System Identification through Expression Optimization

Anonymous Authors[1]

Abstract

With the abundance of natural data from physical systems, much engineering and scientific value comes from an ability to discover the underlying, governing equations of a system, with little prior knowledge. Current approaches for data-driven system identification either find relationships in the data that aren't interpretable, or require significant prior knowledge from the user. This work describes a new approach to system identification that requires minimal user input and discovers governing equations that are parsimonious, generalizable and interpretable. This is enabled by recent advances in expression optimization, allowing for the automated discovery of mathematical expressions from a combinatorically large set of possibilities. Using simulated data, our approach correctly identifies both linear and nonlinear PDEs including the Navier-Stokes equations. It can also generate exact and approximate Koopman eigenfunctions for nonlinear ODEs. The ability to interpret large amounts of data will allow researchers to better understand and control important natural systems, such as the earths climate, for addressing global warming and fluid flow for more efficient energy generation and transportation.

1. Introduction

Over the past several decades machine learning and artificial intelligence has made great strides in learning patterns from large amounts of data. Although accurate, these systems are often uninterpretable by the human researchers who create them. They do not report an explanation for their predictions nor do they generalize well when the task is changed slightly.

Recently, however, strides have been made to create AI systems that, rather than just looking for trends, look for causal explanations of observed data (Bridewell, 2008, Brunton, 2016). If an AI system can correctly determine a simple underlying model for a system then it has the capability of providing an explanatory account of the data as well as the ability to generalize well in predicting behavior under a wider variety of situations. This can help researchers more quickly discover models that explain experimental data.

The goal of this project is to implement a system identifier for the discovery of physical processes in terms understandable by a human researcher (i.e. stated as partial differential equations (PDEs)). The system should work with multi-dimensional data, be robust to noise, and require small amounts of data to operate. This paper is outlined as follows: The next section discusses the approach to system identification that will be used, including how features are generated and selected from observed data. The following section describes two test systems from which synthetic noisy data is obtained to demonstrate the system identifier. The last two sections report the results of the system identifier and a discussion of how it can be improved.

The metric that was chosen to decide between models is the adjusted R^2 value of the fit. The traditional R^2 value always increases when new features are added so it makes our algorithm susceptible to overfitting. Instead, we penalized the addition of more features to the regression by defining the R^2_{adj} as [CITATION]

$$l_f = -R^2_{\text{adj}} = -[R^2 - \alpha(1 - R^2)(n - 1)]$$

where n is the number of features in the model. R^2_{adj} reduces to R^2 when there is only a single

Note that when $n = 1$, this expression reduces to the traditional R^2 value. But when $n > 1$, then the difference between R^2 and 1 is scaled by the number of parameters.

In the case of discovering Koopman modes, parsimony is no longer a consideration so a new objective function needs to be used. Given a linear system with n_g number of Koopman eigenfunctions, the loss function associated with the Koopman operator K and Koopman eigenfunction $\vec{g}$ is given by

$$l_k(K, \vec{g}) = \frac{1}{n_g} ||\vec{g}(\vec{x})^{t+1} - K\vec{g}(\vec{x})^t||_2^2$$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Algorithm 1 Forward Search

```
Input: Loss function $l_f$,
       New feature generator $n_f$,
       Threshold $t$
Initialize $expr = [\,]$,    $lowest = \infty$
repeat
  $expr =$ concatenate($expr,\ n_f(expression)$)
  if $l_f(expr) < lowest$ then
    $lowest = l_f(expr)$
  else
    return $expr$
  end if
until $lowest < t$
```

The scaling by $1/n_g$ is so that that algorithm does not get penalized for adding more Koopman eigenfunctions to explain the system, and instead focuses on closely matching each new state variable on each iteration.

2. System Identification

The goal of identifying a simple model that describes observed data has been researched in the past by Bridewell, 2008, Brunton, 2016. The system identifier presented here, takes elements from those two approaches while expanding the capabilities to the domain of spatio-temporal processes governed by nonlinear PDEs. The simplest form of a general PDE is given by

$$\frac{\partial u(x,t)}{\partial t} = \alpha_1 f_1(u(x,t)) + ... + \alpha_n f_n(u(x,t)) = \boldsymbol{\alpha}^T \boldsymbol{f}$$

where the α_i are constant coefficients and the functions f_i are *feature functions* of the solution $u(x,t)$.

The first step of the system will be to generate a large number of feasible feature functions that could comprise the PDE. The second step is to select a subset of these features that best explain the left hand side of the PDE through linear regression. These steps are discussed in more detail in the next few subsections.

[[Include objective function here]]

2.1. Generation of Features

[[Put forward search here]]

The choice of feature selection in general is a difficult problem. There are an infinite number of possible feature functions and it is impossible to know a priori which functions will comprise the PDE. Since the system identifier is meant to be an aide to researchers and scientists, it is reasonable to ask the user to provide some amount of guidance to the algorithm without explicitly handing over a set of feature functions (which the researcher presumably does not know).

Algorithm 2 Backward Search

```
Input: Loss function $l_f$,
       Expression $expr$,
       Threshold $t$
Initialize $lowest = l_f(expr)$
repeat
  Initialize $losses = [\,]$
  for $feat$ in $expr$ do
    $new_e xpr =$ remove($expr, feat$)
    push($losses, l_f(new_e xpr)$)
  end for
  $i = (losses)$
  if $losses[i] < lowest$ then
    $lowest = losses[i]$
    $expr =$ remove($expr, feat$)
  else
    return $expr$
  end if
until $lowest < t$
```

To this end, the system identifier relies on a domain-specific *grammar*, provided by the user, that gives the rules for generating feature functions. A grammar is a set of production rules that govern a language (or a set of expressions). Each rule can either be *non-terminal*, in which the rule relates generic expressions together (e.g. multiplication), or *terminal*, in which an expression is concretely defined (e.g. the observed data). Each expression in the language can be represented by a tree of operators, each of which is part of the grammar. Once a grammar is defined, expressions can be sampled from the grammar with varying levels of complexity (as defined by the depth of the expression tree).

A sample grammar and expression tree are shown in figure **??**. A convenient way to read the rules of the grammar is to convert it to plain english. Let $\mathbb{R}$ mean "an expression" and $\mapsto$ mean "can be", then the second production rule reads "an expression can be an expression times an expression". The last production rule is where the variables of interest are introduced. This rule reads "an expression can be a velocity component or pressure". Sampling from a grammar is a process of selecting production rules and then filling in any non-terminal expressions until only terminal expressions remain. In the expression tree shown in the right of figure **??**, the first production rule chosen is multiplication. Then, on the left side, the terminal expression u was chosen, and on the right side, the spatial derivative was chosen, followed by the terminal expression u.

To produce the candidate set of features, the user will specify a desired tree depth, d, and all possible expressions with depth $\leq d$ will be produced. Then, this set of features is searched for expressions that evaluate to the same results and any such duplicates are removed. The number of candidate expressions grows exponentially with the depth of the tree, but fortunately, most pdes that govern physical processes have terms that are only at a depth of 4 or less which makes the problem tractable (see Wikipedia list of nonlinear PDEs, 2018). For the model systems, all terms can be produced from an expression depth of 3, which, for the grammar in figure **??** means that a total of 562 features will be considered (222 after removing duplicates). The julia package `ExprRules.jl` was used to build the grammar and to sample expressions from it at the desired depth.

2.2. Expression Optimization

3. Results

3.1. Advection-Diffusion Equation

The first test system is the unsteady 1D advection-diffusion equation which describes the general transport of material in a fluid. The system has the form

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - v\frac{\partial u}{\partial x}$$

$$u(x,0) = u_0 \tag{1}$$

$$u(0,t) = u_L \tag{2}$$

$$\frac{\partial u}{\partial x}(\infty, t) = 0 \tag{3}$$

The exact solution of which is given by Van Genuchten 1982 as

$$u(x,t) = u_0 + \frac{1}{2}(u_L - u_0)\left(\text{erfc}\left[\frac{x-vt}{2\sqrt{Dt}}\right] + \exp(vx/D)\text{erfc}\left[\frac{x+vt}{2\sqrt{Dt}}\right]\right) \tag{4}$$

The solution is plotted in figure 1 for $x > 0, t > 0, D = 1$ and $v = 0.5$.

Grammar

$$\mathbb{R} \mapsto u$$
$$\mathbb{R} \mapsto \frac{d\mathbb{R}}{dx}$$
$$\mathbb{R} \mapsto \mathbb{R} \times \mathbb{R}$$
$$\mathbb{R} \mapsto \mathbb{R}/\mathbb{R}$$

The grammar was used to a depth of 3 which generates a total of 37 possible features.

3.2. Navier-Stokes Equations

The second test system is the incompressible 2D Navier-Stokes equations which describe the motion of a fluid. The

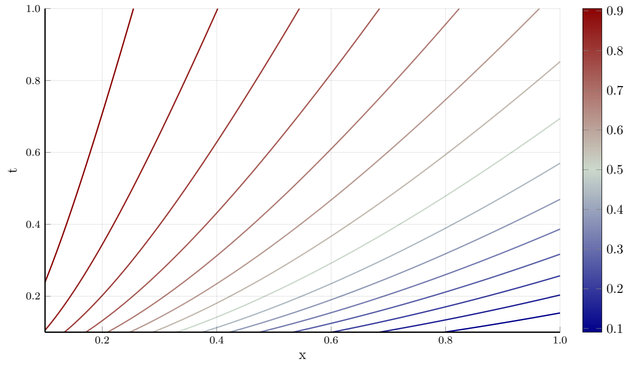


Figure 1. Unsteady data from the 1D advection-diffusion equation

Table 1. Classification accuracies for naive Bayes and flexible Bayes on various data sets.

Sample Points	Noise	Correct Expression	Error in v	Error in D
200	1%	$\surd$	0.78 %	1.54 %
200	5%	$\surd$	5.64 %	2.84 %
200	20%	$\surd$	30.12 %	6.60 %
100	0%	$\surd$	0.84 %	0.29 %
35	0%	$\surd$	6.38 %	2.21 %
10	0%	$\surd$	1.19 %	19.02 %

Navier-Stokes equations are given by

$$\frac{\partial u}{\partial x} + \partial v \partial y = 0 \tag{5}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{6}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial v} + \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{7}$$

where ρ is the fluid density, (u, v) is the fluid velocity, p is the pressure, and μ is the viscosity. The first equation represnts the conservation of mass and the next two equations represent convservation of momentum in the x and y directions, respectively. These equations are nonlinear, have multiple dimensions (x and y) and multiple degrees of freedom (u, v, and p), which makes them a challenging test case for system identification.

Grammar

$$\mathbb{R} \mapsto u \quad | \quad v \quad | \quad p$$
$$\mathbb{R} \mapsto \frac{d\mathbb{R}}{dx} \quad \Big| \quad \frac{d\mathbb{R}}{dy}$$
$$\mathbb{R} \mapsto \mathbb{R} \times \mathbb{R}$$

3.3. Koopman Eigenfunction Discovery

$$
\begin{aligned}
&\mathbb{R} \mapsto \theta \quad | \quad \omega \\
&\mathbb{R} \mapsto \mathbb{R} \times \mathbb{R} \\
&\mathbb{R} \mapsto \sin(\mathbb{G} \times \mathbb{R}) \quad | \quad \sin(\mathbb{G} \times \mathbb{R} + \mathbb{G}) \\
&\mathbb{R} \mapsto \exp(\mathbb{G} \times \mathbb{R}) \\
&\mathbb{R} \mapsto 1/(1 + \exp(-\mathbb{G} \times \mathbb{R})) \\
&\mathbb{R} \mapsto \exp(-(\mathbb{R} - \mathbb{G})^2/\mathbb{G}) \\
&\mathbb{R} \mapsto \mathrm{imag}(\mathbb{R}^{\mathbb{G}}) \\
&\mathbb{R} \mapsto \mathrm{real}(\mathbb{R}^{\mathbb{G}}) \\
&\mathbb{G} \mapsto -\mathbb{G} \quad | \quad \mathbb{G} + \mathbb{G} \quad | \quad \mathbb{G}/\mathbb{G} \\
&\mathbb{G} \mapsto \mathrm{logspace}(-5, 2, 20)
\end{aligned}
$$

Koopman theory [CITATION] states that an nonlinear dynamical system can be converted into a linear system under the state space mapping

$$\vec{x} \rightarrow \vec{g}(\vec{x})$$

where $\vec{g}(x) = \begin{bmatrix} g_1(\vec{x}) & g_2(\vec{x}) & + & ... \end{bmatrix}$ and is, in general infinite dimensional.

Given the time-discrete nonlinear dynamical system given by

$$\vec{x}^{t+1} = \vec{f}(\vec{x}^t)$$

the Koopman transformation yields the linear system

$$\vec{g}(\vec{x})^{t+1} = K\vec{g}(\vec{x})^t$$

A simple nonlinear ODE with an easily found, and finite-dimensional Koopman eigenfunctions is given by [Brunton citation]

$$\frac{dx}{dt} = \mu x \tag{8}$$

$$\frac{dy}{dt} = \lambda(y - x^2) \tag{9}$$

The state-space transformation

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ x^2 \end{bmatrix}$$

linearizes equation 8 to

$$\frac{d}{dt} \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} x \\ y \\ x^2 \end{bmatrix}$$

3.3.1. Nonlinear Pendulum

One of the simplest nonlinear ODEs describes the motion of a pendulum. The equation of motion a pendulum is given by

$$\frac{d^2\theta}{dt^2} + \sin\theta = 0$$

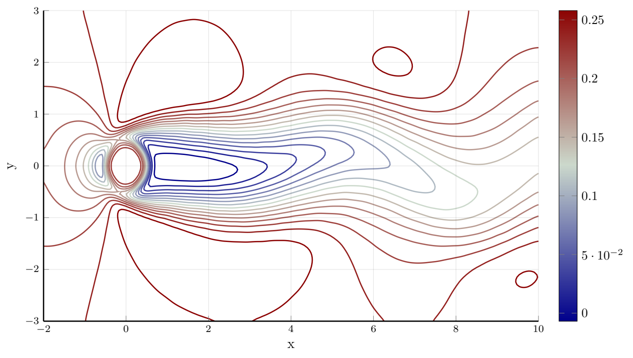
This equation can be converted to a system of nonlinear first-order ODEs given by

$$
\begin{aligned}
\frac{d\theta}{dt} &= \omega \\
\frac{d\omega}{dt} &= -\sin\theta
\end{aligned}
$$

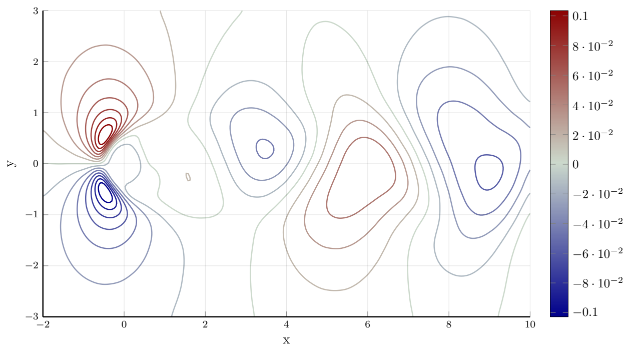
4. Conclusions

References

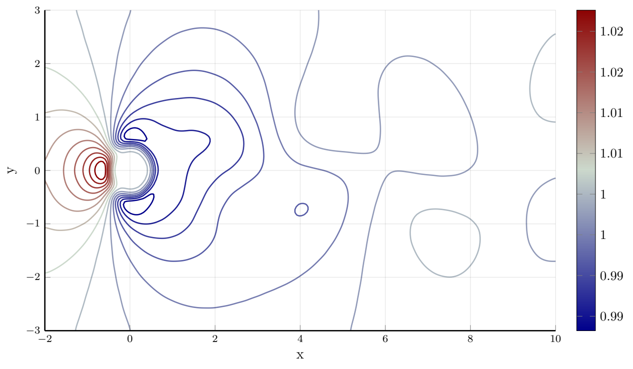
Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.



(a) x-velocity



(b) y-velocity



(c) pressure

Figure 2. Global caption that can reference **??** and **??**