Identifying Isolated Subgraphs:

Classifying Spring-Mass Networks

Keith Schumacher

Data Mining EN 625.740

17 November 2020

Table of Contents

Introduction

Spring-mass networks are utilized across a diverse set of fields: cloth simulations in computer graphics (Provot 1995), protein and macromolecule dynamics in computational biology and chemistry (Togashi 2018), and financial network analysis (Somin 2020). They offer a simple model useful for analyzing network interactions. The goal of this project is to classify masses as belonging to a network structure based on their trajectory in 2-D space.

Data

Data for this project comes from the simulation of vibrating spring-mass networks. Each network is modeled as a series of masses connected by springs. The network can alternatively be described as vertices/nodes connected by edges/links; these terms will be used interchangeably throughout the paper. The motion of each mass is determined by Hooke's Law: the force required to compress or extend a spring is linearly proportional to the distance the spring is compressed or extended. Euler's Method is used to approximate the motion of the system (see euler.py). A step size of .01 was used to simulate the motion over 50,000 steps. Before classification attempts, the data was downsampled; every 100th data point was kept resulting in time series with 500 data points. A visualization was created using the downsampled data to elucidate the dynamics, https://youtu.be/zu7gIVKmhYw (Schumacher 2020) (see network_animate.py and youtube_video.ipynb). The video shows that even at 1% of the original sample rate, the 500 data point trajectory offers enough fidelity to capture the oscillatory dynamics of the network. Therefore it is reasonable to assume the downsampled data will be sufficient for classification purposes.

The algorithm outlined by Cox (2016) is used to calculate the force vector. The force vector changes throughout the simulation and is dependent upon the spatial configuration of vertices. Below are the relevant quantities and equations, they are implemented in matrix_builder.py:

$$e = Ax \quad , \quad y = Ke \quad , \quad f = A^T y \quad , \quad S = A^T K A \quad , \quad f = S x \quad .$$

- e – elongation. A vector quantifying the elongation of each spring. A positive elongation means the spring is stretched beyond its equilibrium.
- A – node-edge incidence matrix. An m by n matrix where m is the number of springs and n is the degrees of freedom for the set of vertices (Figure 1).
- x – position. A vector showing the position of each vertex.
- y – restoring force. The force exerted by each spring.
- K – spring constant matrix. A diagonal matrix with elements corresponding to each spring's spring constant.
- S – stiffness matrix
- f – force vector. Total force exerted on each vertex's degree of freedom.

$$\begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.0 & 0.0 & 0.0 \\ 0.71 & 0.71 & 0.0 & 0.0 & 0.71 & -0.71 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 1.0 & -0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.71 & 0.71 & -0.71 & -0.71 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.0 & -0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.71 & 0.71 & -0.71 & -0.71 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.71 & 0.71 & 0.0 & 0.0 & 0.71 & -0.71 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 1.0 & -0.0 \end{pmatrix}$$

Figure 1: Node-Edge Incidence Matrix: 5 Mass, 8 Spring Network

This project attempts to classify each vertex as belonging to a particular network. A network consists of all vertices such that a path exists connecting any two vertices in the network. Three network topologies are examined (Figure 2). The data set consists of multiple vibrating networks. The networks have different initial conditions and spring constants (though all the springs in a given network have the same spring constant).

Another visualization of the data set with randomly assigned spring constants can be found at https://youtu.be/gT_im7fmows. The first half of video shows 2 networks (vertices and edges). The vertices and edges are color coded either red or blue to indicate to which class the vertices belong. Next the edges and color coding are removed, but the geometry of the network is still present. The general topology (9 masses in a grid) can be inferred by inspection. While possessing knowledge of the network topology makes analysis easier, such a situation does not accurately model many real-world problems. For example, consider analyzing a set of financial assets for network effects. Only the past trajectory of each asset, effectively centered on the asset’s mean value (equilibrium), is known. There is no visually evident relationship between assets based on their their absolute positions. Only the position of an asset with respect to its own equilibrium is important/knowable. Therefore we remove this global position information from the dataset before analysis. A visualization for this form of the data set can be seen near the end of the video. Each vibrating vertex is shown vibrating around its own equilibrium. This is the data this project attempts to classify as belonging to a particular network. Only the 2-D displacement from a vertex's equilibrium point is known.

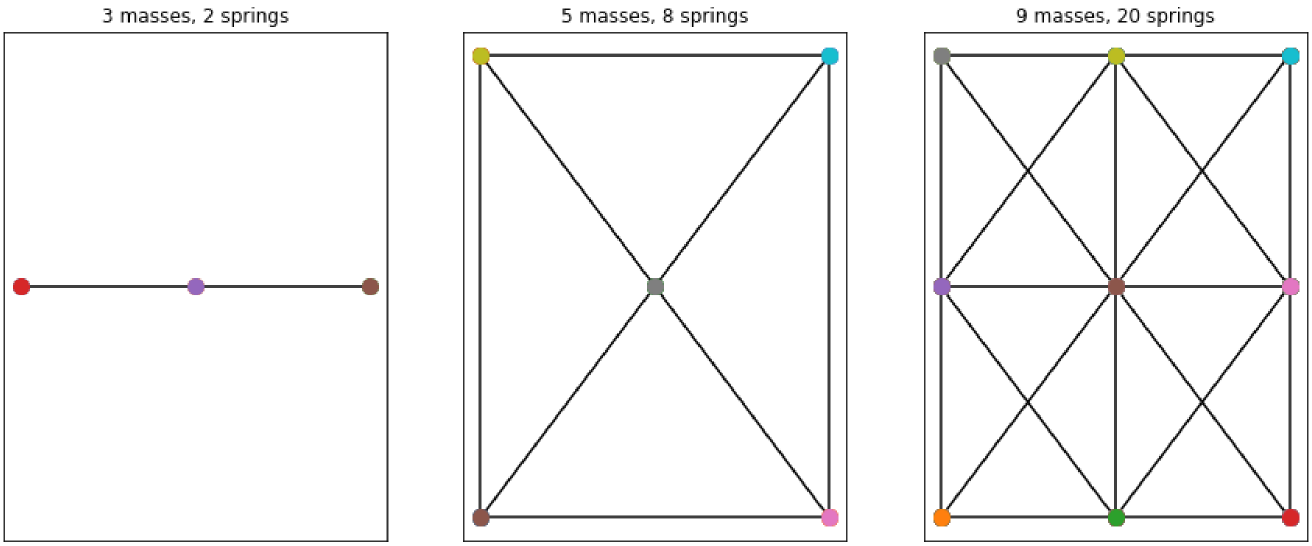

Figure 2: Network Topologies

Classification Techniques

Time series classification algorithms can be grouped by the features they extract: whole series, intervals, shapelets, dictionary-based, spectral, and combination. Whole series-based algorithms attempt to classify based on characteristics across the entire time series. For example, differentiating between a triangle wave and a sine wave when the entire time series consists of one wavelength. Nearest Neighbor with Dynamic Time Warping is a common whole series-based algorithm. Interval-based algorithms break the time series into intervals. Each interval can then be analyzed using whole-series based algorithms. Shapelet-based approaches look for short patterns that define the class. And dictionary-based algorithms look for series of shapelets. Spectral-based methods extract features from the frequency domain instead of the time domain. (Lines et al. 2018)

Figure 3 shows an example of the time series this project attempts to classify. There are no easily discernible characteristics that appear across the entire series or within particular intervals. Also, there are no recognizable shapes/patterns. Based upon knowledge of how the data was created, a spectral algorithm seems most promising for this project. In general, solutions to the system of differential equations that generated the time series, characterized by Hooke’s Law, describe oscillatory motion with frequency based on spring constants. Therefore, it is hypothesized that a spectral method will outperform a temporal method. Time Series Forest and RISE are tested; classification algorithms that draw features from the time domain and frequency domain respectively.

This project utilizes implementations of the Time Series Forest and RISE algorithms provided by the sktime project (Löning et al. 2019). Both Time Series Forest and RISE are univariate algorithms. Sktime provides two methods for working with multivariate time series. 1) Time Series Concatenation – transforms a multivariate time series into a univariate time series by concatenating each dimension of the time series into one larger time series. 2) Column Ensembling – A classification algorithm is applied to each dimension of the time series (possibly different algorithms for different dimensions), then the classification predictions are aggregated. Both methods are utilized in this project.

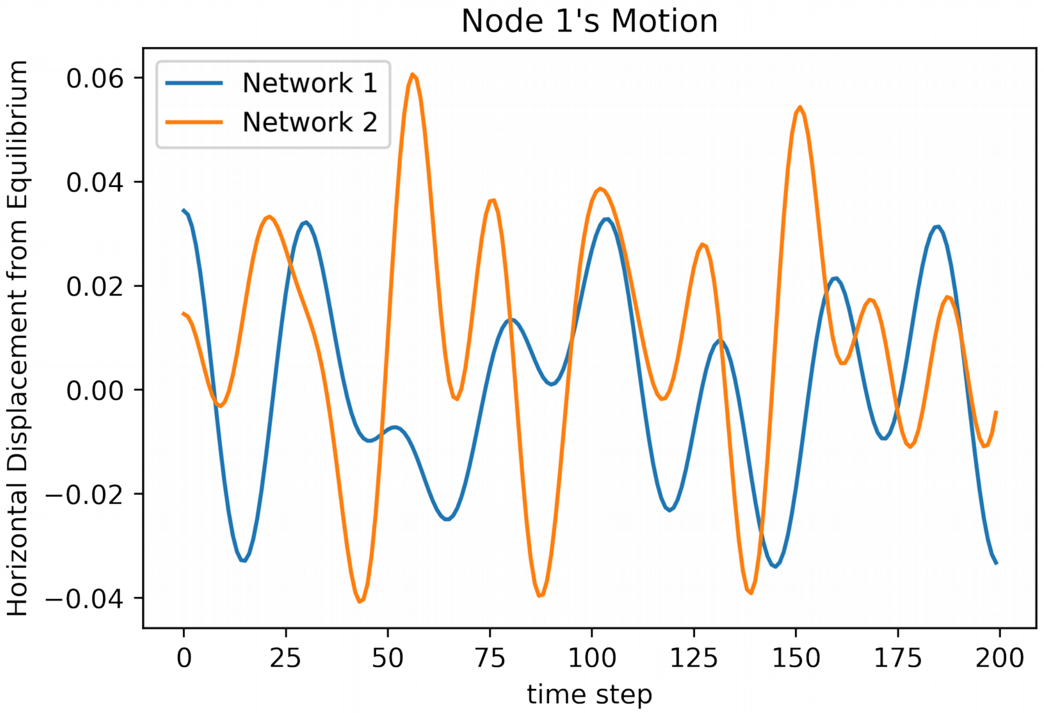

Figure 3: Example Time Series

Time Series Forest (TSF)

The TSF algorithm is a variation on the random forest learning method. The time series is split into many overlapping intervals and a decision tree is contructed for each interval. Three features are calculated on each interval: mean, standard deviation, and slope. The mean is the average value of the time series over an interval. The standard deviation is the deviation of the time series from its mean over an interval. And the slope is the slope of the least squares regression line of the time series over an interval. These features serve as splitting criterion with a threshold chosen to maximize entropy (entropy measures the effectiveness of splitting the classes). The final classification prediction is then taken to be the mode of all predictions made by the decision trees. (Deng et al. 2013)

Random Interval Spectral Ensemble (RISE)

RISE is essentially a frequency domain analogue to TSF; decision trees are built upon random intervals. The difference is in the features extracted from each interval. The power spectrum and auto-correlation of each interval is calculated, then features are derived from the frequency domain

(Lines et al. 2018). “Auto-Correlation features involve concatenation of autocorrelation, partial autocorrelation, and autoregressive terms, and the power spectrum terms are the truncated periodogram (squared Fourier terms)”. (Lines et al. 2016)

Conclusion

TSF and RISE were tested on the vibrating network data described above. Two networks with 9 vertices and 20 edges in 2-dimensions were simulated. Table 1 shows the results of the testing the two algorithms under various conditions. Below is a description of the various elements in Table 1:

- 2 Spring Constants – all the springs in a given network have the same spring constant (.01 and .02).
- Random Spring Constants – All the springs have random spring constants.
- Test Ratio=.5 – 50% of samples are used as test cases (and 50% are used for training). 27 training and 27 test time series.
- Test Ratio=.75 – 75% of samples are used as test cases (and 25% are used for training). 13 training and 41 test time series.
- The initial position of each vertex was random and there was no initial velocity.
- The network dynamics were simulated 3 times. Therefore, the experiment consisted of a classification problem with 54 time series (9 masses * 2 networks * 3 simulations) and 2 classes.
- Each estimator was tested 10 times with samples randomly assigned to the test/training sets. The classification accuracy was then averaged across these 10 tests.

As expected, classification accuracy decreased when spring stiffness was random in each network (as opposed to each spring being equally stiff in each network). TSF performed better when there were more training samples (50% of all samples were used for training). And RISE performed better when there were less training samples (25% of all samples were used for training). The Column Ensemble method outperformed the Column Concatenation Method.

An interesting extension of this project would be to see how damping (the network vibrations slowly die out) and noise (there is a noise term added to each vertex's position) impact the classification algorithms under the various conditions described in Table 1.

Classification Algorithm	**Classification Accuracy**			
	2 Spring Constants		Random Spring Constants	
	Test Ratio=.5	Test Ratio=.75	Test Ratio=.5	Test Ratio=.75
TSF (Column Concatenation)	.98	.84	.96	.69
TSF (Column Ensemble)	.99	.93	1.0	.68
RISE (Column Concatenation)	1.0	.98	.79	.74
RISE (Column Ensemble)	.99	.99	.85	.76

Table 1: TSF vs. RISE Classification Scores

Bibliography

Cox, S.. 2016. Linear Algebra In Situ. Caam.rice.edu. Available at: <https://www.caam.rice.edu/~cox/lais/bundle.pdf> [Accessed 9 November 2020].

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. Information Sciences,239, 142-153. doi:10.1016/j.ins.2013.02.030

Jason Lines, Sarah Taylor, and Anthony Bagnall. 2018. Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. ACM Trans. Knowl. Discov. Data. 12, 5, Article 52 (July 2018), 35 pages.

Keith Schumacher. (November 12, 2020). network classification [Video]. Youtube. https://youtu.be/zu7gIVKmhYw

Lines J., Taylor S., Bagnall A., "HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles for Time Series Classification," 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, 2016, pp. 1041-1046, doi: 10.1109/ICDM.2016.0133.

Löning, M., Bagnall A., Ganesh S., Kazakov V., Lines J., Király F. 2019: “sktime: A Unified Interface for Machine Learning with Time Series”

Somin, S., Altshuler, Y., Gordon, G. et al. 2020. Network Dynamics of a Financial Ecosystem. Sci Rep 10, 4587. https://doi.org/10.1038/s41598-020-61346-y

Togashi, Y., & Flechsi. 2018. Coarse-Grained Protein Dynamics Studies Using Elastic Network Models. International journal of molecular sciences

X. Provot, Deformation Constraints in a Mass-spring Model to Describe Rigid Cloth Behaviour. 1995. Proc of Graphics Interface