Software paper for submission to the Journal of Open Research Software

To complete this template, please replace the blue text with your own. The paper has three main sections: (1) Overview; (2) Availability; (3) Reuse potential.

Please submit the completed paper to: editor.jors@ubiquitypress.com

---

(1) Overview

Title

ERFit: Entropic Regression Fit Matlab Package, for Data-Driven System Identification of Underlying Dynamic Equations

Paper Authors

1. AlMomani, Abd AlRahman R.
2. Bollt, Erik M.

Paper Author Roles and Affiliations

1. Clarkson Center for Complex Systems Science (C^3S^2), and Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY, 13699. 2. Clarkson Center for Complex Systems Science (C^3S^2), and Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY, 13699.

Abstract

Data-driven sparse system identification becomes the general framework for a wide range of problems in science and engineering. It is a problem of growing importance in applied machine learning and artificial intelligence algorithms. In this work, we developed the Entropic Regression Software Package (ERFit), a MATLAB package for sparse system identification using the entropic regression method. The code requires minimal supervision, with a wide range of options that make it adapt easily to different problems in science and engineering. The ERFit is available at `https://github.com/almomaa/ERFit-Package`.

Keywords

Data-Driven, System Identification, Machine Learning, Mathematical Modeling, Nonlinear Dynamics, Sparse Regression, Information Theory, Causality Inference, Entropic Regression.

Introduction

In this paper, we introduce an efficient Matlab implementation for the Entropic Regression method we introduced in AlMomani et al. (2020), AlMomani (2019). Entropic Regression (ER), is a data-driven discovery method for the underlying dynamics using sparse system identification. ER uses the conditional mutual information as an information-theoretic criterion and iteratively select relevant basis functions in a greedy search optimization scheme in terms of information criterion objective. Consider the problem in the matrix form:

$$\dot{X} = F(X) = \Phi(X)\beta \tag{1}$$

where $X \in \mathbb{R}^{N \times d}$ is the measured state variables of the d-dimensional system with N observations, $\dot{X}$ is the vector field estimated from X, $\Phi : \mathbb{R}^{N \times d} \mapsto \mathbb{R}^{N \times K}$ is a function that maps the state variables X, to expanded set of candidate functions (this general form in Eq. 1, cores a wide range ordinary differential equations, writing the vector field as a linear combination of possibly nonlinear basis functions) Carleman (1932), and $\beta \in \mathbb{R}^{K \times d}$ is the parameters matrix. For the sake of clarity we will write $\Phi(X)$ as Φ in the following discussion.

The basis functions do not need to be mutually orthogonal, and this was the main theme in previous approaches on nonlinear SID, with different methods differ mainly on how a model's fit is quantified Crutchfield & McNamara (1987), Hamilton et al. (2015). The different approaches include using standard squared error measures Yao & Bollt (2007), Chen et al. (1989), sparsity-promoting methods Kalouptsidis et al. (2011), Brunton et al. (2016a), Wang et al. (2011, 2016), Kaiser et al. (2018), Brunton et al. (2016b), Tran & Ward (2017) as well as using entropy-based cost functions Guo et al. (2008). Among those, sparsity-promoting methods have proven particularly useful because they tend to avoid the issue of overfitting, thus allowing a large number of basis functions to be included to capture possibly rich dynamical behavior Kalouptsidis et al. (2011), Brunton et al. (2016a), Wang et al. (2011).

The inverse problem is then: Given Φ and $\dot{X}$, find β. The Best Linear Unbiased Estimator (BLUE) Henderson (1975), of the parameters matrix is known to be the least squares solution given by:

$$
\begin{aligned}
\mathcal{L}(\dot{X}, \Phi) &= (\Phi^T \Phi)^{-1} \Phi^T \dot{X} \\
&= \Phi^\dagger \dot{X}
\end{aligned} \tag{2}
$$

where $\Phi^\dagger$ is the pseudoinverse of the matrix Φ. The reconstructed vector field using the least squares solution is given by:

$$
\begin{aligned}
\mathcal{V}(\dot{X}, \Phi) &= \Phi \Phi^\dagger \dot{X} \\
&= \Phi \mathcal{L}(\dot{X}, \Phi)
\end{aligned} \tag{3}
$$

Sparse Regression problem now finding the minimal set of index $s \subset \{1, 2, ..., K\}$, such that $\mathcal{V}(\dot{X}, \Phi_s)$ as close as possible for $\dot{X}$ according to adopted quality measure (or

objective function, cost function, loss function), where $\Phi_s \subset \Phi$ is the matrix with only the columns with index $i \in s$.

The ER method is a greedy search optimization method that contains two stages: Forward ER and Backward ER; in both stages, selection and elimination of basis functions are based on an entropy criterion (conditional mutual information).

Forward Selection

In the forward stage, our objective is to select the subset $s \subset \mathcal{S} = \{1, 2, ..., K\}$, that represent a strong candidate functions. Starting from empty set $s_0 = \emptyset$, the forward selection stage can be written as:

$$
\begin{aligned}
u_k &= arg \max_{i \in \mathcal{S}, i \notin s_{k-1}} I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{s_{k-1}})), \\
s_k &= s_{k-1} + u_k
\end{aligned}
\tag{4}
$$

where $k = 1, ...,$ is the iteration index, u_k is the set of index with the maximum objective function value. Note that $s_0 = \emptyset \implies \mathcal{V}(\dot{X}, \Phi_{s_0}) = \emptyset$ which reduces the conditional mutual information $I(\cdot; \cdot | \cdot)$ to the mutual information $I(\cdot; \cdot)$. The forward stage have a reward function, where at each iteration k, given the information $(\mathcal{V}(\dot{X}, \Phi_{s_{k-1}}))$ we already have from the set s_{k-1}, we are looking for the function that maximally add extra information to the model. The process terminates when the termination condition $HLT1$ met, which we will describe in the following sections.

Backward Elimination

After the termination of the forward selection, we have the set s that has the indices of the strong candidate functions. Eventually, s may have a few non-relevant functions indices that are selected due to a high degree of uncertainty and the rounding error at the end of forward ER. Since we have reduced set of functions indices ($card(s) << K$), it would be inexpensive to perform a validation operation to ensure the accuracy of the model, and the backward ER represent this operation. The backward stage is an elimination stage, where the functions indexed by s re-examined for their information-theoretic relevance and these that are redundant will be removed. In particular, we label the set s as initial set $s_0 = s$ for the backward stage, and we perform the following computations and updates,

$$
\begin{aligned}
u_k &= arg \min_{i \in s_{k-1}} I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{\{s_{k-1}-i\}})), \\
s_k &= s_{k-1} - u_k.
\end{aligned}
\tag{5}
$$

The backward stage has a loss function, where at each iteration k, we examine information that will potentially be lost if we remove the index i from the set s_{k-1}. We continue the elimination process until the termination condition $HLT2$ met.

The result of the backward ER is a set of indices s. We emphasize here that the forward ER stage can greatly reduce the computational complexity of the backward stage, by limiting the elimination search space to a few candidate functions. However, the backward elimination has a lower rate of error than the forward selection, and in case we having a low-dimension system, or we have efficient computations resources, we can skip the forward stage, and apply the backward stage directly with initial set $s = \{1, 2, \ldots, K\}$, and we provide this option in our software package.

The corresponding parameters $\beta \in \mathbb{R}^K$, can be found by updating the vector of zeros $\beta = \mathbf{0}_K$ such that:

$$\beta_s = \mathcal{L}(\dot{X}, \Phi_s) \tag{6}$$

where β_s is the entries of β indexed by the elements of s, which gives that $\|\beta\|_0 = card(s)$. Note that the ER focus on finding the optimal set of basis functions in terms of the conditional mutual information with the vector field, and after finding this set, we find the value of the parameters by the ordinary least squares, without any attempts to apply any advanced techniques for optimizing the parameter's magnitude. In our package, we provide the function (`getSystemHandle.m`), which translate the estimated parameters to a system of ODEs, and return a function handle of the ODE function.

Termination Conditions

In theory, the mutual information $I(x; y|z)$ is always non-negative and equals zero if and only if x and y are statistically independent given z. However, in practice, due to finite sampling and estimation inaccuracies, the estimated mutual information does not always equal to zero even when x and y are independent and can be negative. Thus, one needs a way to decide whether x and y should be deemed independent given the estimated value of $I(x; y|z)$. In Sun et al. (2014), the authors introduced a standard shuffle test, with a "confidence" parameter $\alpha \in [0, 1]$ for tolerance estimation. The shuffle test requires random shuffling of one of the variables n_s times, to build a test statistic. In particular, for the i-th random shuffle, a random permutation $\pi^{(i)} : [T] \rightarrow [T]$ is generated to shuffle one of the variables, say y, which produces a new variable $(\tilde{y}^{(i)})$ where $\tilde{y}^{(i)} = y_{\pi^{(i)}}$; x and z are kept the same. Then, we estimate the mutual information $I(x; \tilde{y}^{(i)}|z)$ using the (partially) permuted variable $(x, \tilde{y}^{(i)}, z)$, for each $i = 1, \ldots, n_s$. For given α, we then compute a threshold value $I_\alpha(x; y|z)$ as the α-percentile from the values of $I(x; \tilde{y}^{(i)}|z)$. If $I(x; y|z) > I_\alpha(x; y|z)$, we conclude that x and y as dependent given z; otherwise independent. This threshold (tolerance), of mutual Independence is adopted in the forward selection and backward elimination stages as the termination condition. We can interpret the tolerance as the minimum effective quantity of information. In this sense, in the forward ER we are selecting the functions as well as they add a significant quantity of information to the model, while in the backward ER, we discarding functions as well as the information added by them is below the minimum effective quantity, or negligible.

Let *tol* bel the estimated tolerance of mutual Independence, then we set the termina-

tion condition in the backward elimination to:

$$HLT2 = \begin{cases} 1, & \text{IF } I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{\{s_{k-1}-i\}})) > tol, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

where $I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{\{s_{k-1}-i\}}))$ follows from Eq. 5.

While in the forward selection stage, we further require the following. Let $I_a = I(\dot{X}; \mathcal{V}(\dot{X}, \Phi))$ be the mutual information between $\dot{X}$, and the recovered signal $\mathcal{V}(\dot{X}, \Phi)$ using **all** the candidate functions, and let $I_s = I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_s))$ be the mutual information between $\dot{X}$, and the recovered signal $\mathcal{V}(\dot{X}, \Phi_s)$ using the candidate functions indicated by the subset of indices s at the end of each iteration of the forward selection. Then, at any iteration, if $(I_a - I_s) < tol$, that indicates that no other function than what has already been included in the subset s contains significant information if included. So, we formulate the termination condition for the forward selection as:

$$HLT1 = \begin{cases} 1, & \text{IF } I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{s_{k-1}})) < tol, \text{ OR } (I_a - I_s) < tol \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where $I(\dot{X}; \mathcal{V}(\dot{X}, \Phi_i) | \mathcal{V}(\dot{X}, \Phi_{s_{k-1}}))$ is what indicated in Eq. 4.

Implementation and architecture

Fig. 1 shows the general structure of the ERFit software. We provide a wide range of options and adaptability of our implementation, helping the users (researchers) easily use the package to discover underlying dynamics in science and engineering. Moreover, we make it possible for the researchers to explore and investigate different alternatives by allowing user-defined functions for the mutual information estimator, derivative estimator, and parameter magnitude estimator based on the recovered sparse structure.

Detailed comments and user guide (using MATLAB live script) are provided within the code, to explain every step of the algorithm, and all possible choices for the options.

Quality control and Benchmarking

The Entropic Regression implementation have been tested on a wide range of systems with varying number of dimensions such as Logistic map (1D), Lorenz (3D), Rossler (3D), coupled network of Logistic map (100D), Kuramoto-Sivenshsky Equations (16D), coupled network of Lorenz system (300D). Note that the dimension mentioned here is the system dimension, which is highly increased after the construction of the candidate functions Library. For example AlMomani (2019), AlMomani et al. (2020), the 16D of Kuramoto-Sivashinsky equation, will result with about 150 dimensions for the regression problem, and the 300D of the Lorenz system network (100 nodes, 3D each), considering the second order power polynomial expansion, will result with 45000+ dimensions for the regression problem.

In our implementation, we provide a detailed user guide using the ERFit package, and we showed the results for different standard benchmarking problems such as time
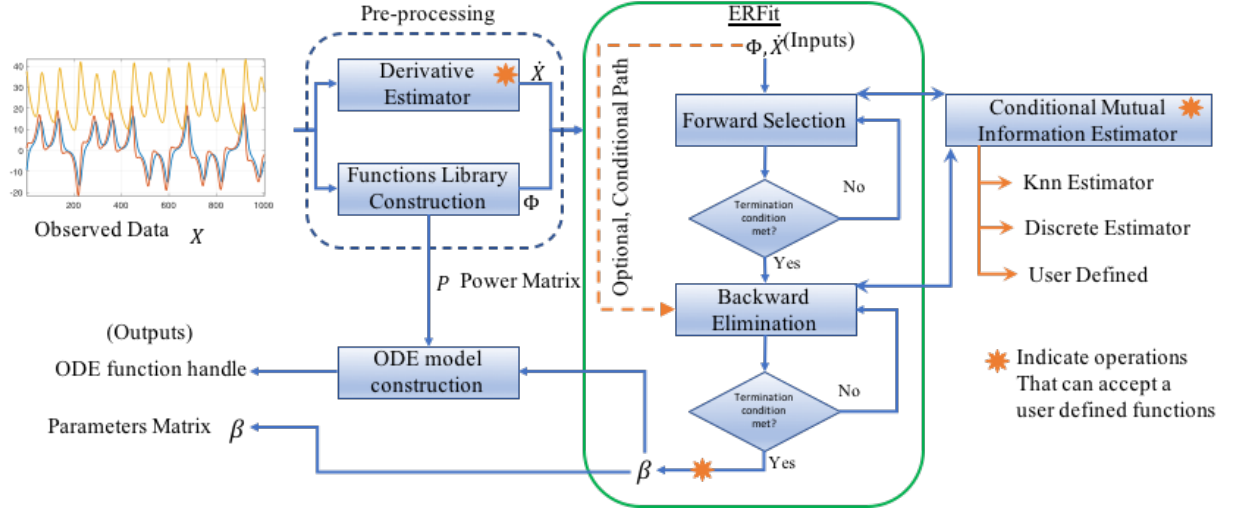
Figure 1: Process flow of the Entropic Regression package (ERFit)

series from Lorenz system and Rossler attractor. In Fig. 2 we present two summarized examples from Rossler and Van der Pol attractors.

(2) Availability

Operating system

The operating system requirements are subject to MATLAB version requirements, which discussed in the Programming language section.

Programming language

This code created with MATLAB 2018b. However, for the ERFit package, we emphasize the use of the basic operations while minimal using built-in functions, and according to Matlab release notes (can be found by running the syntax: `web(fullfile(docroot, 'matlab/release-notes.html'))`, in the command line), the ERFit package is compatible with MATLAB 2016a and newer versions.

Additional system requirements

There are no additional requirements. However, for high dimensional problems, it is recommended to have at least 8GB RAM.

Software location:

Code repository: GitHub.
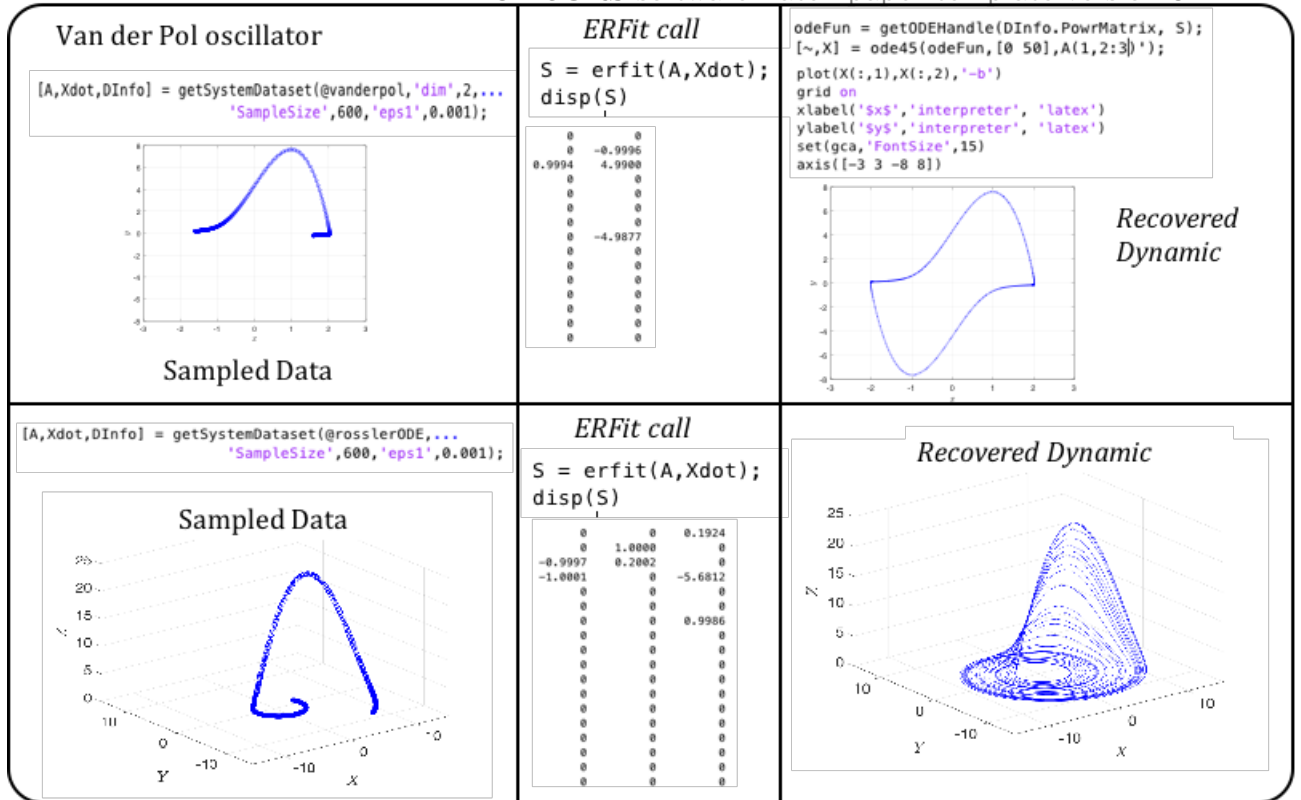
Name: ERFit-Package.

Figure 2: Two illustrative examples for the use and performance of the ERFit.

Persistent identifier: https://github.com/almomaa/ERFit-Package
Licence: GNU GENERAL PUBLIC LICENSE
Date published: 01/06/20

Language

English

(3) Reuse potential

In a wide range of scientific fields, such as but not limited to biology, epidemiology, chemistry, physics, control systems, and causality inference, a core objective is the data-driven discovery of the underlying dynamics by the construction of a mathematical model that helps to analyze and predict the observed system. The `erfit` can be used to construct such a reduced dimension mathematical model. It takes the time series observed in a specific experimental setting and produces a reduced dimension mathematical model that is ready to plug-in to the ode solver and build analysis and predictions. Although the `erfit` requires minimal supervision and experience from the user, we advance and facilitate the possibility that the researchers explore the different variations with a wide range of options. We made it very simple and straight forward to use a user-defined function for the derivative estimation, mutual information estimator, conditional mutual information estimator, and the parameters magnitude estimator based on the recovered

sparse structure.

Funding statement

Competing interests

The authors declare that they have no competing interests.

References

AlMomani, A. A. R. (2019), Prediction Analysis and System Identification of Complex Systems, PhD thesis, Clarkson University.

AlMomani, A. A. R., Sun, J. & Bollt, E. (2020), 'How entropic regression beats the outliers problem in nonlinear system identification', *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30**(1), 013107.

Brunton, S. L., Proctor, J. L. & Kutz, J. N. (2016*a*), 'Discovering governing equations from data by sparse identification of nonlinear dynamical systems', *Proceedings of the National Academy of Sciences* **113**(15), 3932–3937.
URL: *https://www.pnas.org/content/113/15/3932*

Brunton, S. L., Proctor, J. L. & Kutz, J. N. (2016*b*), 'Sparse identification of nonlinear dynamics with control (sindyc)', *IFAC-PapersOnLine* **49**(18), 710–715.

Carleman, T. (1932), 'Application de la théorie des équations intégrales linéaires aux systèmes d'équations différentielles non linéaires', *Acta Math.* **59**, 63–87.
URL: *http://dx.doi.org/10.1007/BF02546499*

Chen, S., Billings, S. A. & Luo, W. (1989), 'Orthogonal least squares methods and their application to non-linear system identification', *International Journal of Control* **50**(5), 1873–1896.
URL: *https://doi.org/10.1080/00207178908953472*

Crutchfield, J. P. & McNamara, B. S. (1987), 'Equations of motion from a data series', *Complex systems* **1**(417-452), 121.

Guo, L.-z., Billings, S. A. & Zhu, D. (2008), 'An extended orthogonal forward regression algorithm for system identification using entropy', *International Journal of Control* **81**(4), 690–699.

Hamilton, F., Berry, T. & Sauer, T. (2015), 'Predicting chaotic time series with a partial model', *Physical Review E* **92**(1), 010902.

Henderson, C. R. (1975), 'Best linear unbiased estimation and prediction under a selection model', *Biometrics* pp. 423–447.

Kaiser, E., Kutz, J. N. & Brunton, S. L. (2018), 'Sparse identification of nonlinear dynamics for model predictive control in the low-data limit', *Proceedings of the Royal Society A* **474**(2219), 20180335.

Kalouptsidis, N., Mileounis, G., Babadi, B. & Tarokh, V. (2011), 'Adaptive algorithms for sparse system identification', *Signal Processing* **91**(8), 1910 – 1919.
URL: *http://www.sciencedirect.com/science/article/pii/S0165168411000697*

Sun, J., Cafaro, C. & Bollt, E. M. (2014), 'Identifying the coupling structure in complex systems through the optimal causation entropy principle', *Entropy* **16**(6), 3416–3433.

Tran, G. & Ward, R. (2017), 'Exact recovery of chaotic systems from highly corrupted data', *Multiscale Modeling & Simulation* **15**, 1108–1129.

Wang, W.-X., Lai, Y.-C. & Grebogi, C. (2016), 'Data based identification and prediction of nonlinear and complex dynamical systems', *Physics Reports* **644**, 1–76.
URL: *http://www.sciencedirect.com/science/article/pii/S037015731630134X*

Wang, W.-X., Yang, R., Lai, Y.-C., Kovanis, V. & Grebogi, C. (2011), 'Predicting catastrophes in nonlinear dynamical systems by compressive sensing', *Physical Review Letters* **106**(15).

Yao, C. & Bollt, E. M. (2007), 'Modeling and nonlinear parameter estimation with kronecker product representation for coupled oscillators and spatiotemporal systems.', *Physica D* **1**(227), 78–99.

REFERENCES