Google Summer of Code Proposal

Towards Support for Reverse Engineering of Neural Circuits and Non-Linear Dimensionality Reduction in Fovea and PyDSTool

Daniel McNeela
University of California, Berkeley
daniel.mcneela@gmail.com

March 21, 2016

Abstract

Computational models of neural circuitry are increasingly superseding single neuron models for understanding dynamic processes in the brain. Recurrent Neural Networks offer an attractive method for simulating, replicating, and characterizing these circuits. PyDSTool currently supports simulation of neural circuity dynamics and linear methods of dimensionality reduction for normalizing chaotic data such as Principle Component Analysis; however, such methods are insufficiently tailored to working with non-linearly distributed data. I propose the implementation of a module targeted at providing non-linear dimensionality reduction via Sammon mappings, principal curves, and locally linear embeddings. In addition, I will create tools in Fovea that allow for the intuitive visualization of data sets resolved by these methods.

I also seek to provide a suite of tools to PyDSTool for constructing and training Recurrent Neural Networks and corresponding solutions for visualizing the operation, feedback mechanisms, network structure, and dynamical systems characteristics (e.g. attractors) of these models. These networks are well-suited for the modelling of neural circuitry and often output highly nonlinear data which can be subsequently smoothed and visualized using the newly-implemented non-linear dimensionality reduction techniques. The capabilities I seek to develop for Fovea will assist in the understanding and reverse engineering of data generated by these RNNs. While the RNN tools and non-linear dimensionality reduction methods will be implemented with neuroscience applications in mind, their use will be valuable across a wide spectrum of domains.

Contents

1 Contact Information

Full Name: Daniel McNeela
Email: daniel.mcneela@gmail.com
Location: San Diego, CA and Berkeley, CA, USA
Personal Website: mcneela.github.io
Github Profile: http://www.github.com/mcneela

2 Project Details

The project comprises four main components, namely

1. Interfacing with existing algorithms for nonlinear dimensionality reduction in PyDSTool.
2. Providing additional tools in Fovea for visualizing data sets normalized by the aforementioned methods.
3. Implementing tools for building and simulating recurrent neural networks in PyDSTool.
4. Providing neural circuitry visualization capability in Fovea based off the implemented RNN functionality.

I will provide a structural outline for each component individually.

2.1 Implementing Non-Linear Dimensionality Reduction Methods in PyDSTool

I would like to provide a diverse sampling of prominent non-linear dimensionality reduction methods and learning algorithms. These tools can then be used by researchers to preprocess experimental data before feeding it into Fovea for visualization. The following implementations will be created:

Sammon Mapping
Given an initial data set X, this straightforward algorithm defines a new *projection* data set Y, such that the distance d^*_{ij} between data points $x_i, x_j \in X$ and the distance d_{ij} between data points $y_i, y_j \in Y$ is minimized according to the following error function:

$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*ij - d_{ij})^2}{d^*_{ij}}$$

I plan on applying gradient descent to minimize this error. The method will likely utilize PyDSTool's current PCA facility for generating a starting configuration for the gradient descent iteration. Interestingly, some modern implementations of Sammon Mapping enlist neural networks, so time permitting, I will look into optimizing the implementation using the newly-developed RNN capability.

Principal Curve Analysis

While the details of principal curve analysis are more involved than those of Sammon mapping, from a high-level view the procedure involves iterating a function-valued sequence $f_n(x)$ until the minimal average of orthogonal distances between function values and data points is achieved. The resulting curve is typically smooth and passes cleanly through the center of the data set while preserving structural non-linearities. More algorithmic details can be found in [1]

Locally Linear Embedding

The Locally Linear Embedding algorithm works as follows. Assign to each vector-valued data point $\vec{X_i}$ in high-dimensional data set X a set of k neighboring data points using an algorithm such as k *nearest neighbors*. Compute a matrix of weights W defining a linear transformation between X and the projection set Y such that the error function

$$\varepsilon(W) = \sum_i \left| \vec{X_i} - \sum_j W_{ij} \vec{X_j} \right|^2$$

is minimized. After the matrix of weights is determined and fixed, a similar error function is minimized in order to optimize the embedding vectors $\vec{Y_i}$ of the dimensionality reduction map $\varphi : X \to Y$. This error function is given by

$$\phi(Y) = \sum_i \left| \vec{Y_i} - \sum_j W_{ij} \vec{Y_j} \right|^2$$

More details about the algorithm can be found in [2]

2.2 Developing Additional Tools for Visualizing Chaotic Data Sets and their Normalizations in Fovea

Currently, individual Fovea plots are generated in separate windows. For the purpose of fine-tuning dimensionality reduction techniques, it is often useful to be able to generate side-by-side comparisons of multiple plots in a single window, either for plotting different dimensions of a high-dimensional data set (e.g. for a set of vectors of the form $[x_1, x_2, x_3, x_4]$, plotting coordinates x_1, x_2, x_3 in one frame and plotting coordinates x_2, x_3, x_4 in another) or for viewing a single plot from different angles simultaneously. To do this, I will make use of matplotlib's `add_subplot` function to provide support for multiple frames when plotting. I will also add the ability to logically group frames and then synchronously manipulate these frames as a unit using other Fovea visualization capabilities. For example, users will be able to generate a data plot, set offset positions for multiple viewing angles, and then apply relative scaling, rotation, or other transformations to these groups of plots as a whole while still retaining the relative offsets in viewing positions.

In terms of the visualization of various nonlinear dimensionality reduction algorithms, I intend to introduce functionality allowing for dynamic switching between plots generated by different algorithms in realtime as well as GUI options allowing for visual tweaking of these algorithms.

For example, I would implement GUI functionality allowing a user to edit in realtime the metric used in error calculations for Sammon mapping in order to finetune three and two dimensional data projections.

2.3 Implementing Recurrent Neural Networks in PyDSTool

Recurrent Neural Networks provide an ideal framework for the modelling of neural circuitry and associated dynamical systems in neuroscience. They work by accepting a sequence of data as input, feeding this data through an arbitrary number of computational layers, and generating an output data sequence based on learning acquired during network training. The defining characteristic of a RNN in comparison to other neural networks is the ability of component network computational units to accept as input feedback from the outputs of other member units. This capability endows RNNs with a form of "memory," allowing for highly complex and dynamic computations. The general definition of an RNN in the neuroscience contex is based off the equation

$$\tau\dot{\mathbf{x}}(t) = -\mathbf{x}(t) + \mathbf{J}\mathbf{r}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{b}$$

Each of the named functions is vector-valued. The ith component of $\mathbf{x}$ can be viewed as the synaptic current at the soma of a biological neuron. The variable $\mathbf{J}$ is not a component of the biological system, but rather a weight matrix that specifies the feedback configuration of the RNN. Similarly, $\mathbf{B}$ is a weight matrix that is applied to the input vector $\mathbf{u}$ of firing rates. The vector $\mathbf{b}$ applies a predetermined bias to each unit in the network, and τ sets the timescale of the system. I plan to model my RNN implementation off those given in [3] [4] and drawing from a deep learning library such as *Theano* as necessary. The network will be trained using minibatch Stochastic Gradient Descent. Given time, I will also implement more specialized versions of the general RNN such as the Long Short Term Memory (LSTM) RNN detailed by Schmidhuber [5].

2.4 Developing Neural Circuitry Visualization Capabilities in Fovea

Recurrent Neural Networks map input data sequences to output data sequences, and a key component in analyzing the computational structure and outcomes of any RNN involves identifying trends in the map and stabilities between the input and output data sequences. The implemented visualization capabilities will assist in reverse engineering and understanding the data generated by the RNN. The key steps in reverse engineering an RNN are identifying its attractors and performing a linearization of network dynamics around these attractors. I will implement functionality in Fovea that will identify attractors in RNN data, highlight these attractors in plots, and illustrate the convergence of neighborhood points to these attractors. Furthermore I will implement GUI functionality for diagrammatic visualization of RNN layers and allow for on-the-fly manipulation of these diagrams. These manipulations will be reflected in realtime in output data plots. I will look to the diagrams and schematics provided in [6] as a starting point in designing Fovea visualization capabilities for RNNs.

3 Deliverables

3.1 Minimal Deliverables

1. A complete Sammon Mapping algorithm implementation.
2. A complete Principle Curve Analysis implementation.
3. A complete Locally Linear Embedding implementation.
4. Support for dimension-toggling in Fovea visualizations.
5. Support for multi-frame plotting in Fovea.
6. GUI support for realtime reduction algorithm tweaking.
7. A Hopfield Network implementation.
8. An implementation of an RNN tailored towards modelling neural circuitry.
9. Attractor visualization in Fovea.
10. Diagrammatic visualization of RNNs.

3.2 Time-Contingent Deliverables

1. Optimization of Sammon Mapping algorithm via RNN's and inclusion of error-function support for additional metrics such as the Bregman distance.
2. Implementation of additional nonlinear dimensionality reduction algorithms such as curvilinear distance analysis, diffusion mapping, and diffeomorphic dimensionality reduction.
3. Support for additional high-dimensional visualization capabilities such as gradient color-mapping and time-series plot transformation.

4 Implementation Timeline

The following table presents a tentative timeline detailing the approximate dates by which I intend to have specific project tasks implemented.

Timeline	Task
Mar. 26 - May 23	Become familiar with current Fovea and PyDSTool capabilities and code bases • Document existing modules, classes, and methods, possibly with the assistance of a documentation system such as *readthedocs*.
Mar. 26 - May 23	Acquire application-specific knowledge of general dynamical systems theory, machine learning, mathematics, and neuroscience. • Work through textbooks on the subject. • Read relevant papers from the field.
Mar. 26 - May 23	Identify Matplotlib features pertinent to RNN visualization. If necessary, modify Fovea's Matplotlib distribution or extend Matplotlib to provide needed baseline functionality.
Mar. 26 - May 23	Introduce both tutorial-based and formal documentation of current Fovea functionality.
Mar. 26 - May 23	Resolve potential inconsistencies between Python 2.x and Python 3.x versions of PyDSTool and Fovea that could affect project design.
Mar.26 - May 23	Draft a design plan adhering to the Agile methodology for future PRs on Trello and Google Docs.
Mar. 26 - May 23	As a first pass, implement the Hopfield Network as a simple case of the more general RNN model. Create example code for modelling associative memory with the network and package with Fovea.
May 23 - May 29	Submit pull request for Sammon Mapping functionality. Provide tests of new algorithm and create a dimensionality reduction example to be packaged with Fovea.
May 30 - Jun. 5	Submit pull request for principle curve analysis. Provide test cases and example to be packaged with Fovea.
Jun. 6 - Jun. 12	Submit pull request for locally linear embedding implementation. Provide test cases and example to be packaged with Fovea.
Jun. 13 - Jun. 19	Implement dimension toggling and multi-frame plotting in Fovea. Provide test cases and example to be packaged.
Jun. 20 - Aug. 2	Implement and finetune recurrent neural network in PyDSTool. Train on appropriate data.
Aug.3 - Aug. 16	Implement RNN visualization capabilities in Fovea. Write test cases and provide examples to be packaged with Fovea.
Aug. 17 - Aug. 23	Tie up loose ends, bug fixing, provide tutorial and formal documentation of newly-implemented features.

It should be noted that the final two tasks in this timeline will likely take significantly more time than the others. I will plan on working on these for the duration of the summer, but will expect to have their implementations finished by the respective dates listed on the timeline.

5 Communication Plan

I will keep in regular contact with Dr. Clewley via email and Skype. Individual project tasks will be created and tracked on the project's Trello board, and we will be identifying highest-priority issues during weekly sprints. The project design will follow the Agile methodology. I also think it would be beneficial to create a Gitter chat page for more general project and feature discussion.

6 Personal Details

6.1 Motivation

I have a very strong interest in computational neuroscience, and I am considering pursuing graduate studies in the field. I am excited to work on this project because it will expose me to a number of research-level topics in neuroscience and allow me to contribute to the field in a significant way by assisting in the development of tools tailored towards both students of neuroscience and researchers doing groundbreaking work in the field. Many of the capabilities that I will implement are the product of extremely new developments in the field for which no suitable, neuroscience-oriented tools currently exist. For example, the RNN implementation described in [4] was just published not a month ago.

6.2 Project Match

In the past, I developed a small application for performing basic linear algebra and matrix operations. In that project, I implemented some involved algorithms such as the Strassen algorithm which is similar to the work I will be doing for this project with implementing non-linear dimensionality reduction.

The project was unique in that it focused on user operability and ease-of-use, and I will apply those same principles to developing intuitive GUI operations and visualization capabilities for Fovea.

6.3 Other GSOC Proposals

I am not currently submitting any other proposals for Google Summer of Code.

6.4 Education

I am a student at the University of California, Berkeley studying Applied Mathematics with a focus in Computer Science. I have a particular interest in computational neuroscience and related areas.

6.5 Experience

I have a strong background in both pure and applied mathematics including linear algebra, differential equations, and analysis. I have extensive experience programming in Python, including a number of projects listed on my CV. I also have experience with other applicable technologies such as Git and C. I am relatively new to computational neuroscience, but my strong math background makes it comparatively easy for me to quickly assimilate and understand the relevant topics and techniques in machine learning, data science, and dynamical systems theory, all of which are rooted in applied mathematics.

6.6 Time Commitments

I will likely be taking a summer class at UC Berkeley, but I do not foresee any major time conflicts arising as a result. I should still have ample time each week to work on my GSOC project.

7 References

[1] T. Hastie and W. Stuetzle, “Principal Curves,” *Journal of the American Statistical Association*, pp. 502–516, 1989. https://web.stanford.edu/~hastie/Papers/Principal_Curves.pdf.

[2] S. Roweis and L. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, pp. 2323–2326, 2000. http://web.stanford.edu/class/ee378b/papers/roweis.pdf.

[3] D. Sussilo, “Neural circuits as computational dynamical systems,” *Current Opinion in Neurobiology*, pp. 156–163, 2014. https://pdfs.semanticscholar.org/40d3/37ea0ff1672f25b1166d822bd0d16bb70745.pdf.

[4] F. Song, G. Yang, and X. Wang, “Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework,” *PLOS — Computational Biology*, pp. 1–30, feb 2016. http://www.cns.nyu.edu/wanglab/publications/pdf/song_ploscb2016.pdf.

[5] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015. http://arxiv.org/pdf/1404.7828v4.pdf.

[6] D. Sussilo and L. Abbott, “Generating coherent patterns of activity from chaotic neural networks,” *Neuron*, pp. 544–557, aug 2009. http://ac.els-cdn.com/S0896627309005479/1-s2.0-S0896627309005479-main.pdf?_tid=c76817ac-eef9-11e5-aa16-00000aacb361&acdnat=1458519406_7ea2c907d25a8d5a2db5ef301f0ea53a.