PySensors: A Python package for sparse sensor placement

7 October 2020

Extra material added in by Brian

TODO: remove extra material

What should the paper contain?

From the JOSS submission guide:

"JOSS welcomes submissions from broadly diverse research areas. For this reason, we require that authors include in the paper some sentences that explain the software functionality and domain of use to a non-specialist reader. We also require that authors explain the research applications of the software. The paper should be between 250-1000 words.

Your paper should include:

- A list of the authors of the software and their affiliations, using the correct format (see the example below).
- A summary describing the high-level functionality and purpose of the software for a diverse, non-specialist audience.
- A clear Statement of Need that illustrates the research purpose of the software.
- A list of key references, including to other software addressing related needs.
- Mention (if applicable) a representative set of past or ongoing research projects using the software and recent scholarly publications enabled by it.
- Acknowledgment of any financial support."

See also the review checklist to get an idea of what the reviewers are looking for.

Compiling this document

Make sure you have pandoc installed.

```
pandoc -s --bibliography paper.bib --filter pandoc-citeproc paper.md -o paper.pdf
```

Code

You can include code as follows

```
def fib(n):
    if n < 2:
        return n
    else:
        return fib(n - 1) + fib(n - 2)
```

Other examples

Here are some other example JOSS papers:

- Example on JOSS website
- pyomeca - this one is a bit long, but shows how figures can be incorporated into a JOSS submission
- All published papers

Summary

TODO: write summary section

Scientists have long quantified empirical observations by developing mathematical models that characterize the observations, have some measure of interpretability, and are capable of making predictions. Dynamical systems models in particular have been widely used to study, explain, and predict system behavior in a wide range of application areas, with examples ranging from Newton's laws of classical mechanics to the Michaelis-Menten kinetics for modeling enzyme kinetics. While governing laws and equations were traditionally derived by hand, the current growth of available measurement data and resulting emphasis on data-driven modeling motivates algorithmic approaches for model discovery. A number of such approaches have been developed in recent years and have generated widespread interest, including Eureqa (Schmidt and Lipson 2009), sure independence screening and sparsifying operator (Ouyang et al. 2018), and the sparse identification of nonlinear dynamics (SINDy) (S. L. Brunton, Proctor, and Kutz 2016). Maximizing the impact of these model discovery methods requires tools to make them widely accessible to scientists across domains and at various levels of mathematical expertise.

`PySINDy` is a Python package for the discovery of governing dynamical systems models from data. In particular, `PySINDy` provides tools for applying the SINDy approach to model discovery (S. L. Brunton, Proctor, and Kutz 2016). Given data in the form of state measurements $\mathbf{x}(t) \in \mathbb{R}^n$, the SINDy method seeks a function $\mathbf{f}$ such that

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t)).$$

SINDy poses this model discovery as a sparse regression problem, wherein relevant terms in $\mathbf{f}$ are selected from a library of candidate functions. Thus, SINDy models balance accuracy and efficiency, resulting in parsimonious models that avoid overfitting while remaining interpretable and generalizable. This approach is straightforward to understand and can be readily customized using different sparse regression algorithms or library functions.

The `PySensors` package can be used by both researchers looking to advance the state of the art and practitioners seeking simple sparse sensor selection methods for their applications of interest. Simple methods and abundant examples help new users to hit the ground running. At the same time modular classes leave flexibility for users to experiment with and plug in new sensor selection algorithms or dimensionality reduction techniques. Users of `scikit-learn` will find `Pysensors` syntax familiar and intuitive. The package is fully compatible with `scikit-learn` and follows object-oriented design principles.

The SINDy method has been widely applied for model identification in applications such as chemical reaction dynamics (Hoffmann, Fröhner, and Noé 2019), nonlinear optics (Sorokina, Sygletos, and Turitsyn 2016), thermal fluids (Loiseau 2019), plasma convection (Dam et al. 2017), numerical algorithms (Thaler, Paehler, and Adams 2019), and structural modeling (Lai and Nagarajaiah 2019). It has also been extended to handle more complex modeling scenarios such as partial differential equations (Schaeffer 2017; Rudy et al. 2017), systems with inputs or control (Kaiser, Kutz, and Brunton 2018), corrupt or limited data (Tran and Ward 2017; Schaeffer, Tran, and Ward 2018), integral formulations (Schaeffer and McCalla 2017; Reinbold, Gurevich, and Grigoriev 2020), physical constraints (Loiseau and Brunton 2018), tensor representations (Gelß et al. 2019), and stochastic systems (Boninsegna, Nüske, and Clementi 2018). However, there is not a definitive standard implementation or package for applying SINDy. Versions of SINDy have been implemented within larger projects such as `sparsereg` (Quade 2018), but no specific implementation has emerged as the most widely adopted and most versions implement only a limited set of features. Researchers have thus typically written their own implementations, resulting in duplicated effort and a lack of standardization. This not only makes it more difficult to apply SINDy to scientific data sets, but also makes it more challenging to benchmark extensions to the method against the original and makes such extensions less accessible to end users. The `PySINDy` package provides a dedicated central codebase where many of the basic SINDy features are implemented, allowing for easy use and standardization. This also makes it straightforward for users to extend the package in a way such that new developments are available to a wider user base.

Features

`PySensors` enables the sparse placement of sensors for two classes of problems: reconstruction and classification. For reconstruction problems the package im-

plements a unified `SensorSelector` class, with methods for efficiently analyzing the effects data or sensor quantity have on reconstruction performance. Often different sensor locations impose variable costs, e.g. if measuring sea-surface temperature, it may be more expensive to place buoys/sensors in the middle of the ocean than close to shore. These costs can be taken into account during sensor selection via a built-in cost-sensitive optimization routine (Clark et al. 2018). For classification tasks, the package implements the Sparse Sensor Placement Optimization for Classification (SSPOC) algorithm (B. W. Brunton et al. 2016), allowing one to optimize sensor placement for classification accuracy. This SSPOC implementation is fully general in the sense that it can be used in conjunction with any linear classifier. Additionally, `PySensors` provides methods to enable straightforward exploration of the impacts of primary hyperparameters.

It is well known (Manohar et al. 2018) that the basis in which one represents measurement data can have a pronounced effect on the sensors that are selected and the quality of the reconstruction. Users can readily switch between different bases typically employed for sparse sensor selection, including PCA modes and random projections. Because `PySensors` was built with `scikit-learn` compatibility in mind, it is easy to use cross-validation to select among possible choices of bases, basis modes, and other hyperparameters.

Finally, included with `PySensors` is a large suite of examples, implemented as Jupyter notebooks. Some of the examples are written in a tutorial format and introduce new users to the objects, methods, and syntax of the package. Other examples demonstrate intermediate-level concepts such as how to visualize model parameters and performance, how to combine `scikit-learn` and `PySensors` objects, selecting appropriate parameter values via cross-validation, and other best-practices. Further notebooks use `PySensors` to solve challenging real-world problems. The notebooks reproduce many of the examples from the papers upon which the package is based (Manohar et al. 2018; Clark et al. 2018; B. W. Brunton et al. 2016). To help users begin applying `PySensors` to their own datasets even faster, interactive versions of every notebook are available on Binder. Overall, the examples will compress the learning curve of learning a new software package.

Acknowledgments

TODO: write acknowledgments section

This project is a fork of `sparsereg` (Quade 2018). SLB acknowledges funding support from the Air Force Office of Scientific Research (AFOSR FA9550-18-1-0200) and the Army Research Office (ARO W911NF-19-1-0045). JNK acknowledges support from the Air Force Office of Scientific Research (AFOSR FA9550-17-1-0329). This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant Number DGE-1256082.

References

Boninsegna, Lorenzo, Feliks Nüske, and Cecilia Clementi. 2018. "Sparse Learning of Stochastic Dynamical Equations." *The Journal of Chemical Physics* 148 (24): 241723. https://doi.org/10.1063/1.5018409.

Brunton, Bingni W, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. 2016. "Sparse Sensor Placement Optimization for Classification." *SIAM Journal on Applied Mathematics* 76 (5): 2099–2122.

Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz. 2016. "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems." *Proceedings of the National Academy of Sciences* 113 (15): 3932–7. https://doi.org/10.1073/pnas.1517384113.

Clark, Emily, Travis Askham, Steven L Brunton, and J Nathan Kutz. 2018. "Greedy Sensor Placement with Cost Constraints." *IEEE Sensors Journal* 19 (7): 2642–56.

Dam, Magnus, Morten Brøns, Jens Juul Rasmussen, Volker Naulin, and Jan S. Hesthaven. 2017. "Sparse Identification of a Predator-Prey System from Simulation Data of a Convection Model." *Physics of Plasmas* 24 (2): 022310. https://doi.org/10.1063/1.4977057.

Gelß, Patrick, Stefan Klus, Jens Eisert, and Christof Schütte. 2019. "Multidimensional Approximation of Nonlinear Dynamical Systems." *Journal of Computational and Nonlinear Dynamics* 14 (6). https://doi.org/10.1115/1.4043148.

Hoffmann, Moritz, Christoph Fröhner, and Frank Noé. 2019. "Reactive SINDy: Discovering Governing Reactions from Concentration Data." *Journal of Chemical Physics* 150 (025101). https://doi.org/10.1063/1.5066099.

Kaiser, Eurika, J Nathan Kutz, and Steven L Brunton. 2018. "Sparse Identification of Nonlinear Dynamics for Model Predictive Control in the Low-Data Limit." *Proceedings of the Royal Society of London A* 474 (2219). https://doi.org/10.1098/rspa.2018.0335.

Lai, Zhilu, and Satish Nagarajaiah. 2019. "Sparse Structural System Identification Method for Nonlinear Dynamic Systems with Hysteresis/Inelastic Behavior." *Mechanical Systems and Signal Processing* 117: 813–42. https://doi.org/10.1016/j.ymssp.2018.08.033.

Loiseau, J.-C., and Steven L. Brunton. 2018. "Constrained Sparse Galerkin Regression." *Journal of Fluid Mechanics* 838: 42–67. https://doi.org/10.1017/jfm.2017.823.

Loiseau, Jean-Christophe. 2019. "Data-Driven Modeling of the Chaotic Thermal Convection in an Annular Thermosyphon." *arXiv Preprint arXiv:1911.07920*.

Manohar, Krithika, Bingni W Brunton, J Nathan Kutz, and Steven L Brunton. 2018. "Data-Driven Sparse Sensor Placement for Reconstruction: Demon-

strating the Benefits of Exploiting Known Patterns." *IEEE Control Systems Magazine* 38 (3): 63–86.

Ouyang, Runhai, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M. Ghiringhelli. 2018. "SISSO: A Compressed-Sensing Method for Identifying the Best Low-Dimensional Descriptor in an Immensity of Offered Candidates." *Physical Review Materials* 2 (8): 083802. https://doi.org/10.1103/physrevmaterials.2.083802.

Quade, Markus. 2018. "Sparsereg - Collection of Modern Sparse Regression Algorithms." https://doi.org/10.5281/zenodo.1173754.

Reinbold, Patrick AK, Daniel R Gurevich, and Roman O Grigoriev. 2020. "Using Noisy or Incomplete Data to Discover Models of Spatiotemporal Dynamics." *Physical Review E* 101 (1): 010203. https://doi.org/10.1103/physreve.101.010203.

Rudy, Samuel H, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. 2017. "Data-Driven Discovery of Partial Differential Equations." *Science Advances* 3 (e1602614). https://doi.org/10.1126/sciadv.1602614.

Schaeffer, Hayden. 2017. "Learning Partial Differential Equations via Data Discovery and Sparse Optimization." In *Proceedings of the Royal Society a*, 473:20160446. 2197. The Royal Society. https://doi.org/10.1098/rspa.2016.0446.

Schaeffer, Hayden, and Scott G McCalla. 2017. "Sparse Model Selection via Integral Terms." *Physical Review E* 96 (2): 023302. https://doi.org/10.1103/physreve.96.023302.

Schaeffer, Hayden, Giang Tran, and Rachel Ward. 2018. "Extracting Sparse High-Dimensional Dynamics from Limited Data." *SIAM Journal on Applied Mathematics* 78 (6): 3279–95. https://doi.org/10.1137/18m116798x.

Schmidt, Michael, and Hod Lipson. 2009. "Distilling Free-Form Natural Laws from Experimental Data." *Science* 324 (5923): 81–85. https://doi.org/10.1126/science.1165893.

Sorokina, Mariia, Stylianos Sygletos, and Sergei Turitsyn. 2016. "Sparse Identification for Nonlinear Optical Communication Systems: SINO Method." *Optics Express* 24 (26): 30433–43. https://doi.org/10.1364/oe.24.030433.

Thaler, Stephan, Ludger Paehler, and Nikolaus A Adams. 2019. "Sparse Identification of Truncation Errors." *Journal of Computational Physics* 397: 108851. https://doi.org/10.1016/j.jcp.2019.07.049.

Tran, Giang, and Rachel Ward. 2017. "Exact Recovery of Chaotic Systems from Highly Corrupted Data." *Multiscale Modeling & Simulation* 15 (3): 1108–29. https://doi.org/10.1137/16m1086637.