github repo: https://github.com/ancorso/sci_discovery

System Identification of Partial Differential Equations

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.

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

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

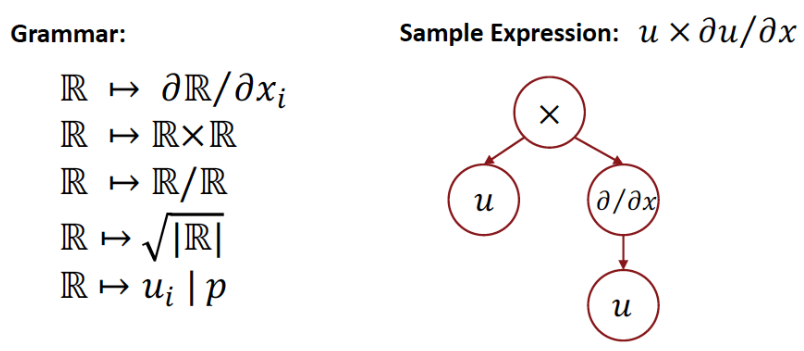
where the α_i are constant coefficients and the functions f_i are *feature functions* of the solution $u(x,t)$. For the purposes of this project we will place a constraint on the feature functions that they have no free parameters associated with them other than α. Therefore, an expression such as $f(u(x,t)) = \partial u^2(x,t)/\partial x$ is a valid feature function while $f(u(x,t)) = u(x,t)/(k + u(x,t))$ is not, unless k is a known constant. This constraint is placed on the problem merely to make the model fitting process as straightforward as possible.

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.

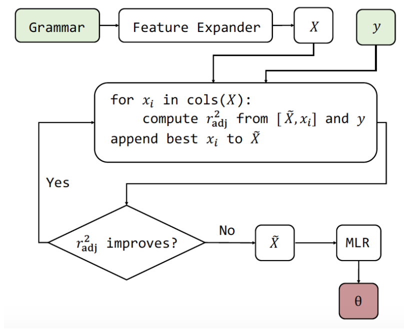
Generation of Features

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).


Grammar:
$\mathbb{R} \mapsto \partial\mathbb{R}/\partial x_i$
$\mathbb{R} \mapsto \mathbb{R}\times\mathbb{R}$
$\mathbb{R} \mapsto \mathbb{R}/\mathbb{R}$
$\mathbb{R} \mapsto \sqrt{|\mathbb{R}|}$
$\mathbb{R} \mapsto u_i \mid p$
Sample Expression: $u \times \partial u/\partial x$
$\times$
u
$\partial/\partial x$
u

(a) Grammar and expression tree


Grammar
Feature Expander
X
y
for x_i in cols(X): compute r^2_{adj} from $[\bar{X}, x_i]$ and y append best x_i to $\bar{X}$
Yes
r^2_{adj} improves?
No
$\bar{X}$
MLR
θ

(b) Flow Chart of feature selection algorithm

A sample grammar and expression tree are shown in figure 1a. 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 1a, 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 1a 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.

Feature Selection

Once the candidate feature functions are computed, the next step of the algorithm is to select the best subset of features that describes the observed data. In the next subsection, an evaluation metric is presented that will be used to determine how good a subset of features is. In the following subsection, the algorithm that chooses the best subset will be described in detail.

Evaluation Metric

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.

Feature Selection Algorithm

In general, the algorithm to find which subset of features has the best R^2_{adj} value is very computationally complex. To check all of the subsets of a set of n features requires 2^n linear regressions. The grammars and depths that would commonly be used would give rise to $n = 100 - 1000$ features which makes the subset search intractable. In place of the brute force search of subsets, a form of forward search was implemented instead. The workflow of this algorithm is shown in figure 1b. A description of the full algorithm is given in the following two paragraphs.

The inputs to the system indentifier are y, the data that needs to be explained (i.e. $\partial u/\partial t$ for the advection-diffusion equation), and the grammar defined by the domain expert which contains the observational data and the production rules to generate features. The grammar is then used to generate many possible features (with the expression depth controlled by the user). The features are stored in an input matrix X which is cleaned of duplicate columns (due to equivalent expressions from the grammar). The columns of the input matrix, X and the output vector, y, are normalized to a mean of 0 and a variance of 1 to avoid problems with differing scales and constant offsets. X and y are then passed into the feature selection algorithm.

The feature selection algorithm starts with an empty set of features $\tilde{X}$. Each feature from X is added, in turn, to $\tilde{X}$ so that a linear regression with y can be performed and the R^2_{adj} value computed. The added feature is then removed and the next feature is added. Once all features in X have been tried, the feature that had the best R^2_{adj} value is permanently added to $\tilde{X}$. On the next iteration, all features (except the feature already used) individually get added to $\tilde{X}$ and the R^2_{adj} is computed for the resulting two-feature input. If there is no feature that caused the R^2_{adj} to increase, then the algorithm stops and returns the set of features that are currently in $\tilde{X}$, otherwise, the new feature with the best R^2_{adj} gets added to $\tilde{X}$ and the process continues.

The algorithm is also capable of returning not just the highest scoring set of features, but the top k scoring sets of features. This is achieved by running the feature selection algorithm k times and keeping a list of the sets of features that were returned on previous iterations. These sets of features are skipped over on the next iteration and the next best combination is found. Searching through more possibilities increases the chances that the algorithm doesn't get stuck in a local optimum due to the greedy selection criterion. The resulting complexity of this algorithm is $\mathcal{O}(nk)$ which is a significant improvement.

Sample Data

In order to evaluate the efficacy of the system identifier, two test systems were selected for trials. These test systems will produce data at varying levels of spatial resolution and with different amounts of noise. The data will then be fed through the system identifier and the results will be compared to the actual underlying system. 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

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

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

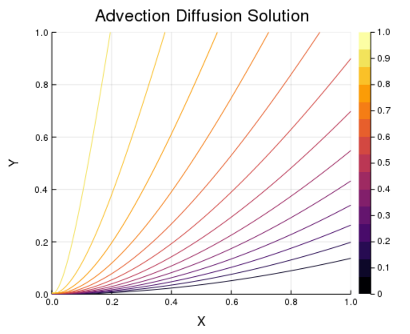
$$u(x,t) = u_0 + \frac{1}{2}(u_L - u_0)\left(\text{erfc}\left[\frac{x - vt}{2\sqrt{Dt}}\right] + \exp(vx/D)\text{erfc}\left[\frac{x + vt}{2\sqrt{Dt}}\right]\right)$$

The solution is plotted in figure 2a for $x > 0$, $t > 0$, $D = 1$ and $v = 0.5$.

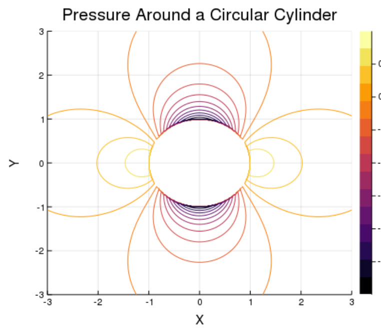
The second test system is the steady 2D Euler equations which govern the flow of an inviscid fluid in two dimensions. The system has the form (written in vector index notation for convenience)

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

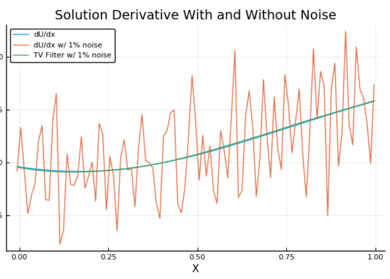
where ρ is the constant density of the fluid, u_i is the i^{th} velocity component and x_i is the i^{th} velocity spatial direction (for $i = 1,\ 2$). The first equation represents conservation of mass in the fluid, while the second equation represents conservation of momentum.



(a) The solution to the advection-diffusion equation in the domain $x > 0$ and $t > 0$



(b) The pressure field of inviscid flow over around a circular cylinder



(c) Total variation filter

Figure 2: Solution to test systems and denoising example

The exact solution of these equations for flow around a circular cylinder of radius R, centered at the origin, is given in cylindrical coordinates as (Anderson 1984)

$$u_r = U_\infty \left(1 - \frac{R^2}{r^2}\right)\cos\theta, \qquad u_\theta = -U_\infty \left(1 + \frac{R^2}{r^2}\right)\sin\theta, \qquad \frac{2p}{\rho U_\infty^2} = 2\frac{R^2}{r^2}\cos 2\theta - \frac{R^4}{r^4}$$

For the purposes of this project, the velocities were converted into Cartesian coordinates using the typical cylindrical coordinate transforms

Adding Noise

In the real-world experimental setting, we expect the data collected to be noisy. To model this, we assume additive white gaussian noise in the measurements of u such that for a given noise level η, we have the standard deviation of the additive noise given by

$$\sigma = \eta \ \mathrm{std}(u) \quad \text{so} \quad \boldsymbol{u}_{\text{noisy}} = \boldsymbol{u} + \boldsymbol{\epsilon} \qquad \epsilon_i \sim \mathcal{N}(0, \sigma)$$

Numerical Differentiation and Filtering

Since the data we a dealing with comes from a spatio-temporal domain, it is required that we analyze the spatial derivatives of the data in order to build an accurate model of the system dynamics. In an experimental setting, the spatial derivatives of the data cannot directly be measured, so instead, we must compute the derivatives numerically using a central difference formula which gives the approximation

$$\frac{\partial u_i}{\partial x} \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x}$$

where i represents the grid index in the spatial dimension. The same can be done in the time dimension.

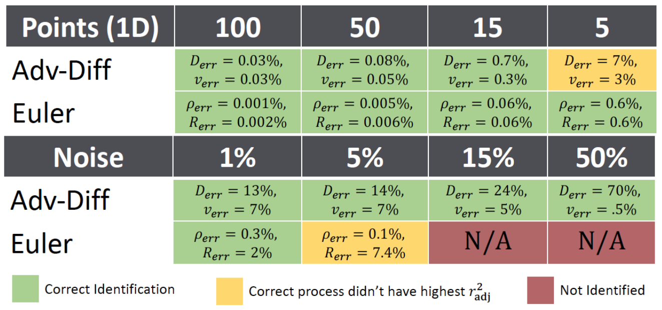
In the case of a noisy measurement, taking the numerical derivative will amplify the noise more than the signal itself so we need a way to denoise the signal before taking the derivative. One excellent way of denoising a signal is to use total-variation regularization (Rudin 1992) which finds the signal u_n such that the objective function $E(x_n, u_n) + \lambda V(u_n)$ is minimized. x_n is the noisy input signal and

$$E(x, u) = \frac{1}{2}\sum_n (x_n - y_n)^2 \quad \text{and} \quad V = \sum_n |y_{n+1} - y_n|$$

are the mean squared error and the total variation respectively. The function `imROF` was used from the julia package `Images.jl` to perform this filtering on the sampled data and after each differentiation is performed. The regularization parameter λ was set to 2. The effect of total-variation filtering on the numerical first derivative is showed in figure 2c, where we can see that noise in the observed data leads to amplification when taking the derivative. After using total variation, denoising, however, the signal is closely recovered.

Test System	Observations	y	Grammar	x_1-Range	x_2-Range	t-Range	t Points
Adv-Diff w/ $D = 1$, $v = 0.5$	u	$\partial u/\partial t$	Fig. 1a	(0,1)	N/A	(0,1)	200
2D Euler w/ $\rho = 1$, $U_\infty = 1$	u, v, p	$\partial p/\partial x$	Fig. 1a less $\div$, $\sqrt{}$	(1,3)	(-1,1)	N/A	N/A

Table 1: Parameters used for system identification of test systems



Points (1D)	100	50	15	5
Adv-Diff	🟩 $D_{err} = 0.03\%$, $v_{err} = 0.03\%$	🟩 $D_{err} = 0.08\%$, $v_{err} = 0.05\%$	🟩 $D_{err} = 0.7\%$, $v_{err} = 0.3\%$	🟨 $D_{err} = 7\%$, $v_{err} = 3\%$
Euler	🟩 $\rho_{err} = 0.001\%$, $R_{err} = 0.002\%$	🟩 $\rho_{err} = 0.005\%$, $R_{err} = 0.006\%$	🟩 $\rho_{err} = 0.06\%$, $R_{err} = 0.06\%$	🟩 $\rho_{err} = 0.6\%$, $R_{err} = 0.6\%$
Noise	**1%**	**5%**	**15%**	**50%**
Adv-Diff	🟩 $D_{err} = 13\%$, $v_{err} = 7\%$	🟩 $D_{err} = 14\%$, $v_{err} = 7\%$	🟩 $D_{err} = 24\%$, $v_{err} = 5\%$	🟩 $D_{err} = 70\%$, $v_{err} = .5\%$
Euler	🟩 $\rho_{err} = 0.3\%$, $R_{err} = 2\%$	🟨 $\rho_{err} = 0.1\%$, $R_{err} = 7.4\%$	🟥 N/A	🟥 N/A

🟩 Correct Identification 🟨 Correct process didn't have highest r^2_{adj} 🟥 Not Identified

Figure 3: Results form the two model problems with variations of the amount of spatial resolution in the data and the amount of noise in the data

Results

The system identifier was run on both test systems with varying levels of spatial resolution (spatial sample points) and amount of noise in the data. The parameters used for each system are shown in table 1.

In the trials for varying the spatial resolution, the data was produced without noise and the number of sample points per spatial dimension was varied from 100 down to 5. When varying the noise, the spatial resolution was fixed at 100 points per dimension and the amount of noise was varied from 1% to 50%. For each configuration, the system identifier was scored on if it induced the correct form of the underlying PDE (i.e. it chose the correct features) and to what degree it computed the correct coefficients of the features. The results of these tests are shown in figure 3. The color of the cell indicates if the select system identifier got the form of the PDE correct. Green means that the highest scoring set of features matched the PDE, yellow means that the correct features were found in the top k expressions but did not score as the highest, and red means the expression was not found at all. The percent error in the coefficients is reported in each cell where the correct PDE expression was induced.

We can see from the data that the system is very robust in the low-data limit and is moderately robust to noisy data. The correct form of the PDE was determined for all spatial resolutions tested – likely because the solutions are smooth enough that an accurate derivative can still be computed with largely spaced sample points. The 1D advection-diffusion equation was always determined correctly even in the presence of large amounts of noise, and only the system parameters suffered in accuracy. The 2D Euler equation did not work as well with noisy data, likely because the noise filtering was less effective in higher dimensions.

Conclusions and Future Work

From this initial investigation we conclude that this system identification system can work for systems that are are unsteady,nonlinear, and have multiple dimensions. Identification is feasible in the low-data limit and for moderate amounts of noise in the observed data. The two main challenges of a system identifier are denoising, and feature selection. The use of a total variation denoiser and a domain specific grammar do much to alleviate these issues but more work can be done on improving the denoising filter when taking spatial derivatives, as well as implementing better algorithms for producing candidate features.

1 References

- Anderson, J. D. (1984). Fundamentals of aerodynamics. McGraw-Hill Companies.
- Bridewell, W., Langley, P., Todorovski, L., & Deroski, S. (2008). Inductive process modeling. Machine learning, 71(1), 1-32.
- Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proceedings of the National Academy of Sciences, 201517384.
- Kochenderfer, M., & Wheeler, T. (2018). Algorithms for Optimization. MIT Press
- Langley, P., Shiran, O., Shrager, J., Todorovski, L., & Pohorille, A. (2006). Constructing explanatory process models from biological data and knowledge. Artificial Intelligence in Medicine, 37(3), 191-201.
- Rudin, L. I., Osher, S., & Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. Physica D: nonlinear phenomena, 60(1-4), 259-268.
- Van Genuchten, M. T., & Alves, W. J. (1982). Analytical solutions of the one-dimensional convective-dispersive solute transport equation (No. 157268). United States Department of Agriculture, Economic Research Service.
- https://en.wikipedia.org/wiki/List_of_nonlinear_partial_differential_equations, 2018