Robust subsampling-based sparse Bayesian inference to tackle four challenges (large noise, outliers, data integration, and extrapolation) in the discovery of physical laws from data

Sheng Zhang[a], Guang Lin[a,b,c,*]

[a]*Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA*
[b]*School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA*
[c]*Department of Statistics, Purdue University, West Lafayette, IN 47907, USA*

Abstract

The derivation of physical laws is a dominant topic in scientific research. We propose a new method capable of discovering the physical laws from data to tackle four challenges in the previous methods. The four challenges are: (1) large noise in the data, (2) outliers in the data, (3) integrating the data collected from different experiments, and (4) extrapolating the solutions to the areas that have no available data. To resolve these four challenges, we try to discover the governing differential equations and develop a model-discovering method based on sparse Bayesian inference and subsampling. The subsampling technique is used for improving the accuracy of the Bayesian learning algorithm here, while it is usually employed for estimating statistics or speeding up algorithms elsewhere. The optimal subsampling size is moderate, neither too small nor too big. Another merit of our method is that it can work with limited data by the virtue of Bayesian inference. We demonstrate how to use our method to tackle the four aforementioned challenges step by step through numerical examples: (1) predator-prey model with noise, (2) shallow water equations with outliers, (3) heat diffusion with random initial and boundary conditions, and (4) fish-harvesting problem with bifurcations. Numerical results show that the robustness and accuracy of our new method is significantly better than the other model-discovering methods and traditional regression methods.

Keywords: machine learning, Bayesian inference, subsampling, outlier, data integration, extrapolation

1. Introduction

The search for physical laws has been a fundamental aim of science for centuries. The physical laws are critical to the understanding of natural phenomena and the prediction of future dynamics. They are either derived by other known physical laws or generalized based on empirical observations of physical behavior. We focus on the second task, which is also called data-driven discovery of governing physical laws. It deals with the case where experimental data are given while the governing physical model is unclear. Traditional methods for discovering physical laws from

*Corresponding author.
Email address: guanglin@purdue.edu (Guang Lin)

data include interpolation and regression. Suppose $x : \mathbb{R} \to \mathbb{R}$ is an unknown physical law. Given data $\{t_i, x(t_i)\}_{i=1}^N$, traditional methods approximate the expression of $x(t)$ in terms of a class of functions of t. This approach has two limitations:

- If the data are collected from different experiments, traditional methods would not be able to use all of the data together to discover the physical laws. For example, free falls from different initial height $[x(t) = x_0 - (1/2)gt^2]$ follow the same physical law but they have different trajectories. Traditional methods can only use the data on the same trajectory to discover the path and predict future motion. However, using all the data can increase the accuracy.

- Traditional methods are incapable of extrapolating to the areas where no data are given.

To resolve these two limitations, we adopt the strategy: first, discover the differential equations that $x(t)$ satisfies; second, solve the differential equations analytically or numerically. For the free fall example, we can use all the data from different trajectories together to discover the differential equation $x'(t) = -gt$ and extrapolate to any other given initial height. This discovery pattern is applicable to a larger class of models than traditional methods and derives the governing differential equations, which provide insights to the governing physical laws behind the observations [1]. Many fundamental laws are formulated in the form of differential equations, such as Maxwell equations in classical electromagnetism, Einstein field equations in general relativity, Schrodinger equation in quantum mechanics, Navier-Stokes equations in fluid dynamics, Boltzmann equation in thermodynamic, predator-prey equations in biology, and Black-Scholes equation in economics. While automated techniques for generating and collecting data from scientific measurements are more and more precise and powerful, automated processes for extracting knowledge in analytic forms from data are limited [2]. Our goal is to develop automated algorithms for extracting the governing differential equations from data.

Consider a differential equation of the form

$$\frac{dx}{dt} = f(t, x), \tag{1}$$

with the unknown function $f(t, x)$. Given the data $\left\{t_i, x_i, x_i'\right\}_{i=1}^N$ collected from a space governed by this differential equation, where $x_i = x(t_i)$ and $x_i' = (dx/dt)(t_i)$, automated algorithms for deriving the expression of $f(t, x)$ are studied from various approaches. One of the approaches assumes that $f(t, x)$ is a linear combination of simple functions of t and x. First, construct a moderately large set of basis-functions that may contain all the terms of $f(t, x)$; then, apply algorithms to select a subset that is exactly all the terms of $f(t, x)$ from the basis-functions and estimate the corresponding weights in the linear combination.

Suppose the basis-functions are chosen as $f_1(t, x), f_2(t, x), \ldots, f_M(t, x)$. Then we need to estimate the weights $w_1, w_2, \ldots, w_M$ in the following linear combination:

$$\frac{dx}{dt} = w_1 f_1(t, x) + w_2 f_2(t, x) + \cdots + w_M f_M(t, x). \tag{2}$$

Given the data $\left\{t_i, x_i, x_i'\right\}_{i=1}^{N}$, where $x_i = x(t_i)$ and $x_i' = (dx/dt)(t_i)$, the above problem becomes a regression problem as follows:

$$\begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_N' \end{bmatrix} = \begin{bmatrix} f_1(t_1, x_1) & f_2(t_1, x_1) & \cdots & f_M(t_1, x_1) \\ f_1(t_2, x_2) & f_2(t_2, x_2) & \cdots & f_M(t_2, x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(t_N, x_N) & f_2(t_N, x_N) & \cdots & f_M(t_N, x_N) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \epsilon, \tag{3}$$

where $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_N]^{\mathrm{T}}$ is the model error. Let

$$\eta = [x_1', \ldots, x_N']^{\mathrm{T}} \tag{4}$$

$$\Phi = \begin{bmatrix} f_1(t_1, x_1) & \cdots & f_M(t_1, x_1) \\ \vdots & \ddots & \vdots \\ f_1(t_N, x_N) & \cdots & f_M(t_N, x_N) \end{bmatrix} \tag{5}$$

$$\mathbf{w} = [w_1, \ldots, w_M]^{\mathrm{T}}. \tag{6}$$

Equation (3) may be written in the vector form as follows:

$$\eta = \Phi\mathbf{w} + \epsilon. \tag{7}$$

Now the problem is to estimate the weight-vector $\mathbf{w}$ given a known vector η and a known matrix Φ.

Since many physical systems have few terms in the equations, the set of basis-functions usually has many more terms than $f(t, x)$: $M \gg \#\{\text{terms in } f(t, x)\}$, which suggests the use of sparse methods to select the subset of basis-functions and estimate the weights. These sparse methods can be sequential threshold least squares (also called sparse identification of nonlinear dynamics (SINDy)) [3, 4], lasso (least absolute shrinkage and selection operator) [5, 6], or threshold sparse Bayesian regression [1]. Sequential threshold least squares does least-square regression and eliminates the terms with small weights iteratively (Algorithm 1); lasso solves the following optimization problem:

$$\min_{\mathbf{w}} \left\{ \frac{1}{2N} \|\eta - \Phi\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 \right\}, \tag{8}$$

where the regularization parameter λ may be fitted by cross-validation (Algorithm 2); threshold sparse Bayesian regression calculates the posterior distribution of $\mathbf{w}$ given the data and then filters out small weights, iteratively until convergence (Algorithm 3). A comparison of these three sparse methods is illustrated in [1] and shows that threshold sparse Bayesian regression is more accurate and robust than the other two methods.

The same mechanism as above also applies to the discovery of general differential equations including higher-order differential equations and implicit differential equations [1], besides the differential equations of the form (1). Nevertheless, the mechanism is described in the pattern (1) here for convenience and simplification, so that more attention is given to the essence of the algorithm itself. In addition, to apply the algorithm to real-world problems, dimensional analysis can be incorporated in the construction of the basis-functions [1]. Any physically meaningful

equation has the same dimensions on every term, which is a property known as dimensional homogeneity. Therefore, when summing up terms in the equations, the addends should be of the same dimension.

Sparse regression methods for data-driven discovery of differential equations are developed recently with a wide range of applications, for example, inferring biological networks [7], sparse identification of a predator-prey system [8], model selection via integral terms [9], extracting high-dimensional dynamics from limited data [10], recovery of chaotic systems from highly corrupted data [11], model selection for dynamical systems via information criteria [12], model predictive control in the low-data limit [13], sparse learning of stochastic dynamical systems [14], model selection for hybrid dynamical systems [15], identification of parametric partial differential equations [16], extracting structured dynamical systems with very few samples [17], constrained Galerkin regression [18], rapid model recovery [19], convergence of the SINDy algorithm [20]. Moreover, other methods for data-driven discovery of differential equations are proposed as well, for instance, deep neural networks [21, 22, 23] and Gaussian process [24]. One of the advantages of the sparse regression methods is the ability to provide explicit formulas of the differential equations, from which further analysis on the systems may be performed, while deep neural networks usually provide "black boxes", in which the mechanism of the systems is not very clearly revealed. Another advantage of the sparse regression methods is that they do not require too much prior knowledge of the differential equations, while Gaussian process methods have restrictions on the form of the differential equations and are used to estimate a few parameters.

Previous developments and applications based on sparse regression methods mostly employ either sequential threshold least squares or lasso, or their variations. One of the reasons why data-driven discovery of differential equations has not yet been applied to industry is the instability of its methods. Previous methods require the data of very high quality, which is usually not the case in industry. Although threshold sparse Bayesian regression is more accurate and robust than the other two methods and provides error bars that quantify the uncertainties [1], its performance is still unsatisfactory if the provided data are of large noise or contain outliers. Therefore, it is instructive to improve the threshold sparse Bayesian regression algorithm and apply it to the fields above, as this will improve the overall performance of the method in most cases. In this paper, we develop a subsampling-based technique for improving the threshold sparse Bayesian regression algorithm, so that the new algorithm is robust to large noise and outliers. Note that subsampling methods are usually employed for estimating statistics [25] or speeding up algorithms [4] in the literatures, but the subsampling method in this paper is used for improving the accuracy of the Bayesian learning algorithm. In practice, denoising techniques can be used to reduce part of the noise and outliers in the data before our algorithm is performed.

The remainder of this paper is structured as follows. In Section 2, we introduce the threshold sparse Bayesian regression algorithm. In Section 3, we detail our new subsampling-based algorithm. In Section 4, we investigate the robustness of our new algorithm through an example of discovering the predator-prey model with noisy data. In Section 5, we discuss how to use our new algorithm to remove outliers, with an example of discovering the shallow water equations using the data corrupted by outliers. In Section 6, we tackle the challenge of data integration through an example of discovering the heat diffusion model with random initial and boundary conditions. In Section 7, we

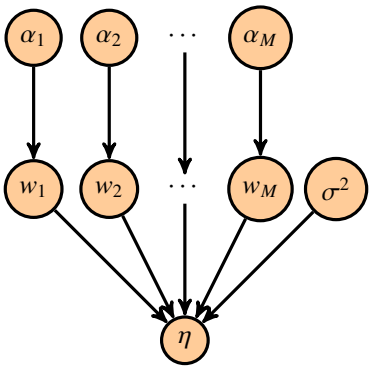

Figure 1: Graphical structure of the sparse Bayesian model.

tackle the challenge of extrapolation through an example of discovering the fish-harvesting model with bifurcations. Finally, the summary is given in Section 8.

2. Threshold sparse Bayesian regression

2.1. Bayesian hierarchical model setup

Let η be a known $N \times 1$ vector, Φ be a known $N \times M$ matrix, $\mathbf{w} = [w_1, w_2, \ldots, w_M]^{\mathrm{T}}$ be the weight-vector to be estimated sparsely, and ϵ be the model error:

$$\eta = \Phi\mathbf{w} + \epsilon. \tag{9}$$

We adopt a sparse Bayesian framework based on RVM (relevance vector machine [26], which is motivated by automatic relevance determination [27, 28]) to estimate the weight-vector $\mathbf{w}$. The Bayesian framework assumes that the model errors are modeled as independent and identically distributed zero-mean Gaussian with variance σ^2. The variance may be specified beforehand, but in this paper it is fitted by the data. The model gives a multivariate Gaussian likelihood on the vector η:

$$p\left(\eta|\mathbf{w}, \sigma^2\right) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left\{-\frac{||\eta - \Phi\mathbf{w}||^2}{2\sigma^2}\right\}. \tag{10}$$

Now we introduce a Gaussian prior over the weight-vector. The prior is governed by a set of hyper-parameters, one hyper-parameter associated with each component of the weight-vector:

$$p\left(\mathbf{w}|\alpha\right) = \prod_{j=1}^{M} \mathcal{N}\left(w_j|0, \alpha_j^{-1}\right), \tag{11}$$

where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_M]^{\mathrm{T}}$. The values of the hyper-parameters are estimated from the data. See Figure 1 for the graphical structure of this model.

2.2. Inference

The posterior over all unknown parameters given the data can be decomposed as follows:

$$p\left(\mathbf{w}, \alpha, \sigma^2|\eta\right) = p\left(\mathbf{w}|\eta, \alpha, \sigma^2\right) p\left(\alpha, \sigma^2|\eta\right). \tag{12}$$

As analytic computations cannot be performed in full, we approximate $p\left(\alpha, \sigma^2|\eta\right)$ using the Dirac delta function at the maximum likelihood estimation:

$$\begin{aligned}
\left(\hat{\alpha}_{\mathrm{ML}}, \hat{\sigma}^2_{\mathrm{ML}}\right) &= \arg\max_{\alpha,\sigma^2} \left\{p\left(\eta|\alpha,\sigma^2\right)\right\} \\
&= \arg\max_{\alpha,\sigma^2} \left\{\int p\left(\eta, \mathbf{w}|\alpha,\sigma^2\right) d\mathbf{w}\right\} \\
&= \arg\max_{\alpha,\sigma^2} \left\{\int p\left(\eta|\mathbf{w},\sigma^2\right) p\left(\mathbf{w}|\alpha\right) d\mathbf{w}\right\} \\
&= \arg\max_{\alpha,\sigma^2} \left\{(2\pi)^{-N/2} \left|\sigma^2\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^{\mathrm{T}}\right|^{-1/2} \exp\left\{-\frac{1}{2}\eta^{\mathrm{T}}\left(\sigma^2\mathbf{I} + \Phi\mathbf{A}^{-1}\Phi^{\mathrm{T}}\right)^{-1}\eta\right\}\right\}, \qquad (13)
\end{aligned}$$

with $\mathbf{A} = \mathrm{diag}\left(\alpha_1, \alpha_2, \ldots, \alpha_M\right)$. We may use the Dirac delta function as an approximation on the basis that this point-estimate is representative of the posterior in the sense that the integral calculation for the posterior using the point-estimate is roughly equal to the one obtained by sampling from the full posterior distribution [26]. This maximization is a type-II maximum likelihood and can be calculated using a fast method [29]. Now, we may integrate out α and σ^2 to get the posterior over the weight-vector:

$$\begin{aligned}
p\left(\mathbf{w}|\eta\right) &= \iint p\left(\mathbf{w}, \alpha, \sigma^2|\eta\right) d\alpha d\sigma^2 \\
&= \iint p\left(\mathbf{w}|\eta, \alpha, \sigma^2\right) p\left(\alpha, \sigma^2|\eta\right) d\alpha d\sigma^2 \\
&\approx \iint p\left(\mathbf{w}|\eta, \alpha, \sigma^2\right) \delta\left(\hat{\alpha}_{\mathrm{ML}}, \hat{\sigma}^2_{\mathrm{ML}}\right) d\alpha d\sigma^2 \\
&= p\left(\mathbf{w}|\eta, \hat{\alpha}_{\mathrm{ML}}, \hat{\sigma}^2_{\mathrm{ML}}\right) \\
&= \frac{p\left(\eta|\mathbf{w}, \hat{\sigma}^2_{\mathrm{ML}}\right) p\left(\mathbf{w}|\hat{\alpha}_{\mathrm{ML}}\right)}{p\left(\eta|\hat{\alpha}_{\mathrm{ML}}, \hat{\sigma}^2_{\mathrm{ML}}\right)} \qquad \text{[Bayes' rule]} \\
&= (2\pi)^{-M/2} \left|\hat{\Sigma}\right|^{-1/2} \exp\left\{-\frac{1}{2}\left(\mathbf{w} - \hat{\mu}\right)^{\mathrm{T}} \hat{\Sigma}^{-1} \left(\mathbf{w} - \hat{\mu}\right)\right\} \\
&= \mathcal{N}\left(\mathbf{w}|\hat{\mu}, \hat{\Sigma}\right), \qquad (14)
\end{aligned}$$

in which the posterior covariance and mean are:

$$\hat{\Sigma} = \left[\hat{\sigma}^{-2}_{\mathrm{ML}}\Phi^{\mathrm{T}}\Phi + \mathrm{diag}\left(\hat{\alpha}_{\mathrm{ML}}\right)\right]^{-1} \qquad (15)$$

$$\hat{\mu} = \hat{\sigma}^{-2}_{\mathrm{ML}}\hat{\Sigma}\Phi^{\mathrm{T}}\eta. \qquad (16)$$

Therefore the posterior for each weight can be deduced from (14):

$$p(w_j|\eta) = \mathcal{N}\left(w_j|\hat{\mu}_j, \hat{\Sigma}_{jj}\right), \qquad (17)$$

with mean $\hat{\mu}_j$ and standard deviation $\hat{\Sigma}^{1/2}_{jj}$. Thus the mean posterior prediction of the weight-vector $\mathbf{w}$ and other quantities that we want to obtain are determined by the values of η and Φ. This is the beauty of the Bayesian approach: there is no need to determine a regularization parameter via expensive cross-validation, and moreover likelihood values and confidence intervals for the solution can be easily calculated [30]. Another merit of the Bayesian approach is that it can work with limited data since it incorporates prior information into the problem to supplement limited data.

2.3. Implementation

In practice the optimal values of many hyper-parameters α_j in (13) are infinite [26], and thus from (15)-(17) the posterior distributions of many components of the weight-vector are sharply peaked at zero. This leads to the sparsity of the resulting weight-vector. To further encourage the accuracy and robustness, a threshold $\delta \geq 0$ is placed on the model to filter out possible disturbance present in the weight-vector. Then the weight-vector is reestimated using the remaining terms, iteratively until convergence. The entire procedure is summarized in Algorithm 3. A discussion about how to choose the threshold and its impact on the solution is detailed in [1].As the threshold is a parameter representing the model complexity, we may use machine learning algorithms such as cross-validation to determine it. Note that in Algorithm 3, $\hat{\mu}$ is more and more sparse after each loop by design. Thus the convergence of the algorithm is guaranteed given the convergence of the calculation of maximum likelihood in (13).

In this paper, we define an error bar value that quantifies the quality of the posterior estimated model as follows:

$$\text{error bar} = \sum_{\substack{j=1 \\ \hat{\mu}_j \neq 0}}^{M} \frac{\hat{\Sigma}_{jj}}{\hat{\mu}_j^2}, \tag{18}$$

where each estimated variance $\hat{\Sigma}_{jj}$ is divided by the square of the corresponding estimated mean $\hat{\mu}_j^2$ to normalize the variance on each weight. This definition adds up all the normalized variances of each weight present in the result, and penalizes the unsureness of the estimations. In other words, a smaller error bar value means smaller normalized variances and higher posterior confidence, and implies higher model quality. If given a set of candidate models for the data, the preferred model would be the one with the minimum error bar value.

As a comparison, other sparse regression algorithms are listed: sequential threshold least squares (Algorithm 1) and lasso (Algorithm 2).

3. Subsampling-based threshold sparse Bayesian regression

3.1. Motivation

In the regression problem (3), when we have more data than the number of unknown weights: $N > M$, we may use a subset of the data $\left\{t_i, x_i, x_i'\right\}_{i=1}^{N}$ to estimate the weights. We do this on the basis that the data sets collected from real-world cases may contain outliers or a percentage of data points of large noise. Classical methods for parameter estimation, such as least squares, fit a model to all of the presented data. These methods have no internal mechanism for detecting or discarding outliers. Other robust regression methods designed to overcome the limitations of traditional methods include iteratively reweighted least squares [31, 32], random sample consensus [33], least absolute deviations [34], Theil-Sen estimator [35, 36], repeated median regression [37, 38], but they do not fit very well into the framework of data-driven discovery of differential equations. Classical methods for parameter estimation are averaging methods based on the assumption (the smoothing assumption) that there will always be enough good values to smooth out any gross noise [33]. In practice the data may contain more noise than what the good values can

Algorithm 1: Sequential threshold least squares: $\eta = \Phi\mathbf{w} + \epsilon$

Input: η, Φ, threshold

Output: $\hat{\mu}$

Solve $\hat{\mu}$ in $(\Phi^{\mathrm{T}}\Phi)\hat{\mu} = \Phi^{\mathrm{T}}\eta$;

For components of $\hat{\mu}$ with absolute value less than the threshold, set them as 0;

while $\hat{\mu} \neq \mathbf{0}$ **do**

  Delete the columns of Φ whose corresponding weight is 0, getting Φ';

  Solve $\hat{\mu}'$ in $(\Phi'^{\mathrm{T}}\Phi')\hat{\mu}' = \Phi'^{\mathrm{T}}\eta$;

  Update the corresponding components of $\hat{\mu}$ using $\hat{\mu}'$;

  For components of $\hat{\mu}$ with absolute value less than the threshold, set them as 0;

  if $\hat{\mu}$ *is the same as the one on the last loop* **then**

    break;

  end

end

Algorithm 2: Lasso: $\eta = \Phi\mathbf{w} + \epsilon$

Input: η, Φ

Output: $\hat{\mu}$

$\hat{\mu} = \arg\min_{\mathbf{w}}\{\frac{1}{2N}\|\eta - \Phi\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1\}$, where λ is fitted by five-fold cross-validation with minimum mean squared error (MSE) on validation sets.

ever compensate, breaking the smoothing assumption. To deal with this situation, an effective way is to discard the "bad" data points (outliers or those of large noise), and use the rest "good" data points to run estimations. Based on this idea, we propose an algorithm called subsampling-based threshold sparse Bayesian regression (SubTSBR) in this paper. This method filters out bad data points by the means of random subsampling.

3.2. Implementation

SubTSBR approaches the problem by randomly selecting data points to estimate the weights and using a measure (error bar) to evaluate the estimations. When an estimation is "good" (small error bar), our algorithm identifies the corresponding selected data points as good data points and the estimation as the final result. To be specific, our algorithm is given a user-preset subsampling size S ($< N$) and the number of loops L (≥ 1) at the very beginning. For each loop, a subset of the data consisting of S data points is randomly selected: $\left\{t_{k_i}, x_{k_i}, x'_{k_i}\right\}_{i=1}^{S}$ $\left(\subset \left\{t_i, x_i, x'_i\right\}_{i=1}^{N}\right)$ and

Algorithm 3: Threshold sparse Bayesian regression: $\eta = \Phi\mathbf{w} + \epsilon$

Input: η, Φ, threshold

Output: $\hat{\mu}$, $\hat{\Sigma}$

Calculate the posterior distribution $p(\mathbf{w}|\eta)$ in $\eta = \Phi\mathbf{w}$, and let the mean be $\hat{\mu}$;

For components of $\hat{\mu}$ with absolute value less than the threshold, set them as 0;

while $\hat{\mu} \neq \mathbf{0}$ **do**

  Delete the columns of Φ whose corresponding weight is 0, and let the result be Φ';

  Calculate the posterior distribution $p(\mathbf{w}'|\eta)$ in $\eta = \Phi'\mathbf{w}'$, and let the mean be $\hat{\mu}'$;

  Update the corresponding components of $\hat{\mu}$ using $\hat{\mu}'$;

  For components of $\hat{\mu}$ with absolute value less than the threshold, set them as 0;

  if $\hat{\mu}$ *is the same as the one on the last loop* **then**

    break;

  end

end

Set the submatrix of $\hat{\Sigma}$ corresponding to non-zero components of $\hat{\mu}$ as the last estimated posterior variance in the preceding procedure, and set the other elements of $\hat{\Sigma}$ as 0.

used to estimate the weights $w_1, w_2, \ldots, w_M$ in the following regression problem:

$$\begin{bmatrix} x'_{k_1} \\ x'_{k_2} \\ \vdots \\ x'_{k_S} \end{bmatrix} = \begin{bmatrix} f_1(t_{k_1}, x_{k_1}) & f_2(t_{k_1}, x_{k_1}) & \cdots & f_M(t_{k_1}, x_{k_1}) \\ f_1(t_{k_2}, x_{k_2}) & f_2(t_{k_2}, x_{k_2}) & \cdots & f_M(t_{k_2}, x_{k_2}) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(t_{k_S}, x_{k_S}) & f_2(t_{k_S}, x_{k_S}) & \cdots & f_M(t_{k_S}, x_{k_S}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \epsilon, \tag{19}$$

where $f_1(t, x), f_2(t, x), \ldots, f_M(t, x)$ are the basis-functions and ϵ is the model error. This regression problem can be symbolized into the form as follows:

$$\eta = \Phi\mathbf{w} + \epsilon, \tag{20}$$

which is (9). By running Algorithm 3, we obtain a differential equation:

$$\frac{dx}{dt} = \hat{\mu}_1 f_1(t, x) + \hat{\mu}_2 f_2(t, x) + \cdots + \hat{\mu}_M f_M(t, x), \tag{21}$$

along with an error bar calculated by (18). After repeating this procedure L times with different randomly selected data points, the differential equation with the smallest error bar among all the loops is chosen as the final result of the whole subsampling algorithm. Our algorithm has two user-preset parameters: the subsampling size and the number of loops. Their impact on the accuracy of the final result is discussed in Section 4 with an example. Note that the above mechanism is described in the pattern (1) for convenience and simplification. It also applies to higher-order

Algorithm 4: Subsampling-based threshold sparse Bayesian regression: $\eta = \Phi\mathbf{w} + \epsilon$

Input: η, Φ, threshold, subsampling size S, the number of loops L

Output: $\hat{\mu}$, $\hat{\Sigma}$

Let $I_{N\times N}$ be the $N \times N$ identity matrix, where N is the number of rows in Φ;

for $r = 1$ *to* L **do**

  Let P_r be an $S \times N$ submatrix of $I_{N\times N}$ with randomly chosen rows;

  Use Algorithm 3 to solve the problem $P_r\eta = P_r\Phi\mathbf{w} + \epsilon$, getting $\hat{\mu}_r, \hat{\Sigma}_r$;

  Calculate [error bar]$_r$ using $\hat{\mu}_r, \hat{\Sigma}_r$ and (18);

end

Let $R = \arg\min_r\{[\text{error bar}]_r\}$;

Let $\hat{\mu} = \hat{\mu}_R$ and $\hat{\Sigma} = \hat{\Sigma}_R$.

differential equations and implicit differential equations, as long as the differential equations can be symbolized into the form (20). The SubTSBR procedure is summarized in Algorithm 4, where the for-loop can be coded parallelly.

3.3. Why it works

The numerical results in this paper show that our subsampling algorithm can improve the overall accuracy in the discovery of differential equations, and the error bar (18) is capable of evaluating the estimations. The given data $\left\{t_i, x_i, x_i'\right\}_{i=1}^N$ contain a part of data points of small noise and a part of data points of large noise. When a subset consisting of only data points of small noise is selected, our algorithm would estimate the weights well and indicate that this is the case by showing a small error bar (18). As we do not know which data points are of small noise and which data points are of large noise before the model is discovered, we select a subset from the data randomly, repeating multiple times. When it happens that the selected data points are of small noise, we would have a good estimation of the weights and at the same time recognize this case.

The numerical results also show that in order to attain optimal performance, the subsampling size should be moderate, neither too small nor too big, while the number of loops is as big as possible when computing time permits. As the correctness of the governing differential equations is crucial, computing time can be a secondary issue to consider. In practice, we can increase the number of loops gradually and stop the algorithm when the smallest error bar among all the loops drops below a certain preset value or the smallest error bar stops decreasing.

4. The challenge of large noise

4.1. Problem description

The noise in the data can hinder model-discovering algorithms from getting the correct results. Here we detail the mechanism of our algorithm, tune the parameters, and investigate the robustness against noise.

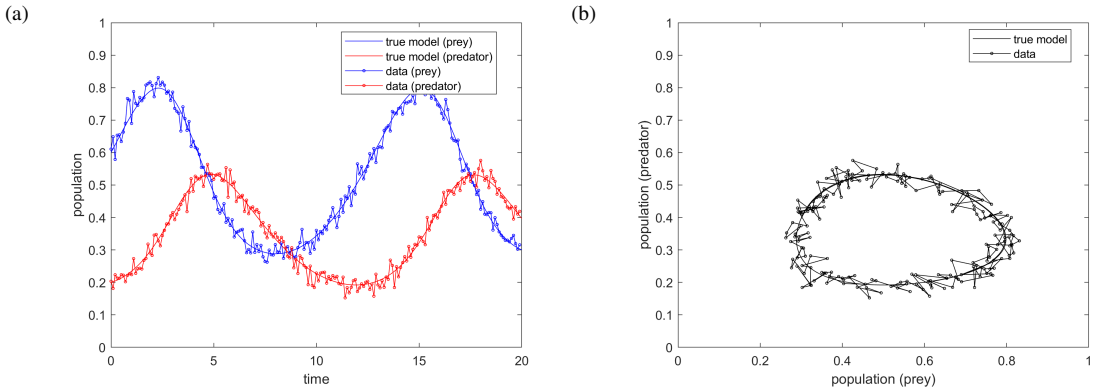

Figure 2: The true predator-prey model and the noisy data. (a) Population vs time. (b) Population(predator) vs population(prey).

4.2. Example: the predator-prey model with noise

The predator-prey model is a system of a pair of first-order nonlinear differential equations and is frequently used to describe the interaction between two species, one as a predator and the other as prey. The population change by time is as follows:

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{22}$$

$$\frac{dy}{dt} = \delta xy - \gamma y, \tag{23}$$

where x is the number of the prey, y is the number of the predator, and $\alpha, \beta, \delta, \gamma$ are positive real parameters describing the interaction of the two species. In this example, we fix the parameters as follows:

$$\frac{dx}{dt} = \frac{1}{2}x - \frac{3}{2}xy \tag{24}$$

$$\frac{dy}{dt} = xy - \frac{1}{2}y. \tag{25}$$

We assume that we do not know about the formula of the system (24) - (25), neither the terms nor the parameters, and try to discover the model using noisy data.

4.2.1. Data collection

We first generate 200 data points from the system (24) - (25), with the initial value $x_0 = 0.6$ and $y_0 = 0.2$, during time $t = 0$ to $t = 20$. Then independent and identically distributed white noise $\mathcal{N}(0, 0.02^2)$ is added to all the data x and all the data y. See Figure 2 for the true model and the noisy data.

4.2.2. Calculate numerical derivatives using total-variation derivative

Now we need to calculate the derivatives of the data to estimate the left-hand-side terms in (24) - (25). The noise in the data would be amplified greatly if the derivatives are calculated using numerical differentiation. See Figure 3a

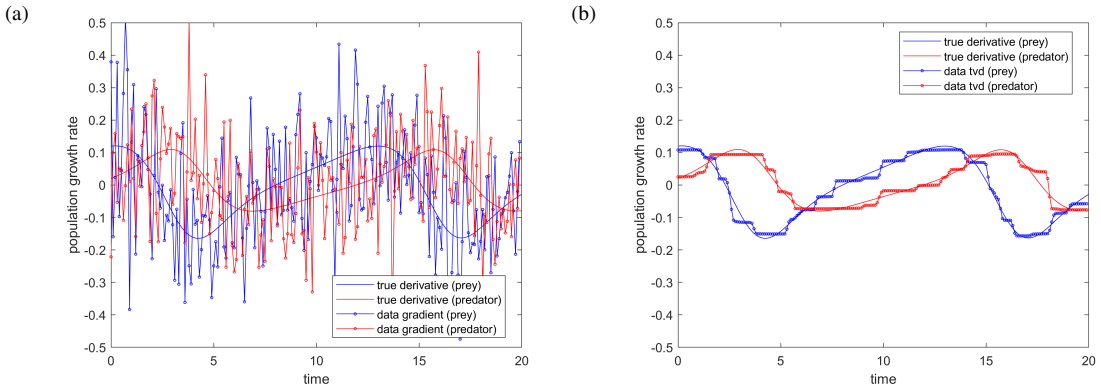

Figure 3: (a) The true derivatives of the data and the approximated derivatives using gradient. (b) The true derivatives of the data and the approximated derivatives using total-variation derivative (tvd).

for the approximated derivatives using gradient. Therefore, we use total-variation derivative [39, 40] instead. For a real function $f(t)$ on $[0, L]$, total-variation derivative computes the derivatives of f as the minimizer of the functional:

$$F(u) = \lambda \int_0^L |u'| + \frac{1}{2} \int_0^L |Au + f(0) - f|^2, \tag{26}$$

where $Au(t) = \int_0^t u$ is the operator of antidifferentiation and λ is a regularization parameter that controls the balance between the two terms. The numerical implementation is introduced in [40]. See Figure 3b for the approximated derivatives of $x(t)$ and $y(t)$ using total-variation derivative with $\lambda = 0.02$.

As we can see in Figure 3, the robust differentiation method is critical for getting high-quality derivatives. Besides total-variation derivative, many other methods are available for robust differentiation. Another approach is to use denoising techniques to reduce the noise in the data before taking derivatives. For example, a neural network is used to denoise data and approximate derivatives in [41]. Those methods may have better performance in practical use, depending on the situation. Here we do not use denoising techniques for the sake of demonstrating the robustness of our algorithm against noise. In practice, the denoised data may still contain noise. Our algorithm can be used following the denoising processes and may achieve good results.

4.2.3. Discover the model using different sparse algorithms

If sequential threshold least squares (Algorithm 1) is applied to discover the model (24) - (25), we get the following result:

$$\begin{aligned} \frac{dx}{dt} &= -1.122 + 3.539x + 5.896y - 4.243x^2 - 6.604xy - 13.612y^2 + 1.770x^3 + 3.844x^2y \\ &\quad +1.874xy^2 + 11.588y^3 \end{aligned} \tag{27}$$

$$\begin{aligned} \frac{dy}{dt} &= 0.230 - 1.613x - 0.427y + 2.636x^2 + 3.302xy - 1.781y^2 - 1.525x^3 - 1.080x^2y \\ &\quad -1.735xy^2 + 2.408y^3. \end{aligned} \tag{28}$$

If lasso (Algorithm 2) is used, we have:

$$\frac{dx}{dt} = 0.085 + 0.219x - 0.363y + 0.001xy - 0.043y^2 + 0.074x^3 - 0.265x^2y - 1.592xy^2 + 0.773y^3 \quad (29)$$

$$\frac{dy}{dt} = -0.037 + 0.154x - 0.349y + 0.100xy - 0.087y^2 - 0.051x^3 + 0.220x^2y + 0.953xy^2 - 0.030y^3. \quad (30)$$

If TSBR (Algorithm 3) is used, we get:

$$\frac{dx}{dt} = 0.230(0.018)x + 0.443(0.104)y - 2.448(0.434)y^2 - 1.929(0.130)xy^2 + 3.132(0.442)y^3 \quad (31)$$

$$\frac{dy}{dt} = -0.641(0.033)y + 1.609(0.133)xy - 0.578(0.122)x^2y, \quad (32)$$

with error bars 0.1174 and 0.0538, where the numbers in front of each term read as "mean (standard deviation)" of the corresponding weights. Note that sequential threshold least squares, lasso, and TSBR use all 200 data points at the same time to discover the model respectively.

In contrast, if SubTSBR (Algorithm 4) is applied to discover the model, we have:

$$\frac{dx}{dt} = 0.491(0.015)x - 1.458(0.041)xy \quad (33)$$

$$\frac{dy}{dt} = -0.487(0.017)y + 0.971(0.031)xy, \quad (34)$$

with error bars 0.0018 and 0.0022. The approximated weights in (33) - (34) are slightly smaller in absolute value than the true weights in (24) - (25) because when we calculate the numerical derivatives, total-variation derivative (26) smooths out some variation in the derivatives. This defect does not have much impact on the result and can be addressed by using different methods to calculate the derivatives or collecting the derivatives along with the data directly. Here the subsampling size is 60 and the number of loops is 30. All methods in this example have the threshold set at 0.1 and use monomials generated by $\{1, x, y\}$ up to degree three as basis-functions (10 terms in total). The result of SubTSBR (33) - (34) approximates the true system (24) - (25) significantly better than the other sparse algorithms. Although the data contain a considerable amount of noise, SubTSBR successfully finds the exact terms in the true system and accurately estimates the parameters. See Figure 4 for the final selected data points in SubTSBR. See Figure 5 for the dynamics calculated by the systems derived from each algorithm. Note that since the data are collected from $t = 0$ to $t = 20$, Figure 5a shows the approximation and Figure 5b shows the prediction.

4.2.4. Basis-selection success rate vs subsampling size and the number of loops

In SubTSBR (Algorithm 4), we have two parameters to set, one of which is the subsampling size and the other is the number of loops. Here we first investigate the impact on the basis-selection success rate by different subsampling sizes. In Figure 6a, each curve is drawn by fixing the number of loops. Then given each subsampling size, SubTSBR is applied to the data set collected above. This method is performed 1000 times for each fixed number of loops and subsampling size. Then the percentage of successful identification of the exact terms in the system (24) - (25) is calculated and plotted. In the discovery process, the most difficult part is to identify the exact terms in the system. If the exact terms are successfully identified, it is usually easy to estimate the weights.

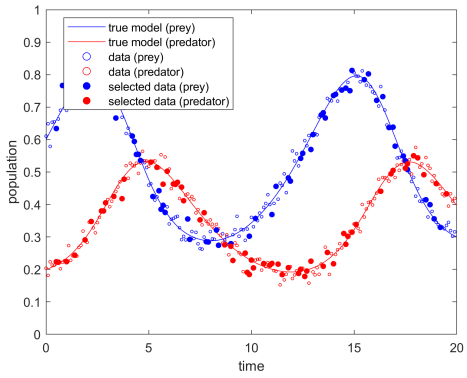

Figure 4: The final selected data points in SubTSBR.

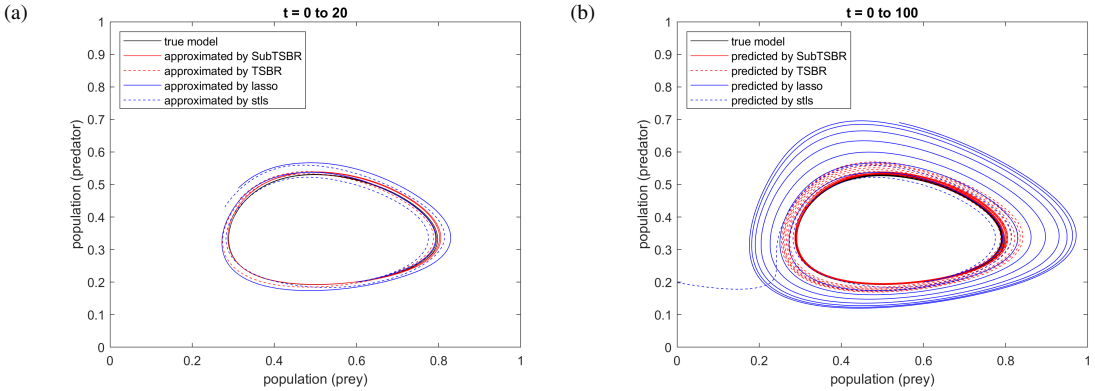

Figure 5: (a) Approximated dynamics by SubTSBR, TSBR, lasso, and sequential threshold least squares from $t = 0$ to $t = 20$. (b) Predicted dynamics from $t = 0$ to $t = 100$.

Figure 6a shows that for each fixed number of loops, basis-selection success rate goes up and then down when the subsampling size increases. When the subsampling size equals 200, all the data points are used and SubTSBR is equivalent to TSBR (Algorithm 3). In this case the true terms cannot be identified. In addition, for each chosen number of loops there is an optimal subsampling size, and the optimal subsampling size increases by the number of loops.

Now we investigate the impact on the basis-selection success rate by different numbers of loops. In Figure 6b, each curve is drawn by fixing the subsampling size. Then given each number of loops, SubTSBR is applied to the data set to discover the model. The discovery is done 1000 times for each fixed subsampling size and number of loops. Then the percentage of successful identification of the exact terms in the system (24) - (25) is calculated and plotted.

Figure 6b shows that for each fixed subsampling size, basis-selection success rate keeps going up when the number of loops increases. In addition, the larger the subsampling size is, more loops are needed for the basis-selection success rate to reach a certain level. This is because our data set is polluted by Gaussian noise and naturally contains some data points of large noise and some of small noise. As the subsampling size gets bigger, it is less likely for each of the

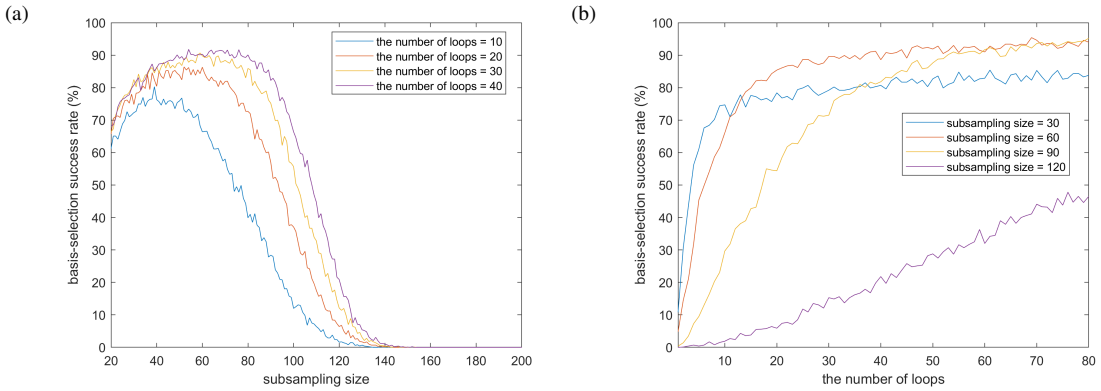

Figure 6: The total number of data points is 200. (a) Basis-selection success rate vs subsampling size, with different numbers of loops. (b) Basis-selection success rate vs the number of loops, with different subsampling sizes.

random subsets of data to exclude the data points of large noise. When more loops are used, the likelihood for one of the loops to exclude the data points of large noise increases. As long as one of the loops excludes the data points of large noise, this loop may successfully select the true basis functions and have the smallest error bar. If it happens, the final result would come from this loop and SubTSBR selects the true basis functions successfully in this one of 1000 runs. This explains why the curves of smaller subsampling size go up faster.

On the other hand, when more data points are used, the noise inside the subsamples gets smoothed out easier in the regression (20). If all the included data points in the subsample are of small noise, then the result from a larger subsampling size may be a better one. This explains why the saturated basis-selection success rate is higher for larger subsampling size within a certain range. In conclusion, there is tradeoff for larger subsampling size—it is more difficult to include only data points of small noise while it is easier to smooth out the noise inside the subsample.

4.2.5. Adjusted error bar and auto-fitting of subsampling size

In real-world applications, the case is usually more complicated and the problem of setting the best subsampling size is subtle. Since the true equations are unknown, the basis-selection success rate cannot be calculated. Therefore, drawing a curve like the ones in Figure 6 to find the best subsampling size is not available. Here we define an adjusted error bar as an indicator for the quality of the approximated model and fit the subsampling size automatically.

The error bar defined in (18) depends on the number of data points used and the quality of the approximated model. When the subsampling size is within a reasonable range, the quality of the approximated model does not differ too much, so the error bar is dominated by the number of data points (see Figure 7). By the formula (18), the error bar is negatively correlated with the number of data points. If we want to compare results among different subsampling sizes, we need to adjust the error bar such that it is independent of the subsampling size. Inspired by the rate of convergence of Monte Carlo method, we give the following empirical formula:

$$\text{adjusted error bar} = [\text{error bar}] \times [\text{subsampling size}]^{0.5}. \tag{35}$$

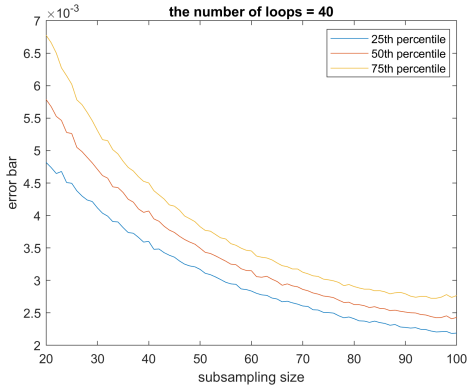

Figure 7: Error bar vs subsampling size. The total number of data points is 200.

In Figure 8, we can see that the adjusted error bar almost only depends on the quality of the approximated model. Note that we have the best models when the subsampling size is between 60 and 80 (see Figure 6a). Meanwhile, the percentile curves in Figure 8 indicate that the optimal subsampling sizes are in about that range. This observation confirms that the adjusted error bar depends on the quality of the discovered model. The better the model is, the smaller the adjusted error bar is.

Now, we do not have to set the subsampling size at the beginning but we may try different subsampling sizes to discover the model, and the best result can be selected from all the results with different subsampling sizes. (In Figure 7 and Figure 8, for each fixed number of loops and subsampling size, we run SubTSBR 1000 times and discover 1000 models with their error bars or adjusted error bars. Then the 25th, 50th, and 75th percentiles of these error bars or adjusted error bars are plotted.)

4.2.6. A new data set with larger noise

Here we use a new data set with white noise $\mathcal{N}(0, 0.05^2)$ to discover the predator-prey model (24) - (25). All other settings remain the same. Corresponding to Figure 2, the noisy data are presented in Figure 9a and Figure 9b. Corresponding to Figure 6, the basis-selection success rates are presented in Figure 9c and Figure 9d. With larger noise, the chance of successfully picking out the true terms from the basis-functions is lower. As a result, more loops would be needed in this case. Figure 9d only demonstrates the results with the numbers of loops up to 80, but many more loops may be used in practical problems. As for computation time, it takes about 15 seconds to discover the dynamical system in this example with subsampling size 60 and the number of loops 1000 on one core of the CPU Intel i7-6700HQ (coded in MATLAB 2018a).

4.2.7. Another new data set with smaller noise

Now we use another new data set with white noise $\mathcal{N}(0, 0.005^2)$ to discover the predator-prey model (24) - (25). All other settings remain the same. The numerical results are presented in Figure 10. In this case, we have a large portion of data points of small noise to use, so a 100% basis-selection success rate can be reached with just 20 loops.

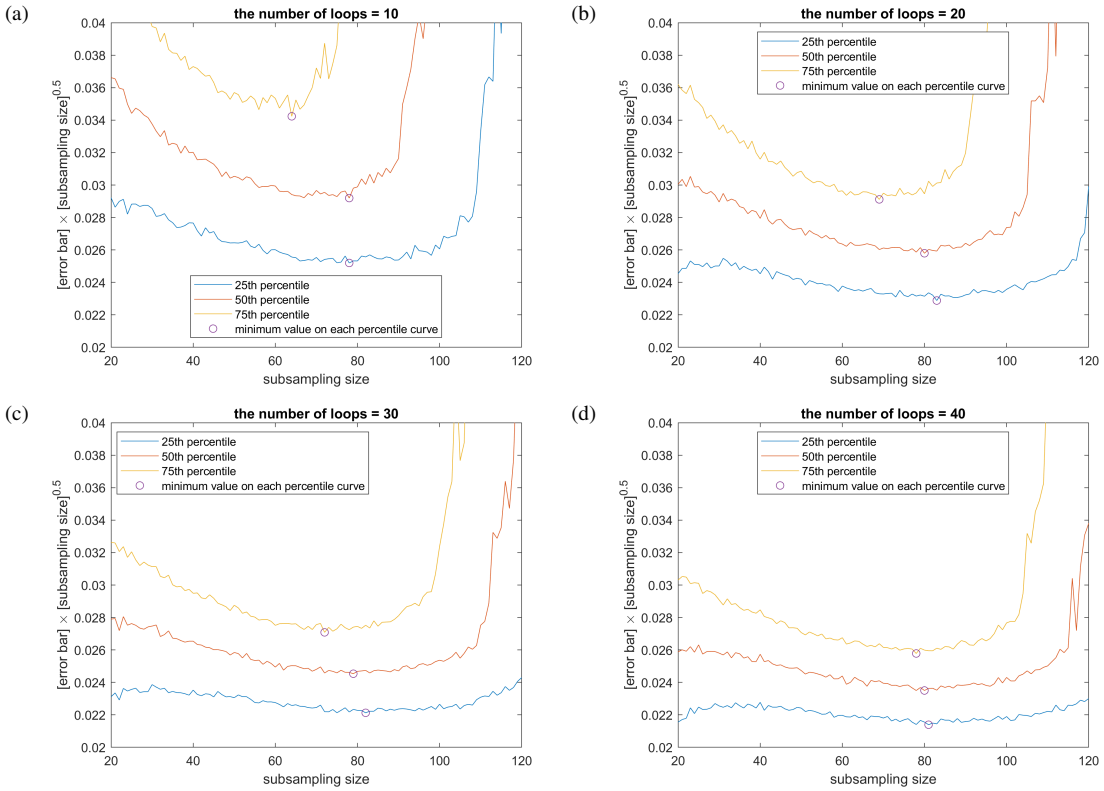

Figure 8: Adjusted error bar vs subsampling size, with different numbers of loops. The total number of data points is 200.

Also, the noise inside the subsamples is small, so we do not need a large subsampling size to smooth out the noise inside the subsamples. The basis-selection success rate is very high even with a subsampling size of 20.

5. The challenge of outliers

5.1. Problem description

The outliers in the data can cause serious problems in the estimations and should be removed. The subsampling procedure in Algorithm 4 is designed to be resistant to outliers. Here we calculate how many loops are needed to exclude the outliers from a data set. Then we apply our theory to an example.

5.2. The number of loops needed to remove outliers

Suppose we are given N data points, a portion p of which are outliers. Suppose the subsampling size is S. We try to determine the number of loops L such that with confidence q, at least one of the L randomly selected subsets of the data does not contain any outlier.

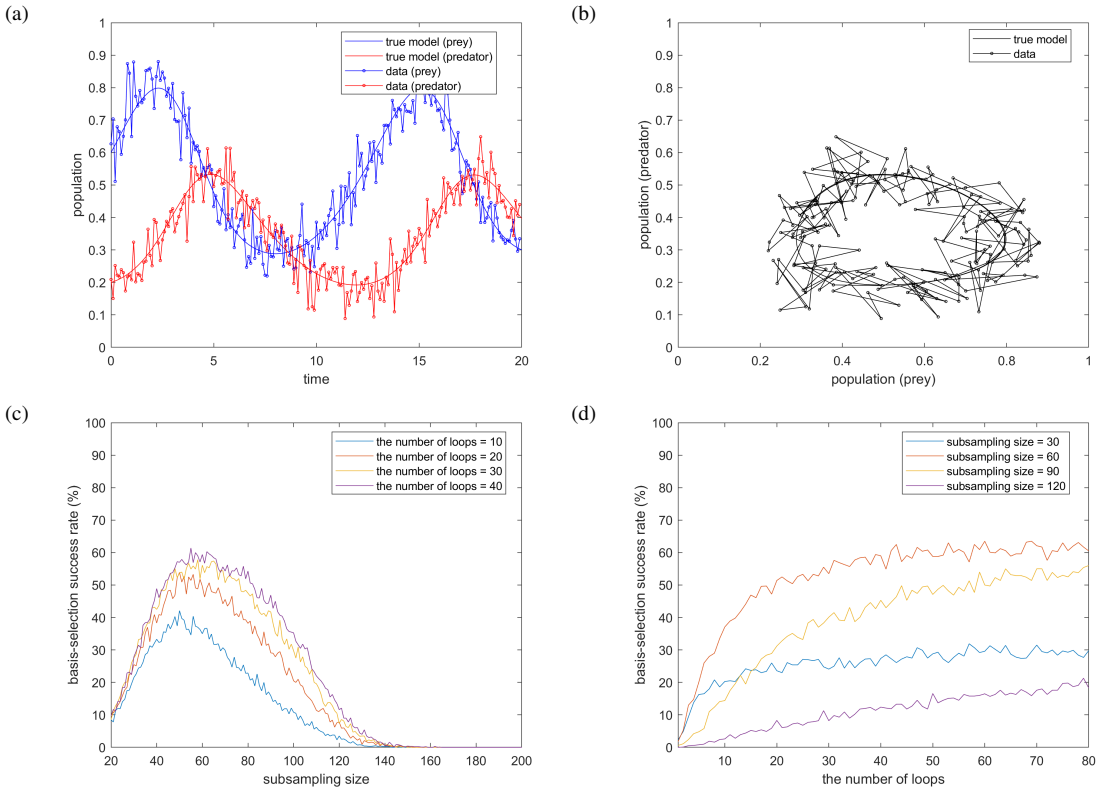

Figure 9: (a) and (b) The noisy data with larger noise than the ones in Figure 2. (c) and (d) The basis-selection success rates with larger noise than the ones in Figure 6.

The number of outliers is pN and the number of "good" data points is $(1-p)N$. For a random subset of size S not containing any outlier, the probability is

$$(1-p)\frac{(1-p)N-1}{N-1}\frac{(1-p)N-2}{N-2}\cdots\frac{(1-p)N-S+1}{N-S+1}. \tag{36}$$

When $p \ll 1$ and $S \ll N$, the probability is approximately

$$(1-p)^S. \tag{37}$$

For a random subset of size S containing at least one outlier, the probability is

$$1-(1-p)^S. \tag{38}$$

For L random subsets of size S each containing at least one outlier, the probability is

$$[1-(1-p)^S]^L. \tag{39}$$

For L random subsets of size S at least one subset not containing any outlier, the probability is

$$1-[1-(1-p)^S]^L. \tag{40}$$

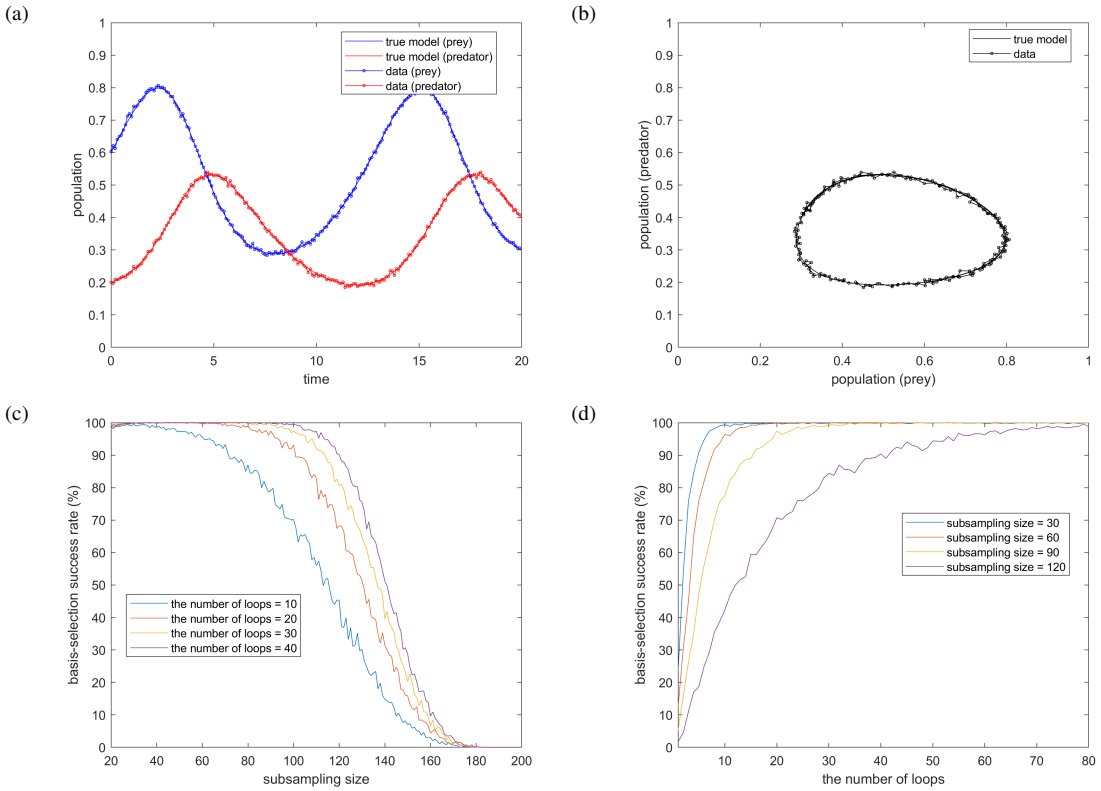

Figure 10: (a) and (b) The noisy data with smaller noise than the ones in Figure 2. (c) and (d) The basis-selection success rates with smaller noise than the ones in Figure 6.

Set the probability greater than or equal to q:

$$1 - [1 - (1 - p)^S]^L \geq q. \tag{41}$$

Then we have

$$L \geq \frac{\log(1 - q)}{\log(1 - (1 - p)^S)}. \tag{42}$$

See Figure 11 for the relationship between the subsampling size S, the portion of outliers p, and the minimum number of loops L when $q = 0.99$.

5.3. Example: shallow water equations with outliers

Consider the following 2-D conservative form of shallow water equations:

$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = 0 \tag{43}$$

$$\frac{\partial (hu)}{\partial t} + \frac{\partial (hu^2 + (1/2)gh^2)}{\partial x} + \frac{\partial (huv)}{\partial y} = 0 \tag{44}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial (hv^2 + (1/2)gh^2)}{\partial y} = 0 \tag{45}$$

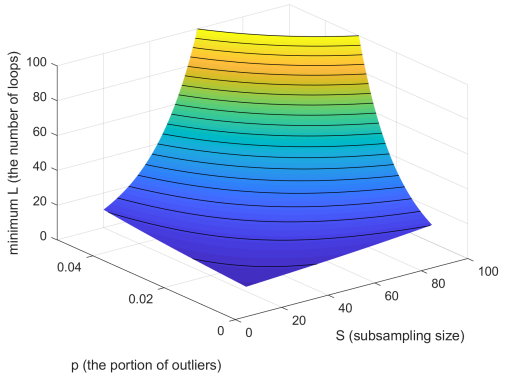

Figure 11: The relationship between the subsampling size S, the portion of outliers p, and the minimum number of loops L when $q = 0.99$ in (42). The black curves are the contours with fixed minimum L.

on $(x, y) \in [0, 39] \times [0, 39]$ and $t \in [0, \infty)$, with reflective boundary conditions and a water drop initiating gravity waves, where h is the total fluid column height, (u, v) is the fluid's horizontal flow velocity averaged across the vertical column, and $g = 9.8\,\mathrm{m\,s^{-2}}$ is the gravitational acceleration. The first equation can be derived from mass conservation, the last two from momentum conservation. Here, we have made the assumption that the fluid density is a constant.

5.3.1. Data collection

We generate the numerical solution to the shallow water equations using Lax-Wendroff finite difference method with $\Delta x = \Delta y = 1$ and $\Delta t = 0.02$. See Figure 12. The data are collected at $t = 36$ and the partial derivatives $\partial h/\partial x$, $\partial u/\partial x$, $\partial v/\partial x$, $\partial h/\partial y$, $\partial u/\partial y$, and $\partial v/\partial y$ are calculated by the three-point central-difference formula. The calculation of the partial derivatives $\partial h/\partial t$, $\partial u/\partial t$, and $\partial v/\partial t$ uses the points from two adjacent time frames. Assume that only the central 36×36 part of the data is made accessible and 2% of them are added on the values of h, u, and v by independent and identically distributed random error $\sim \mathcal{U}(0.5, 1)$, the uniform distribution on $[0.5, 1]$. There are $36 \times 36 = 1296$ accessible data points. See Figure 12d. Thus, the accessible data to discover the model are:

$$\left\{h_i, u_i, v_i, \left(\frac{\partial h}{\partial t}\right)_i, \left(\frac{\partial u}{\partial t}\right)_i, \left(\frac{\partial v}{\partial t}\right)_i, \left(\frac{\partial h}{\partial x}\right)_i, \left(\frac{\partial u}{\partial x}\right)_i, \left(\frac{\partial v}{\partial x}\right)_i, \left(\frac{\partial h}{\partial y}\right)_i, \left(\frac{\partial u}{\partial y}\right)_i, \left(\frac{\partial v}{\partial y}\right)_i\right\}_{i=1}^{1296}, \tag{46}$$

2% of which are outliers.

5.3.2. Discovery of the model

We apply the dimensional analysis method introduced in [1] to construct the basis-functions of the same dimension as $\partial h/\partial t$ $(\mathrm{m\,s^{-1}})$ to discover it: $h(\partial u/\partial x)$, $h(\partial v/\partial x)$, $h(\partial u/\partial y)$, $h(\partial v/\partial y)$, u, $u(\partial h/\partial x)$, $u(\partial h/\partial y)$, v, $v(\partial h/\partial x)$, $v(\partial h/\partial y)$, $(\partial h/\partial t)(\partial h/\partial x)$, $(\partial h/\partial t)(\partial h/\partial y)$. Similarly, we can construct the basis-functions of the same dimension as $\partial u/\partial t$ and $\partial v/\partial t$ $(\mathrm{m\,s^{-2}})$ to discover them: $u(\partial u/\partial x)$, $u(\partial v/\partial x)$, $u(\partial u/\partial y)$, $u(\partial v/\partial y)$, $v(\partial u/\partial x)$, $v(\partial v/\partial x)$, $v(\partial u/\partial y)$, $v(\partial v/\partial y)$, $(\partial h/\partial t)(\partial u/\partial x)$, $(\partial h/\partial t)(\partial v/\partial x)$, $(\partial h/\partial t)(\partial u/\partial y)$, $(\partial h/\partial t)(\partial v/\partial y)$, $g(\partial h/\partial x)$, $g(\partial h/\partial y)$. The threshold for all algorithms

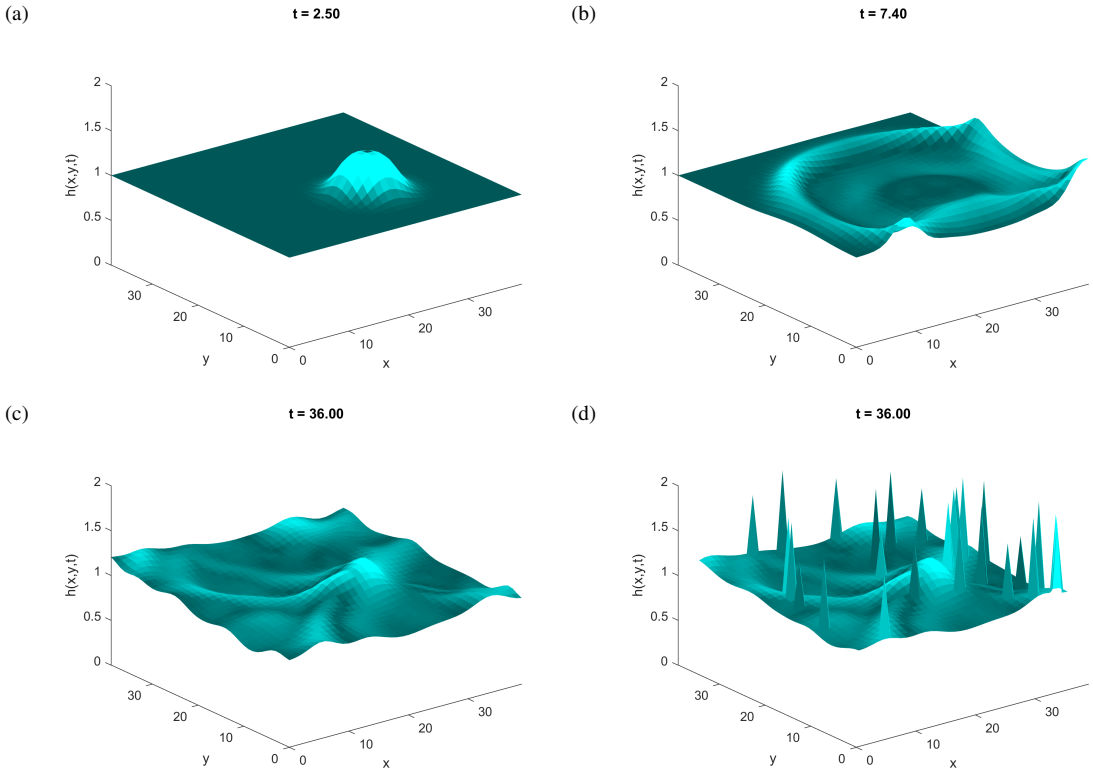

Figure 12: Surface plot displays height colored by momentum. (a) A water drop falls into the pool. (b) The gravity waves are traveling and being reflected by the boundary. (c) The water surface state when the data are collected. (d) The accessible data are corrupted by outliers.

is set at 0.1. If sequential threshold least squares (Algorithm 1) is applied, we get the following result:

$$\frac{\partial h}{\partial t} = -0.990h\frac{\partial u}{\partial x} - 0.988h\frac{\partial v}{\partial y} - 0.710u\frac{\partial h}{\partial x} + 0.395u\frac{\partial h}{\partial y} + 0.409v\frac{\partial h}{\partial x} - 0.709v\frac{\partial h}{\partial y} + 0.267\frac{\partial h}{\partial t}\frac{\partial h}{\partial y} \tag{47}$$

$$\frac{\partial u}{\partial t} = -0.748u\frac{\partial u}{\partial x} - 1.931u\frac{\partial v}{\partial x} + 2.173u\frac{\partial u}{\partial y} + 0.269v\frac{\partial u}{\partial x} + 1.506v\frac{\partial v}{\partial x} - 2.114v\frac{\partial u}{\partial y} + 1.907\frac{\partial h}{\partial t}\frac{\partial v}{\partial x} - 1.956\frac{\partial h}{\partial t}\frac{\partial u}{\partial y} - 1.011g\frac{\partial h}{\partial x} \tag{48}$$

$$\frac{\partial v}{\partial t} = -0.480u\frac{\partial v}{\partial x} - 0.155u\frac{\partial u}{\partial y} + 0.118u\frac{\partial v}{\partial y} + 1.093v\frac{\partial v}{\partial x} - 0.743v\frac{\partial u}{\partial y} - 0.723v\frac{\partial v}{\partial y} - 0.195\frac{\partial h}{\partial t}\frac{\partial u}{\partial x} + 3.128\frac{\partial h}{\partial t}\frac{\partial v}{\partial x} - 3.018\frac{\partial h}{\partial t}\frac{\partial u}{\partial y} + 0.195\frac{\partial h}{\partial t}\frac{\partial v}{\partial y} - 1.007g\frac{\partial h}{\partial y}. \tag{49}$$

If lasso (Algorithm 2) is used, we have:

$$\begin{aligned}\frac{\partial h}{\partial t} &= -0.988h\frac{\partial u}{\partial x} + 0.001h\frac{\partial v}{\partial x} - 0.987h\frac{\partial v}{\partial y} + 0.004u - 0.733u\frac{\partial h}{\partial x} + 0.358u\frac{\partial h}{\partial y} + 0.409v\frac{\partial h}{\partial x} \\ &\quad -0.710v\frac{\partial h}{\partial y} + 0.180\frac{\partial h}{\partial t}\frac{\partial h}{\partial y} \qquad (50)\\ \frac{\partial u}{\partial t} &= -0.587u\frac{\partial u}{\partial x} + 0.049u\frac{\partial v}{\partial x} + 0.017u\frac{\partial u}{\partial y} + 0.112v\frac{\partial u}{\partial x} - 0.466v\frac{\partial v}{\partial x} - 0.005v\frac{\partial u}{\partial y} - 1.011g\frac{\partial h}{\partial x} \qquad (51)\\ \frac{\partial v}{\partial t} &= -0.469u\frac{\partial v}{\partial x} + 0.049u\frac{\partial v}{\partial y} + 0.165v\frac{\partial v}{\partial x} - 0.621v\frac{\partial v}{\partial y} - 1.006g\frac{\partial h}{\partial y}. \qquad (52)\end{aligned}$$

If TSBR (Algorithm 3) is used, we get:

$$\begin{aligned}\frac{\partial h}{\partial t} &= -0.990h\frac{\partial u}{\partial x} - 0.988h\frac{\partial v}{\partial y} - 0.705u\frac{\partial h}{\partial x} + 0.375u\frac{\partial h}{\partial y} + 0.402v\frac{\partial h}{\partial x} - 0.696v\frac{\partial h}{\partial y} + 0.207\frac{\partial h}{\partial t}\frac{\partial h}{\partial y} \qquad (53)\\ \frac{\partial h}{\partial t} &= -0.750u\frac{\partial u}{\partial x} - 2.042u\frac{\partial v}{\partial x} + 2.283u\frac{\partial u}{\partial y} + 0.265v\frac{\partial u}{\partial x} + 1.557v\frac{\partial v}{\partial x} - 2.172v\frac{\partial u}{\partial y} - 1.011g\frac{\partial h}{\partial x} \qquad (54)\\ \frac{\partial v}{\partial t} &= -0.635u\frac{\partial v}{\partial x} + 0.118u\frac{\partial v}{\partial y} + 1.106v\frac{\partial v}{\partial x} - 0.767v\frac{\partial u}{\partial y} - 0.725v\frac{\partial v}{\partial y} - 0.189\frac{\partial h}{\partial t}\frac{\partial u}{\partial x} + 0.157\frac{\partial h}{\partial t}\frac{\partial v}{\partial x} \\ &\quad +0.174\frac{\partial h}{\partial t}\frac{\partial v}{\partial y} - 1.007g\frac{\partial h}{\partial y}. \qquad (55)\end{aligned}$$

In contrast, if SubTSBR (Algorithm 4) is applied to discover the model, we have:

$$\begin{aligned}\frac{\partial h}{\partial t} &= -1.001h\frac{\partial u}{\partial x} - 0.996h\frac{\partial v}{\partial y} - 0.932u\frac{\partial h}{\partial x} - 0.986v\frac{\partial h}{\partial y} \qquad (56)\\ \frac{\partial u}{\partial t} &= -0.939u\frac{\partial u}{\partial x} - 1.051v\frac{\partial u}{\partial y} - 0.998g\frac{\partial h}{\partial x} \qquad (57)\\ \frac{\partial v}{\partial t} &= -1.023u\frac{\partial v}{\partial x} - 0.980v\frac{\partial v}{\partial y} - 1.002g\frac{\partial h}{\partial y}, \qquad (58)\end{aligned}$$

where the subsampling size is 162, which is one eighth of all the accessible data, and the number of loops is 120, which is the minimum L calculated by (42) with confidence 0.99. Note that the system of equations (43) - (45) is equivalent to

$$\begin{aligned}\frac{\partial h}{\partial t} + h\frac{\partial u}{\partial x} + h\frac{\partial v}{\partial y} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} &= 0 \qquad (59)\\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial h}{\partial x} &= 0 \qquad (60)\\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial h}{\partial y} &= 0. \qquad (61)\end{aligned}$$

Therefore, SubTSBR gives the best result.

6. The challenge of data integration

6.1. Problem description

Data integration is the process of combining the data from different sources into meaningful and valuable information. In many cases, collecting enough data from a single experiment is difficult to achieve due to limited resources.

For instance, the experiments may need to be done at multiple different time or different locations. When the data are from multiple experiments, although they are generated by the same model, the initial conditions or boundary conditions used to generate them may be different or even unmeasurable. Traditional interpolation and regression methods are not applicable to these cases since these methods require all the data to be from the same curve. A significant advantage of our method of discovering governing differential equations is that the data are allowed to be from different experiments, as long as the model behind them is the same.

On top of that, if the initial condition and boundary condition can be formulated into algebraic equations and we are given data at the initial state and boundary, we may symbolize the initial condition and boundary condition into the form (7) and discover them using our method. The only difference in this case is that the algebraic equations do not have any derivative term.

Finally, with the discovered differential equation, initial condition, and boundary condition, we may reconstruct the solutions to the model and make predictions.

6.2. Example: heat diffusion with random initial and boundary conditions

Consider the following 1-D heat diffusion equation:

$$\frac{\partial u}{\partial t} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2} \tag{62}$$

on $x \in [0, 5]$ and $t \in [0, \infty)$ with random initial condition:

$$u(x, 0) = -\frac{1}{2}\xi_1 x(x - 5) \tag{63}$$

and random boundary condition:

$$u(0, t) = \xi_2 \sin(2t) - \xi_3^2 \cos t + \xi_3^2 \tag{64}$$

$$u(5, t) = \xi_2\xi_3 \sin t - \xi_3 \sin(t + \frac{\pi}{4}) + \frac{\xi_3\sqrt{2}}{2}, \tag{65}$$

where ξ_1, ξ_2, ξ_3 are independent random variables:

- $\xi_1 \sim \mathcal{U}(0, 1)$, the uniform distribution on $[0, 1]$;
- $\xi_2 \sim \mathcal{U}(0, 1)$, the uniform distribution on $[0, 1]$;
- $\xi_3 \sim \mathcal{N}(0, 0.5^2)$, the normal distribution with mean 0 and standard deviation 0.5.

When $\xi_1 = \xi_2 = \xi_3 = 0.5$, the solution of the heat diffusion equation is displayed in Figure 13.

6.2.1. Data collection and discovery of the model

We collect data at the grid points on $x \in [0, 5]$ and $t \in [0, 5]$ illustrated in Figure 13b from 20 solutions generated by 20 sets of independent random variables $\{\xi_1, \xi_2, \xi_3\}$. There are $11 \times 11 \times 20 = 2420$ data points. Then we calculate

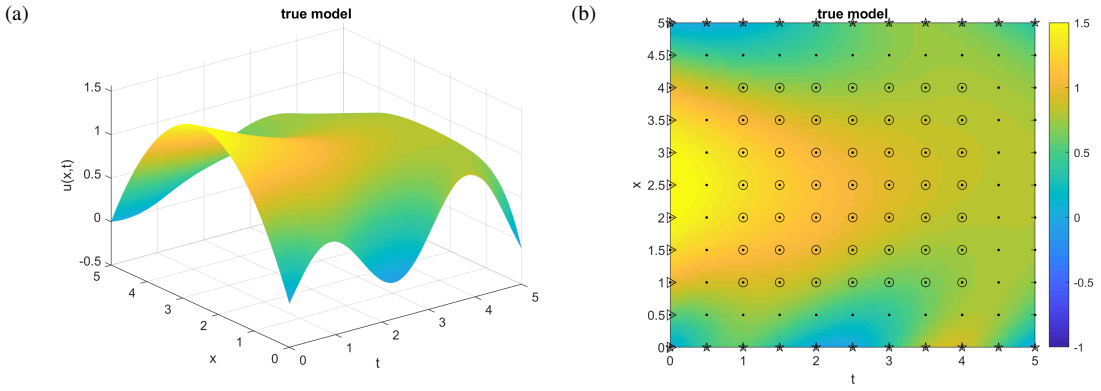

Figure 13: Solution of the 1-D heat diffusion equation (62) with $\xi_1 = \xi_2 = \xi_3 = 0.5$. (a) Surface plot. (b) Image plot with scaled colors. Grid points: [•] all data points to collect from the model [◯] data points used to discover PDE [⋆] data points used to discover boundary condition [▷] data points used to discover initial condition.

derivatives using the five-point central-difference formula. Note that the derivatives are only calculated at the interior grid points (marked as [◯] in Figure 13b). Next we discover the PDE using our algorithm SubTSBR with subsampling size 245, which is one fourth of all interior grid points, and 30 loops. The basis-functions are monomials generated by $\{1, x, t, u, \partial u/\partial x, \partial^2 u/\partial x^2\}$ up to degree 3. There are 56 terms. The result is:

$$\frac{\partial u}{\partial t} = 0.498\,\frac{\partial^2 u}{\partial x^2}. \tag{66}$$

After that, we discover the boundary condition using SubTSBR with subsampling size 55, which is one fourth of all lower boundary points or upper boundary points, and 30 loops. The basis-functions are monomials generated by $\{1, \xi_2, \xi_3, t, \sin t, \cos t\}$ up to degree 3. There are 56 terms. The result is:

$$u(0,t) = 1.000\,\xi_3^2 + 2.000\,\xi_2 \sin t \cos t - 1.000\,\xi_3^2 \cos t \tag{67}$$

$$u(5,t) = 0.707\,\xi_3 - 0.707\,\xi_3 \sin t - 0.707\,\xi_3 \cos t + 1.000\,\xi_2\xi_3 \sin t. \tag{68}$$

Next we discover the initial condition using SubTSBR with subsampling size 55, which is one fourth of all initial points, and 30 loops. The basis-functions are monomials generated by $\{1, \xi_1, x, \sin x, \cos x\}$ up to degree 3. There are 35 terms. The result is:

$$u(x,0) = 2.500\,\xi_1 x - 0.500\,\xi_1 x^2. \tag{69}$$

6.2.2. Prediction

Now we predict the solution when $\xi_1 = \xi_2 = \xi_3 = 0.5$. Fix the ξ_1, ξ_2, ξ_3 values in (67) - (69) and solve the PDE (66) on $x \in [0, 5]$ and $t \in [0, 15]$. We get the solution of the heat diffusion equation predicted by discovering PDE in Figure 14c and 14d. The true model is solved by (62) and fixing $\xi_1 = \xi_2 = \xi_3 = 0.5$ in (63) - (65). Its solution is displayed in Figure 14a and 14b. As a comparison, the solution predicted by least-squares regression is displayed in Figure 14e and 14f, where $u(x, t, \xi_1, \xi_2, \xi_3)$ is fitted by a linear combination of monomials generated

t	MSE by discovering PDE	MSE by least-squares regression
0	0.000000	0.003998
1.5	0.000065	0.001693
3.0	0.000094	0.001908
4.5	0.000076	0.002779
6.0	0.000107	0.021333
7.5	0.000105	0.641847
9.0	0.000094	3.882689
10.5	0.000055	10.631797
12.0	0.000058	18.003597
13.5	0.000062	24.832532
15.0	0.000048	36.988403

Table 1: Mean squared error (MSE) of the predicted solutions in Figure 14 by discovering PDE and least-squares regression at different time t.

by $\{1, x, \sin x, \cos x, t, \sin t, \cos t, \xi_1, \xi_2, \xi_3\}$ up to degree 3. There are 220 terms. Here we use the same data set for discovering PDE to do the regression. The solution is drawn by fixing $\xi_1 = \xi_2 = \xi_3 = 0.5$: $u(x, t, 0.5, 0.5, 0.5)$. Figure 14 shows that both of discovering PDE and least-squares regression approximate the true model well on $t \in [0, 5]$, but discovering PDE predicts the true model much better on $t \in [5, 15]$. Least-squares regression starts to fail when $t \geq 6$ because the data are collected within $t \in [0, 5]$ and least-squares regression is a kind of interpolation method. It does not work well outside the region with known data. In contrast, discovering PDE is a kind of extrapolation method, which works beyond the original observation range. See Figure 15 for the solutions at different time t and Table 1 for the mean squared error (MSE) at different time t.

Note that we do not use any data from $\xi_1 = \xi_2 = \xi_3 = 0.5$ to discover the PDE, initial condition, or boundary condition. Instead, we use the data from 20 randomly generated sets of independent random variables $\{\xi_1, \xi_2, \xi_3\}$. Predictions at other ξ_1, ξ_2, ξ_3 values can be derived in the same way by fixing the ξ_1, ξ_2, ξ_3 values in (67) - (69) and solving (66). The prediction at $\{\xi_1, \xi_2, \xi_3\}$ does not need any data from the same $\{\xi_1, \xi_2, \xi_3\}$.

Also note that we do not need any information about ξ_1, ξ_2, ξ_3 to discover the PDE. If the values of ξ_1, ξ_2, ξ_3 are unknown, we can still discover the PDE but unable to discover the initial condition or boundary condition in this example. If given a new initial condition and boundary condition, we can still make prediction using the discovered PDE, while we are not able to do so using regression methods.

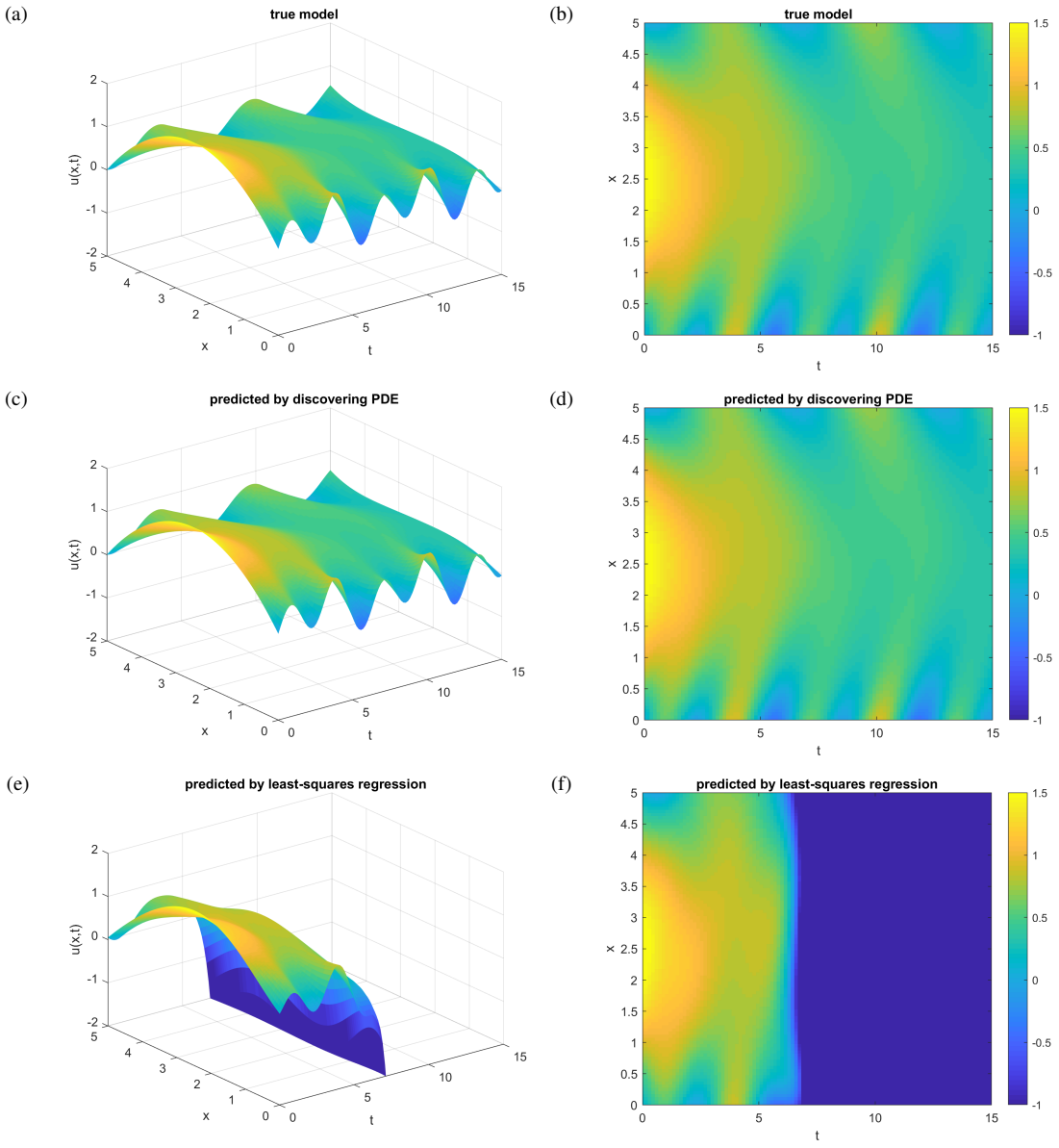

Figure 14: (a) (b) The true model solved by (62) with initial and boundary conditions (63) - (65). (c) (d) The solution predicted by discovering PDE, solved by (66) with initial and boundary conditions (67) - (69). (e) (f) The solution predicted by least-squares regression. All with $\xi_1 = \xi_2 = \xi_3 = 0.5$.

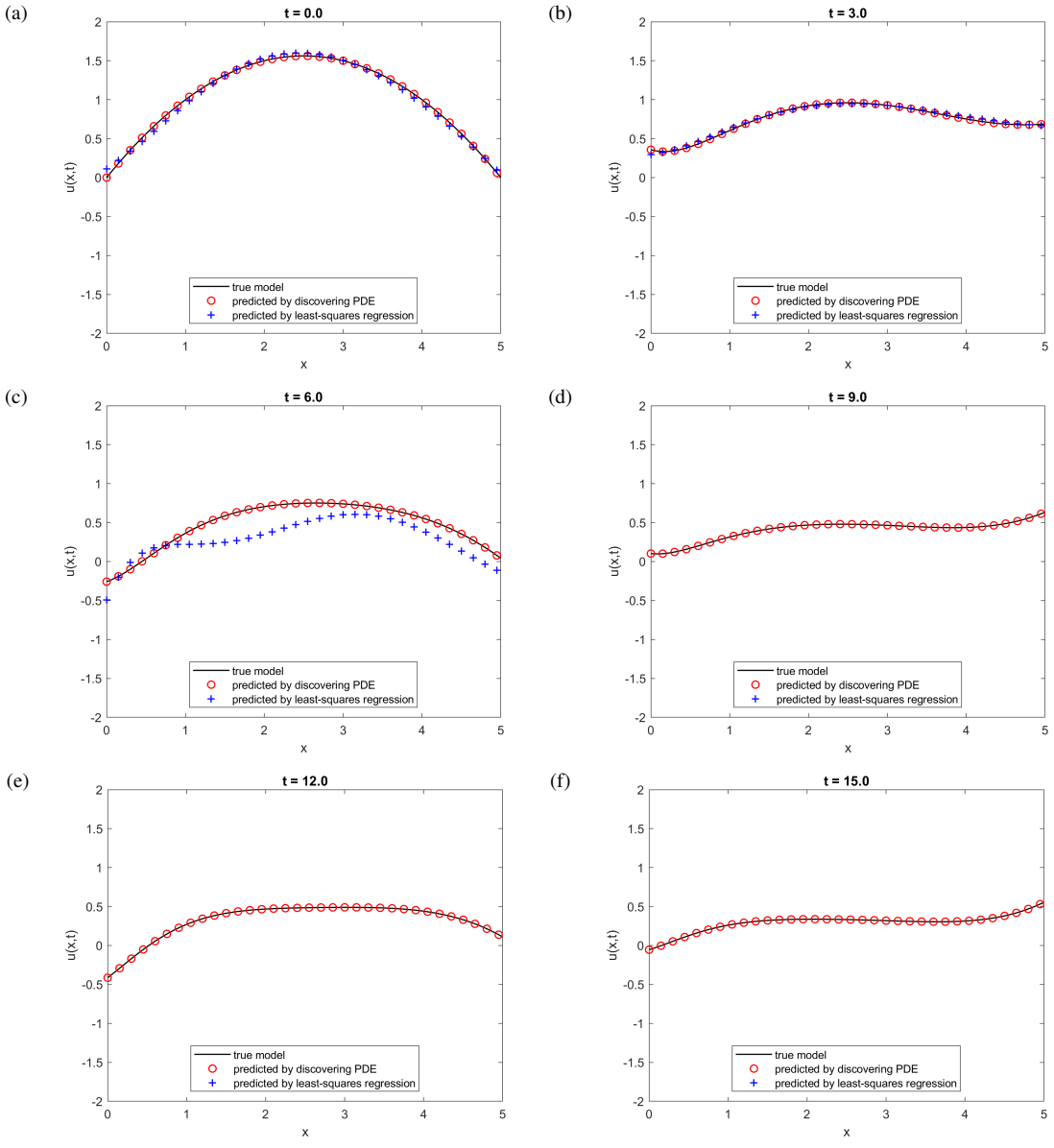

Figure 15: The true model, the solution predicted by discovering PDE, and the solution predicted by least-squares regression, at different time t. All settings are the same as Figure 14.

7. The challenge of extrapolation

7.1. Problem description

Here we demonstrate how to tackle the challenge of extrapolation through an example of discovering a differential equation with bifurcations. When the differential equations have bifurcations, the solutions may have different behavior in different areas. If the data are given within a region, traditional interpolation and regression methods may not be able to capture the behavior or make predictions outside that region. On the contrary, the method of discovering differential equations is not impacted by bifurcations and it can extrapolate to the areas where no data are given.

7.2. Example: fish-harvesting problem with bifurcations

Consider the following fish-harvesting problem:

$$\frac{dx}{dt} = x(4 - x) - H, \tag{70}$$

where $x(t)$ is the population of the fish at time t and $H \geq 0$ is the constant rate at which the fish are harvested. In this example, we fix $H = 3$:

$$\frac{dx}{dt} = x(4 - x) - 3. \tag{71}$$

Setting $dx/dt = 0$, we have:

$$x(4 - x) - 3 = 0, \tag{72}$$

whose solutions are

$$x = 1 \text{ and } x = 3. \tag{73}$$

When the fish population x is between 1 and 3, the population grows up; otherwise, it goes down. See Figure 16b.

7.2.1. Data collection and discovery of the model

We generate data on $t \in [0, 2]$ using five random initial values $x_0 \sim \mathcal{U}(1, 3)$, the uniform distribution on $[1, 3]$, and we collect data at the nodes illustrated in Figure 16a. Then the derivatives are calculated by the five-point central-difference formula. Note that the derivatives are not calculated at the first two and last two data points in each curve. There are 80 data points with derivative. Next we discover the ODE using our algorithm SubTSBR with subsampling size 40, a half of all the data points with derivative, and 30 loops. The basis-functions are monomials generated by $\{1, t, x\}$ up to degree 10. There are 66 terms. The result is:

$$\frac{dx}{dt} = -2.9998 + 3.9998x - 1.0000x^2. \tag{74}$$

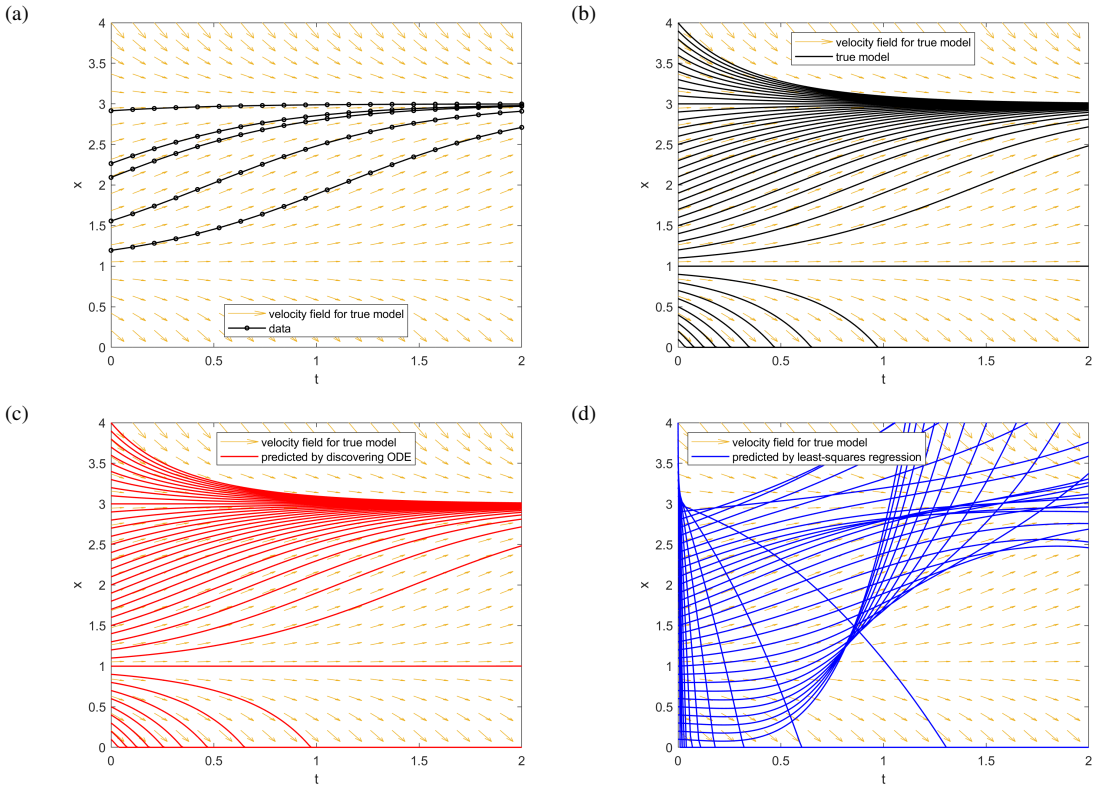

Figure 16: (a) The data are generated by five random initial values and collected at the nodes. (b) The solutions to the true model (71). (c) The solutions to the discovered model (74). (d) The solutions calculated by least-squares regression.

7.2.2. Prediction

Now we predict the solutions using the discovered ODE (74) and 40 evenly spaced initial values x_0. See Figure 16c. The solutions calculated by the true model with the same initial values are shown in Figure 16b. As a comparison, the solutions predicted by least-squares regression are displayed in Figure 16d, where the data are all 100 nodes illustrated in Figure 16a, and $u(t, x_0)$ is fitted by a linear combination of monomials generated by $\{1, t, x_0\}$ up to degree 11 with the constraint $u(0, x_0) = x_0$. There are 66 coefficients to be estimated. Then the solutions are drawn by fixing x_0 at each value.

Although the data are generated by initial values x_0 between 1 and 3, the discovered model (74) extrapolates to other initial values and almost perfectly predicts the behavior of the true model. By contrast, least-squares regression barely approximates the behavior of the true model in the area where the data are given and is not able to extrapolate.

8. Summary

In this paper, we have analyzed four challenges in the data-driven discovery of physical laws: (1) large noise in the data, (2) outliers in the data, (3) integrating the data collected from different experiments, and (4) extrapolating the

solutions to the areas that have no available data. To tackle these challenges, we have adopted the strategy: first, discover the governing differential equations; second, solve the differential equations analytically or numerically. On top of that, we have proposed a new model-discovering algorithm, the subsampling-based threshold sparse Bayesian regression algorithm. This new algorithm is a generalization and improvement of the original threshold sparse Bayesian regression algorithm first introduced in [1]. Our new algorithm is designed to be robust to large noise and outliers via incorporating a subsampling technique in the sparse Bayesian inference. The new algorithm has two user-preset parameters, the subsampling size and the number of loops. The subsampling size should be moderate and can be fitted automatically using the adjusted error bar defined in this paper, while the number of loops is the bigger the better. In practice, we can increase the number of loops gradually and stop the algorithm when the smallest error bar among all the loops drops below a certain preset value or the smallest error bar stops decreasing. The minimum number of loops needed to remove a certain percentage of outliers has also been calculated.

Four examples have been given in this paper: the predator-prey model with noise, shallow water equations with outliers, heat diffusion with random initial and boundary conditions, and fish-harvesting problem with bifurcation. They demonstrate that our new algorithm is able to overcome the four aforementioned challenges and is significantly better than the other model-discovering methods (sequential threshold least squares, lasso, and the original threshold sparse Bayesian regression algorithm) and traditional regression method (least squares). In practice, denoising processes may be applied followed by our new algorithm to attain optimum performance. On top of that, the success of the subsampling strategy proposed in this paper justifies the effectiveness of the defined error bar as an indicator for the quality of the discovered model.

Acknowledgments

We acknowledge the support from the National Science Foundation (DMS-1555072, DMS-1736364, and DMS-1821233). We thank Elizabeth A. Robbins for proofreading this manuscript.

[1] S. Zhang, G. Lin, Robust data-driven discovery of governing physical laws with error bars, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 474 (2217) (2018) 20180305. doi:10.1098/rspa.2018.0305.
URL http://rspa.royalsocietypublishing.org/lookup/doi/10.1098/rspa.2018.0305

[2] M. Schmidt, H. Lipson, Distilling Free-Form Natural Laws from Experimental Data, Science 324 (5923) (2009) 81–85. doi:10.1126/science.1165893.
URL http://www.sciencemag.org/lookup/doi/10.1126/science.1165893

[3] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences 113 (15) (2016) 3932–3937. doi:10.1073/pnas.1517384113.
URL http://www.pnas.org/lookup/doi/10.1073/pnas.1517384113

[4] S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, Science Advances 3 (4) (2017) e1602614. doi:10.1126/sciadv.1602614.
URL http://advances.sciencemag.org/lookup/doi/10.1126/sciadv.1602614

[5] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) (1996) 267–288.

[6] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 473 (2197) (2017) 20160446. doi:10.1098/rspa.2016.0446.
URL http://rspa.royalsocietypublishing.org/lookup/doi/10.1098/rspa.2016.0446

[7] N. M. Mangan, S. L. Brunton, J. L. Proctor, J. N. Kutz, Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics, IEEE Transactions on Molecular, Biological and Multi-Scale Communications 2 (1) (2016) 52–63. doi:10.1109/TMBMC.2016.2633265.
URL http://ieeexplore.ieee.org/document/7809160/

[8] M. Dam, M. Brns, J. Juul Rasmussen, V. Naulin, J. S. Hesthaven, Sparse identification of a predator-prey system from simulation data of a convection model, Physics of Plasmas 24 (2) (2017) 022310. doi:10.1063/1.4977057.
URL http://aip.scitation.org/doi/10.1063/1.4977057

[9] H. Schaeffer, S. G. McCalla, Sparse model selection via integral terms, Physical Review E 96 (2). doi:10.1103/PhysRevE.96.023302.
URL http://link.aps.org/doi/10.1103/PhysRevE.96.023302

[10] H. Schaeffer, G. Tran, R. Ward, Extracting Sparse High-Dimensional Dynamics from Limited Data, arXiv:1707.08528 [math]ArXiv: 1707.08528.
URL http://arxiv.org/abs/1707.08528

[11] G. Tran, R. Ward, Exact Recovery of Chaotic Systems from Highly Corrupted Data, Multiscale Modeling & Simulation 15 (3) (2017) 1108–1129. doi:10.1137/16M1086637.
URL http://epubs.siam.org/doi/10.1137/16M1086637

[12] N. M. Mangan, J. N. Kutz, S. L. Brunton, 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) (2017) 20170009. doi:10.1098/rspa.2017.0009.
URL http://rspa.royalsocietypublishing.org/lookup/doi/10.1098/rspa.2017.0009

[13] E. Kaiser, J. N. Kutz, S. L. Brunton, Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, arXiv:1711.05501 [physics]ArXiv: 1711.05501.
URL http://arxiv.org/abs/1711.05501

[14] L. Boninsegna, F. Nske, C. Clementi, Sparse learning of stochastic dynamical equations, The Journal of Chemical Physics 148 (24) (2018) 241723. doi:10.1063/1.5018409.
URL http://aip.scitation.org/doi/10.1063/1.5018409

[15] N. M. Mangan, T. Askham, S. L. Brunton, J. N. Kutz, J. L. Proctor, Model selection for hybrid dynamical systems via sparse regression, arXiv:1808.03251 [math]ArXiv: 1808.03251.

URL http://arxiv.org/abs/1808.03251

[16] S. Rudy, A. Alla, S. L. Brunton, J. N. Kutz, Data-driven identification of parametric partial differential equations, arXiv:1806.00732 [math]ArXiv: 1806.00732.
URL http://arxiv.org/abs/1806.00732

[17] H. Schaeffer, G. Tran, R. Ward, L. Zhang, Extracting structured dynamical systems using sparse optimization with very few samples, arXiv:1805.04158 [cs, math]ArXiv: 1805.04158.
URL http://arxiv.org/abs/1805.04158

[18] J.-C. Loiseau, S. L. Brunton, Constrained sparse Galerkin regression, Journal of Fluid Mechanics 838 (2018) 42–67. doi:10.1017/jfm.2017.823.
URL https://www.cambridge.org/core/product/identifier/S0022112017008230/type/journal_article

[19] M. Quade, M. Abel, J. N. Kutz, S. L. Brunton, Sparse Identification of Nonlinear Dynamics for Rapid Model Recovery, Chaos: An Interdisciplinary Journal of Nonlinear Science 28 (6) (2018) 063116, arXiv: 1803.00894. doi:10.1063/1.5027470.
URL http://arxiv.org/abs/1803.00894

[20] L. Zhang, H. Schaeffer, On the Convergence of the SINDy Algorithm, arXiv:1805.06445 [cs, math]ArXiv: 1805.06445.
URL http://arxiv.org/abs/1805.06445

[21] S. H. Rudy, J. N. Kutz, S. L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, arXiv:1808.02578 [math]ArXiv: 1808.02578.
URL http://arxiv.org/abs/1808.02578

[22] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv:1711.10561 [cs, math, stat]ArXiv: 1711.10561.
URL http://arxiv.org/abs/1711.10561

[23] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, arXiv:1711.10566 [cs, math, stat]ArXiv: 1711.10566.
URL http://arxiv.org/abs/1711.10566

[24] M. Raissi, G. E. Karniadakis, Machine Learning of Linear Differential Equations using Gaussian Processes, Journal of Computational Physics 348 (2017) 683–693, arXiv: 1701.02440. doi:10.1016/j.jcp.2017.07.050.
URL http://arxiv.org/abs/1701.02440

[25] B. Efron, C. Stein, The Jackknife Estimate of Variance, The Annals of Statistics 9 (3) (1981) 586–596. doi:10.1214/aos/1176345462.
URL http://projecteuclid.org/euclid.aos/1176345462

[26] M. E. Tipping, Sparse Bayesian learning and the relevance vector machine, Journal of machine learning research 1 (Jun) (2001) 211–244.

[27] D. J. C. MacKay, Bayesian Methods for Backpropagation Networks, in: E. Domany, J. L. van Hemmen, K. Schulten, E. Domany, J. L. van Hemmen, K. Schulten (Eds.), Models of Neural Networks III, Springer New York, New York, NY, 1996, pp. 211–254. doi:10.1007/978-1-4612-0723-8_6.
URL http://link.springer.com/10.1007/978-1-4612-0723-8_6

[28] R. M. Neal, Bayesian Learning for Neural Networks, Vol. 118 of Lecture Notes in Statistics, Springer New York, New York, NY, 1996. doi:10.1007/978-1-4612-0745-0.
URL http://link.springer.com/10.1007/978-1-4612-0745-0

[29] M. E. Tipping, A. C. Faul, Fast marginal likelihood maximisation for sparse Bayesian models, in: AISTATS, 2003.

[30] A. Schmolck, R. Everson, Smooth relevance vector machine: a smoothness prior extension of the RVM, Machine Learning 68 (2) (2007) 107–135. doi:10.1007/s10994-007-5012-z.
URL http://link.springer.com/10.1007/s10994-007-5012-z

[31] P. W. Holland, R. E. Welsch, Robust regression using iteratively reweighted least-squares, Communications in Statistics - Theory and Methods 6 (9) (1977) 813–827. doi:10.1080/03610927708827533.

URL http://www.tandfonline.com/doi/abs/10.1080/03610927708827533

[32] R. Chartrand, Wotao Yin, Iteratively reweighted algorithms for compressive sensing, in: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, Las Vegas, NV, USA, 2008, pp. 3869–3872. doi:10.1109/ICASSP.2008.4518498.
URL http://ieeexplore.ieee.org/document/4518498/

[33] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395. doi:10.1145/358669.358692.
URL http://portal.acm.org/citation.cfm?doid=358669.358692

[34] O. J. Karst, Linear Curve Fitting Using Least Deviations, Journal of the American Statistical Association 53 (281) (1958) 118. doi:10.2307/2282572.
URL https://www.jstor.org/stable/2282572?origin=crossref

[35] H. Theil, A Rank-Invariant Method of Linear and Polynomial Regression Analysis, in: A. J. H. Hallet, J. Marquez, B. Raj, J. Koerts (Eds.), Henri Theils Contributions to Economics and Econometrics, Vol. 23, Springer Netherlands, Dordrecht, 1992, pp. 345–381. doi:10.1007/978-94-011-2546-8_20.
URL http://www.springerlink.com/index/10.1007/978-94-011-2546-8_20

[36] P. K. Sen, Estimates of the Regression Coefficient Based on Kendall's Tau, Journal of the American Statistical Association 63 (324) (1968) 1379. doi:10.2307/2285891.
URL https://www.jstor.org/stable/2285891?origin=crossref

[37] A. F. Siegel, Robust regression using repeated medians, Biometrika 69 (1) (1982) 242–244. doi:10.1093/biomet/69.1.242.
URL https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/69.1.242

[38] P. J. Rousseeuw, Least Median of Squares Regression, Journal of the American Statistical Association 79 (388) (1984) 871–880. doi:10.1080/01621459.1984.10477105.
URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1984.10477105

[39] L. I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D: Nonlinear Phenomena 60 (1-4) (1992) 259–268. doi:10.1016/0167-2789(92)90242-F.
URL http://linkinghub.elsevier.com/retrieve/pii/016727899290242F

[40] R. Chartrand, Numerical Differentiation of Noisy, Nonsmooth Data, ISRN Applied Mathematics 2011 (2011) 1–11. doi:10.5402/2011/164564.
URL https://www.hindawi.com/archive/2011/164564/

[41] J. Lagergren, J. T. Nardini, G. M. Lavigne, E. M. Rutter, K. B. Flores, Learning partial differential equations for biological transport models from noisy spatiotemporal data, arXiv:1902.04733 [math]ArXiv: 1902.04733.
URL http://arxiv.org/abs/1902.04733