FixedPointFinder: A Tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks

Matthew D. Golub[1,2] **and David Sussillo**[1,2,3,4]

1 Department of Electrical Engineering, Stanford University **2** Stanford Neurosciences Institute, Stanford University **3** Google Brain **4** Work done while at Stanford University

DOI: 10.21105/joss.01003

Software

- Review
- Repository
- Archive

Submitted: 17 September 2018
Published: 01 November 2018

Summary

Recurrent neural networks (RNNs) are powerful function approximators that can be designed or trained to solve a variety of computational tasks. Such tasks require the transformation of a set of time-varying input signals into a set of time-varying output signals. Neuroscientists are increasingly interested in using RNNs to explain complex relationships present in recorded neural activity (Pandarinath et al., 2018) and to propose dynamical mechanisms through which a population of neurons might implement a computation (Mante, Sussillo, Shenoy, & Newsome, 2013; Remington, Narain, Hosseini, & Jazayeri, 2018). Once fit to neural recordings or trained to solve a task of interest, an RNN can be reverse-engineered to understand how a computation is implemented in a high-dimensional recurrent neural system, which can suggest hypotheses for how the task might be solved by the brain.

Perhaps the most critical step in this reverse-engineering is identifying the fixed points of the RNN's internal dynamics. These fixed points, along with local linearizations of the RNN dynamics about those fixed points, provide a mechanistic description of how the network implements a computation. Identifying the fixed points of an RNN requires optimization tools that have previously been tailored for specific, "vanilla" or Hopfield-like RNN architectures (Sussillo & Barak, 2013, Katz & Reggia (2018)). These approaches rely on hard-coded analytic derivatives (e.g., of the hyperbolic tangent function) and thus can be cumbersome for complicated models (e.g., gated architectures), for studies that involve multiple different models, or when frequent architectural changes are required during the development of a model. While numerical derivatives can partially alleviate these difficulties, the additional computational costs typically make them infeasible for anything but the smallest of models.

Here, we introduce `FixedPointFinder`, an open-source Tensorflow toolbox for finding fixed points and linearized dynamics in arbitrary RNN architectures. `FixedPointFinder` leverages Tensorflow's efficient automatic differentiation back-end to perform the optimizations and Jacobian calculations required for fixed-point identification and characterization. `FixedPointFinder` works with any arbitrary RNN architecture that conforms to Tensorflow's `RNNCell` API, including popular gated architectures (e.g., long short-term memory (LSTM) or gated recurrent units (GRU)), traditional "vanilla" architectures, as well as custom researcher-developed architectures.

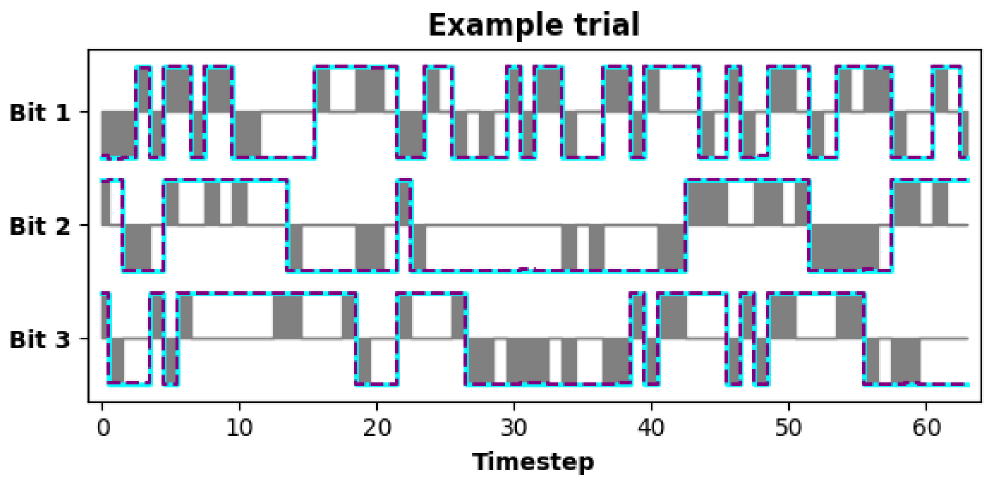

Figure 1: Inputs (gray), target outputs (cyan), and outputs of a trained LSTM RNN (purple) from an example trial of the flip-flop task. Signed input pulses (gray) flip the corresponding bit's state (green) whenever an input pulse has the opposite sign of the current bit state (e.g., if gray goes high when green is low). The RNN has been trained to nearly perfectly reproduce the target memory state (purple closely overlaps cyan).

Example

`FixedPointFinder` includes an end-to-end example that trains a Tensorflow RNN to solve a task and then identifies and visualizes the fixed points of the trained RNN. The task is the "flip-flop" task previously described in Sussillo & Barak (2013). Briefly, the task is to implement a 3-bit binary memory, in which each of 3 input channels delivers signed transient pulses (-1 or +1) to a corresponding bit of the memory, and an input pulse flips the state of that memory bit (also -1 or +1) whenever a pulse's sign is opposite of the current state of the bit. The example trains a 16-unit LSTM RNN to solve this task (Fig. 1). Once the RNN is trained, the example uses `FixedPointFinder` to identify and characterize the trained RNN's fixed points. Finally, the example produces a visualization of these results (Fig. 2). In addition to demonstrating a working use of `FixedPointFinder`, this example provides a testbed for experimenting with different RNN architectures (e.g., numbers of recurrent units, LSTMs vs. GRUs vs. vanilla RNNs) and characterizing how these lower-level model design choices manifest in the higher-level dynamical implementation used to solve a task.

Availability

`FixedPointFinder` is publicly available under the Apache 2.0 license at https://github.com/mattgolub/fixed-point-finder.

Acknowledgements

This work was supported by the Stanford Neurosciences Institute, the Office of Naval Research Grant #N00014-18-1-2158, and the Simons Foundation Grant #543049.

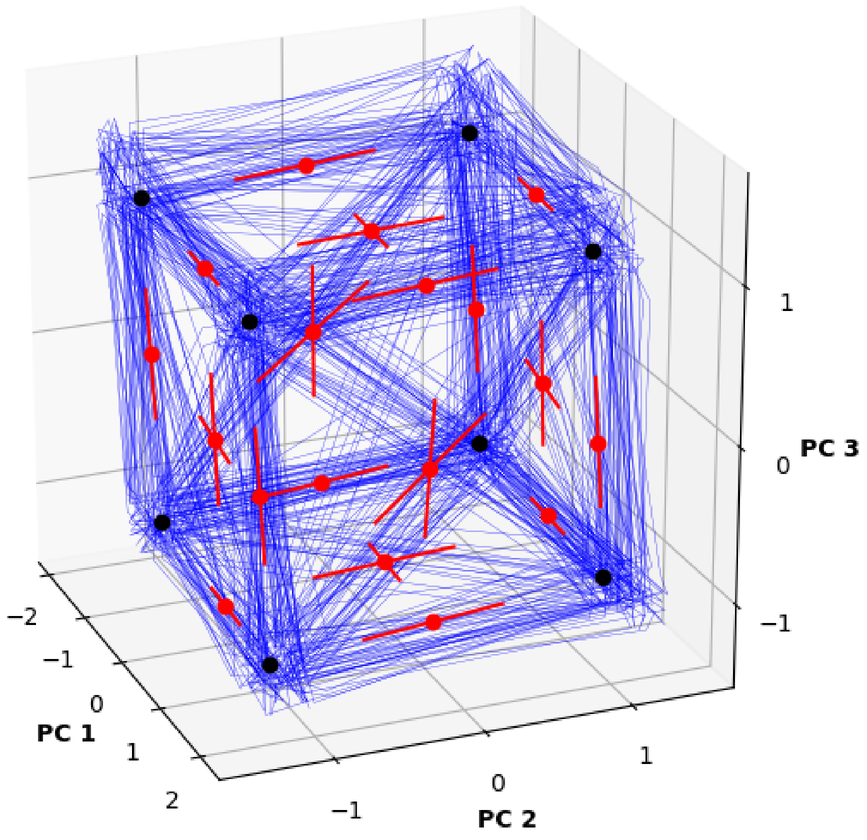

Figure 2: Fixed-point structure of an LSTM RNN trained to solve the flip-flop task. `FixedPointFinder` identified 8 stable fixed points (black points), each of which corresponds to a unique state of the 3-bit memory. `FixedPointFinder` also identified a number of unstable fixed points (red points) along with their unstable modes (red lines), which mediate the set of state transitions trained into the RNN's dynamics. Here, each unstable fixed point is a "saddle" in the RNN's dynamical flow field, and the corresponding unstable modes indicate the directions that nearby states are repelled from the fixed point. State trajectories from example trials (blue) traverse about these fixed points. All quantities are visualized in the 3-dimensional space determined by the top 3 principal components computed across 128 example trials.

References

Katz, G. E., & Reggia, J. A. (2018). Using directional fibers to locate fixed points of recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8), 3636–3646. doi:10.1109/TNNLS.2017.2733544

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78. doi:10.1038/nature12742

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, *15*(10), 805–815. doi:10.1038/s41592-018-0109-9

Remington, E. D., Narain, D., Hosseini, E. A., & Jazayeri, M. (2018). Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, *98*(5), 1005–1019. doi:10.1016/j.neuron.2018.05.020

Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649. doi:10.1162/NECO_a_00409