery of PDEs on experimental data.

5.1 Challenges and questions unanswered

Model discovery is a young and exciting field, and as with all young fields, many limitations, challenges and questions remain. As such I list these in no particular order below - I’ve mentioned some of them before, others might have occured to you while reading this thesis and a few of them are of a more philosophical nature.

- Perhaps the single biggest barrier to apply model discovery on novel, experimental data is the lack of methods able to handle data with spatially and temporally varying coefficient fields. Initial work [Rudy et al., 2018, Chen and Lin, 2021] has focussed on leveraging group sparsity, but so far this approach works only for a single varying dimension - sufficient for temporal dependence, but not for spatial dependence (2D at least). Scaling this approach to spatio-temporal dependence yields a compressed sensing problem: every sample becomes a separate regression problem ($n = 1$) with many features ($p \gg n$). An alternative approach would be to extend on DeepMoD, and use a neural network to model the field. This implicitly encodes a smoothness bias into the fields and initial results have been promising. However, to perform actual model discovery would require some criterion for deciding when a field can be considered inactive.

- Contrarily, perhaps one of the biggest opportunities for model discovery would be the synthesis of data from multiple sources. Rarely does a single experiment tell the whole story: we need multiple experiments to capture all of a systems dynamics. Initial work using group sparsity [de Silva et al., 2019, Tod et al., 2021] only scratches the surface of what is possible. How can data from multiple experiments with different length and time scales be integrated? Initial work on combining timescales [Champion et al., 2019a] on ODEs is promising, but has not been expanded to PDEs, nor to (multiple) lengthscales. Even more interesting would be to combine data from different experimental setups and instruments, also known as multimodal fusion.

- Related to the data synthesis challenge is that of sampling, active learning and experimental design. Sampling has been studied in-depth from the perspective of signal reconstruction [Brunton et al., 2013], but not from the viewpoint of model discovery - a subtly different problem. Is there an optimal sampling strategy for model discovery, and if so, what would it be? Having more knowledge about this could form the basis for an active learning approach, where the model itself suggests where to sample to optimize model discovery. The most ambitious form of this would be experiment design, with an agent suggesting novel experiments designed to aid model discovery.

- All of the neural networks used in our work were simple MLPs - no normalization, batching, or attention mechanisms. How much do the implicit biases of the function approximator impact the discovered equation? Various works show that incorporating underlying symmetries and invariances in the network improves performance [Cranmer et al., 2020, Greydanus et al., 2019, Finzi et al., 2021], but in the context of model discovery these would have to be discovered from data.

- Model discovery is usually applied directly on the observable data; we observe some quantity u and also wish to find a model for u. In some cases the observable is not fully observed - the model is described by an n-dimensional state, and we only observe some of those dimensions -, or not at all what we wish to model - we observe single particles but want to model a density-. This significantly complicates model discovery, as it requires first constructing or inferring these 'latent' dynamics. This is likely to yield biased estimates and, in our experience, model discovery deals poorly with bias in data. Making model discovery more robust at dealing with bias is a sorely understudied subject.

- A reflection on in what kind of settings model discovery can be useful is needed. Most papers (including our own) use model discovery on systems where the dynamics are already known, and relatively simple. In what kind of data could model discovery be truly useful? Can the complex, high dimensional data associated with the modern world by described by relatively simple, physics-inspired PDEs? Is model discovery only useful to discover effective models, or would it be able to discover new, more fundamental dynamics?

Bibliography

Steven Atkinson. Bayesian Hidden Physics Models: Uncertainty Quantification for Discovery of Nonlinear Partial Differential Operators from Data. *arXiv:2006.04228 [cs, stat]*, June 2020. URL http://arxiv.org/abs/2006.04228. arXiv: 2006.04228.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. *arXiv:1909.01377 [cs, stat]*, October 2019. URL http://arxiv.org/abs/1909.01377. arXiv: 1909.01377.

Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, January 2009. ISSN 1936-4954. doi: 10.1137/080716542. URL http://epubs.siam.org/doi/10.1137/080716542.

Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. *arXiv:1803.05649 [cs, stat]*, March 2018. URL http://arxiv.org/abs/1803.05649. arXiv: 1803.05649.

J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, June 2007. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0609476104. URL http://www.pnas.org/cgi/doi/10.1073/pnas.0609476104.

Gert-Jan Both and Remy Kusters. Temporal Normalizing Flows. *arXiv:1912.09092 [physics, stat]*, December 2019. URL http://arxiv.org/abs/1912.09092. arXiv: 1912.09092.

Gert-Jan Both and Remy Kusters. Fully differentiable model discovery. *arXiv:2106.04886 [cs, stat]*, June 2021. URL http://arxiv.org/abs/2106.04886. arXiv: 2106.04886.

Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. DeepMoD: Deep learning for Model Discovery in noisy data. *arXiv:1904.09406 [physics, q-bio, stat]*, April 2019. URL http://arxiv.org/abs/1904.09406. arXiv: 1904.09406.

Gert-Jan Both, Georges Tod, and Remy Kusters. Model discovery in the sparse sampling regime. *arXiv:2105.00400 [physics, stat]*, May 2021a. URL http://arxiv.org/abs/2105.00400. arXiv: 2105.00400.

Gert-Jan Both, Gijs Vermarien, and Remy Kusters. Sparsely constrained neural networks for model discovery of PDEs. *arXiv:2011.04336 [physics]*, May 2021b. URL http://arxiv.org/abs/2011.04336. arXiv: 2011.04336.

B. W. Brunton, S. L. Brunton, J. L. Proctor, and J. N. Kutz. Optimal Sensor Placement and Enhanced Sparsity for Classification. *arXiv:1310.4217 [cs]*, October 2013. URL http://arxiv.org/abs/1310.4217. arXiv: 1310.4217.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1517384113. URL http://www.pnas.org/lookup/doi/10.1073/pnas.1517384113.

Kathleen Champion, Steven L. Brunton, and J. Nathan Kutz. Discovery of Nonlinear Multiscale Systems: Sampling Strategies and Embeddings. *SIAM Journal on Applied Dynamical Systems*, 18(1):312–333, January 2019a. ISSN 1536-0040. doi: 10.1137/18M1188227. URL http://arxiv.org/abs/1805.07411. arXiv: 1805.07411.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *arXiv:1904.02107 [stat]*, March 2019b. URL http://arxiv.org/abs/1904.02107. arXiv: 1904.02107.

Kathleen Champion, Peng Zheng, Aleksandr Y. Aravkin, Steven L. Brunton, and J. Nathan Kutz. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *arXiv:1906.10612 [physics]*, June 2019c. URL http://arxiv.org/abs/1906.10612. arXiv: 1906.10612.

Aoxue Chen and Guang Lin. Robust data-driven discovery of partial differential equations with time-dependent coefficients. *arXiv:2102.01432 [cs, stat]*, February 2021. URL http://arxiv.org/abs/2102.01432. arXiv: 2102.01432.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *arXiv:1806.07366 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.07366. arXiv: 1806.07366.

Roberto Cipolla, Yarin Gal, and Alex Kendall. Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *2018*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7482–7491, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00781. URL https://ieeexplore.ieee.org/document/8578879/.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. *arXiv:2003.04630 [physics, stat]*, March 2020. URL http://arxiv.org/abs/2003.04630. arXiv: 2003.04630.

Brian de Silva, David M. Higdon, Steven L. Brunton, and J. Nathan Kutz. Discovery of Physics from Data: Universal Laws and Discrepancy Models. *arXiv:1906.07906 [physics, stat]*, June 2019. URL http://arxiv.org/abs/1906.07906. arXiv: 1906.07906.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A Practical Method for Constructing Equivariant Multilayer Perceptrons for Arbitrary Matrix Groups. *arXiv:2104.09459 [cs, math, stat]*, April 2021. URL http://arxiv.org/abs/2104.09459. arXiv: 2104.09459.

Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian Neural Networks. *arXiv:1906.01563 [cs]*, September 2019. URL http://arxiv.org/abs/1906.01563. arXiv: 1906.01563.

Roger Guimerà, Ignasi Reichardt, Antoni Aguilar-Mogas, Francesco A. Massucci, Manuel Miranda, Jordi Pallarès, and Marta Sales-Pardo. A Bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, 6(5):eaav6971, January 2020. ISSN 2375-2548. doi: 10.1126/sciadv.aav6971. URL https://advances.sciencemag.org/lookup/doi/10.1126/sciadv.aav6971.

Ingvild M. Helgøy and Yushu Li. A Noise-Robust Fast Sparse Bayesian Learning Model. *arXiv:1908.07220 [cs, stat]*, May 2020. URL http://arxiv.org/abs/1908.07220. arXiv: 1908.07220.

Zheyuan Hu, Ameya D. Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization? *arXiv:2109.09444 [cs, math, stat]*, September 2021. URL http://arxiv.org/abs/2109.09444. arXiv: 2109.09444.

Junzhou Huang and Tong Zhang. The Benefit of Group Sparsity. *arXiv:0901.2962 [math, stat]*, March 2009. URL http://arxiv.org/abs/0901.2962. arXiv: 0901.2962.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from Data. *arXiv:1710.09668 [cs, math, stat]*, October 2017. URL http://arxiv.org/abs/1710.09668. arXiv: 1710.09668.

Zichao Long, Yiping Lu, and Bin Dong. PDE-Net 2.0: Learning PDEs from Data with A Numeric-Symbolic Hybrid Deep Network. *Journal of Computational Physics*, 399:108925, December 2019. ISSN 00219991. doi: 10.1016/j.jcp.2019.108925. URL http://arxiv.org/abs/1812.04426. arXiv: 1812.04426.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. *arXiv:1712.01312 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1712.01312. arXiv: 1712.01312.

Suryanarayana Maddu, Bevan L. Cheeseman, Ivo F. Sbalzarini, and Christian L. Müller. Stability selection enables robust learning of partial differential equations from limited noisy data. *arXiv:1907.07810 [physics]*, July 2019. URL http://arxiv.org/abs/1907.07810. arXiv: 1907.07810.

Suryanarayana Maddu, Bevan L. Cheeseman, Christian L. Müller, and Ivo F. Sbalzarini. Learning physically consistent mathematical models from data using group sparsity. *arXiv:2012.06391 [cs, q-bio, stat]*, December 2020. URL http://arxiv.org/abs/2012.06391. arXiv: 2012.06391.

Suryanarayana Maddu, Dominik Sturm, Christian L. M üller, and Ivo F. Sbalzarini. Inverse-Dirichlet Weighting Enables Reliable Training of Physics Informed Neural Networks. *arXiv:2107.00940 [physics, q-bio]*, July 2021. URL http://arxiv.org/abs/2107.00940. arXiv: 2107.00940.

N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 473(2204):20170009, August 2017. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2017.0009. URL http://rspa.royalsocietypublishing.org/lookup/doi/10.1098/rspa.2017.0009.

Michail Maslyaev, Alexander Hvatov, and Anna Kalyuzhnaya. Data-driven PDE discovery with evolutionary approach. *arXiv:1903.08011 [cs, math]*, 11540:635–641, 2019. doi: 10.1007/978-3-030-22750-0_61. URL http://arxiv.org/abs/1903.08011. arXiv: 1903.08011.

Levi McClenny and Ulisses Braga-Neto. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism. *arXiv:2009.04544 [cs, stat]*, September 2020. URL http://arxiv.org/abs/2009.04544. arXiv: 2009.04544.

Nicolai Meinshausen and Peter Buehlmann. Stability Selection. *arXiv:0809.2932 [stat]*, May 2009. URL http://arxiv.org/abs/0809.2932. arXiv: 0809.2932.

Rajdip Nayek, Ramon Fuentes, Keith Worden, and Elizabeth J. Cross. On spike-and-slab priors for Bayesian equation discovery of nonlinear dynamical systems via sparse linear regression. *arXiv:2012.01937 [cs, eess, stat]*, December 2020. URL http://arxiv.org/abs/2012.01937. arXiv: 2012.01937.

K. J. Painter. Mathematical models for chemotaxis and their applications in self-organisation phenomena. *arXiv:1806.08627 [q-bio]*, June 2018. URL http://arxiv.org/abs/1806.08627. arXiv: 1806.08627.

Brenden K. Petersen. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv:1912.04871 [cs, stat]*, February 2020. URL http://arxiv.org/abs/1912.04871. arXiv: 1912.04871.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal Differential Equations for Scientific Machine Learning. *arXiv:2001.04385 [cs, math, q-bio, stat]*, January 2020. URL http://arxiv.org/abs/2001.04385. arXiv: 2001.04385.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv:1711.10561 [cs, math, stat]*, November 2017a. URL http://arxiv.org/abs/1711.10561. arXiv: 1711.10561.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv:1711.10566 [cs, math, stat]*, November 2017b. URL http://arxiv.org/abs/1711.10566. arXiv: 1711.10566.

Bharath Ramsundar, Dilip Krishnamurthy, and Venkatasubramanian Viswanathan. Differentiable Physics: A Position Piece. *arXiv:2109.07573 [physics]*, September 2021. URL http://arxiv.org/abs/2109.07573. arXiv: 2109.07573.

Adil Rasheed, Omer San, and Trond Kvamsdal. Digital Twin: Values, Challenges and Enablers From a Modeling Perspective. *IEEE Access*, 8:21980–22012, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2970143. URL https://ieeexplore.ieee.org/document/8972429/.

Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*, May 2015. URL http://arxiv.org/abs/1505.05770. arXiv: 1505.05770.

Samuel Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-driven identification of parametric partial differential equations.

arXiv:1806.00732 [math], June 2018. URL http://arxiv.org/abs/1806.00732. arXiv: 1806.00732.

Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, April 2017. ISSN 2375-2548. doi: 10.1126/sciadv.1602614. URL http://advances.sciencemag.org/lookup/doi/10.1126/sciadv.1602614.

Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, January 2017. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2016.0446. URL https://royalsocietypublishing.org/doi/10.1098/rspa.2016.0446.

Michael Schmidt and Hod Lipson. Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923):81–85, April 2009. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1165893. URL https://www.sciencemag.org/lookup/doi/10.1126/science.1165893.

Sungyong Seo and Yan Liu. Differentiable Physics-informed Graph Networks. *arXiv:1902.02950 [cs, stat]*, February 2019. URL http://arxiv.org/abs/1902.02950. arXiv: 1902.02950.

Khemraj Shukla, Ameya D. Jagtap, and George Em Karniadakis. Parallel Physics-Informed Neural Networks via Domain Decomposition. *arXiv:2104.10013 [cs]*, September 2021. URL http://arxiv.org/abs/2104.10013. arXiv: 2104.10013.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *arXiv:2006.09661 [cs, eess]*, June 2020. URL http://arxiv.org/abs/2006.09661. arXiv: 2006.09661.

Simo Särkkä. The Use of Gaussian Processes in System Identification. *arXiv:1907.06066 [cs, eess, stat]*, July 2019. URL http://arxiv.org/abs/1907.06066. arXiv: 1907.06066.

Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective: Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, June 2011. ISSN 13697412. doi: 10.1111/j.1467-9868.2011.00771.x. URL http://doi.wiley.com/10.1111/j.1467-9868.2011.00771.x.

Michael Tipping. tipping01a.pdf. *Journal of Machine Learning Research*, I, 2001. URL https://www.jmlr.org/papers/volume1/tipping01a/tipping01a.pdf.

Georges Tod, Gert-Jan Both, and Remy Kusters. Discovering PDEs from Multiple Experiments. *arXiv:2109.11939 [physics, stat]*, September 2021. URL http://arxiv.org/abs/2109.11939. arXiv: 2109.11939.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: a Physics-Inspired Method for Symbolic Regression. *arXiv:1905.11481 [hep-th, physics:physics]*, May 2019. URL http://arxiv.org/abs/1905.11481. arXiv: 1905.11481.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *arXiv:2006.10782 [physics, stat]*, December 2020. URL http://arxiv.org/abs/2006.10782. arXiv: 2006.10782.

Floris Van Breugel, J. Nathan Kutz, and Bingni W. Brunton. Numerical Differentiation of Noisy Data: A Unifying Multi-Objective Optimization Framework. *IEEE Access*, 8:196865–196877, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3034077. Conference Name: IEEE Access.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv:2001.04536 [cs, math, stat]*, January 2020a. URL http://arxiv.org/abs/2001.04536. arXiv: 2001.04536.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *arXiv:2007.14527 [cs, math, stat]*, July 2020b. URL http://arxiv.org/abs/2007.14527. arXiv: 2007.14527.

Antoine Wehenkel and Gilles Louppe. Unconstrained Monotonic Neural Networks. *arXiv:1908.05164 [cs, stat]*, August 2019. URL http://arxiv.org/abs/1908.05164. arXiv: 1908.05164.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data. *arXiv:2003.06097 [cs, stat]*, March 2020. doi: 10.1016/j.jcp.2020.109913. URL http://arxiv.org/abs/2003.06097. arXiv: 2003.06097.

Ye Yuan, Junlin Li, Liang Li, Frank Jiang, Xiuchuan Tang, Fumin Zhang, Sheng Liu, Jorge Goncalves, Henning U. Voss, Xiuting Li, Jürgen Kurths, and Han Ding. Machine Discovery of Partial Differential Equations from Spatiotemporal Data. *arXiv:1909.06730 [physics, stat]*, September 2019. URL http://arxiv.org/abs/1909.06730. arXiv: 1909.06730.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. page 32.

Peng Zhao and Bin Yu. On Model Selection Consistency of Lasso. page 23.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, December 2006. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000000735. URL https://www.tandfonline.com/doi/full/10.1198/016214506000000735.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2005.00503.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00503.x. _eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2005.00503.x.

RÉSUMÉ

La découverte de modèles autonome vise à découvrir des équations différentielles à base d'un ensemble de données. Elle est souvent abordée comme un problème de régression éparse, en sélectionnant les termes active et en construisant l'équation inconnue à partir d'un ensemble de contributions candidates.Dans le cas des équations différentielles partielles typiquement consistent de dérivées d'ordre supérieur, qui sont calculées à l'aide de la différentiation numérique. Il est donc difficile de calculer les candidates avec précision pour des données expérimentales, qui sont souvent bruyantes et éparses. Dans cette thèse, nous développons une méthode de découverte de modèles basée sur les réseaux de neurones. La base de notre approche est un réseau neuronal qui interpole et débruite les données, et contraint le réseau à une équation donnée - un modèle connu sous le nom de réseaux neuronaux informés de la physique (Physics Informed Neural Networks:PINNS). Simultanément, nous utilisons la régression éparse pour apprendre cette équation contraignante au fur et à mesure de l'entraînement du réseau neuronal, ce qui permet d'obtenir le modèle sous-jacent. Dans la première partie de cette thèse, nous montrons qu'une telle approche améliore considérablement la robustesse de la découverte de modèles par rapport aux réseaux neuronaux non contraints ou à d'autres approches de découverte de modèles. Dans la deuxième partie, nous présentons un cadre modulaire, montrant comment ces réseaux contraints peuvent utiliser n'importe quel algorithme de régression éparse. Dans la troisième partie, nous nous appuyons sur les deux premières parties pour réaliser un algorithme de découverte de modèle entièrement différentiable en contraignant un réseau neuronal avec un apprentissage bayésien éparse. De plus, nous introduisons les flux de normalisation conditionnels (Normalizing flows) et montrons comment ils peuvent être utilisés pour déduire des distributions de probabilité dépendant du temps. Dans l'ensemble, notre travail montre l'importance d'une modélisation précise des données pour la découverte de modèles, et renforce l'argument en faveur de substituts de réseaux neuronaux contraints par la physique pour la découverte des équations différentielles a base de données expérimentales.

MOTS CLÉS

Régression parcimonieuse - Réseaux neuronaux informés par la physique - Découverte de modèles

ABSTRACT

Model discovery aims at autonomously discovering equations underlying a dataset. It is often approached as a sparse regression problem, selecting terms and constructing the unknown equation from a set of candidate feautures. In the case of partial differential equations, these features consist of higher-order derivatives, which are calculated using numerical differentiation. This makes it challenging to accurately calculate the candidate feature in experimental data, which is noisy and sparse. In this thesis we develop a neural network-based model discovery method, with a focus on discovering partial differential equations from noisy and sparse data. The foundation of our approach is a neural network which interpolates and denoises the data, and constrain the network to a given equation - a model known as physics informed neural networks. Simultaneously, we employ sparse regression to learn this constraining equation as the neural network is trained, yielding the underlying model. In the first part of this thesis we show that such an approach significantly improves the robustness of model discovery compared to unconstrained neural networks or other model discovery approaches. Improving on this, in the second part we present a modular framework, showing how these constrained networks can utilize any sparse regression algorithm. In the third part, we build upon the first two parts to achieve a fully differentiable model discovery algorithm by constraining a neural network with Sparse Bayesian Learning. Additionally, we introduce Conditional Normalizing Flows and show how they can be used to infer time-dependent probability distributions. Taken together, our work shows the importance of accurately modelling data for model discovery, and strongly establishes the argument for physics-constrained, neural network-based surrogates for model discovery of PDEs on experimental data.

KEYWORDS

Sparse regression - Physics Informed Neural Networks - Model discovery