Learning Interpretable Collective Variables for Spreading Processes on Networks

Marvin Lücke and Stefanie Winkelmann Zuse Institute Berlin

Jobst Heitzig

FutureLab on Game Theory and Networks of Interacting Agents,
Potsdam Institute for Climate Impact Research and
Zuse Institute Berlin

Nora Molkenthin

Complexity Science Department, Potsdam Institute for Climate Impact Research

Péter Koltai

Department of Mathematics, University of Bayreuth

Collective variables (CVs) are low-dimensional projections of high-dimensional system states. They are used to gain insights into complex emergent dynamical behaviors of processes on networks. The relation between CVs and network measures is not well understood and its derivation typically requires detailed knowledge of both the dynamical system and the network topology. In this work, we present a data-driven method for algorithmically learning and understanding CVs for binary-state spreading processes on networks of arbitrary topology. We demonstrate our method using four example networks: the stochastic block model, a ring-shaped graph, a random regular graph, and a scale-free network generated by the Albert–Barabási model. Our results deliver evidence for the existence of low-dimensional CVs even in cases that are not yet understood theoretically.

Introduction. Networks of interacting agents are widely used to model socio-dynamical phenomena [1] such as the spreading of a disease [2–4] or the diffusion of a (political) opinion within a society [5, 6]. In such networks, nodes represent individual agents, and edges represent some form of social interaction. Each node has a state that evolves over time depending on the states of neighboring nodes. Often, stochastic effects are included to account for uncertainty in the dynamics and for the unpredictability of agents. These types of spreading processes are at the core of numerous open problems in a wide range of disciplines, such as understanding social collective behavior [7], assessing systemic risk in financial systems [8], or controlling modern power grids [9].

One approach to elevate our understanding of these models is to seek a low-dimensional representation of the system that captures the fundamental dynamics on timescales of interest. The projection into this low-dimensional space is called a *collective variable* (CV), and the projected dynamics is called the *effective dynamics*. Good CVs retain the essential information about the system's behavior, reducing the dimensionality and enabling a more efficient analysis and prediction.

For discrete-state spreading processes on certain simple networks, e.g., complete graphs and dense Erdős–Rényi random graphs, it is known that the shares of nodes in each state constitute a good CV, and its evolution is given by an ordinary differential equation in the meanfield limit [10]. There are many other results, valid for a varying range of networks and models, that describe the evolution of the shares of states in terms of (par-

tial) differential equations in the mean-field or hydrodynamic limit [11–14]. Another popular choice of CVs are the counts of certain network motifs, e.g., the number of links between nodes of different states (typically called pair-approximation [15-17]), and moment-closure methods can be used to approximate their evolution [18– 22]. For the special case of binary-state dynamics, standard mean-field or pair-approximation theories have been complemented by higher-order master equations which improve accuracy by introducing fractions of neighbors of one or the other state, resolved with respect to the degrees of the nodes [23–27]. However, there is no allencompassing theory relating any network topology and any process occurring on it to resulting CVs. Hence, a constructive computational approach – like the one we will present – can elucidate cases that theoretical results do not vet cover. Moreover, the above-mentioned techniques postulate a candidate CV based on (physical) insights about the system, whereas our procedure does not require such intuition.

In this work, we extend the transition manifold approach [28, 29] to learn CVs based on simulation data and to systematically find the relationship of the learned CVs to topological features of the network (see Fig. 1). The transition manifold approach assumes and exploits that the transition density functions of the system accumulate around a low-dimensional manifold, from which a CV can be inferred. The approach is designed such that most information of the density propagation of the process is retained under projection onto the CV [28]. To make it applicable to binary-state spreading processes on

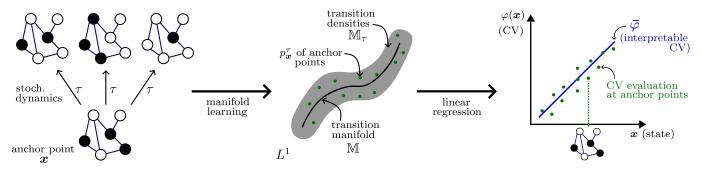


FIG. 1: Illustration of our method. Left: The random process is described by its distribution, sampled through S samples per initial network state (anchor). Middle: These distributions are used to learn a low-dimensional parametrization of the transition manifold. Right: A regression step allows for interpretability of the learned CV.

networks, we develop a technique for evenly sampling the state space and we add a linear regression step to produce interpretable CVs.

While goals similar to ours have been pursued in the literature before, to our best knowledge, this is the first work to learn CVs from data for networks of interacting agents. A crucial difference to previous works which learn reduced (also called surrogate or emergent) spaces [30–33] from data – even in the agent-based context [34] – is that they perform the reduction primarily based on the information of a state, and not on its dynamics. Once a good CV has been learned with our method, a surrogate dynamical model for its evolution could be determined and analyzed by tools also utilized in these references, in particular by linear transition operators associated with the dynamics [35–37]. Other approaches for data-based analysis and reduced modelling of interacting agent systems postulate CVs instead of learning them [38–42].

Recently, deep learning techniques have become popular for finding low-dimensional variables and surrogate dynamical models [35, 36, 43–48]. Artificial neural networks can represent coordinates from a large general class, but the dynamical conditions necessary for them to perform well remain implicit in these methods. Our approach, however, relies on explicit dynamical assumptions that are validated during the data-driven computation.

The transition manifold. Consider a fixed network of N nodes, on which each node $i \in \{1, \ldots, N\}$ has a discrete state $x_i \in \mathbb{S}$. The state space of the process is thus $\mathbb{X} := \mathbb{S}^N$, and its elements are system states $\boldsymbol{x} = (x_1, \ldots, x_N)$. Given a system state $\boldsymbol{x} \in \mathbb{X}$ and time $\tau \geq 0$, the transition density function $p_{\boldsymbol{x}}^{\tau} \in L^1(\mathbb{X}) =: L^1$ is defined such that $p_{\boldsymbol{x}}^{\tau}(\boldsymbol{y})$ is the probability that the system is in state \boldsymbol{y} at time τ after having started in state \boldsymbol{x} at time 0. (The term density comes from the original theory for continuous state spaces. We use L^1 to emphasize that these are vectors with entries summing to 1.) The transition manifold approach [28, 29] exploits the observation that for certain systems and an appropriate

choice of τ , the set

$$\mathbb{M}_{\tau} := \{ p_{\boldsymbol{x}}^{\tau} \mid \boldsymbol{x} \in \mathbb{X} \} \subset L^1 \tag{1}$$

is close to a d-dimensional submanifold $\mathbb{M} \subset L^1$ called the transition manifold. The lag-time τ needs to be longer than the "relaxation time" towards \mathbb{M} and shorter than the time it takes to eventually converge to a stationary distribution (see supplemental material [49, section S.2] for details).

As a consequence, one can show that there exists a d-dimensional collective variable $\varphi: \mathbb{X} \to \mathbb{R}^d$, such that for all $x \in \mathbb{X}$

$$p_{\boldsymbol{x}}^{\tau} \approx \tilde{p}_{\varphi(\boldsymbol{x})}^{\tau},$$
 (2)

for some function $\tilde{p}_{(\cdot)}^{\tau}$. Hence, the essential information needed to characterize the dynamics is captured by the collective variable φ . A coordinate function φ satisfying this is for instance a "parametrization" of the manifold \mathbb{M} , in the sense that the assignment $\mathbb{M} \to \mathbb{X} \to \mathbb{R}^d$, $p_{\boldsymbol{x}}^{\tau} \mapsto \boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$ is one-to-one, cf. [28, 29]. Typically, the dimension d of the reduced state is significantly smaller than the dimension of the original state.

From now on, we consider binary-state dynamics, i.e., $\mathbb{S} = \{0,1\}$. While this already covers a wide range of interesting dynamics, we expect that most of the following can be extended to an arbitrary number of states, at the cost of additional technicalities. For binary-state dynamics on a complete network, to give an example, a good collective variable is often given by the share of nodes in state 1, i.e., d=1 and $\varphi(\boldsymbol{x})=\sum_i x_i/N$ [10]. This is related to the quantity called magnetization in the similar Ising model [50].

Learning interpretable CVs. We propose the following method, which consists of three steps (Fig. 1).

Step 1. We choose a diverse set of dynamically relevant anchor points $\boldsymbol{x}^1, \dots, \boldsymbol{x}^K \in \mathbb{X}$ in which the CV is going to be computed in the first instance. Diversity of the points is crucial in the sense that their respective transition densities cover \mathbb{M}_{τ} sufficiently well. Otherwise, the parametrization learned in the next step would

yield a result biased by the insufficient coverage, that is potentially not a CV for the entire process. A precise quantification of sufficient coverage cannot be stated in generality as it depends on the system at hand and on the desired quality of the CVs. We employ an algorithm that prioritizes sampling anchor points containing communities of nodes with the same state, as this strategy produced the best results for the spreading processes we examined, see supplemental material [49, section S.2] for details.

Step 2. Next, we approximate a "parametrization" φ of the transition manifold \mathbb{M} from simulation data. For each anchor point \boldsymbol{x}^k we conduct $S \in \mathbb{N}$ simulations of the process of length τ , yielding S samples for each transition density $p_{\boldsymbol{x}^k}^{\tau}$. Using this data, we obtain evaluations of the collective variable φ at the anchor points $\boldsymbol{x}^1,\ldots,\boldsymbol{x}^K$ by applying a manifold learning technique. Here, the dimension d of the CV is also an output. See supplemental material [49, section S.2] for further details on the approximation of the transition manifold.

The necessary number K of anchor points and S of simulations per anchor point depend on the size and complexity of the network. For the examples below (N < 1000), we found $K \approx 1000$ and $S \approx 100$ adequate. This very small number of anchor points compared to the number 2^N of all possible states is sufficient due to the targeted sampling method we employ (cf. step 1). We observed a substantial robustness of the results in varying the method's hyperparameters, but an entirely automatic procedure for their selection has yet to be designed. The output of this second step of the method is evaluations of the d-dimensional CV φ at the anchor points, $\varphi(\boldsymbol{x}^1), \ldots, \varphi(\boldsymbol{x}^K) \in \mathbb{R}^d$.

Step 3. The third step of the method aims at determining the meaning of the CV and finding a reasonable map $\bar{\varphi}: \{0,1\}^N \to \mathbb{R}^d$ that extends it to states \boldsymbol{x} outside of the original data set. Motivated by the fact that, for binary-state dynamics, the share of nodes in state 1 in (parts of) the network is known to be a good CV in specific cases [10], we propose maps $\bar{\varphi}$ of the form

$$\bar{\varphi}(\boldsymbol{x}) = \begin{pmatrix} \bar{\varphi}_1(\boldsymbol{x}) \\ \vdots \\ \bar{\varphi}_d(\boldsymbol{x}) \end{pmatrix}, \quad \bar{\varphi}_j(\boldsymbol{x}) = \sum_{i=1}^N \Lambda_{j,i} \ x_i, \qquad (3)$$

where $\Lambda \in \mathbb{R}^{d \times N}$ is a parameter matrix. For example, choosing d=1 and $\Lambda=(1,\ldots,1)$ yields a map describing the total count of state 1. In the different context of coupled ODEs, a CV similar to (3) was examined in Refs. [51, 52].

We find optimal parameters Λ by employing linear regression to fit $\bar{\varphi}$ to the computed CV values in the anchors, $\varphi(\boldsymbol{x}^1), \ldots, \varphi(\boldsymbol{x}^K)$, from step 2. To prevent overfitting we use the *graph total variation* regularizer, which penalizes variation of Λ along edges (see supplemental material [49, section S.3]). The reason for this choice

is that each node in densely interconnected clusters is expected to contribute similarly to the CV. Even for networks containing hubs we can maintain this regularizer by bootstrapping our result to modify the ansatz functions, cf. Example 4 below. There we use linear basis functions for a first regression step, observe strong correlation in Λ to the network's degrees, and modify the ansatz functions accordingly. If suggested by physical intuition, one can also entirely deviate from the linear ansatz functions we propose.

In the following examples binary-state spreading processes are studied, in which each node experiences memoryless random evolution in continuous time, with transition rates determined by a constant exploration rate (noise) and an "influence" rate based on the states in the node's neighborhood, see supplemental material [49, section S.1] for details. The investigations below refer to the noisy voter model [53, 54], yet the supplemental material also includes findings related to alternative dynamics.

Example 1: Stochastic block model. We examine a network of N = 900 nodes that is constructed using the stochastic block model. The network consists of three clusters such that cluster 1 and 2 are densely connected, cluster 1 and 3 are connected only sparsely, and cluster 2 and 3 are not connected at all, cf. Fig. 2. We expect the optimal CV to be d = 3-dimensional and contain the counts of 1's in each cluster. This CV is exact in the sense that for $N \to \infty$ it satisfies a mean-field equation [10]. Applying our method and plotting the resulting CV point cloud $\{\varphi(\boldsymbol{x}^1),\ldots,\varphi(\boldsymbol{x}^K)\}\subset\mathbb{R}^3$ yields an approximately cuboid shaped transition manifold. We found that the vertices of this cuboid correspond to extreme states \boldsymbol{x} in which for each cluster either all or no nodes have state 1, cf. Fig. 2. To discover the meaning of the three coordinates $\varphi_1, \varphi_2, \varphi_3$, we calculate the optimal fit according to the regression problem proposed in step 3 of the method, which yields a collective variable $\bar{\varphi}(x) = \Lambda x$ with optimal parameters Λ shown in Fig. 3. The entries of the first row $\Lambda_{1,:}$ are approxi-

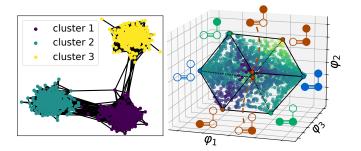


FIG. 2: For the stochastic block model network (left), the transition manifold is a 3-dimensional cuboid (right). The vertices of the cuboid correspond to extreme states \boldsymbol{x} where for each cluster either all (filled circle) or no nodes (empty circle) have state 1.

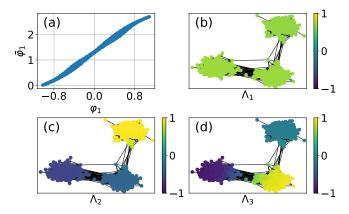


FIG. 3: Optimal Λ from (3) for Example 1. (a): Data $\varphi_1(\boldsymbol{x}^k)$ versus optimal fit $\bar{\varphi}_1$. (b)-(d): Optimal Λ entries for the respective coordinates plotted as color values on the network.

mately equal and thus $\bar{\varphi}_1$ describes the count of 1's in the whole network. The optimal $\Lambda_{2,:}$ is positive and constant within cluster 3 and negative and constant within clusters 1 and 2. Thus, $\bar{\varphi}_2$, calculates how the 1's are distributed between clusters $\{1,2\}$ and $\{3\}$. Finally, $\Lambda_{3,:}$ is positive in cluster 1, negative in cluster 2, and approximately 0 in cluster 3, which implies that $\bar{\varphi}_3$ measures how the 1's are distributed between clusters 1 and 2, regardless of the number of 1's in cluster 3. Hence, the learned CV $\bar{\varphi}$ includes exactly the information that was predicted by theory [10] for large N, i.e., the counts of 1's for each cluster, but the coordinates are ordered by dynamical prevalence. (For instance, coordinate 3 is the least prevalent because information flows quickly between the two densely connected clusters 1 and 2.)

An interesting question would be to consider the change of the CV and especially its dimension with increasing edge density between the clusters. In particular, do structural transitions in the CV coincide with the so-called *detectability threshold* of the stochastic block model [55–57], where the edge statistics become indistinguishable from an Erdős–Rényi random graph model? This will be addressed in future work.

Example 2: Ring-shaped network. We apply our method to a ring-shaped network of N=50 nodes. Examining the point cloud $\{\varphi(\boldsymbol{x}^1),\dots,\varphi(\boldsymbol{x}^K)\}$ for different choices of d, we can not identify a low-dimensional transition manifold as increasing d keeps adding valuable information. To keep the CV dimension reasonably small, we choose d=5. (This yields CVs of reasonable quality, cf. supplemental material [49, section S.5].)

Solving the regression problem in step 3 yields a Λ that is constant in the first coordinate, i.e., the most important information is again the total count of 1's, see Fig. 4. The subsequent $\Lambda_{j,i}$ are pairs of sine and cosine functions of the node index i, starting with one oscillation for coordinates i = 2, 3 and then doubling the frequency

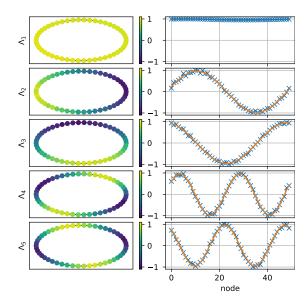


FIG. 4: Left: optimal $\Lambda_{i,:}$ plotted as color values on the ring-shaped network. Right: $\Lambda_{i,:}$ (blue crosses) and a sine fit (orange line). The collective variables φ_i represent the real Fourier coefficients of the distribution of 1's on the ring, since $\varphi_i(\boldsymbol{x}) \approx \Lambda_{i,:} \boldsymbol{x}$ with the $\Lambda_{i,:}$ being sines and cosines of increasing frequencies.

for coordinates j=4,5. Hence, the collective variable $\bar{\varphi}$ measures the distribution of 1's on the ring, with increasing precision as we let d increase. This structure mimics Fourier coefficients, which suggests that (in the limit of infinitely many nodes) the optimal collective variable measures the position-dependent concentration of 1's as a density function on the ring. This result agrees well with other works considering ring-shaped or lattice networks, e.g. [11, 13, 14], which find that the concentration of 1's is governed by a diffusive PDE in the hydrodynamic limit. The CV of the system thus being a function on the ring, any finite-dimensional approximation has a truncation error. However, orthogonal trigonometric polynomials are a natural (and in an L^2 -sense optimal) choice, found by our method.

Example 3: Random 3-regular network. One challenge in reduced modelling of spreading processes on random regular graphs is that edges are correlated. If the degree grows indefinitely with the network size, it was shown for the voter model that the share of state 1 is an asymptotically perfect CV [10]. In the same study it was observed numerically that for small degrees this CV still seems to support an effective dynamics, which deviates from the one obtained by mean-field approximation. Our method applied to a random 3-regular network validates the observation by reproducing this CV; see the supplemental material [49, section S.4].

Example 4: Albert–Barabási network. Finally, we apply our method to a network generated by the Albert–Barabási model [58]. In the preferential attachment al-

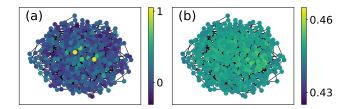


FIG. 5: (a) For the Albert–Barabási network, the optimal Λ as in (3) assigns a large weight to nodes with high degree. (b) After pre-weighting with node degree, cf. (4), the optimal Λ is constant. Hence, the collective variable describes the degree-weighted count of 1's.

gorithm each new node is connected to m=2 existing nodes, that are randomly picked with probability proportional to their degree. This procedure yields (asymptotically) a scale-free network. Applying our method results in a point cloud $\{\varphi(\boldsymbol{x}^1),\ldots,\varphi(\boldsymbol{x}^K)\}$ that indicates a d=1-dimensional transition manifold, see [49, Fig. S.3]. The optimal $\Lambda \in \mathbb{R}^N$ according to the linear regression problem in step 3 assigns a large positive weight to nodes of high degree, whereas nodes with small degree have small or even negative weight, cf. Fig. 5. This conflicts with our choice of graph total variation regularizer that favors solutions for which Λ is equal for neighboring nodes. We tackle this issue by applying a pre-weighting of each node i with its degree d_i :

$$\bar{\varphi}(\boldsymbol{x}) = \sum_{i=1}^{N} \Lambda_i \ d_i \ x_i. \tag{4}$$

The optimal Λ for (4) becomes approximately constant, and hence the CV measures the degree-weighted count of state 1 in the system, cf. Fig. 5. Multiple experiments for varying parameters confirmed this result, provided the preferential attachment parameter is chosen $m \geq 2$. (For m=1 the resulting networks exhibit a significantly larger diameter [59]. As a consequence, the degree-weighted count does not seem to sufficiently characterize the dynamics.) We are not aware of any theoretical work showing that the degree-weighted count is a good CV for (binary-state or other) spreading processes on Albert–Barabási networks, although refs. [60, 61] hint at the significance of this observable.

Validation. A numerical validation of the CVs learned in the above examples is presented in the supplemental material [49, section S.5].

Conclusion. We propose a method to learn interpretable CVs for spreading processes on networks without the need for prior expert knowledge about the network topology or the dynamical process. This method consists of the following steps: First we sample anchor (network) states, from which we start many short simulations. Then we approximate the transition manifold and extend the learned CVs to unseen data using (total-

variation-regularized) linear regression. The CVs are interpretable since the inferred parameters indicate the function and significance of features of the network structure. We have demonstrated this method for four different network topologies and two types of spreading dynamics (see supplemental material [49, section S.4]) and have thus shown its flexibility and usefulness. Although out of scope for the current manuscript, we expect that the method can be generalized to processes with more than two discrete states as well as inhomogeneous agent dynamics.

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy via the Berlin Mathematics Research Center MATH+ (EXC2046/ project ID: 390685689).

- C. Castellano, S. Fortunato, and V. Loreto, Rev. Mod. Phys. 81, 591 (2009).
- [2] I. Z. Kiss, J. C. Miller, and P. L. Simon, *Mathematics of Epidemics on Networks* (Springer International Publishing, 2017).
- [3] R. Pastor-Satorras, C. Castellano, P. VanMieghem, and A. Vespignani, Rev. Mod. Phys. 87, 925 (2015).
- [4] S. Mauras, V. Cohen-Addad, G. Duboc, M. D. la Tour, P. Frasca, C. Mathieu, L. Opatowski, and L. Viennot, PLoS Comput. Biol. 17 (2021), 10.1371/journal.pcbi.1009264.
- 5] S. Banisch and E. Olbrich, J. Math. Sociol. **43**, 76 (2019).
- [6] A. Das, S. Gollapudi, and K. Munagala, in Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14 (Association for Computing Machinery, 2014) p. 403–412.
- [7] J. B. Bak-Coleman, M. Alfano, W. Barfuss, C. T. Bergstrom, M. A. Centeno, I. D. Couzin, J. F. Donges, M. Galesic, A. S. Gersick, J. Jacquet, A. B. Kao, R. E. Moran, P. Romanczuk, D. I. Rubenstein, K. J. Tombak, J. J. V. Bavel, and E. U. Weber, PNAS 118, e2025764118 (2021).
- [8] F. Caccioli, P. Barucca, and T. Kobayashi, J. Comput. Soc. Sci. 1, 81 (2017).
- [9] D. Witthaut, F. Hellmann, J. Kurths, S. Kettemann, H. Meyer-Ortmanns, and M. Timme, Rev. Mod. Phys. 94, 015005 (2022).
- [10] M. Lücke, J. Heitzig, P. Koltai, N. Molkenthin, and S. Winkelmann, Stoch. Process. their Appl. 166, 104220 (2023).
- [11] E. Presutti and H. Spohn, Ann. Probab. 11 (1983), 10.1214/aop/1176993437.
- [12] D. Keliger, I. Horváth, and B. Takács, Stoch. Process. their Appl. 148, 324 (2022).
- [13] R. Durrett and W.-T. L. Fan, Ann. Appl. Probab. 26 (2016), 10.1214/16-aap1181.
- [14] W.-T. L. Fan, Bernoulli **27** (2021), 10.3150/20-bej1296.
- [15] M. J. Keeling, D. Rand, and A. Morris, Proc. R. Soc. London, Ser. B 264, 1149 (1997).
- [16] F. Vazquez and V. M. Eguíluz, New J. Phys. 10, 063011 (2008).

- [17] E. Pugliese and C. Castellano, Europhys. Lett. 88, 58004 (2009).
- [18] A.-L. Do and T. Gross, in Adaptive Networks: Theory, Models and Applications (Springer, 2009) pp. 191–208.
- [19] M. Taylor, P. L. Simon, D. M. Green, T. House, and I. Z. Kiss, J. Math. Biol. 64, 1021 (2012).
- [20] P. Moretti, S. Y. Liu, A. Baronchelli, and R. Pastor-Satorras, Eur. Phys. J. B 85 (2012), 10.1140/epjb/e2012-20501-1.
- [21] C. Kuehn, in *Understanding Complex Systems* (Springer International Publishing, 2016) pp. 253–271.
- [22] A. F. Peralta, A. Carro, M. S. Miguel, and R. Toral, New J. Phys. 20 (2018), 10.1088/1367-2630/aae7f5.
- [23] K. T. Eames and M. J. Keeling, PNAS 99, 13330 (2002).
- [24] V. Marceau, P.-A. Noël, L. Hébert-Dufresne, A. Allard, and L. J. Dubé, Phys. Rev. E 82, 036116 (2010).
- [25] J. P. Gleeson, Phys. Rev. Lett. 107, 068701 (2011).
- [26] J. Lindquist, J. Ma, P. Van den Driessche, and F. H. Willeboordse, J. Math. Biol. 62, 143 (2011).
- [27] J. P. Gleeson, Phys. Rev. X 3, 021004 (2013).
- [28] A. Bittracher, P. Koltai, S. Klus, R. Banisch, M. Dellnitz, and C. Schütte, J. Nonlinear Sci. 28, 471 (2018).
- [29] A. Bittracher, S. Klus, B. Hamzi, P. Koltai, and C. Schütte, J. Nonlinear Sci. 31 (2021), 10.1007/s00332-020-09668-z.
- [30] R. R. Coifman, I. G. Kevrekidis, S. Lafon, M. Maggioni, and B. Nadler, Multiscale Modeling & Simulation 7, 842 (2008).
- [31] M. A. Rohrdanz, W. Zheng, M. Maggioni, and C. Clementi, J. Chem. Phys. 134 (2011), https://doi.org/10.1063/1.3569857.
- [32] T. Berry, D. Giannakis, and J. Harlim, Phys. Rev. E 91, 032915 (2015).
- [33] P. Koltai and S. Weiss, Nonlinearity **33**, 1723 (2020).
- [34] F. Kemeth, T. Bertalan, T. Thiem, F. Dietrich, S. J. Moon, C. Laing, and I. Kevrekidis, Nat. Commun. 13, 3318 (2022).
- [35] B. Lusch, J. N. Kutz, and S. L. Brunton, Nat. Commun. 9, 4950 (2018).
- [36] A. Mardt, L. Pasquali, H. Wu, and F. Noé, Nat. Commun. 9, 1 (2018).
- [37] G. Froyland, D. Giannakis, B. R. Lintner, M. Pike, and J. Slawinska, Nat. Commun. 12, 6570 (2021).
- [38] F. Lu, M. Zhong, S. Tang, and M. Maggioni, Proc. Nat. Acad. Sci. 116, 14424 (2019).
- [39] M. Maggioni, J. Miller, and M. Zhong, arXiv:1912.11123 (2019), arXiv:1912.11123.
- [40] J.-H. Niemann, S. Klus, and C. Schütte, PLoS ONE 16 (2021), 10.1371/journal.pone.0250970.
- [41] N. Wulkow, P. Koltai, and C. Schütte, J. Nonlinear Sci. 31 (2021), 10.1007/s00332-020-09673-2.
- [42] L. Helfmann, J. Heitzig, P. Koltai, J. Kurths, and C. Schütte, Eur. Phys. J. Spec. Top. 230, 3249 (2021).
- [43] S. Brandt, F. Sittel, M. Ernst, and G. Stock, J. Phys. Chem. Lett. 9, 2144 (2018).

- [44] C. Wehmeyer and F. Noé, J. Chem. Phys. 148, 241703 (2018).
- [45] M. Raissi, P. Perdikaris, and G. E. Karniadakis, arXiv preprint, arXiv:1801.01236 (2018), 10.48550/arXiv.1801.01236.
- [46] W. Chen, H. Sidky, and A. L. Ferguson, J. Chem. Phys. 150 (2019), 10.1063/1.5092521.
- [47] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, PNAS 116, 22445 (2019).
- [48] S. E. Otto and C. W. Rowley, SIAM J. Appl. Dyn. Syst. 18, 558 (2019).
- [49] See Supplemental Material below.
- [50] B. A. Cipra, Am. Math. Mon. **94**, 937 (1987).
- [51] J. Gao, B. Barzel, and A.-L. Barabási, Nature 530, 307 (2016).
- [52] E. Laurence, N. Doyon, L. J. Dubé, and P. Desrosiers, Phys. Rev. X 9, 011042 (2019).
- [53] B. L. Granovsky and N. Madras, Stoch. Process. their Appl. 55, 23 (1995).
- [54] A. Carro, R. Toral, and M. S. Miguel, Scientific Reports 6 (2016), 10.1038/srep24775.
- [55] J. Reichardt and M. Leone, Phys. Rev. Lett. 101, 078701 (2008).
- [56] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. Lett. 107, 065701 (2011).
- [57] J. Banks, C. Moore, J. Neeman, and P. Netrapalli, in 29th Annual Conference on Learning Theory, Proceedings of Machine Learning Research, Vol. 49 (2016) pp. 383–416.
- [58] R. Albert and A.-L. Barabási, Rev. Mod. Phys. 74, 47 (2002).
- [59] B. Bollobás and O. Riordan, Combinatorica 24, 5 (2004).
- [60] K. Suchecki, V. M. Eguíluz, and M. S. Miguel, Europhys. Lett. 69, 228 (2005).
- [61] G. Bianconi, Phys. Lett. A 303, 166 (2002).
- [62] M. Porter and J. Gleeson, Dynamical Systems on Networks (Springer International Publishing, 2016).
- [63] A. Bittracher, M. Mollenhauer, P. Koltai, and C. Schütte, Multiscale Model. Simul. 21, 449 (2023).
- [64] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, Found. Trends Mach. Learn. 10, 1 (2017).
- [65] R. R. Coifman and S. Lafon, Appl. Comput. Harmon. Anal. 21 (2006), 10.1016/j.acha.2006.04.006.
- [66] A. Bittracher and C. Schütte, in Advances in Dynamics, Optimization and Computation: A volume dedicated to Michael Dellnitz on the occasion of his 60th birthday (Springer, 2020) pp. 132–150.
- [67] R. R. Coifman, Y. Shkolnisky, F. J. Sigworth, and A. Singer, IEEE Trans. Image Process. 17, 1891 (2008).
- [68] R. J. Tibshirani and J. Taylor, Ann. Stat. 39 (2011), 10.1214/11-aos878.
- [69] T. Hastie, J. Friedman, and R. Tibshirani, The Elements of Statistical Learning (Springer New York, 2001).
- [70] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, Math. Program. Comput. 12, 637 (2020).

Supplemental Material: Learning Interpretable Collective Variables for Spreading Processes on Networks

S1. Supplementary Details on Binary-State Spreading Processes

Our method for learning interpretable collective variables can be applied to arbitrary (binary-state) spreading processes on networks. To illustrate this method, we apply it to two popular dynamical models: a version of the so-called *voter model* and the *threshold model* [62]. Both generate Markovian (i.e., memoryless) processes.

Voter model. In the continuous-time noisy voter model (CNVM) on a simple graph containing N nodes, each node $i \in \{1, ..., N\}$ has a discrete state $x_i \in \mathbb{S}$, where \mathbb{S} is a finite set of possible states. In the context of the voter model, these states are also called *opinions*. The state space of the CNVM is $\mathbb{X} := \mathbb{S}^N$, and its elements are system states $\boldsymbol{x} = (x_1, ..., x_N)$. The CNVM describes the mechanism that nodes adopt the opinion of randomly chosen neighbors. At random times occurring at a fixed rate, every node is influenced by a (uniformly chosen) random neighbor. Hence, the rate at which a node switches to a certain opinion is proportional to the share of that opinion in its neighborhood. Simultaneously, the nodes also conduct a random opinion exploration independent of the opinions of their neighbors. More precisely, the rate at which a node i transitions from state $m \in \mathbb{S}$ to state $n \in \mathbb{S}$, $n \neq m$, is defined as

$$r_{m,n}\frac{d_{i,n}(\boldsymbol{x})}{d_i} + \tilde{r}_{m,n},\tag{S1}$$

where $d_{i,n}(\boldsymbol{x})$ is the number of neighbors of node i that have state n, d_i is the degree of node i, and $r_{m,n}, \tilde{r}_{m,n} \geq 0$ are model parameters. The rate $r_{m,n}$ determines how fast transitions from opinion m to n occur due to the above described imitation of neighbors. The additional rate $\tilde{r}_{m,n}$ enables "explorative" transitions from m to n independently of the neighborhood and thus controls the amount of noise in the system. Thus, by Markovianity, we have

Prob
$$[x_i(t+dt) = n \mid x_i(t) = m] = \left(r_{m,n} \frac{d_{i,n}(\mathbf{x})}{d_i} + \tilde{r}_{m,n}\right) dt + \mathcal{O}(dt^2)$$

as $dt \to 0$.

In all examples presented below and in the main text, we studied the CNVM with two possible opinions $\mathbb{S} = \{0, 1\}$ and model parameters

$$r = \begin{pmatrix} - & 0.99 \\ 1.01 & - \end{pmatrix}, \quad \tilde{r} = \begin{pmatrix} - & 0.005 \\ 0.005 & - \end{pmatrix}.$$
 (S2)

Threshold model. In contrast to the voter model, where transition rates are proportional to the share of opinions in the neighborhood, the threshold model assumes that a switch to a different opinion only occurs if that opinion is already sufficiently established in the neighborhood. More precisely, in the threshold model a node i transitions from state $m \in \{0,1\}$ to state n=1-m at the rate

$$\begin{cases} r_{m,n} + \tilde{r}_{m,n}, & \frac{d_{i,n}(\mathbf{x})}{d_i} \ge b_{m,n} \\ \tilde{r}_{m,n}, & \text{else} \end{cases}$$
 (S3)

where $d_{i,n}(\boldsymbol{x})$ is the number of neighbors of node i that have state n, d_i is the degree of node i, and $r_{m,n}, \tilde{r}_{m,n} \geq 0$ are model parameters. The value $b_{m,n} \in [0,1]$ is the threshold at which a node changes its opinion from m to n. Similarly to the voter model, the additional rates $\tilde{r}_{m,n}$ control the noise in the system.

In the examples presented below, we used the following parameters:

$$r_{0,1} = r_{1,0} = 1, \quad \tilde{r}_{0,1} = \tilde{r}_{1,0} = 0.1, \quad b_{0,1} = b_{1,0} = 0.5.$$
 (S4)

S2. Supplementary Details on the Transition Manifold

The choice of lag time. For a very small lag time τ compared to the timescales of the system, the transition density functions are close to Dirac distributions, i.e., $p_{\boldsymbol{x}}^{\tau} \approx \delta_{\boldsymbol{x}}$ for all \boldsymbol{x} . Hence, the set

$$\mathbb{M}_{\tau} := \{ p_{\boldsymbol{x}}^{\tau} \mid \boldsymbol{x} \in \{0, 1\}^{N} \} \subset L^{1}$$
(S5)

consists of 2^N clearly distinct points, so that for $\boldsymbol{x} \neq \boldsymbol{y}$ we have $\|p_{\boldsymbol{x}}^{\tau} - p_{\boldsymbol{y}}^{\tau}\| \gg 0$. For a very large time τ on the other hand, the elements $p_{\boldsymbol{x}}^{\tau}$ of \mathbb{M}_{τ} are all close to the stationary distribution $\rho \in L^1$ of the process, i.e., for all \boldsymbol{x} we have $\|p_{\boldsymbol{x}}^{\tau} - \rho\| \approx 0$. (Existence of the stationary distribution is guaranteed if $\tilde{r}_{0,1}, \tilde{r}_{1,0} > 0$ for the processes considered here, as this implies that the resulting Markov process is irreducible and recurrent.) For intermediate values of τ , we often observe that the set \mathbb{M}_{τ} is close to a d-dimensional submanifold $\mathbb{M} \subset L^1$ called the t-ransition t

$$\inf_{f \in \mathbb{M}} \|f - p_{\boldsymbol{x}}^{\tau}\| \le \varepsilon \tag{S6}$$

for all $x \in \mathbb{X}$. Thus, the transition manifold \mathbb{M} and its dimension d depend on τ . However, it has been observed [28, 29, 63] that for systems exhibiting a low-dimensional collective variable, a transition manifold of a certain dimension is robust with respect to τ , in the sense that for a wide range of lag times the manifold \mathbb{M} differs only slightly and its dimension remains constant. Moreover, by Chapman–Kolmogorov, a consequence of Equation (2) from the main document is that for $t \geq 0$ we have

$$\tilde{p}_{\varphi(\boldsymbol{x})}^{\tau+t}(\boldsymbol{y}) := \int \tilde{p}_{\varphi(\boldsymbol{x})}^{\tau}(\boldsymbol{z}) p_{\boldsymbol{z}}^{t}(\boldsymbol{y}) d\boldsymbol{z} \approx \int p_{\boldsymbol{x}}^{\tau}(\boldsymbol{z}) p_{\boldsymbol{z}}^{t}(\boldsymbol{y}) d\boldsymbol{z} = p_{\boldsymbol{x}}^{\tau+t}(\boldsymbol{y}), \tag{S7}$$

i.e., the collective variable φ determined for the lag time τ characterizes the dynamics also for all times $\tau + t \geq \tau$.

As many model simulations of length τ must be sampled for approximating the transition manifold in practice, it is advantageous to choose the minimum τ within the above-mentioned range of lag times. However, if the lag time is chosen too small, the fast processes of the system will not equilibrate in that time frame, resulting in an unnecessarily large dimension d and a CV that is too fine-grained. In the case of systems with unknown time scales, it is advisable to examine trajectories to infer suitable lag times and to approximate the transition manifold for different τ , comparing their dimensions d. For the CNVM and the threshold model, we found that a lag time of the order such that nodes are expected to experience at least one state transition produces satisfactory results, i.e., $\tau \approx (r_{m,n} + \tilde{r}_{m,n})^{-1}$. Nevertheless, minor modifications may be required for specific examples due to the effects of network topology.

Approximating the transition manifold from data. As discussed in the main text, a CV can be obtained by learning a parametrization of the transition manifold $\mathbb{M} \subset L^1$ from data. The approach we suggest was first presented in [29] and requires the calculation of distances between transition densities in the form of the maximum mean discrepancy (MMD) [64]. The MMD measures the difference between the means of two distributions after mapping them into a reproducing kernel Hilbert space (or feature space) \mathbb{H} , which is induced by a positive definite kernel function $\kappa: \mathbb{X} \times \mathbb{X} \to \mathbb{R}$. The kernel κ also induces a feature map $\phi: \mathbb{X} \to \mathbb{H}$ such that

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_{\mathbb{H}}.$$
 (S8)

The MMD between two transition densities $p_{\boldsymbol{x}}^{\tau}$ and $p_{\boldsymbol{y}}^{\tau}$ is then defined as

$$MMD^{2}(p_{x}^{\tau}, p_{y}^{\tau}) := \|\mathbb{E}[\phi(\mathbf{X}(\tau, x))] - \mathbb{E}[\phi(\mathbf{X}(\tau, y))]\|_{\mathbb{H}}^{2},$$
(S9)

where $\mathbf{X}(\tau, \boldsymbol{x})$ denotes a random variable with distribution $p_{\boldsymbol{x}}^{\tau}$. Here, $\mathbb{E}[\phi(\mathbf{X}(\tau, \boldsymbol{x}))] = \int \phi(\boldsymbol{z})p_{\boldsymbol{x}}^{\tau}(d\boldsymbol{z})$ is a Hilbert-space valued integral. Its computation is, however, not required for the evaluation of the MMD: Given (S8), we can rewrite the definition of the MMD using the kernel, which is often referred to as the "kernel trick":

$$MMD^{2}(p_{\boldsymbol{x}}^{\tau}, p_{\boldsymbol{y}}^{\tau}) = \mathbb{E}\left[\kappa(\mathbf{X}(\tau, \boldsymbol{x}), \tilde{\mathbf{X}}(\tau, \boldsymbol{x}))\right] + \mathbb{E}\left[\kappa(\mathbf{X}(\tau, \boldsymbol{y}), \tilde{\mathbf{X}}(\tau, \boldsymbol{y}))\right] - 2 \mathbb{E}\left[\kappa(\mathbf{X}(\tau, \boldsymbol{x}), \tilde{\mathbf{X}}(\tau, \boldsymbol{y}))\right],$$
(S10)

where $\mathbf{X}(\tau, \boldsymbol{x}), \tilde{\mathbf{X}}(\tau, \boldsymbol{x})$ are independent random variables with distribution $p_{\boldsymbol{x}}^{\tau}$, and analogously for \boldsymbol{y} . This enables us to approximate the MMD by averaging over kernel function evaluations at samples of $p_{\boldsymbol{x}}^{\tau}$ and $p_{\boldsymbol{y}}^{\tau}$.

Given the set of anchor points $x^1, \dots, x^K \in \{0, 1\}^N$, we define the distance matrix $\Delta \in \mathbb{R}^{K \times K}$, which contains the squared MMD for all pairs of states, i.e.,

$$\Delta_{k_1,k_2} := \text{MMD}^2(p_{x^{k_1}}^{\tau}, p_{x^{k_2}}^{\tau}). \tag{S11}$$

To estimate Δ , we perform $S \in \mathbb{N}$ simulations of duration τ starting in each anchor point \boldsymbol{x}^k and denote the end points of these S simulations by $\boldsymbol{y}^{(k,1)}, \ldots, \boldsymbol{y}^{(k,S)} \in \{0,1\}^N$. Then we construct the kernel matrix $M \in \mathbb{R}^{K \times K}$ with

$$M_{k_1,k_2} := \frac{1}{S^2} \sum_{s_1,s_2=1}^{S} \kappa(\boldsymbol{y}^{(k_1,s_1)}, \boldsymbol{y}^{(k_2,s_2)}), \tag{S12}$$

from which we obtain the estimation

$$\Delta_{k_1,k_2} \approx M_{k_1,k_1} + M_{k_2,k_2} - 2 M_{k_1,k_2}. \tag{S13}$$

Finally, we apply a distance-based manifold learning algorithm to the distance matrix Δ , which yields an approximation to the d-dimensional CV φ evaluated at the anchor points \boldsymbol{x}^k , i.e., $\varphi(\boldsymbol{x}^1), \dots, \varphi(\boldsymbol{x}^K) \in \mathbb{R}^d$.

We chose the diffusion maps method [65] as the manifold learning algorithm. In the diffusion maps algorithm, we use a gaussian kernel with bandwidth 1 to estimate local similarity between data points, and employ a Fokker-Planck diffusion to construct a random walk on these points. This means, $\alpha = 1/2$ in [65]. A low-dimensional representation of the data is then obtained from the dominant eigenvectors of the resulting diffusion matrix. Numerical experiments have shown that our results are robust with respect to the choice of bandwidth and diffusion type parameter α .

We note that the original transition manifold theory [28, 66] requires Equation (2) from the main text to hold in a certain weighted L^2 norm. The justification that computations with MMD as a distance measure deliver this approximation property too has been provided in [29, Section 3] for dynamics exhibiting a timescale separation; see in particular Remark 3.16 therein.

S3. Supplementary Details on the Algorithmic Realization

In this section we provide details on each of the three steps of our method presented in the main text.

Step 1. In the first step of the method, it is crucial to choose a diverse set of dynamically relevant anchor points $x^1, \ldots, x^K \in \mathbb{X}$, in the sense that their respective transition densities cover \mathbb{M}_{τ} sufficiently well. For example, picking random states $x^k \sim \text{Unif}(\{0,1\}^N)$ does often not produce a desirable set of anchor points, as mostly states with about 50% 1's are sampled (by the law of large numbers). Thus, we would not gain any insights on the behavior of the system in states that have substantially more or less 1's than 50%. Instead, we suggest to construct states x that contain communities of nodes with the same state because these are especially dynamically stable in most spreading processes. As configurations of nodes that involve many alternating states tend to dissolve quickly under spreading dynamics, the system is predominantly observed in states with uniform clusters (if there are clusters), and the best understanding of long term dynamics is extracted by putting emphasis on such more relevant states.

In order to sample such a diverse set of anchor points $\boldsymbol{x}^1,\ldots,\boldsymbol{x}^K$, we employ Algorithm 1. We start with an empty (uninitialized) state \boldsymbol{x} and assign some random seed nodes to be of state 0 and 1. Then, we iteratively assign the state 0 to neighbors of nodes with state 0, and 1 to neighbors of nodes with state 1, until a random target count $c \in \{0,\ldots,N\}$ of 1's and N-c of 0's is reached. For each sample \boldsymbol{x}^k we use a random number of seeds between 1 and $N_{\text{seed}}=5$. Larger or more intricate networks may require a bigger number of seeds.

Algorithm 1 Sampling an anchor point x

```
1: x \leftarrow \text{empty array of size } N
 2: c \leftarrow \text{sample Unif}(\{0, \dots, N\})

    b target count of state 1

 3: n \leftarrow \text{sample Unif}(\{1, \dots, N_{\text{seed}}\})
                                                                                                          ▷ number of seeds
                                                                                 \triangleright reduce num. of seeds (if necessary)
 4: n \leftarrow \min\{n, c, N - c\}
 5: i_1, \ldots, i_{2n} \leftarrow \text{random indices from } \{1, \ldots, N\}
 6: x[i_1], \ldots, x[i_n] \leftarrow 0
                                                                                                    ▷ initialize seed points
 7: x[i_{n+1}], \ldots, x[i_{2n}] \leftarrow 1
                                                                                                    ▷ initialize seed points
    while (count of state 0) < N - c or (count of state 1) < c do
 9:
         if (count of state 0) < N - c then
10:
             i \leftarrow \text{index of random uninitialized neighbor of a node with state } 0
11:
             (If no such i exists, pick any random uninitialized node)
12:
             x[i] \leftarrow 0
         end if
13:
         if (count of state 1) < c then
14:
             i \leftarrow \text{index of random uninitialized neighbor of a node with state 1}
15:
16:
             (If no such i exists, pick any random uninitialized node)
17:
18:
         end if
19:
    end while
    return x
```

Step 2. In the second step of the method, we approximate a "parametrization" φ of the transition manifold \mathbb{M} from simulation data. To this end, most efficient methods require the computation of local distances. A particularly advantageous distance between densities $p_{\boldsymbol{x}}^{\tau}$ and $p_{\boldsymbol{y}}^{\tau}$ is the maximum mean discrepancy (MMD), as it can be efficiently approximated using samples of the densities [29, 64]. For each anchor point \boldsymbol{x}^k we conduct S simulations of the spreading process of length τ , yielding S samples for each transition density $p_{\boldsymbol{x}^k}^{\tau}$. Using this data we approximate the distance matrix $\Delta \in \mathbb{R}^{K \times K}$ that contains the pairwise MMDs between the densities associated to the anchor points, which is described in detail in section S2. Finally, we obtain evaluations of the collective variable φ at the anchor points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^K$ by applying a manifold learning algorithm to Δ . We choose the diffusion maps method [65] for this purpose. The dimension d of the CV can be obtained from the diffusion maps algorithm [67], or can be inferred by inspecting the network structure and plots of the transition manifold. Moreover, it is computationally cheap to test different dimensions d after the distance matrix Δ has been computed. The output of this second step of the method is evaluations of the d-dimensional CV φ at the anchor points, $\varphi(\boldsymbol{x}^1), \ldots, \varphi(\boldsymbol{x}^K) \in \mathbb{R}^d$.

Step 3. We learn a CV $\bar{\varphi}$ of the form

$$\bar{\varphi}(\boldsymbol{x}) = \begin{pmatrix} \bar{\varphi}_1(\boldsymbol{x}) \\ \vdots \\ \bar{\varphi}_d(\boldsymbol{x}) \end{pmatrix}, \quad \bar{\varphi}_j(\boldsymbol{x}) = \sum_{i=1}^N \Lambda_{j,i} \ x_i, \tag{S14}$$

by employing linear regression to find the optimal parameters Λ . To this end, we define the matrix $\Phi:=(\varphi(\boldsymbol{x}^1),\ldots,\varphi(\boldsymbol{x}^K))\in\mathbb{R}^{d\times K}$, the matrix $X\in\mathbb{R}^{N\times K},\,X_{i,k}:=x_i^k$, and note that

$$\bar{\Phi} := (\bar{\varphi}(\boldsymbol{x}^1), \dots, \bar{\varphi}(\boldsymbol{x}^K)) = \Lambda X \in \mathbb{R}^{d \times K}.$$
 (S15)

Optimal parameters Λ , that yield a maximal correlation between our fit Φ and the data Φ , are then found by solving the centered linear regression problem for the rows of Λ

$$\Lambda_{j,:} = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\operatorname{argmin}} \|EX^T \boldsymbol{\lambda} - E\Phi_{j,:}\|, \quad j = 1, \dots, d.$$
 (S16)

The operator $E := I - \frac{1}{K} \mathbf{1}(\mathbf{1}^T) \in \mathbb{R}^{K \times K}$, where I is the identity matrix and $\mathbf{1} = (1, \dots, 1)^T$, centers a vector around its mean. To prevent overfitting of the parameters Λ to the data, we use the *graph total variation* regularizer

$$TV(\Lambda_{j,:}) := \sum_{(i,k)\in\mathcal{E}} |\Lambda_{j,i} - \Lambda_{j,k}|,$$
(S17)

where \mathcal{E} is the edge set of the graph. This yields the generalized LASSO [68] linear regression problem

$$\Lambda_{j,:} = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\operatorname{argmin}} \|EX^T \boldsymbol{\lambda} - E\Phi_{j,:}\| + \alpha \cdot \text{TV}(\boldsymbol{\lambda}).$$
 (S18)

The strength $\alpha > 0$ of the penalty can be optimized for a given data set via cross-validation [69]. Due to the regularization, the solution $\Lambda_{j,:}$ tends to be constant within network communities, which reinforces the interpretation of $\bar{\varphi}$ to measure the count of 1's in densely connected regions of the network. The optimization problem (S18) is convex and can be efficiently solved using off-the-shelf solvers. We employ the OSQP solver [70].

For most network structures that we studied, the ansatz functions presented in (S14) resulted in CVs of good quality. In some cases however, it can be beneficial to pre-weight each node. For example, for the Albert–Barabási network (Example 4 in the main text) we noticed a strong correlation of the optimal (in the sense of (S18)) weights Λ with the node degree. This conflicts with the graph total variation regularizer because the network contains many edges between nodes of substantially different degree. We successfully tackled this issue by incorporating a pre-weighting of each node with its degree into the ansatz functions:

$$\bar{\varphi}(\boldsymbol{x}) = \sum_{i=1}^{N} \Lambda_i \ d_i \ x_i, \tag{S19}$$

where d_i denotes the degree of node i. Transferring this pre-weighting to the regression problem (S18) yields

$$\Lambda = \underset{\boldsymbol{\lambda} \in \mathbb{R}^N}{\operatorname{argmin}} \|EX^T D \boldsymbol{\lambda} - E\Phi\| + \alpha \cdot \text{TV}(\boldsymbol{\lambda}), \tag{S20}$$

where $D = \operatorname{diag}(d_1, \ldots, d_N) \in \mathbb{R}^{N \times N}$.

If our suggested ansatz functions do not yield satisfying results for a particular network, they could also be completely replaced by functions that might be better suited for that problem.

S4. Supplementary Details on the Numerical Examples

In this section we provide additional details on the numerical examples presented in the main text, which all employ the continuous-time noisy voter model (CNVM) as the dynamical model. Moreover, we apply our method to a threshold model on the same network structures as in the main text. (For details on the models, see section S1.) The resulting CVs are identical, which indicates that the network plays a significant role in the behavior of spreading processes on networks – maybe even a more important role than the specific spreading model employed (voter model, threshold model, Glauber dynamics, etc.).

The Python code for the numerical examples is available at

https://github.com/lueckem/spreading-processes-CVs.

The necessary model simulations were conducted using our open-source Python package SPoNet (Spreading Processes on Networks, https://github.com/lueckem/SPoNet).

For the calculation of the distance matrix Δ , cf. (S13), we used the Gaussian kernel

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}{\sigma^2}\right). \tag{S21}$$

The bandwidth σ was set to $\sqrt{N/2}$, where N is the number of nodes of the respective network. The other parameter values of the examples are summarized in Table I. The experiments were conducted on a 16-core CPU with 32 GB of memory and took less than 20 minutes each, including the simulations of the CNVM, the manifold learning, and the linear regression with cross-validation. The most costly step ($\sim 70\%$ of runtime) in these examples was the calculation of the transition manifold parametrization, i.e., the calculation of the distance matrix (S13) and application of the diffusion maps algorithm.

Results for the threshold model. As stated before, employing the threshold model instead of the voter model results in identical transition manifolds and CVs in the three example networks.

For the stochastic block model network (Example 1 in the main text), the transition manifold is again a three dimensional cuboid. The CV learned by our method is illustrated in Fig. S1.

For the ring network (Example 2 in the main text) there is no low dimensional transition manifold and the CV again represents Fourier coefficients, cf. Fig. S2.

For the Albert–Barabási network (Example 4 in the main text) the transition manifold is again one-dimensional, as shown in Fig. S3.

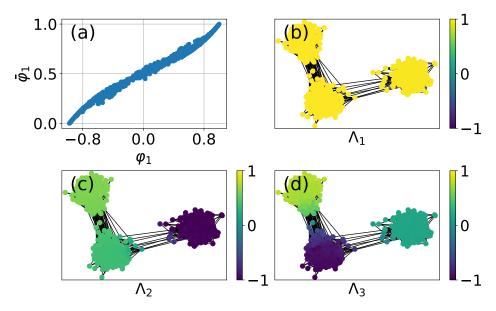


FIG. S1: Optimal Λ of the threshold model on the stochastic block model network. (a): Data $\varphi_1(\boldsymbol{x}^k)$ versus optimal fit $\bar{\varphi}_1$. (b)-(d): Optimal Λ entries for the respective coordinates plotted as color values on the network.

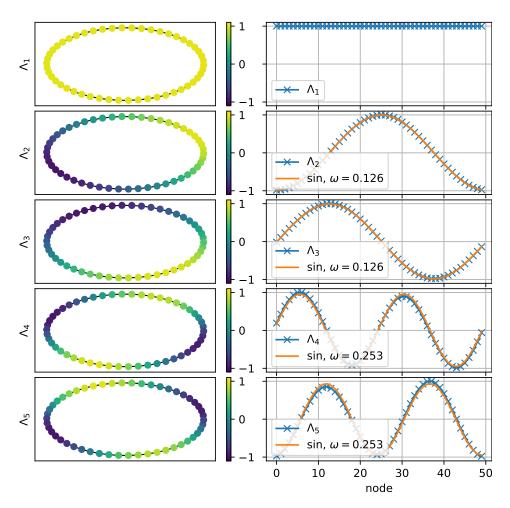


FIG. S2: Left: optimal $\Lambda_{i,:}$ plotted as color values on the ring-shaped network. Right: $\Lambda_{i,:}$ (blue crosses) and a sine fit (orange line).

Parameter	stochastic block model	ring	3-regular	Albert–Barabási
N	900	50	500	500
K	2000	2000	1000	1000
S	100	300	100	100
τ (voter model)	2	5	4	4
τ (threshold model)	1	3	8	2
d	3	5	1	1

TABLE I: Parameter values used in the examples.

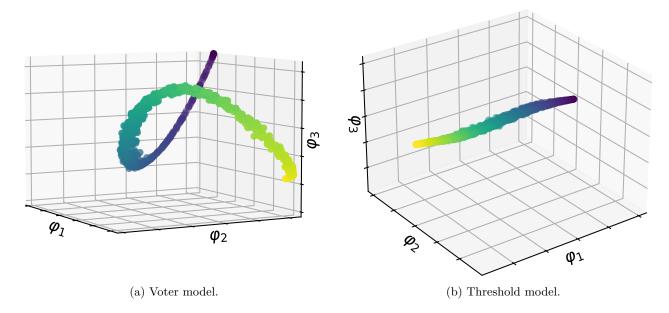


FIG. S3: In the Albert-Barabási example the transition manifold is one-dimensional.

Random regular graphs. We have also studied the voter model and the threshold model on a uniformly random 3-regular graph with N=500 nodes (Example 3 in the main text). The statistical correlation between the edges (caused by regularity) poses a severe challenge to reduced modelling approaches [25].

The resulting transition manifold is one-dimensional and the learned CV calculates the global share of 1's, cf. Fig S4. Hence, the CV is identical to the optimal CV in the complete graph and Erdős–Rényi random graph (with sufficiently large edge density) setting [10]. However, the dynamics on random 3-regular graphs behaves (quantitatively) differently than on a complete graph. In particular, the known mean-field equation, which constitutes the large population limit of the voter model on complete networks, is not valid for random regular graphs. We are not aware of a similar dynamical equation for the share of 1's in the random regular graph setting but the one-dimensionality of the transition manifold indicates its existence.

S5. Validation

Recall that the transition manifold approach is seeking for a low-dimensional parametrization φ of the set of all transition densities $p_{\boldsymbol{x}}^{\tau}$, $\boldsymbol{x} \in \{0,1\}^N$. Thus, for two states \boldsymbol{x} and \boldsymbol{y} the distance between $p_{\boldsymbol{x}}^{\tau}$ and $p_{\boldsymbol{y}}^{\tau}$ should correlate with the distance between $\varphi(\boldsymbol{x})$ and $\varphi(\boldsymbol{y})$. Thus, the quality of a collective variable φ can be assessed using the following heuristic. Given a small $\varepsilon > 0$ and two states $\boldsymbol{x}^1, \boldsymbol{x}^2 \in \{0,1\}^N$ with $\varphi(\boldsymbol{x}^1) \approx \varphi(\boldsymbol{x}^2)$, we define the time t^* as

$$t^* := \inf\{t \ge 0 \mid \forall \tau \ge t : \|p_{x^1}^{\tau} - p_{x^2}^{\tau}\| < \varepsilon\}. \tag{S22}$$

This is well-defined because we assume that all p_x^{τ} converge to a unique stationary distribution for $\tau \to \infty$. If t^* is rather large, this implies that the CV is very coarse because initial states with similar CV values may lead to quite different behavior over long timescales. But a good CV has the property that starting the system in x^1 versus in x^2

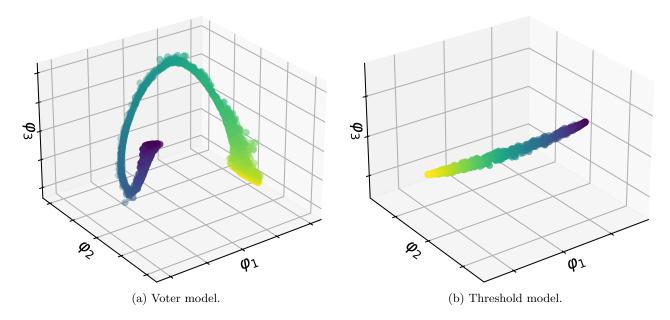


FIG. S4: For the random 3-regular graph the transition manifold is one-dimensional.

should make almost no difference after a short time, and hence, good CVs exhibit a small t^* . However, this property alone does not sufficiently characterize the quality of the CV (for example, $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ implies $t^* = 0$, but it does not simplify the state at all). Hence, we additionally define a third state \boldsymbol{x}^3 with a clearly distinct reduced state $\|\varphi(\boldsymbol{x}^3) - \varphi(\boldsymbol{x}^1)\| \gg 0$. Then the CV is of good quality if the associated distributions $p_{\boldsymbol{x}^1}^{\tau}$ and $p_{\boldsymbol{x}^3}^{\tau}$ are also substantially different for a long time, i.e.,

$$\hat{t} := \inf\{t \ge 0 \mid \forall \tau \ge t : \|p_{x^1}^{\tau} - p_{x^3}^{\tau}\| < \varepsilon\}$$
 (S23)

should be large. If \hat{t} is rather small, this implies that the CV is too fine-grained because it assigns different values to initial states that lead to a similar dynamics after a short time. The best possible CV achieves both the smallest t^* for all choices of $\mathbf{x}^1, \mathbf{x}^2$ and the largest \hat{t} for all choices of $\mathbf{x}^1, \mathbf{x}^3$, as it exactly filters out the state information with a short timescale impact, but keeps the information that leads to fundamentally different dynamics on a long timescale.

We propose the following numerical method for validation. After the approximation $\bar{\varphi}$ of the collective variable has been calculated by our method, we pick three states $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3 \in \{0,1\}^N$ as discussed above, i.e, such that $\bar{\varphi}(\boldsymbol{x}^1) \approx \bar{\varphi}(\boldsymbol{x}^2)$ and $\bar{\varphi}(\boldsymbol{x}^3)$ is substantially different. Then we compare the distributions $p_{\boldsymbol{x}^1}^{\tau}$, $p_{\boldsymbol{x}^2}^{\tau}$, and $p_{\boldsymbol{x}^3}^{\tau}$ via their maximum mean discrepancy (MMD)

$$MMD^{2}(\boldsymbol{x}^{i}, \boldsymbol{x}^{j}; t) := \mathbb{E}\left[\kappa\left(\mathbf{X}(t, \boldsymbol{x}^{i}), \tilde{\mathbf{X}}(t, \boldsymbol{x}^{i})\right)\right] + \mathbb{E}\left[\kappa\left(\mathbf{X}(t, \boldsymbol{x}^{j}), \tilde{\mathbf{X}}(t, \boldsymbol{x}^{j})\right)\right] - 2 \mathbb{E}\left[\kappa\left(\mathbf{X}(t, \boldsymbol{x}^{i}), \tilde{\mathbf{X}}(t, \boldsymbol{x}^{j})\right)\right],$$
(S24)

where $\mathbf{X}(t, \boldsymbol{x})$ and $\tilde{\mathbf{X}}(t, \boldsymbol{x})$ are independent random variables with distribution $p_{\boldsymbol{x}}^t$, and $\kappa(\cdot, \cdot)$ is a Gaussian kernel. If the learned CV $\bar{\varphi}$ is good, we expect that $\mathrm{MMD}(\boldsymbol{x}^1, \boldsymbol{x}^2; t)$ goes to 0 quickly (as t^* is small), while both $\mathrm{MMD}(\boldsymbol{x}^1, \boldsymbol{x}^3; t)$ and $\mathrm{MMD}(\boldsymbol{x}^2, \boldsymbol{x}^3; t)$ stay large for a long time (as \hat{t} is large). We estimate the three MMDs by replacing the expectation in (S24) with averages from model simulations. To prove the quality of the learned CV $\bar{\varphi}$, one would have to conduct this experiment for all possible choices of $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3$, which is not feasible. However, we argue that doing it for a few (random) choices is a good indicator of whether the CV is correct.

A weaker (but easier to compute) indicator of the quality of a CV is given by the differences between the projected

flip states of nodes until $\bar{\varphi}(\mathbf{x}^1) \approx \bar{\varphi}(\mathbf{x}^2)$.

¹ Given x^1 , we sample x^2 using a Markov chain Monte Carlo method. Starting with a uniformly random x^2 , we randomly

distributions, i.e.,

$$\operatorname{MMD}_{\varphi}^{2}(\boldsymbol{x}^{i}, \boldsymbol{x}^{j}; t) := \mathbb{E}\left[\kappa\left(\bar{\varphi}(\mathbf{X}(t, \boldsymbol{x}^{i})), \bar{\varphi}(\tilde{\mathbf{X}}(t, \boldsymbol{x}^{i}))\right)\right] + \mathbb{E}\left[\kappa\left(\bar{\varphi}(\mathbf{X}(t, \boldsymbol{x}^{j})), \bar{\varphi}(\tilde{\mathbf{X}}(t, \boldsymbol{x}^{j}))\right)\right] - 2 \mathbb{E}\left[\kappa\left(\bar{\varphi}(\mathbf{X}(t, \boldsymbol{x}^{i})), \bar{\varphi}(\tilde{\mathbf{X}}(t, \boldsymbol{x}^{j}))\right)\right].$$
(S25)

(The kernels κ in (S24) and in (S25) are clearly different objects, as the state \boldsymbol{x} and reduced state $\bar{\varphi}(\boldsymbol{x})$ have different dimensions.) In contrast to (S24), we expect that $\mathrm{MMD}_{\varphi}(\boldsymbol{x}^1,\boldsymbol{x}^2;t)$ is close to 0 for all times t, as states with the same CV value should lead to identical effective (projected) dynamics. The differences $\mathrm{MMD}_{\varphi}(\boldsymbol{x}^1,\boldsymbol{x}^3;t)$ and $\mathrm{MMD}_{\varphi}(\boldsymbol{x}^2,\boldsymbol{x}^3;t)$ should again be large for small and intermediate t. For very large t, even they should approach zero as the distribution of all $\mathbf{X}(t,\boldsymbol{x}^j)$ converge to the stationary distribution.

Results. For the stochastic block model example, we picked states x^1 and x^2 such that the share of 1's is 10% in cluster 1, 0% in cluster 2, and 50% in cluster 3, see Fig. S5. In state x^3 the total number of 1's is the same, but they are distributed uniformly, irrespective of the clusters. For both the voter model dynamics and the threshold model dynamics the time t^* is very small ($t^* \approx 5$) compared to \hat{t} . This indicates that the learned CV is good, as discussed above. An interesting difference between the voter model and the threshold model on this network is that the metastable states (where each cluster is dominated by one opinion) persist much longer in the threshold model. As x^1 and x^3 typically move into different metastable states, their MMD stays large much longer in the threshold model.

For the ring network, state x^1 is chosen randomly (i.i.d. for every node), x^2 is chosen such that the collective variable matches, and x^3 is given by a random permutation of node states of x^1 , see Fig. S6. Hence, state x^1 and x^3 have the same number of 1's. Here, the time t^* is rather large compared to \hat{t} , which shows that this CV is too coarse. This is not surprising because we capped the CV dimension at five, whereas the transition manifold approach suggested a higher dimension, see the main text. One could improve the quality of this CV (i.e., reduce t^*) at the cost of increasing its dimension and hence complexity.

For the Albert–Barabási network, state x^2 is such that the top 10% of nodes with largest degree are in state 1, whereas in x^3 the 10% of nodes with smallest degree are in state 1, see Fig. S7. Hence, the total number of 1's is identical, but the CV, which measures the degree-weighted count of 1's, differs substantially. State x^1 is chosen to have the same degree-weighted count of 1's as state x^2 . This experiment indicates that the learned CV is good because t^* is small compared to \hat{t} . Moreover, we again observe that due to the longer persistence of metastable states in the threshold model, the MMD between x^1 and x^3 stays much larger in the threshold model than in the voter model.

Finally, we validate that the simple share of 1's is indeed a good CV in the case of the random 3-regular network in Fig. S8. States x^1 and x^2 are random permutations with the same number of 1's, whereas x^3 has a different number of 1's. The time t^* is again small compared to \hat{t} , which implies that the CV is good.

Robustness of the results. For all considered network topologies and spreading processes on them we have tested the robustness of our results with respect to variations in the model parameters r, \tilde{r} . We observed that the results reported here all remained valid qualitatively as long as the parameter values were not varied by many orders of magnitude.

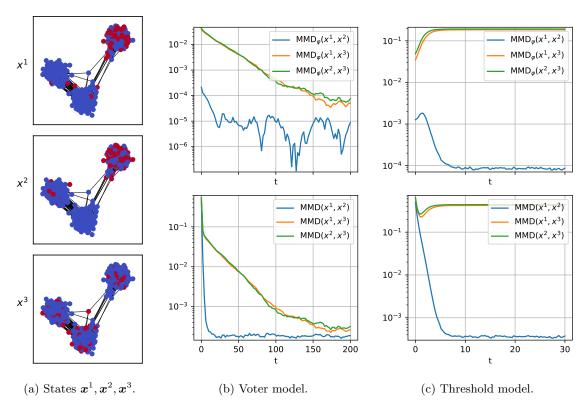


FIG. S5: Validation for the stochastic block model example. For a definition of MMD and MMD $_{\varphi}$ see (S24) and (S25).

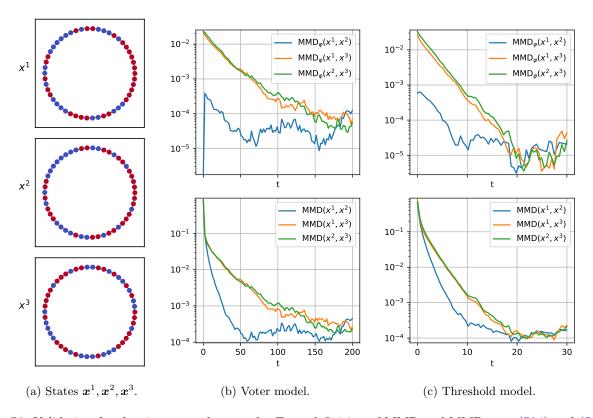


FIG. S6: Validation for the ring network example. For a definition of MMD and MMD $_{\varphi}$ see (S24) and (S25).

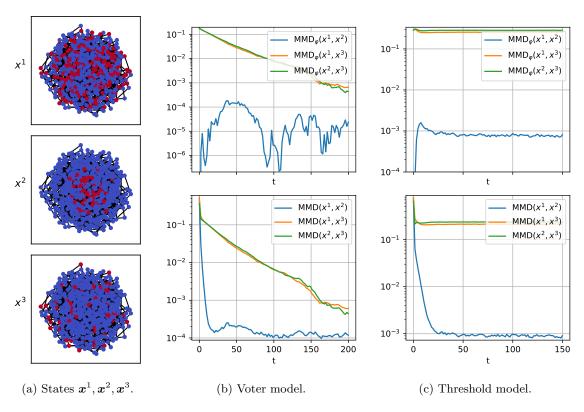


FIG. S7: Validation for the Albert–Barabási network example. For a definition of MMD and MMD $_{\varphi}$ see (S24) and (S25).

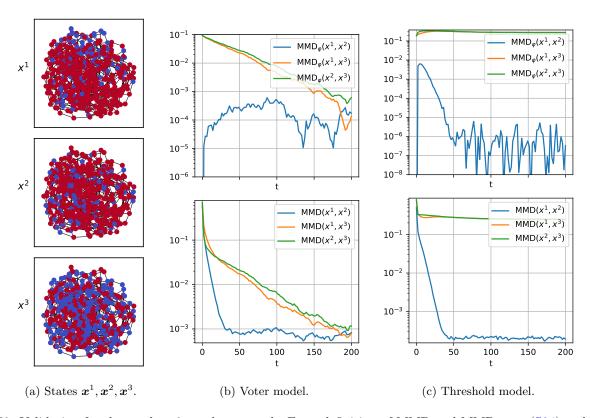


FIG. S8: Validation for the random 3-regular network. For a definition of MMD and MMD $_{\varphi}$ see (S24) and (S25).