The Journal of Open Source Software

PySensors 2.0: A Python Package for Sparse Sensor Placement

Niharika Karnik [1], **Yash Bhangale** [1], **Mohammad G. Abdo** [3], **Andrei A. Klishin** [4], **Joshua J. Cogliati** [3], **Bingni W. Brunton** [5], **J. Nathan Kutz** [2], **Steven L. Brunton** [1], **and Krithika Manohar** [1]

1 Department of Mechanical Engineering, University of Washington, USA **2** Department of Applied Mathematics, University of Washington, USA **3** Idaho National Laboratory, USA **4** Department of Mechanical Engineering, University of Hawai'i at Mānoa, USA **5** Department of Biology, University of Washington, USA

DOI: 10.21105/joss.09265

Software

- Review
- Repository
- Archive

Editor: Kanishka B. Narayan

Reviewers:

- @isdanni
- @henrykironde

Submitted: 07 October 2025
Published: 17 February 2026

Summary

PySensors is a Python package for the optimization of sparse sensor placement for reconstruction and classification tasks. This major update to PySensors introduces novel spatial constraints for sensor placement, including the enforcement of maximum or exact sensor counts in specific regions, predetermined sensor locations, and minimum distances between sensors. This functionality is extended to support custom basis inputs, enabling integration of any data-driven or spectral basis. We also use a thermodynamic approach that goes beyond a single "optimal" sensor configuration and maps the complete landscape of sensor interactions induced by the training data. This comprehensive view facilitates integration with external selection criteria and enables assessment of sensor replacement impacts. The new optimization technique also accounts for over- and under-sampling of sensors, utilizing a regularized least squares approach for robust reconstruction. Additionally, we implement noise-induced uncertainty quantification of the estimation error and provide pointwise uncertainty heat maps to guide deployment decisions. To highlight these additions, we outline the mathematical algorithms and theory underlying these new capabilities. The usage of these new features is illustrated with code examples and practical advice for implementation across multiple application domains. Finally, we outline a roadmap of potential extensions to further strengthen the package's functionality and applicability to emerging sensing challenges.

Statement of need

The scalable optimization of sensor placement is critical for efficient monitoring, control, and decision-making in complex engineering systems. Sensor measurements are necessary for real-time estimation of high-dimensional fluid flows (Erichson et al., 2020), large-scale flexible structures (Manohar, Hogan, et al., 2018), and temperature and pressure fields across geophysical (Alonso et al., 2010) and nuclear energy systems (Karnik et al., 2024). Accurate real-time tracking of key system variables depends sensitively on the locations of sensors deployed within the system, which have to be optimized for the desired task. In general, the selection of an optimal subset of sensors among candidate locations is NP-hard, necessitating heuristic or greedy approaches such as compressed sensing (Donoho, 2006) or greedy algorithms that exploit submodularity (Krause et al., 2008).

The Journal of Open Source Software

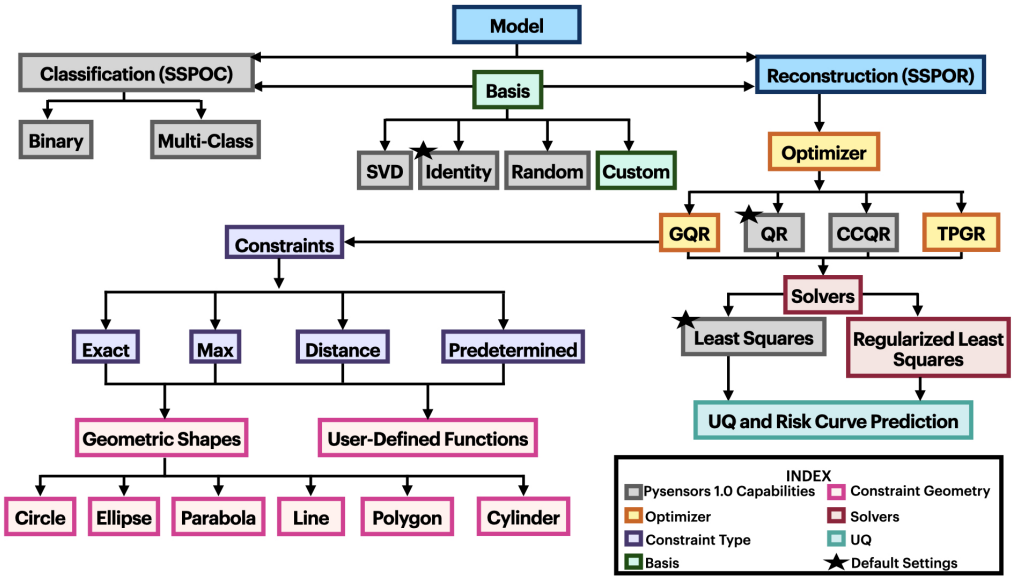

Figure 1: An overview image of capabilities of PySensors

`PySensors` is a Python package (Silva et al., 2021) dedicated to solving the complex challenge of optimal sensor placement in data-driven systems. It implements advanced sparse optimization algorithms that use dimensionality reduction techniques to identify the most informative measurement locations with remarkable efficiency (Brunton et al., 2016; Clark et al., 2020; Manohar, Brunton, et al., 2018). It helps users identify the best locations for sensors when working with complex high dimensional data, focusing on both reconstruction (Manohar, Brunton, et al., 2018) and classification (Brunton et al., 2016) tasks. The package follows `scikit-learn` conventions for user-friendly access while offering advanced customization options for experienced users. Other sensor placement packages such as `Chama` (Klise et al., 2017), `Polire` (Narayanan et al., 2020), and `OSPS` toolbox (Yi et al., 2011) focus primarily on event detection, Gaussian process modeling, and structural health monitoring respectively, while `PySensors` specifically targets signal reconstruction and classification applications.

The original `PySensors` code provided an implementation of `Sparse Sensor Placement for Reconstruction (SSPOR)` using the `QR` optimizer for optimal sensor selection in unconstrained placement settings in which the number of sensors p is equivalent to the number of modes r. The previous version offers only limited support for constraints and for specifying the number of sensors, which restricts its applicability to many real-world problems. The `Cost-Constrained QR (CCQR)` framework incorporates heterogeneous cost functions into the optimization process to accommodate practical deployment constraints. The `Sparse Sensor Placement Optimization for Classification (SSPOC)` framework identifies minimal sensor configurations that can classify high dimensional signals $\mathbf{x} \in \mathbb{R}^n$ as belonging to one of c classes. Data is projected into a reduced spectral or data-driven basis $r \ll n$ to enable efficient optimization. Different basis functions such as `Identity`, `SVD`, and `RandomProjection` implemented in `PySensors` can significantly impact sensor selection effectiveness and reconstruction quality (Manohar, Brunton, et al., 2018).

This new version of `Pysensors` focuses specifically on practical engineering applications where measurement data is inherently noisy and spatial deployment constraints are unavoidable. Key improvements include constraint-aware optimization methods that handle spatial restrictions and sensor density limitations. This version of `PySensors` implements methodologies introduced in Klishin et al. (2023) and Karnik et al. (2024) to make them accessible to scientists and engineers in all domains. These enhancements transform `PySensors` from a purely academic tool into a practical platform for solving real-world sensing challenges while maintaining

mathematical rigor. Our code is thoroughly documented and contains extensive examples. `PySensors` is completely open-source and adheres to PEP8 stylistic standards.

New Features

`PySensors` 1.0 implements two sensor placement methods for reconstruction: (1) `QR`, a greedy optimization technique that places sensors throughout an unconstrained domain, and (2) `CCQR`, a cost-constrained optimization technique that accepts a user-defined vector of sensor costs throughout the domain, penalizing larger-cost locations for sensor placement.

Traditional QR factorization presents challenges for under-sampling ($p < r$) and over-sampling ($p > r$) regimes in which the number of sensors p is strictly greater or less than the model rank r, respectively, and cannot enforce complex spatial constraints. `PySensors 2.0` addresses these limitations through two new optimization algorithms: `Generalized QR (GQR)` and `Two Point GReedy (TPGR)`.

Implementing spatially constrained sensor placement requires a deeper intervention in the underlying QR optimization framework. To address this requirement, we add the generalized QR `GQR` optimization functionality based on recent work (Karnik et al., 2024), which provides the architectural flexibility needed to handle complex spatial constraints. We enhance the algorithm's capabilities by implementing diverse spatial constraints options: `maxn`, `exactn`, `predetermined`, and `distance`. Pre-defined constraint regions include 'Circle', 'Ellipse', 'Polygon', 'Parabola', 'Line', and 'Cylinder'. Additionally, users can define a custom constraint using a `.py` python file or an equation string.

In the context of reduced order models with truncation rank r, over-sampling refers to deploying more sensors than the dimensionality of the reduced subspace (i.e., $p > r$ sensors), while under-sampling involves using fewer sensors than the model rank ($p < r$). The objective that the `QR` optimizer is based on suffers from two limitations: it is not defined for the under-sampling regime, and the underlying optimization is difficult to interpret and visualize directly. Klishin et al. (2023) resolves these with a new optimization technique that accounts for over- and under-sampling scenarios in sensor placement and decomposing the resulting objective into sums over the placed sensors

$$\mathcal{H} \equiv -\log\det(\mathbf{S}^{-2} + (\mathbb{S}\mathbf{\Psi}_r)^T(\mathbb{S}\mathbf{\Psi}_r)/\eta^2) \approx E_b + \sum_{i\in\gamma} h_i + \sum_{i\neq j\in\gamma} J_{ij}$$

The Two Point Greedy (TPGR) optimizer uses the above approximate objective, which allows the user to specify any number of sensors p for a given mode number r. In contrast, the QR pivoting algorithm always returns exactly $p = r$ sensors, ordered by decreasing importance, with any additional sensors selected randomly. Beyond this flexibility, TPGR can also generate sensor placement energy landscapes, providing insight into the relative quality of different sensor configurations.

`PySensors 2.0` adds the Regularized Least Squares solution derived in Klishin et al. (2023):

$$\mathbf{A}_{RLS} = \left(\mathbf{S}^{-2} + (\mathbb{S}\mathbf{\Psi}_r)^T(\mathbb{S}\mathbf{\Psi}_r)/\eta^2\right)^{-1}(\mathbb{S}\mathbf{\Psi}_r)^T/\eta^2,$$

This Regularized Least Squares solution is the new default reconstruction solver for `PySensors 2.0`, replacing the Least Squares solution via the Moore-Penrose pseudoinverse. We implement uncertainty quantification metrics from Klishin et al. (2023) to assess the propagation of measurement noise through reconstruction algorithms and provide robust pointwise error estimates in reconstruction outputs. In addition, this version adds support for custom bases, so that researchers can transform their data into an alternative basis such as dynamic mode decomposition (DMD) modes before passing it to `PySensors`.

Finally, we demonstrate these new functionalities using new notebook examples for nuclear energy component systems, and update and streamline existing notebooks for sea surface temperature and image reconstruction.

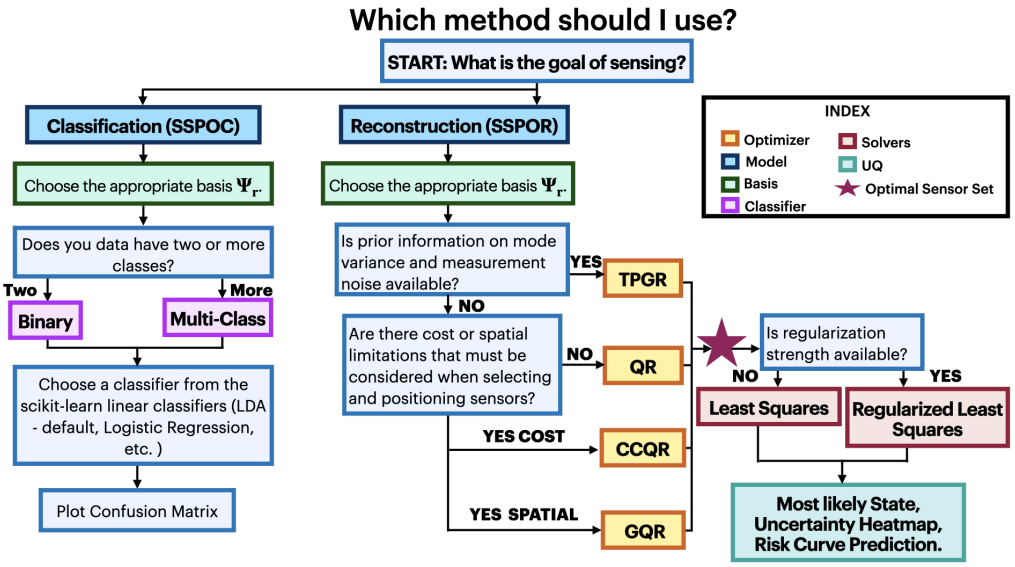

Figure 2: A flowchart to suggest which method to use

Acknowledgments

The authors would like to thank B. de Silva for valuable feedback. The authors acknowledge support from the Boeing Company; the National Science Foundation through the Mathematical Foundations of Digital Twins (MATH-DT) program under Award Nos. 2529361 and 2529362; NSF AI Institute in Dynamic Systems under grant 2112085 and the Idaho National Laboratory (INL) Laboratory Directed Research & Development (LDRD) Program under DOE Idaho Operations Office Contract DE-AC07-05ID14517 for LDRD-22A1059-091FP.

References

Alonso, M. T., López-Dekker, P., & Mallorquí, J. J. (2010). A novel strategy for radar imaging based on compressive sensing. *IEEE Transactions on Geoscience and Remote Sensing*, *48*(12), 4285–4295. https://doi.org/10.1109/TGRS.2010.2051231

Brunton, B. W., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Sparse sensor placement optimization for classification. *SIAM Journal on Applied Mathematics*, *76*(5), 2099–2122. https://doi.org/10.1137/15M1036713

Clark, E., Brunton, S. L., & Kutz, J. N. (2020). Multi-fidelity sensor selection: Greedy algorithms to place cheap and expensive sensors with cost constraints. *IEEE Sensors Journal*, *21*(1), 600–611. https://doi.org/10.1109/JSEN.2020.3013094

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306. https://doi.org/10.1109/TIT.2006.871582

Erichson, N. B., Mathelin, L., Yao, Z., Brunton, S. L., Mahoney, M. W., & Kutz, J. N. (2020). Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, *476*(2238), 20200097. https://doi.org/10.1098/rspa.2020.0097

Karnik, N., Abdo, M. G., Estrada-Perez, C. E., Yoo, J. S., Cogliati, J. J., Skifton, R. S., Calderoni, P., Brunton, S. L., & Manohar, K. (2024). Constrained optimization of sensor placement for nuclear digital twins. *IEEE Sensors Journal*. https://doi.org/10.1109/JSEN.2024.3368875

Klise, K. A., Nicholson, B., & Laird, C. D. (2017). Sensor placement optimization using chama. *Number SAND2017-11472. Albuquerque, NM: Sandia National Laboratories*. https://doi.org/10.2172/1405271

Klishin, A. A., Kutz, J. N., & Manohar, K. (2023). Data-induced interactions of sparse sensors. *arXiv Preprint arXiv:2307.11838*. https://doi.org/10.48550/arXiv.2307.11838

Krause, A., Singh, A., & Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, *9*(2). https://dl.acm.org/doi/10.5555/1390681.1390689

Manohar, K., Brunton, B. W., Kutz, J. N., & Brunton, S. L. (2018). Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, *38*(3), 63–86. https://doi.org/10.1109/MCS.2018.2810460

Manohar, K., Hogan, T., Buttrick, J., Banerjee, A. G., Kutz, J. N., & Brunton, S. L. (2018). Predicting shim gaps in aircraft assembly with machine learning and sparse sensing. *Journal of Manufacturing Systems*, *48*, 87–95. https://doi.org/10.1016/j.jmsy.2018.01.011

Narayanan, S. D., Patel, Z. B., Agnihotri, A., & Batra, N. (2020). A toolkit for spatial interpolation and sensor placement. *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 653–654. https://doi.org/10.1145/3384419.3430407

Silva, B. M. de, Manohar, K., Clark, E., Brunton, B. W., Kutz, J. N., & Brunton, S. L. (2021). PySensors: A Python package for sparse sensor placement. *Journal of Open Source Software*, *6*(58), 2828. https://doi.org/10.21105/joss.02828

Yi, T.-H., Li, H.-N., & Gu, M. (2011). Optimal sensor placement for structural health monitoring based on multiple optimization strategies. *The Structural Design of Tall and Special Buildings*, *20*(7), 881–900. https://doi.org/10.1002/tal.712