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

$$R^2_{\text{adj}} = R^2 - (1 - R^2)(n-1)^2 - \lambda(n-1)$$

where n is the number of features in the model.

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 squared. The third term in the expression was included for the cases when R^2 was very close to or exactly one (as can be the case when dealing with noise-free synthetic data). If $R^2 = 1$, the second term vanishes, and there would be no way to distinguish between models of different complexity. Thus, a small value of λ is chosen to break ties in favor of the simpler model. In the following tests, $\lambda = 0.001$ was used.

[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.

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

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).

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

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} - v\frac{\partial u}{\partial x} \quad withbcs \quad u(x,0) = u_0, \quad u(0,t) = u_L, \quad \frac{\partial u}{\partial x}(\infty, t)$$

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

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

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

3.2. Navier-Stokes Equations

[[Change this to the viscous incompressible flow equations]] The second test system is the steady 2D Euler equations which govern the flow of an inviscid fluid in two dimensions. The system has the form (written in vector index notation for convenience)

$$\frac{\partial u_i}{\partial x_i} = 0, \qquad u_j\frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\partial p \partial x_i, \quad withbc \quad \vec{u}(-\infty, y) = (U_\infty, 0)$$

where ρ is the constant density of the fluid, u_i is the i^{th} velocity component and x_i is the i^{th} velocity spatial direction

(for $i = 1,\ 2$). The first equation represents conservation of mass in the fluid, while the second equation represents conservation of momentum.

3.3. Koopman Eigenfunction Discovery

4. Conclusions

References

Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification*. John Wiley and Sons, 2nd edition, 2000.

Kearns, M. J. *Computational Complexity of Machine Learning*. PhD thesis, Department of Computer Science, Harvard University, 1989.

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.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds.). *Machine Learning: An Artificial Intelligence Approach, Vol. I.* Tioga, Palo Alto, CA, 1983.

Mitchell, T. M. The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA, 1980.

Newell, A. and Rosenbloom, P. S. Mechanisms of skill acquisition and the law of practice. In Anderson, J. R. (ed.), *Cognitive Skills and Their Acquisition*, chapter 1, pp. 1–51. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1981.

Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959.