Efficient PAC Learnability of Dynamical Systems Over Multilayer Networks

Zirou Qiu,[1,2] Abhijin Adiga, [2] Madhav V. Marathe,[1,2] S. S. Ravi,[2,3]
Daniel J. Rosenkrantz,[2,3] Richard E. Stearns,[2,3] Anil Vullikanti[1,2] [1]

[1]Computer Science Dept., University of Virginia.
[2]Biocomplexity Institute and Initiative, University of Virginia.
[3]Computer Science Dept., University at Albany – SUNY.

Abstract

Abstract. Networked dynamical systems are widely used as formal models of real-world cascading phenomena, such as the spread of diseases and information. Prior research has addressed the problem of learning the behavior of an unknown dynamical system when the underlying network has a single layer. In this work, we study the learnability of dynamical systems over *multilayer* networks, which are more realistic and challenging. First, we present an efficient PAC learning algorithm with provable guarantees to show that the learner only requires a small number of training examples to infer an unknown system. We further provide a tight analysis of the Natarajan dimension which measures the model complexity. Asymptotically, our bound on the Nararajan dimension is tight for *almost all* multilayer graphs. The techniques and insights from our work provide the theoretical foundations for future investigations of learning problems for multilayer dynamical systems.

Conference version. The paper is accepted at `ICML-2024`.

1 Introduction

Networked dynamical systems are mathematical frameworks for numerous cascade processes, including the spread of social behaviors, information, diseases, and biological phenomena [6, 35, 43, 57, 55, 39, 37]. In general, such a system consists of an *underlying graph* where vertices are entities (e.g., individuals, genes), and edges represent relationships between the entities. In modeling the contagion propagation, each vertex maintains a *state* and a set of *interaction functions* (i.e., behavior), which specify how the state evolves over time. Overall, the system dynamics proceeds in discrete time, with vertices updating their states using interaction functions.

Inferring the unknown components of dynamical systems is an active research area [11, 18, 13, 2, 42, 46]. One direction focuses on learning the unknown *interaction functions* of vertices. Interaction functions are crucial to the system dynamics; they specify the decision-making rules that entities employ to update their states. An illustrative example is the class of threshold interaction functions [28], which are widely used to model the spread of social contagions [41, 64]. Under this framework, each entity in the network has a decision threshold that represents the tipping point for a behavioral (i.e., state) shift. For instance, in rumor propagation, a person's belief changes when the number of neighbors believing in the rumor reaches a threshold [60]. Overall, the interaction functions define the mechanism of the cascade process, which also describes the system's global behavior [21].

Existing methods for learning interaction functions only apply to the case when the underlying graph has a *single-layer*. Nevertheless, such single-layer frameworks often are oversimplifications of reality, as real-world networks encompass diverse types of connections that are not adequately captured by single-layer graphs [38]. To our knowledge, the learning problem for the more complex and realistic *multilayer* setting (that gives rise to multi-relational networks) has not received attention in the literature. In this work, we fill this gap through a formal study of **the learnability of dynamical systems over multilayer networks**.

The multilayer setting. The graph in our target system consists of multiple layers with generally *different* set of edges in each layer. This is a classic setting for multilayer networks, and the capacity of such networks to model complex real-world phenomena has been widely recognized (e.g., [29, 38, 48]). Notably, multilayer networks allow heterogeneous ties between vertices, with edges in each layer capturing a particular type of interaction (e.g., close friend, acquaintance) [48, 38, 20]. Further, the multilayer framework enables the modeling of more realistic and complex cascades that involve *cross-layer interactions* [19, 54, 7]. These cascades are characterized by increasingly intricate interaction functions of the system.

Problem description. Consider a multilayer networked system where the interaction functions of vertices are *unknown*. By inferring the missing functions, we aim to learn a system that captures the behavior of the true unknown system, with performance guarantees under the Probably Approximately Correct (PAC) model [61]. We learn from *snapshots of the true system's dynamics*, a common scheme considered in related papers (e.g., [10, 65, 13]). To further measure the expressive power of the hypothesis class and characterize the sample complexity of learning, we examine the *Natarajan dimension* [47] of the model, a well-known generalization of the VC dimension [63]. Overall, we aim to address the following questions: (i) *How to efficiently learn multilayer systems?* (ii) *What is the complexity of our model for learning multilayer systems?*

Challenges. The multilayer setting poses new challenges. First, the number of hypotheses grows exponentially in the number of layers, thus requiring a learner to search a much larger space. Further, a learner is tasked with extracting information from the complex *cross-layer interactions*. For example, while the training data (snapshots of dynamics) indicates a vertex's state change, it *does not* indicate which layer(s) triggered the change. Additionally, analyzing a learner's performance is challenging, as incorrect predictions could be caused by any combination of layers. The intertwined connections between vertices in the multilayer setting further complicate the analysis of model complexity. Collectively, these factors distinctly set apart our problem from its single-layer counterpart.

Our contributions are as follows:

- **Efficient learning.** We show that a small training set is sufficient to efficiently PAC learn a multilayer system. Specifically, we obtain the following results. (i) We develop an efficient PAC learning algorithm with provable guarantees: w.p. at least $1 - \delta$, the prediction error is at most ϵ, for any $\epsilon, \delta > 0$. (ii) For any fixed ϵ and δ, the number of training examples used by our algorithm is only $O(\sigma k \log(\sigma k))$, where k is the number of layers and σ is the number of vertices with unknown interaction functions. Thus, when σ is fixed, the size of an adequate training set does *not* increase with the network size or density. This result also provides an upper bound on the sample complexity that is tighter than the general information-theoretic bound [30]. (iii) We extend the proposed learner to the Probably Mostly Approximately Correct setting [4] and prove that the amount of training data can be further reduced when the learner is allowed to make approximated predictions. (iv) Using real-world and synthetic multilayer networks, we experimentally explore the relationship between the PAC algorithm's performance and system parameters under various scenarios.

- **Model complexity.** We provide a tight analysis of the Natarajan dimension (Ndim), which measures the expressive power of the learning model. (i) We present a novel combinatorial structure and establish its *equivalence* to shatterable sets. (ii) Based on this connection, we develop an (efficient) method for constructing shatterable sets and show that when restricting the system to an individual layer, Ndim is *exactly* σ, the number of vertices with unknown interaction functions. This precise characterization could be of independent interest. (iii) We then extend the argument to show that for a k-layer system, Ndim is between σ and $k\sigma$ and present classes of instances where the bounds are tight. This result also provides a lower bound on the sample complexity. (iv) Lastly, using a probabilistic argument, we show that our upper bound $k\sigma$ is asymptotically tight almost surely: for *almost all graphs*, Ndim is exactly $k\sigma$.

Related work. For learning *single-layer* networks, many researchers have developed methods to address problems related to *cascade inference* (e.g., learning the diffusion functions at vertices, edge parameters, source vertices, and contagion states of vertices) by observing propagation dynamics. For instance, Chen et al. [10] study the problem of learning both the edge probability and source vertices under the celebrated independent cascade model. In another work by Conitzer et al. [13], they examine the setting where the opinions (states) of vertices are to be learned in stochastic cascades under the PAC scheme (PMAC to be more precise). Lokhov [42] investigates the issue of parameter reconstruction for a special diffusion model, where they learn from real cascades. Other representative examples on learning single-layer systems include [11, 14, 16, 18, 22, 27, 32, 33, 36, 46, 65]. Questions on learning the network topology from the cascades have also been studied; see, for example [34, 51, 1, 23, 45, 26, 58]. To our knowledge, the problem of learning the interaction functions of networked *multilayer* systems has not been examined.

The paper that is most closely related work is by Adiga et al. [2] in ICML 2019, where the PAC learnability of threshold interaction functions in *single-layer* networked systems is studied. They present a consistent learner when there are only positive examples and show the hardness of learning when negative examples are also included. They also bound the sample complexity based on the VC dimension. As mentioned earlier, the multilayer setting introduces new *challenges* that do not arise in the single-layer setting. For these reasons, we note that the results and techniques in [2] cannot be directly applied to our multilayer setting, which requires new techniques developed in our work.

2 Preliminaries

Our setting aligns with the existing research on learning networked systems. For the readers' reference, we have included a list of settings used in several related papers in the Appendix (Section 6.1).

2.1 Multilayer Networked Dynamical Systems

We now present the Multilayer Dynamical Systems as a formal model for cascades on multilayer networks.

Multilayer networks. All the graphs considered are undirected. For any integer $k \geq 1$, let $[k]$ denote the set $\{1, \ldots, k\}$. A *multilayer network* [38] is a sequence of graphs $\mathcal{M} = \{\mathcal{G}_i\}_{i=1}^{k}$, $\mathcal{G}_i = (\mathcal{V}, \mathcal{E}_i)$, where $\mathcal{V}$ is a set of n vertices *shared* by all graphs in $\mathcal{M}$, and $\mathcal{E}_i$ is the set of edges in $\mathcal{G}_i$, $i \in [k]$. The edge sets of graphs in $\mathcal{M}$ are generally *different*. Overall, one can view $\mathcal{M}$ as a k-layer network where $\mathcal{G}_i \in \mathcal{M}$ is the ith layer.

Dynamical systems. Dynamical systems on multilayer networks are generalizations of systems over single-layer networks. A *Multilayer Synchronous Dynamical System* (**MSyDS**) over the Boolean domain $\mathbb{B} = \{0, 1\}$ is a triple $h^* = (\mathcal{M}, \mathcal{F}, \Psi)$:

- $\mathcal{M} = \{\mathcal{G}_i\}_{i=1}^{k}$ is an underlying multilayer network with k layers. Each vertex has a state from $\mathbb{B}$.
- $\mathcal{F} = \{f_{i,v} : i \in [k], v \in \mathcal{V}\}$ is a collection of functions, with $f_{i,v}$ denoting vertex v's **interaction function** on the ith layer $\mathcal{G}_i$.
- $\Psi = \{\psi_v : v \in \mathcal{V}\}$ is a collection of functions, with ψ_v denoting the **master function** of vertex v.

The system dynamics proceeds in discrete time. Starting from an initial configuration of vertex states, at each step, vertices update states *synchronously* using interaction functions and master functions. Specifically, for any $t \geq 0$, the state of each vertex v at time $t + 1$ is computed as follows:

- For each $\mathcal{G}_i \in \mathcal{M}$, the interaction function $f_{i,v} \in \mathcal{F}$ is evaluated; the inputs are the time-t states of vertices in v's closed neighborhood (i.e., v and its neighbors) in $\mathcal{G}_i$; $f_{i,v}$ then outputs a value in $\mathbb{B}$. This gives a k-vector $\mathbf{W}_v$ for each v, where $\mathbf{W}_v(i)$ is the output of $f_{i,v}$, $i \in [k]$.
- Next, the master function ψ_v is evaluated, with $\mathbf{W}_v$ as the input. The output of ψ_v, which is a value in $\mathbb{B}$, is the **next state** of v (i.e., its state at time $t + 1$).

Interaction functions. We focus on *threshold* interaction functions, a classic framework to model the spread of social contagions [53, 10, 64, 28]. In particular, each $v \in \mathcal{V}$ has an integer threshold $\tau_i(v) \in [0, \deg_i(v) + 2]$ for each layer $\mathcal{G}_i$, $i \in [k]$; $\deg_i(v)$ is the degree of v in $\mathcal{G}_i$. The interaction function $f_{i,v} \in \mathcal{F}$ outputs 1 if the number of active (i.e., state-1) vertices in v's closed neighborhood in $\mathcal{G}_i$ is **at least** $\tau_i(v)$; $f_{i,v}$ outputs 0 otherwise. If $f_{i,v}$ outputs 1, we say that the **threshold condition** of v is satisfied on $\mathcal{G}_i$.

Master functions. The two classes of master functions proposed in the literature are `OR` and `AND` [50, 40, 9]. When function ψ_v is `OR`, the next state of v is 1 iff **there exists** a layer $i \in [k]$ where the interaction function $f_{i,v}$ evaluates to 1. In other words, v's next state is 1 iff its threshold condition is satisfied in at least one layer. Analogously, for `AND` functions, the next state of v is 1 iff $f_{i,v}$ evaluates to 1 in all the layers.

An illustrative example. Consider the scenario of a rumor spreading on a 2-layer social network with two types of ties (e.g., “friends” and “co-workers”). A person’s belief in the rumor changes if the number of neighbors in any one (i.e., the OR master function) of the layers believing in the rumor reaches a certain decision threshold; however, the person’s thresholds for the two layers may be different due to the difference in the strengths of the social ties.

Configuration. A **configuration** $\mathcal{C}$ is an n-bit binary vector that specifies the state of each vertex at a given time step. We use $\mathcal{C}(v)$ to denote the state of vertex v in $\mathcal{C}$. A configuration $\mathcal{C}'$ is the **successor** of $\mathcal{C}$ under a system h^* if $\mathcal{C}'$ evolves from $\mathcal{C}$ in one time step; this is denoted by $\mathcal{C}' = h^*(\mathcal{C})$. Overall, the evolution of system h^* can be represented as a time-ordered sequence of configurations. An example of an evolution from $\mathcal{C}$ to $\mathcal{C}'$ is shown in Fig. 1. We note that the goal of this figure was only to present a toy example of a multilayer system that makes it easier for readers to understand the formal definitions. As for examples of realistic large multilayer systems, we refer the reviewer to references such as [38] and [29].

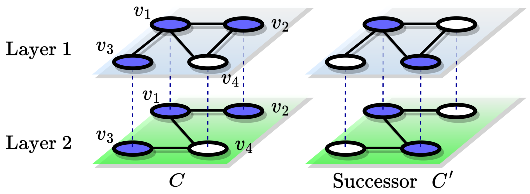


Figure 1: A 2-layer threshold system with OR master functions. Threshold values of vertices v_1 to v_4 in layer 1 are $(2,3,3,2)$, and in layer 2 are $(3,3,2,1)$. State-1 vertices are in blue. The configuration $\mathcal{C} = (1,1,1,0)$, and its successor is $\mathcal{C}' = (1,0,0,1)$.

2.2 The Learning Setting

There is a ground truth MSyDS h^*. The learner is only given partial information about h^*, where the interaction functions (on all layers) of a subset $\mathcal{V}' \subseteq \mathcal{V}$ of vertices are **unknown**. Let $\sigma = |\mathcal{V}'|$. The *hypothesis class* $\mathcal{H}$ consists of $\Theta(n^{\sigma k})$ MSyDSs over all possible threshold values of vertices in $\mathcal{V}'$. The goal of a learner is to infer an MSyDS $h \in \mathcal{H}$ that is a good approximation of h^* by inferring the unknown interaction functions. When $\mathcal{V}' = \mathcal{V}$, the thresholds of all vertices must be learned.

Training. We learn the target system h^* from snapshots of its dynamics under the PAC framework. Formally, let $\mathcal{X} = \{0,1\}^n$ be the set of all n-bit binary vectors. Let $\mathcal{T} = \{(\mathcal{C}_j, \mathcal{C}'_j)\}_{j=1}^q$ be a *training set* of q examples, which consists of the snapshots of system dynamics. Following the PAC setting, examples in $\mathcal{T}$ are *configuration pairs*, where $\mathcal{C}_j$ is drawn i.i.d. from an *unknown* distribution $\mathcal{D}$, and $\mathcal{C}'_j = h^*(\mathcal{C}_j)$ is the successor of $\mathcal{C}_j$ under h^*. We use $\mathcal{T} \sim \mathcal{D}^q$ to denote such a training set.

Intuitively, the training set $\mathcal{T}$ comprises snapshots of the freely-evolved system dynamics. The learner is **not** involved in generating $\mathcal{T}$. For instance, learning from a *trajectory* can be seen as a special case of our PAC scheme. Specifically, when $\mathcal{T}$ consists of configurations on a trajectory Y, it suggests a special sampling distribution (unknown to the learner), where only configurations on Y are sampled with positive probability. Importantly, our proposed learner works under *any* sampling distribution.

Predictions. Given a new sample $\mathcal{C} \sim \mathcal{D}$, a hypothesis $h \in \mathcal{H}$ makes a successful prediction if $h(\mathcal{C}) = h^*(\mathcal{C})$.

The **true error** of a hypothesis h is defined as $L_{(\mathcal{D},h^*)}(h) := \Pr_{\mathcal{C}\sim\mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})]$. In the PAC model, when $\mathcal{T}$ is sufficiently large, the goal of a learner is to output an $h \in \mathcal{H}$ s.t. with probability at least $1-\delta$ over $\mathcal{T} \sim \mathcal{D}^q$, it holds that $L_{(\mathcal{D},h^*)}(h) \le \epsilon$, for any given $\epsilon, \delta \in (0,1)$. The minimum number of training examples needed by any PAC learner to learn $\mathcal{H}$ is called the *sample complexity* of $\mathcal{H}$.

Natarajan dimension. Learning a hypothesis $h \in \mathcal{H}$ in our context can be cast as a *multiclass classification* problem, where h maps a configuration $\mathcal{C}$ to one of the possible 2^n configurations (classes). To characterize the *model complexity* and the *expressive power* of the hypothesis class $\mathcal{H}$, we turn to the Natarajan dimension [47], which extends the VC dimension to multiclass settings. Formally, the **Natarajan dimension** of $\mathcal{H}$, denoted by $\text{Ndim}(\mathcal{H})$, is the maximum size of a *shatterable* set. A set $\mathcal{R} \subset \mathcal{X}$ is **shattered** by $\mathcal{H}$ if for every $\mathcal{C} \in \mathcal{R}$, there are two associated configurations, $\mathcal{C}^A, \mathcal{C}^B \in \mathcal{X}$, s.t. (i) $\mathcal{C}^A \neq \mathcal{C}^B$, and (ii) for every subset $\mathcal{R}' \subseteq \mathcal{R}$, there exists $h \in \mathcal{H}$ where $\forall \mathcal{C} \in \mathcal{R}'$, $h(\mathcal{C}) = \mathcal{C}^A$ and $\forall \mathcal{C} \in \mathcal{R} \setminus \mathcal{R}'$, $h(\mathcal{C}) = \mathcal{C}^B$.

3 PAC Learnability of Multilayer Systems

In this section, we establish the efficient PAC learnability of the hypothesis class $\mathcal{H}$, defined in Section 2. We first propose a learner that efficiently infers an unknown multilayer system. We then show that a training set of size $\lceil 1/\epsilon \cdot \sigma k \cdot \log(\sigma k/\delta) \rceil$ is sufficient for the learner to achieve the (ϵ, δ)-PAC guarantee. Lastly, we prove that our algorithm can also handle the more general PMAC learning setting [4], which permits learners to make approximate predictions. Due to space limits, **full proofs appear in the Appendix (Section 6.2)**. We present proofs for learning interaction functions under the `OR` master function. The results for `AND` master functions follow by duality (see Appendix, Section 6.2).

3.1 An Efficient PAC Learner

For a configuration $\mathcal{C}$, a vertex $v \in \mathcal{V}$ and a layer $i \in [k]$, let $\Gamma_i[\mathcal{C}, v]$ be the number of 1's in the input to the interaction function $f_{i,v}$ under $\mathcal{C}$ (i.e., the number of state-1 vertices in v's closed neighborhood in $\mathcal{G}_i$). We call $\Gamma_i[\mathcal{C}, v]$ the **score** of v in $\mathcal{G}_i$ under $\mathcal{C}$. Let $\tau_i^h(v)$ be the *learned threshold* of v for the ith layer in h, and let $\tau_i^{h^*}(v)$ be v's *true threshold* in the target system h^*.

PAC **Learner.** Our algorithm learns a hypothesis $h \in \mathcal{H}$ by inferring the unknown thresholds in the target system h^*. Let $\mathcal{T} \sim \mathcal{D}^q$ be a given training set. For each vertex $v \in \mathcal{V}'$ with unknown thresholds on each layer $\mathcal{G}_i \in \mathcal{M}$, under `OR` master functions, we assign

$$\tau_i^h(v) = \max_{(\mathcal{C},\mathcal{C}') \in \mathcal{T} : \mathcal{C}'(v)=0} \{\Gamma_i[\mathcal{C}, v]\} + 1. \tag{1}$$

If $\mathcal{C}'(v) = 1$ for all $(\mathcal{C}, \mathcal{C}') \in \mathcal{T}$, we set $\tau_i^h(v) = 0$.

If the master function is `AND`, then we assign $\tau_i^h(v) = \min_{(\mathcal{C},\mathcal{C}') \in \mathcal{T} : \mathcal{C}'(v)=1} \{\Gamma_i[\mathcal{C}, v]\}$. If $\mathcal{C}'(v) = 0$ for all $(\mathcal{C}, \mathcal{C}') \in \mathcal{T}$, we set $\tau_i^h(v) = \deg_i(v) + 2$, where $\deg_i(v)$ is the degree of v in $\mathcal{G}_i$.

Lastly, the algorithm returns the corresponding system $h \in \mathcal{H}$. One can easily verify that h has zero empirical risk. Further, the running time is $O(nk \cdot |\mathcal{T}|)$. Thus, the algorithm is an efficient consistent learner for $\mathcal{H}$. Since $\mathcal{H}$ is finite, it follows that the class $\mathcal{H}$ is efficiently PAC learnable [56].

Theorem 3.1. The class $\mathcal{H}$ is efficiently PAC learnable.

3.2 The Sample Complexity

We now show that our algorithm requires a small training set to PAC-learn $\mathcal{H}$. To begin with, a well-known general result in [30] implies that the sample complexity $m_{\mathcal{H}}(\delta, \epsilon)$ of learning $\mathcal{H}$ is upper bounded by $(1/\epsilon) \log(|\mathcal{H}|/\delta)$, where $\epsilon, \delta \in (0,1)$ are the two PAC parameters. In our case, one can derive that

$$m_{\mathcal{H}}(\delta, \epsilon) \leq \frac{1}{\epsilon} \cdot \Big(\sigma k \cdot \log(d_{\text{avg}}(\mathcal{V}')) + \log(\frac{1}{\delta}) \Big), \tag{2}$$

where $\sigma = |\mathcal{V}'|$, and $d_{\text{avg}}(\mathcal{V}')$ is the average degree of the vertices of $\mathcal{V}'$ in the network where layers are merged into a single layer with parallel edges removed.

A new bound. As one might expect, bound (2) depends explicitly on the average degree since a denser network leads to a larger hypothesis class, requiring a larger training set. Nevertheless, we now establish an alternative bound on the training set size $|\mathcal{T}|$ for our algorithm. Notably, this bound *does not depend explicitly on any graph parameters* (e.g., d_{avg}) except for the number of layers k.

The key lemma. For a configuration $\mathcal{C} \sim \mathcal{D}$, and a vertex v, let $B(\mathcal{C}, v)$ denote the event that "$h^*(\mathcal{C})(v) = 0$", i.e., in the true system h^*, $\mathcal{C}$ does not satisfy the threshold condition of v on any layer. For a layer $i \in [k]$ and a system $h \in \mathcal{H}$, let $A(\mathcal{C}, i, v, h)$ be the "bad" event that (1) "the threshold condition of v on the ith layer is satisfied under h", and (2) "$B(\mathcal{C}, v)$ occurs". Namely, h makes a wrong prediction for the state of v in $\mathcal{G}_i$. Formally, $A(\mathcal{C}, i, v, h)$ is defined as "$\Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)$ and $\forall j \in [k]$, $\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v)$".

We now bound the probability (over $\mathcal{T} \sim \mathcal{D}^q$) of our algorithm learning a "bad" $h \in \mathcal{H}$ s.t. $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \alpha$, for a given $\alpha \in (0,1)$.

Lemma 3.2. For a $v \in \mathcal{V}'$ and an $i \in [k]$, suppose $\tau_i^{h^*}(v) \geq 1$. Let $h \in \mathcal{H}$ be a hypothesis learned from a training set $\mathcal{T}$ of size $q \geq 1$. For a given $\alpha \in (0,1)$

(1) If all integers $\rho_i(v) \in [0, \tau_i^{h^*}(v))$ satisfy $\Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v)$ and $\Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] < \alpha$. Then $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] < \alpha$.

(2) If (1) does not hold, that is, there is a $\rho_i(v) \in [0, \tau_i^{h^*}(v))$ such that $\Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v)$ and $\Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] \geq \alpha$, then $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \alpha$ holds with probability **at most** $(1-\alpha)^q$ over $\mathcal{T} \sim \mathcal{D}^q$.

For an error rate $\alpha \in (0,1)$, Lemma 3.2 states that the probability (over $\mathcal{T} \sim \mathcal{D}^q$) of the algorithm learning a "bad" hypothesis h where $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \alpha$ is at most $(1-\alpha)^q$. Using this lemma, we now present the result on the size of an adequate training set for our algorithm.

Theorem 3.3. For any $\epsilon, \delta \in (0,1)$, with a training set of size $q = \lceil 1/\epsilon \cdot \sigma k \cdot \log(\sigma k/\delta) \rceil$, the proposed algorithm learns a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta$ (over $\mathcal{T} \sim \mathcal{D}^q$), $\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] < \epsilon$.

Proof sketch. We first prove that the learned hypothesis h makes a mistake on a vertex v if and only if $h(\mathcal{C})(v) = 1$ and $h^*(\mathcal{C})(v) = 0$, which corresponds to the event $A(\mathcal{C}, i, v, h)$ for some $i \in [k]$. Then, using Lemma 3.2, we show that with probability at most $k \cdot (1 - \epsilon/(\sigma k))^q$ over $\mathcal{T} \sim \mathcal{D}^q$, the loss satisfies $\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C})(v) \neq h^*(\mathcal{C})(v)] \geq \epsilon/\sigma$. It follows that with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $\sigma k \cdot (1 - \epsilon/(\sigma k))^q$, we have

$$\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] \geq \epsilon$$

Setting

$$q = \lceil \frac{1}{\epsilon} \cdot \sigma k \cdot \log(\frac{\sigma k}{\delta}) \rceil$$

one can verify that $\sigma k \cdot (1 - \epsilon/(\sigma k))^q \leq \delta$. Equivalently, with probability at least $1 - \delta$ over $\mathcal{T} \sim \mathcal{D}^q$, we have $\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] < \epsilon$. ■

Implication on the sample complexity. Theorem 3.3 provides an upper bound on the sample complexity $m_{\mathcal{H}}(\delta, \epsilon)$ of learning $\mathcal{H}$. Specifically, we have

$$m_{\mathcal{H}}(\delta, \epsilon) \leq \lceil \frac{1}{\epsilon} \cdot \sigma k \cdot \log(\frac{\sigma k}{\delta}) \rceil. \tag{3}$$

Remark. With the proposed learner, Theorem 3.3 shows that an adequate number of examples to PAC-learn $\mathcal{H}$ does *not* explicitly depend on the average degree or the size of the multilayer graph. Thus, when other parameters are fixed, the sample complexity of learning $\mathcal{H}$ does not grow as the graph increases in size or density (even though $\mathcal{H}$ itself grows exponentially), making the algorithm scalable to larger networks. Further, our bound in Theorem 3.3 is tighter than Ineq (2) in several regimes. For instance, for a fixed σ, the bound in Ineq (2) grows as $d_{avg}(\mathcal{V}')$ gets larger; on the other hand, our bound remains unchanged.

3.3 Extension to the PMAC Model

Our learner can operate in a more general Probably Mostly Approximately Correct (PMAC) framework [4]. In this setting, a learner aims to make predictions that are good *approximations* for the true values, allowing small errors in the predictions. The PMAC model is used in many contexts such as learning submodular functions [52, 4] and cascade inference [13].

Formulation. In PMAC model, the learned hypothesis h makes a successful prediction if $h(\mathcal{C})$ agrees with $h^*(\mathcal{C})$ on the states of more than $(1 - \beta)$ fraction of the vertices in $\mathcal{V}'$, for a given approximation factor $\beta \in (0,1)$. Formally, let $W(h(\mathcal{C}), h^*(\mathcal{C}))$ be the number of vertices in $\mathcal{V}'$ whose states in $h(\mathcal{C})$ are *different* from those in $h^*(\mathcal{C})$. For given $\epsilon, \delta, \beta \in (0,1)$, the goal is to learn a hypothesis $h \in \mathcal{H}$ such that with probability at least $1 - \delta$ over $\mathcal{T} \sim \mathcal{D}^q$, $\Pr_{\mathcal{C} \sim \mathcal{D}}[W(h(\mathcal{C}), h^*(\mathcal{C})) \geq \beta\sigma] \leq \epsilon$, where "$W(h(\mathcal{C}), h^*(\mathcal{C})) \geq \beta\sigma$" is the "bad" event that $h(\mathcal{C})$ does not approximate $h^*(\mathcal{C})$ to the desired factor.

We prove that under PMAC setting, the size of the training set for our algorithm (in Section 3.1) to learn an unknown system is significantly reduced.

> **Theorem 3.4.** For any given $\epsilon, \delta, \beta \in (0,1)$, with a training set $\mathcal{T}$ of size $q = \lceil 1/\epsilon \cdot 1/\beta \cdot k \cdot \log(\sigma k/\delta) \rceil$, the proposed algorithm (Section 3.1) learns an $h \in \mathcal{H}$ such that with probability at least $1 - \delta$ over $\mathcal{T} \sim \mathcal{D}^q$, h satisfies that $\Pr_{\mathcal{C} \sim \mathcal{D}}[W(h(\mathcal{C}), h^*(\mathcal{C})) \geq \beta\sigma] \ \leq \ \epsilon$.

Proof sketch. Let $A(\mathcal{C}, v, h)$ is the "bad" event where $h(\mathcal{C})(v) \neq h^*(\mathcal{C})(v)$ for a vertex $v \in \mathcal{V}'$ and $\mathcal{C} \sim \mathcal{D}$. Using Lemma (3.2), where we set $\alpha = (\epsilon\beta)/k$, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $k \cdot (1 - (\epsilon\beta)/k)^q$, we have $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq (\epsilon\beta)$ for any vertex $v \in \mathcal{V}'$. Thus, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $\sigma k \cdot (1 - (\epsilon\beta)/k)^q$, there exists a vertex $v \in \mathcal{V}'$ such that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \ \epsilon\beta$. By the linearity of expectation, we have

$$\mathbb{E}_{\mathcal{C} \sim \mathcal{D}}[W(h(\mathcal{C}), h^*(\mathcal{C}))] < \epsilon\beta \cdot \sigma$$

Then, using Markov Inequality, it follows that with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at least $1 - \sigma k \cdot (1 - (\epsilon\beta)/k)^q$, the learned $h \in \mathcal{H}$ satisfies $\Pr_{\mathcal{C} \sim \mathcal{D}}[W(h(\mathcal{C}), h^*(\mathcal{C})) \geq \beta\sigma] \leq \epsilon$. Lastly, Setting

$$q = \lceil 1/\epsilon \cdot 1/\beta \cdot k \cdot \log(\sigma k/\delta) \rceil$$

we have $1 - \sigma k \cdot (1 - (\epsilon\beta)/k)^q \ \geq \ 1 - \delta$. ■

Remark. Compared to the bound on the PAC sample complexity in Theorem 3.3, the size of an adequate training set under the PMAC model is reduced by a multiplicative factor of $\sigma\beta$. For instance, when β is a constant (i.e., to obtain a constant-factor approximation), the training set size is decreased by a linear (w.r.t. σ) factor. This demonstrates a trade-off between the quality of the prediction and the size of the training data: when our learner is allowed to make approximate predictions, it requires a much lower number of examples to learn an appropriate hypothesis.

4 Tight Analysis of Model Complexity

The Natarajan dimension (Ndim) measures the expressiveness of a hypothesis class and characterizes the complexity of learning [47]. The higher the value of Ndim, the greater the expressive power of the learning model. Further, Ndim informs us about the requisite sample size for learning good hypotheses. In this section, we examine Ndim of the hypothesis class $\mathcal{H}$ for multilayer systems, denoted by Ndim$(\mathcal{H})$, and develop the following results. **See Appendix (Section 6.3) for full proofs.**

1. We present an efficient method for constructing shatterable sets. Using this method, we establish that Ndim$(\mathcal{H})$ is *exactly* σ when the underlying network has only one layer. Previously, Adiga et al. [2] showed that for the single-layer case where $\sigma = n$, the VC dimension of $\mathcal{H}$ is at least $n/4$. Our precise characterization of Ndim$(\mathcal{H})$ provides both an improvement over their bound [2] and an extension to arbitrary σ.
2. We show that for multilayer networks, Ndim$(\mathcal{H})$ is bounded between σ and $k\sigma$. This proof uses an extended version of our technique for the single-layer case. Further, we present classes of instances where the bounds

are tight. Our results also show that the best possible lower bound of sample complexity that one can obtain using this approach is $\Omega((\sigma + \log(1/\delta))/\epsilon)$.

3. We further tighten our analysis by proving that asymptotically, for **almost all** graphs (i.e., almost surely) with n vertices and $k \geq 2$ layers, $\text{Ndim}(\mathcal{H})$ of the corresponding hypothesis class $\mathcal{H}$ is *exactly* σk.

4.1 An Exact Characterization for a Single Layer

Let h^* be the true system with a single-layer network. We present a combinatorial characterization of shatterable sets, which allows us to obtain an exact value for $\text{Ndim}(\mathcal{H})$. We begin with some definitions.

Definition 4.1 (Landmark Vertices). Suppose the underlying network has a single layer. Given a set $\mathcal{R} \subseteq \mathcal{X}$, a vertex $v \in \mathcal{V}'$ is a ***landmark*** vertex for a configuration $\mathcal{C} \in \mathcal{R}$ if the score $\Gamma[\mathcal{C}, v] \neq \Gamma[\hat{\mathcal{C}}, v]$ for all $\hat{\mathcal{C}} \in \mathcal{R} \setminus \{\mathcal{C}\}$.

The landmark vertices play a key role in $\mathcal{R}$ being shatterable. Given $\mathcal{R} \subseteq \mathcal{X}$, let $\mathcal{W}(\mathcal{R}) \subseteq \mathcal{V}'$ be the (possibly empty) set of vertices that are landmark vertices for at least one configuration in $\mathcal{R}$.

Definition 4.2 (Canonical Set). A set $\mathcal{R} \subseteq \mathcal{X}$ is ***canonical*** w.r.t. $\mathcal{H}$ if there is an injective mapping from $\mathcal{R}$ to $\mathcal{W}(\mathcal{R})$ s.t. each $\mathcal{C} \in \mathcal{R}$ maps to a landmark vertex of $\mathcal{C}$.

By the definition of shattering (see Section 2), each $\mathcal{C}$ in a shatterable set $\mathcal{R}$ is associated with two configurations, denoted by $\mathcal{C}^A$ and $\mathcal{C}^B$, where $\mathcal{C}^A \neq \mathcal{C}^B$.

Definition 4.3 (Contested Vertices). We call a vertex v **contested** for a $\mathcal{C} \in \mathcal{R}$ if $\mathcal{C}^A(v) \neq \mathcal{C}^B(v)$.

By linking landmark vertices to contested vertices, our next lemma shows that for a single-layer system, the property of a set being canonical is *equivalent* to being shatterable.

> **Lemma 4.4.** When the underlying network has a single layer, a set $\mathcal{R} \subseteq \mathcal{X}$ can be shattered by $\mathcal{H}$ **if and only if** $\mathcal{R}$ is canonical w.r.t. $\mathcal{H}$.

Proof sketch. ($\Rightarrow$) Suppose $\mathcal{H}$ shatters $\mathcal{R}$. We want to show that $\mathcal{R}$ is canonical. We first establish the following claims (1) *All contested vertices are in* $\mathcal{V}'$. (2) *No two configurations in* $\mathcal{R}$ *have a common contested vertex. Further, a* ***contested*** *vertex for a configuration* $\mathcal{C} \in \mathcal{R}$ *is also a* ***landmark*** *vertex for* $\mathcal{C}$. From the above claims, it follows that there exists an injective mapping from $\mathcal{R}$ to $\mathcal{W}(\mathcal{R})$; i.e., $\mathcal{R}$ is canonical. ($\Leftarrow$) Suppose that $\mathcal{R} \subseteq \mathcal{X}$ is canonical w.r.t $\mathcal{H}$. To show that $\mathcal{H}$ shatters $\mathcal{R}$, we present a method to construct the associated configurations, $\mathcal{C}^A$ and $\mathcal{C}^B$, of each $\mathcal{C} \in \mathcal{R}$ by specifying the states of vertices. We then show that the shattering conditions are satisfied under our selected $\mathcal{C}^A$ and $\mathcal{C}^B$. ■

Remark. By definition, the size of a canonical set is at most $|\mathcal{V}'| = \sigma$. From Lemma 4.4, it follows that $\text{Ndim}(\mathcal{H})$, which is the maximum size of a shatterable set, is **at most** σ when the underlying network has a single layer.

Next, we present an efficient method in Theorem 4.5 for constructing a canonical set of size σ based on depth-first search (see proof in the Appendix). Consequently, for *any* underlying single-layer network, **there exists a shatterable set of size exactly** σ. Formally:

Theorem 4.5. When the underlying network has a single layer, a shatterable set of size σ can be (efficiently) constructed. Thus, $\text{Ndim}(\mathcal{H}) = \sigma$.

Proof sketch. To construct a canonical set $\mathcal{R} \subset \mathcal{X}$ with σ configurations, $\mathcal{R}$ is initially empty. Let $\mathcal{G}' = \mathcal{G}[\mathcal{V}']$ be the subgraph induced on $\mathcal{V}'$. Starting from any vertex $v_1 \in \mathcal{V}'$, the algorithm carries out a *depth-first* traversal on $\mathcal{G}'$, while maintaining a stack $\mathcal{K}$ of vertices. Let v_i, $i \in [\sigma]$, denote the ith discovered vertex, $1 \leq i \leq \sigma$. When v_i is discovered, the configuration $\mathcal{C}_{v_i}$ added to $\mathcal{R}$ is constructed as follows: $\mathcal{C}_{v_i}(v) = 1$ if $v \in \mathcal{K}$ and $\mathcal{C}_{v_i}(v) = 0$ otherwise. The algorithm terminates when all vertices in $\mathcal{V}'$ are visited. Clearly $|\mathcal{R}| = \sigma$. We then show that v_i is a landmark vertex of $\mathcal{C}_{v_i}$, thereby establishing that $\mathcal{R}$ is canonical. By Lemma 4.4, $\mathcal{R}$ is also shatterable. ■

Remark. Theorem 4.5 is interesting since for many problems [62, 5], including those on learning networked systems [2], known results on such dimensions provide only bounds rather than exact values. The graph-theoretic machinery we have introduced (e.g., canonical set) to aid our analysis may be of independent interest in studying other learning problems for dynamical systems.

4.2 Bounds on Ndim for Multilayer Systems

Using the results developed in the previous section, we now derive bounds on $\text{Ndim}(\mathcal{H})$ for systems with $k \geq 2$ layers. We first prove that $\text{Ndim}(\mathcal{H}) \leq k\sigma$ by showing that each $v \in \mathcal{V}'$ is *contested* for at most k configurations in any shatterable set. We then show that $\text{Ndim}(\mathcal{H}) \geq \sigma$ by establishing that any shatterable set obtained by restricting the system to any single layer in $\mathcal{M}$ is also shatterable over the multilayer network $\mathcal{M}$. We first state the upper bound:

Lemma 4.6. If the network has $k \geq 2$ layers, then the size of any shatterable set $\mathcal{R}$ is at most $k\sigma$.

To establish a lower bound of σ on the size of a shatterable set, we prove the following lemma.

Lemma 4.7. Suppose h^* is an MSyDS whose underlying network has $k \geq 2$ layers. Let $\hat{h}^*$ be a single-layer system obtained from h^* by using the network in any layer $i \in [k]$. If a set $\mathcal{R}$ is shatterable by the hypothesis class of $\hat{h}^*$, then it is also shatterable by the hypothesis class of h^*.

By Theorem 4.5, when the underlying network has a single layer, there exists a shatterable set of size σ. Thus, Lemma 4.7 implies that there also exists a shatterable set of size σ for the multilayer setting. Overall, we obtain the following bounds on $\text{Ndim}(\mathcal{H})$:

Theorem 4.8. Suppose the underlying network has $k \geq 2$ layers. Then $\sigma \leq \text{Ndim}(\mathcal{H}) \leq k\sigma$.

Remark. There are classes of multilayer systems where the bounds are tight. To match the lower bound, consider the class of k-layer networks $\mathcal{M} = \{\mathcal{G}_1, ..., \mathcal{G}_k\}$ where for each vertex v, the following condition holds: *the vertex v's neighbors in $\mathcal{G}_{i+1}$ form a superset of its neighbors in $\mathcal{G}_i$, with exactly one extra neighbor in $\mathcal{G}_{i+1}$*, $i = 1, ..., k-1$. It can be verified that in such a system, given any shatterable set R, each vertex with unknown interaction functions can be contested for at most one configuration in this set. Therefore, $|R|$ is at most σ and Ndim $= \sigma$. On the other hand, in Section 4.3, we show that the upper bound $k\sigma$ is tight for *almost all* threshold multilayer systems.

Bounds on the sample complexity. By a result in [56], our bounds on Ndim$(\mathcal{H})$ in Theorem 4.8 also imply the following bounds on the sample complexity $m_{\mathcal{H}}(\delta, \epsilon)$: (i) Lower bound: $c_1 \frac{\sigma + \log(1/\delta)}{\epsilon}$; (2) Upper bound: $\frac{1}{\epsilon} \cdot \left(c_2 \cdot k\sigma \cdot \log\left(\frac{k\sigma}{\epsilon}\right) + k\sigma^2 + \log\left(\frac{1}{\delta}\right)\right)$ for some constants $c_1, c_2 \geq 0$.

Remark. The above upper bound is weaker than our bound in Theorem 3.3. Notably, the above bound has a dominant term $O(k\sigma^2)$, while our bound in Theorem 3.3 is $O(k\sigma \log(k\sigma))$. Further, when k is a constant (a realistic scenario in real-world networks by the Dunbar's number [24]), the upper bound in Theorem 3.3 is within a factor $O(\log(\sigma))$ of the lower bound stated above.

4.3 The Asymptotic Behavior of Ndim

We have shown that Ndim$(\mathcal{H}) \leq k\sigma$ for any k-layer system. In this section, we further explore the complexity of the hypothesis class and prove that asymptotically (i.e., as $n \to \infty$), for *almost all* graphs with n vertices and $k \geq 2$ layers, Ndim$(\mathcal{H})$ of the corresponding hypothesis class is *exactly* $k\sigma$, which matches our upper bound (Theorem 4.8).

Approach overview. Given a multilayer network $\mathcal{M}$, we first define a special set $\mathcal{Q}$ of vertex-layer pairs in $\mathcal{M}$. We then show that for each such set $\mathcal{Q}$, there is a shatterable set of size $|\mathcal{Q}|$. Next, using a probabilistic argument, we prove that in the space of all k-layer graphs with n vertices, the proportion of graphs that admit such sets $\mathcal{Q}$ of size $k\sigma$ asymptotically (w.r.t n) tends to 1. Next, consider a graph $\mathcal{M}$ chosen uniformly at random from the space of all k-layer graphs with n vertices. We prove that, asymptotically with probability 1 (i.e., almost surely), $\mathcal{M}$ admits such a special set $\mathcal{Q}$ that contains *all the $k\sigma$ vertex-layer pairs*, thus implying the existence of a shatterable set of size $k\sigma$.

A special vertex-layer set. For a k-layer network $\mathcal{M}$ and a subset $\mathcal{V}'$ of vertices, let $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ be a set of vertex-layer pairs $(v, i), v \in \mathcal{V}', i \in [k]$, such that every $(v, i) \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ satisfies the following condition: $N_{\mathcal{M}}[v, i] \setminus N_{\mathcal{M}}[v', i'] \neq \varnothing$ for all pairs $(v', i') \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}, (v', i') \neq (v, i)$, where $N_{\mathcal{M}}[v, i]$ is the closed neighborhood of v in the ith layer of $\mathcal{M}$. We first establish the correspondence between such a set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ and a shatterable set.

> **Lemma 4.9.** Given a multilayer network $\mathcal{M}$ and a subset $\mathcal{V}'$ of vertices, for each set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$, there is a shatterable set of size $|\mathcal{Q}_{\mathcal{M},\mathcal{V}'}|$ for the corresponding hypothesis class over $\mathcal{M}$, where thresholds of vertices in $\mathcal{V}'$ are unknown.

Our next lemma shows that, asymptotically, almost all graphs $\mathcal{M}$ with n vertices and k layers have a set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ of size $k\sigma$, where $\mathcal{V}'$ is a subset of vertices and $\sigma = |\mathcal{V}'|$.

> **Lemma 4.10.** Given $n \geq 1$, $k \geq 2$, and $\mathcal{V}' \subseteq [n]$, in the space of all k-layer graphs with n vertices, the proportion of graphs that admits a set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ of size $k\sigma$ is at least $1 - 4 \cdot (\sigma k)^2 \cdot (\frac{3}{4})^n$.

Proof sketch. Let $\mathcal{G}_{n,k,1/2}$ be the space of k-layer graphs with n vertices. Let $\mathcal{M} \sim \mathcal{G}_{n,k,1/2}$ be a graph chosen uniformly at random. We first show that the probability of any pair (v, i) violating the condition of $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ is at most $4 \cdot (\frac{3}{4})^n$. It follows that w.p. at most $8 \cdot \binom{\sigma k}{2} \cdot (\frac{3}{4})^n \leq 4 \cdot (\sigma k)^2 \cdot (\frac{3}{4})^n$, there exists such an undesirable pair. Consequently, w.p. at least $1 - 4 \cdot (\sigma k)^2 \cdot (\frac{3}{4})^n$, $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ contains all the $k\sigma$ pairs. We remark that such probability can be interpreted as the proportion of graphs in the space of all k-layer graphs with n vertices. This concludes the proof. ■

Collectively, by Lemmas 4.9 and 4.10, we conclude that for any $k \geq 2$, asymptotically in n, almost all the hypothesis classes of threshold dynamical systems over k-layer graphs have Ndim exactly $k\sigma$.

> **Theorem 4.11.** Given $k \geq 2$, as n approaches infinity, almost all the hypothesis classes of threshold dynamical systems over k-layer graphs have Ndim exactly σk.

Remark. The term *almost all* is standard in probabilistic methods (e.g., see [3, 25]). Formally, a property holds for almost all graphs if asymptotically (w.r.t n) with probability one, a random graph (drawn the space of all n-vertex graphs) possesses that property. In our case, such probability is $1 - 4 \cdot (\sigma k)^2 \cdot (\frac{3}{4})^n \sim 1 - o(1)$, which approaches 1 quickly due to the exponent of n. We also empirically found that for this proportion to be close to 1, n only needs to be around $1,000$ (see **Appendix, Section 6.4**).

5 Experimental Analysis

We present experimental studies on the relationships between model parameters and the empirical performance of our PAC algorithm. Here, we study the performance of the algorithm on different networks [44, 49, 59, 12], shown in Table 1. In particular, The objective of our experiments is two-fold: (i) We examine the effect of different model parameters on the empirical loss, as these parameters appear in the derived sample complexity bound; (ii) We empirically verify that the loss decreases as more samples are provided.

Dataset	Type	k	n	m	Avg. deg.
`Aarhus`	Social	5	61	620	20.33
`CKM-Phy`	Social	3	246	$1,551$	12.61
`Multi-Gnp`	Random	2	500	$7,495$	15
`PPI`	Biology	7	900	$12,870$	28.6
`Twitter`	Social	2	2000	$10,233$	10.23

Table 1: *List of multilayer networks.* Parameters k, n, and m are the number of layers, the number of vertices, and the total number of edges in a network, respectively.

Training and testing. For each network, we have a target system h^* where the threshold of each vertex $v \in \mathcal{V}$ on each layer i is in $[0, \deg_i(v) + 2]$. For each such h^*, a training set $\mathcal{T} = \{(\mathcal{C}_i, h^*(\mathcal{C}_i))\}_{i=1}^q$ is constructed, where each $\mathcal{C}_i$ is sampled from a distribution $\mathcal{D}$. We consider *distributions* where the state of each vertex in $\mathcal{C}_i \in \mathcal{T}$ is 0 w.p. p and 1 w.p. $1-p$, for a $p \in \{0.1, 0.5, 0.9\}$. Intuitively, a higher p implies a distribution that is more biased towards vertices in state 0. Our PAC algorithm then uses $\mathcal{T}$ to learn a hypothesis $h \in \mathcal{H}$ where all the thresholds are inferred. To evaluate the solution quality, we sample $10,000$ configurations from $\mathcal{D}$, and compute the *empirical loss* ℓ, which is the fraction of sampled configurations $\mathcal{C}$'s where $h(\mathcal{C}) \neq h^*(\mathcal{C})$.

In presenting the results, each data point is the average over 50 learned hypotheses.

5.1 Experimental Results

Impacts of model parameters. We first examine the relationships between the loss ℓ and the training set size $|\mathcal{T}|$, over three distributions specified by different values of p. Experimental results using the `Multi-Gnp` network (Table 1) are in Fig 2(a), where *the interaction functions of all vertices must be learned* (i.e., $\sigma = n$).

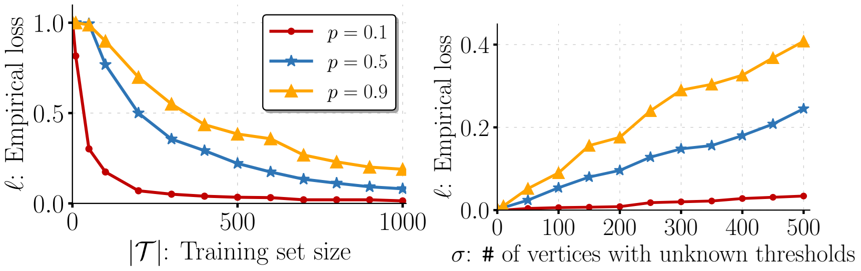


Figure 2: (a): ℓ vs $|\mathcal{T}|$ and (b): ℓ vs σ, over different distributions specified by p. The underlying network is `Multi-Gnp` (Table 1). The stdev for all data points is less than 0.09.

Observations. In Figure 2(a), the loss ℓ decreases as $|\mathcal{T}|$ increases. Such a relationship is expected since a larger sample usually provides more information about the underlying system. Further, for each value of $|\mathcal{T}|$, the loss increases as the distribution of samples becomes skewed towards vertices having state 0. One reason for this behavior is that when states in $\mathcal{C} \sim \mathcal{D}$ consist mostly of 0's, the scores of vertices under $\mathcal{C}$ could be far from their true thresholds. Since our algorithm learns based on the scores, it will require more examples to infer an appropriate system.

Next, we study the relationship between ℓ and σ, under a fixed $|\mathcal{T}| = 500$ over different distributions. The results for the `Multi-Gnp` network are shown in Fig 2(b). Specifically, observe that ℓ increases with σ. This is because a larger σ leads to an (exponentially) larger hypothesis space. Since the amount of training data (i.e., $|\mathcal{T}|$) is fixed, a learned hypothesis would incur a higher loss when σ is larger. Nevertheless, even though $|\mathcal{H}|$ is exponential in σ, the loss ℓ of our algorithm grows much more slowly.

Impact of the number of layers. We study the effect of the graph structure on ℓ. We first examine the relationship between ℓ and $|\mathcal{T}|$ using real-world multilayer networks where *the thresholds of all vertices are to be learned*, and $\mathcal{D}$ is the uniform distribution. The results are in Fig 3(a).

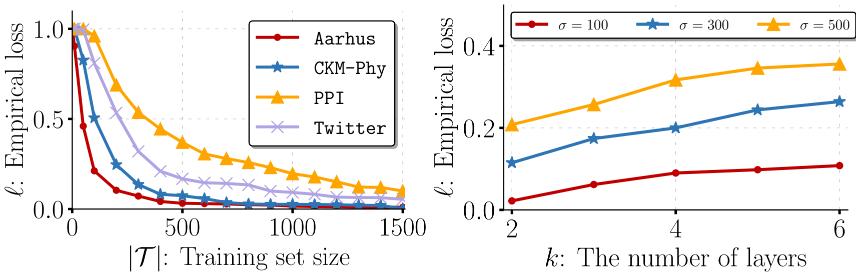


Figure 3: (a): ℓ vs $|\mathcal{T}|$, over different real-world networks (Table 1), and (b): ℓ vs k over different values of σ, where the underlying network is `Gnp`. The stdev for all data points is less than 0.08.

Observations. From Fig 3(a), we observe a joint effect of σ and k on the loss ℓ. In particular, if the network has more vertices (thus a larger $\sigma = n$), the learned hypothesis h usually has a higher loss ℓ, as one would expect. Further, even though the `Twitter` network has more vertices than the `PPI` network, the latter network has more layers. Since the size of the hypothesis class is exponential w.r.t k, for the same $|\mathcal{T}|$, observe that the h under the PPI network incurs a higher loss. Next, we study the effect of k on the loss ℓ using multilayer Gnp networks of size 500 and average degree (on each layer) of 10. We increase the number of layers from 2 to 6 while fixing $|\mathcal{T}|$ at 500. The result is shown in Fig 3(b) for three values of σ. Overall, we observe a positive correlation between k and ℓ; this is because a larger k leads to a larger hypothesis space.

6 Future Work

One direction for future work is to improve our lower bound on the sample complexity for PAC learnability. The second direction is to tighten the gap between the lower and upper bounds on the Natarajan dimension for multilayer systems using other techniques. Another promising direction is to consider a noisy setting where labels in the training set (i.e., the successor configurations) may be incorrect with a small probability. Lastly, we note that real-world networks are closer to graphs with special structures, such as multilayer *scale-free* or *small-world* networks. The multilayer expander graphs and graphs with fixed spectral dimensions are also very interesting due to their theoretical significance. Therefore, we believe that it is important to understand the asymptotic properties of Ndim on such graphs. There are existing works on the asymptotic properties of some other mathematical structures (e.g., cliques [15], number of triangles [8]) on scale-free graphs and small-world graphs [8]. However, we note that the problems studied in these references are very different from our learning problem. Moreover, the networks considered in these references have only a single layer. We believe that obtaining results for Ndim on special classes of multilayer graphs such as multilayer scale-free, small-world, and multilayer expanders poses challenges that are likely to require the development of new theoretical tools.

References

[1] B. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi. Trace complexity of network inference. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 491–499. ACM, 2013.

[2] Abhijin Adiga, Chris J Kuhlman, Madhav Marathe, S Ravi, and Anil Vullikanti. PAC learnability of node functions in networked dynamical systems. In *International Conference on Machine Learning*, pages 82–91. PMLR, 2019.

[3] Noga Alon and Joel H Spencer. *The probabilistic method.* John Wiley & Sons, 2016.

[4] Maria-Florina Balcan and Nicholas JA Harvey. Learning submodular functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 793–802, 2011.

[5] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research*, 20(1):2285–2301, 2019.

[6] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics. *Physics Reports*, 874:1–92, 2020.

[7] Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics reports*, 544(1):1–122, 2014.

[8] Béla Bollobás and Oliver M Riordan. Mathematical results on scale-free random graphs. *Handbook of graphs and networks: from the genome to the internet*, pages 1–34, 2003.

[9] Charles D Brummitt, Kyu-Min Lee, and K-I Goh. Multiplexity-facilitated cascades in networks. *Physical Review E*, 85(4):045102, 2012.

[10] Wei Chen, Xiaoming Sun, Jialin Zhang, and Zhijie Zhang. Network inference and influence maximization from samples. In *International Conference on Machine Learning*, pages 1707–1716. PMLR, 2021.

[11] Yanxi Chen and H. Vincent Poor. Learning mixtures of linear dynamical systems. *International Conference on Machine Learning*, 162:3507–3557, 2022.

[12] James Coleman, Elihu Katz, and Herbert Menzel. The diffusion of an innovation among physicians. *Sociometry*, 20(4):253–270, 1957.

[13] Vincent Conitzer, Debmalya Panigrahi, and Hanrui Zhang. Learning opinions in social networks. In *International Conference on Machine Learning*, pages 2122–2132. PMLR, 2020.

[14] Vincent Conitzer, Debmalya Panigrahi, and Hanrui Zhang. Learning influence adoption in heterogeneous networks. In *Proc. AAAI*, pages 6411–6419. AAAI, 2022.

[15] Fraser Daly, Alastair Haig, and Seva Shneer. Asymptotics for cliques in scale-free random graphs. *arXiv preprint arXiv:2008.11557*, 2020.

[16] Hadi Daneshmand, Manuel Gomez-Rodriguez, Le Song, and Bernhard Schoelkopf. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *International conference on machine learning*, pages 793–801. PMLR, 2014.

[17] Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. Multiclass learnability and the erm principle. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 207–232. JMLR Workshop and Conference Proceedings, 2011.

[18] Quinlan E Dawkins, Tianxi Li, and Haifeng Xu. Diffusion source identification on networks with statistical confidence. In *International Conference on Machine Learning*, pages 2500–2509. PMLR, 2021.

[19] Manlio De Domenico, Clara Granell, Mason A Porter, and Alex Arenas. The physics of spreading processes in multilayer networks. *Nature Physics*, 12(10):901–906, 2016.

[20] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[21] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. *Proceedings of the national academy of Sciences*, 113(3):554–559, 2016.

[22] Nan Du, Yingyu Liang, Maria Balcan, and Le Song. Influence function learning in information diffusion networks. In *International Conference on Machine Learning*, pages 2016–2024. PMLR, 2014.

[23] Nan Du, Le Song, Ming Yuan, and Alex Smola. Learning networks of heterogeneous influence. *Advances in neural information processing systems*, 25, 2012.

[24] Robin IM Dunbar. Coevolution of neocortical size, group size and language in humans. *Behavioral and brain sciences*, 16(4):681–694, 1993.

[25] Paul Erdős and Robin J Wilson. On the chromatic index of almost all graphs. *Journal of combinatorial theory, series B*, 23(2-3):255–257, 1977.

[26] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1019–1028. ACM, 2010.

[27] S. González-Bailón, J. Borge-Holthoefer, A. Rivero, and Y. Moreno. The dynamics of protest recruitment through an online network. *Scientific Reports*, 1:7 pages, 2011.

[28] Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6):1420–1443, 1978.

[29] Zaynab Hammoud and Frank Kramer. Multilayer networks: aspects, implementations, and application in biomedicine. *Big Data Analytics*, 5(1):1–18, 2020.

[30] David Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial intelligence*, 36(2):177–221, 1988.

[31] Shushan He, Hongyuan Zha, and Xiaojing Ye. Network diffusions via neural mean-field dynamics. *Advances in Neural Information Processing Systems*, 33:2171–2183, 2020.

[32] Xinran He, Ke Xu, David Kempe, and Yan Liu. Learning influence functions from incomplete observations. *Advances in Neural Information Processing Systems*, 29, 2016.

[33] Lisa Hellerstein and Rocco A Servedio. On PAC learning algorithms for rich boolean function classes. *Theoretical Computer Science*, 384(1):66–76, 2007.

[34] Hao Huang, Qian Yan, Lu Chen, Yunjun Gao, and Christian S Jensen. Statistical inference of diffusion networks. *IEEE Transactions on Knowledge and Data Engineering*, 33(2):742–753, 2019.

[35] Zhiwei Ji, Ke Yan, Wenyang Li, Haigen Hu, and Xiaoliang Zhu. Mathematical and computational modeling in complex biological systems. *BioMed research international*, 2017, 2017.

[36] Dimitris Kalimeris, Yaron Singer, Karthik Subbian, and Udi Weinsberg. Learning diffusion using hyperparameters. In *International Conference on Machine Learning*, pages 2420–2428. PMLR, 2018.

[37] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. Random Boolean network models and the yeast transcriptional network. *Proc. National Academy of Sciences (PNAS)*, 100(25):14796–14799, Dec. 2003.

[38] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[39] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theoretical Biology*, 229:523–537, 2004.

[40] Kyu-Min Lee, Charles D Brummitt, and K-I Goh. Threshold cascades with response heterogeneity in multiplex networks. *Physical Review E*, 90(6):062816, 2014.

[41] Shuai Li, Fang Kong, Kejie Tang, Qizhi Li, and Wei Chen. Online influence maximization under linear threshold model. *Advances in Neural Information Processing Systems*, 33:1192–1204, 2020.

[42] Andrey Lokhov. Reconstructing parameters of spreading models from partial observations. *Advances in Neural Information Processing Systems*, 29, 2016.

[43] Kristian Lum, Samarth Swarup, Stephen Eubank, and James Hawdon. The contagious nature of imprisonment: an agent-based model to explain racial disparities in incarceration rates. *Journal of The Royal Society Interface*, 11(98):2014.0409, 2014.

[44] Matteo Magnani, Barbora Micenkova, and Luca Rossi. Combinatorial analysis of multiple networks. *arXiv preprint arXiv:1303.4986*, 2013.

[45] Seth Myers and Jure Leskovec. On the convexity of latent social network inference. *Advances in neural information processing systems*, 23, 2010.

[46] H. Narasimhan, D. C. Parkes, and Y. Singer. Learnability of influence in networks. In *Advances in Neural Information Processing Systems*, pages 3186–3194, 2015.

[47] Balas K Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

[48] Mark Newman. *Networks*. Oxford university press, 2018.

[49] Elisa Omodei, Manlio De Domenico, and Alex Arenas. Characterizing interactions in online social networks during exceptional events. *Frontiers in Physics*, 3:59, 2015.

[50] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[51] Jean Pouget-Abadie and Thibaut Horel. Inferring graphs from cascades: A sparse recovery framework. In *International Conference on Machine Learning*, pages 977–986. PMLR, 2015.

[52] Nir Rosenfeld, Eric Balkanski, Amir Globerson, and Yaron Singer. Learning to optimize combinatorial functions. In *International Conference on Machine Learning*, pages 4374–4383. PMLR, 2018.

[53] Daniel J Rosenkrantz, Abhijin Adiga, Madhav Marathe, Zirou Qiu, SS Ravi, Richard Stearns, and Anil Vullikanti. Efficiently learning the topology and behavior of a networked dynamical system via active queries. In *International Conference on Machine Learning*, pages 18796–18808. PMLR, 2022.

[54] Mostafa Salehi, Rajesh Sharma, Moreno Marzolla, Matteo Magnani, Payam Siyari, and Danilo Montesi. Spreading processes in multilayer networks. *IEEE Transactions on Network Science and Engineering*, 2(2):65–83, 2015.

[55] Thomas C Schelling. *Micromotives and macrobehavior*. WW Norton & Company, 2006.

[56] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, 2014.

[57] Michael W Sneddon, James R Faeder, and Thierry Emonet. Efficient modeling, simulation and coarse-graining of biological complexity with nfsim. *Nature methods*, 8(2):177–183, 2011.

[58] S. Soundarajan and J. E. Hopcroft. Recovering social networks from contagion information. In *Proceedings of the 7th Annual Conference on Theory and Models of Computation*, pages 419–430. Springer, 2010.

[59] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006.

[60] Daniel Trpevski, Wallace KS Tang, and Ljupco Kocarev. Model for rumor spreading over networks. *Physical Review E*, 81(5):056102, 2010.

[61] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[62] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the VC-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[63] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[64] Duncan J Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

[65] Mateusz Wilinski and Andrey Lokhov. Prediction-centric learning of independent cascade dynamics from partial observations. In *International Conference on Machine Learning*, pages 11182–11192. PMLR, 2021.

Appendix

6.1 The Settings of Existing Works

Our problem setting follows the line of existing research on learning networked systems. Here, we present the settings of a few illustrative papers on learning networked systems that span multiple authors and domains. These references are also cited in the main paper.

Vertex States	Update Scheme	Time Scale	Interaction Function	Venue
Binary	Synchronous	Discrete	Deterministic	AAAI-2022 [14]
Binary	Synchronous	Discrete	Threshold	ICML-2022 [53]
Binary	Synchronous	Discrete	Threshold	ICML-2021 [10]
Binary	Synchronous	Discrete	Susceptible-Infected	ICML-2021 [18]
Binary	Synchronous	Discrete	Independent Cascade	ICML-2021 [65]
Binary	Synchronous	Continuous	Probablistic	NeurIPS-2020 [31]
Binary	Synchronous	Discrete	Threshold	NeurIPS-2020 [41]
Binary	Synchronous	Discrete	Deterministic	ICML-2020 [13]
Binary	Synchronous	Discrete	Threshold	ICML-2019 [2]
Binary	Synchronous	Discrete	Threshold & Independent Cascade	NeurIPS-2016 [32]
Binary	Synchronous	Discrete	Threshold & Independent Cascade	NeurIPS-2015 [46]

Table 2: The problem settings of a few illustrative papers on learning networked systems.

6.2 Additional Material for Section 3

Duality between OR and AND master functions with respect to learning

The AND and OR master functions can be treated similarly in our context. For the OR master function, the state of a vertex v is 1 if the interaction function in at least one layer outputs a 1. Similarly, for the AND master function, the state of a vertex v is 0 if the interaction function on at least one layer outputs a 0. Due to this duality, all our results for OR master functions carry over to AND master functions.

Proofs in Section 3.2

Recall that $\tau_i^h(v)$ and $\tau_i^{h^*}(v)$ are the thresholds of v on the ith layer in a learned system (hypothesis) h and in the true system h^*, respectively. Fix a vertex v and layer $i \in [k]$. For a configuration $\mathcal{C} \sim \mathcal{D}$, let $B(\mathcal{C}, v)$ denote the event "the threshold condition for v is **not** satisfied in any of the layers under $\mathcal{C}$ in the true system h^*". For an $h \in \mathcal{H}$ and a configuration $\mathcal{C}$, let $A(\mathcal{C}, i, v, h)$ be the event such that (1) the threshold condition of v on the ith layer is satisfied under $\mathcal{C}$ in h, and (2) the event $B(\mathcal{C}, v)$ occurs. Formally, $A(\mathcal{C}, i, v, h)$ is the event "$\Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)$ for the given ith layer and $\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v),\ \forall j \in [k]$".

Lemma 3.2. *For a $v \in \mathcal{V}'$ and an $i \in [k]$, suppose $\tau_i^{h^*}(v) \geq 1$. Let $h \in \mathcal{H}$ be a hypothesis learned from a training set $\mathcal{T}$ of size $q \geq 1$. For a given $\alpha \in (0,1)$,*

(1) If all integer $\rho_i(v) \in [0, \tau_i^{h^*}(v))$ satisfy:

$$\Pr_{\mathcal{C}\sim\mathcal{D}}[\underbrace{\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v),\ \forall j \in [k]}_{\textit{Event } B(\mathcal{C}, v)} \textit{ and } \Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] < \alpha \tag{4}$$

then $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] < \alpha$.

(2) If (1) does not hold, that is, there is a $\rho_i(v)$ such that

$$\Pr_{\mathcal{C}\sim\mathcal{D}}[\underbrace{\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v),\ \forall j \in [k]}_{\textit{Event } B(\mathcal{C}, v)} \textit{ and } \Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] \geq \alpha \tag{5}$$

*then the condition $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \alpha$ holds with probability **at most** $(1-\alpha)^q$ over $\mathcal{T} \sim \mathcal{D}^q$.*

Proof. We first consider the case where $\Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v)$ and $\Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] < \alpha$ for all integer $\rho_i(v) \in [0, \tau_i^{h^*}(v))$. This case implies that

$$\Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq 0] < \alpha \tag{6}$$

Let h be the learned hypothesis by our algorithm. We now argue that $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] < \alpha$. In particular, note that the learned threshold $\tau_i^h(v)$ is always in the range $[0, \tau_i^{h^*}(v)]$. If $\tau_i^h(v) = \tau_i^{h^*}(v)$, the event $A(\mathcal{C}, i, v, h)$ does not occur. On the other hand, if $\tau_i^h(v) < \tau_i^{h^*}(v)$, then the event $\Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)$ is contained in the event $\Gamma_i[\mathcal{C}, v] \geq 0$; thus,

$$\begin{aligned} \Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] &= \Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)] && (7) \\ &\leq \Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq 0] && (8) \\ &< \alpha && (9) \end{aligned}$$

Now consider the second case as stated in Ineq (5). Let $\rho_i(v)$ be the **maximal** integer in $[0, \tau_i^{h^*}(v))$ such that $\Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v)$ and $\Gamma_i[\mathcal{C}, v] \geq \rho_i(v)] \geq \alpha$. We now establish the claim:

Claim 6.0.1. If $\exists\ (\mathcal{C}, \mathcal{C}') \in \mathcal{T}$ s.t. $B(\mathcal{C}, v)$ occurs and $\Gamma_i[\mathcal{C}, v] \geq \rho_i(v)$, then the algorithm learns an $h \in \mathcal{H}$ s.t. $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] < \alpha$.

Suppose such a pair $(\mathcal{C}, \mathcal{C}')$ exists in $\mathcal{T}$. Let h be the hypothesis returned by our algorithm using $\mathcal{T}$. Note that since $\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v),\ \forall j \in [k]$ (i.e., the event $B(\mathcal{C}, v)$), we must have $\mathcal{C}'(v) = 0$. By the definition of the PAC algorithm in the main manuscript, it follows that the learned threshold satisfies:

$$\tau_i^h(v) \geq \rho_i(v) + 1 \tag{10}$$

Recall that $A(\mathcal{C}, i, v, h)$ is the event "$\Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)$ and event $B(\mathcal{C}, v)$ occurs". If $\rho_i(v) = \tau_i^{h^*}(v) - 1$, then we learned the true threshold (i.e., $\tau_i^h(v) = \tau_i^{h^*}(v)$), and thus the event $A(\mathcal{C}, i, v, h)$ does not occur. On the other hand, if $\rho_i(v) < \tau_i^{h^*}(v) - 1$, then

$$\begin{aligned}
\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] &= \Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)] && (11)\\
&\leq \Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq \rho_i(v) + 1] && (12)\\
&< \alpha && (13)
\end{aligned}$$

where the last inequality follows from the maximality of $\rho_i(v)$. This establishes the Claim. To complete the argument for the second case, let $\eta = \Pr_{\mathcal{C}\sim\mathcal{D}}[B(\mathcal{C}, v) \text{ and } \Gamma_i[\mathcal{C}, v] \geq \rho_i(v)]$. The probability (over $\mathcal{T} \sim \mathcal{D}^q$) that no such configuration $\mathcal{C}$ exists in $\mathcal{T}$ is $(1-\eta)^q \leq (1-\alpha)^q$, where the inequality follows from Eq (5). Therefore, with probability at most $(1-\alpha)^q$, it holds that $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \alpha$ for the learned hypothesis h. This concludes the proof. ■

Theorem 3.3. *For any $\epsilon, \delta \in (0, 1)$, with a training set of size $q = \lceil 1/\epsilon \cdot \sigma k \cdot \log(\sigma k/\delta) \rceil$, the proposed algorithm learns a hypothesis $h \in \mathcal{H}$ such that with probabilty at least $1 - \delta$ (over $\mathcal{T} \sim \mathcal{D}^q$),*

$$\Pr_{\mathcal{C}\sim\mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] < \epsilon \tag{14}$$

Proof. For a vertex $v \in \mathcal{V}'$, an $h \in \mathcal{H}$ learned by the proposed algorithm, and a configuration $\mathcal{C} \sim \mathcal{D}$, recall that $h(\mathcal{C})$ denotes the successor of $\mathcal{C}$ under the system (hypothesis) h; $h(\mathcal{C})(v)$ is the state of v in $h(\mathcal{C})$. Let $A(\mathcal{C}, v, h)$ be the bad event where $h(\mathcal{C})(v) \neq h^*(\mathcal{C})(v)$, that is, the next state of v predicted by h is wrong.

For any layer $i \in [k]$, by the mechanisms of the PAC algorithm in the main manuscript:

$$\tau_i^h(v) = \max_{(\mathcal{C}, \mathcal{C}') \in \mathcal{T} : \mathcal{C}'(v) = 0} \{\Gamma_i[\mathcal{C}, v]\} + 1. \tag{15}$$

We remark that the learned threshold $\tau_i^h(v)$ is *at most* the value of the true threshold $\tau_i^{h^*}(v)$.

We first establish the claim:

Claim 6.0.2. The event $A(\mathcal{C}, v, h)$ occurs if and only if $h(\mathcal{C})(v) = 1$ and $h^*(\mathcal{C})(v) = 0$.

The necessity is trivially true. To prove sufficiency, we show that the case where $h(\mathcal{C})(v) = 0$ and $h^*(\mathcal{C})(v) = 1$ never occurs. Note that if $h(\mathcal{C})(v) = 0$, under `OR` master functions, the threshold condition of v is **not** satisfied in any layer under h. That is, $\Gamma_j[\mathcal{C}, v] < \tau_j^h(v), \ \forall j \in [k]$. Since $\tau_j^h(v) \leq \tau_j^{h^*}(v), \forall j \in [k]$, it follows that the threshold condition of v is also **not** satisfied in any of the layers under h^*. Therefore, if $h(\mathcal{C})(v) = 0$, then we must have $h^*(\mathcal{C})(v) = 0$. This completes the proof of Claim 6.0.2.

Based on Claim 6.0.2, a useful interpretation of the event $A(C, v, h)$ is that the threshold condition of v is satisfied in **at least one** layer under $\mathcal{C}$ in h, but in the true system h^*, the threshold condition of v is **not** satisfied in any of the layers.

To arrive at the result in Ineq (14), we first bound the probability (over $\mathcal{T} \sim \mathcal{D}^q$) of learning a bad h where $\Pr_{\mathcal{C}\sim\mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \epsilon/(\sigma k)$. For a $\mathcal{C} \sim \mathcal{D}$, and a layer $i \in [k]$, recall that $A(\mathcal{C}, i, v, h)$ is the event

“$\Gamma_i[\mathcal{C}, v] \geq \tau_i^h(v)$ and $\Gamma_j[\mathcal{C}, v] < \tau_j^{h^*}(v),\ \ \forall j \in [k]$”. Note that the event $A(\mathcal{C}, v, h)$ occurs if and only if $A(\mathcal{C}, i, v, h)$ occurs for at least one layer $i \in [k]$.

For any layer $i \in [k]$, if the true threshold $\tau_i^{h^*}(v) = 0$, the event $A(\mathcal{C}, i, v, h)$ will never happen as the algorithm always learns the correct threshold, i.e., $\tau_i^h(v) = \tau_i^{h^*}(v)$. Now suppose $\tau_i^{h^*}(v) \geq 1$. We can apply Lemma 3.2 with $\alpha = \epsilon/(\sigma k)$ and conclude that with probability **at most** $(1 - \epsilon/(\sigma k))^q$ over the choices of q examples $\mathcal{T} \sim \mathcal{D}^q$, the learned h is “bad”; that is, $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \epsilon/(\sigma k)$. Overall, when considering all the layers in the network, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $k \cdot (1 - \epsilon/(\sigma k))^q$, there exists a layer $i \in [k]$ such that

$$\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \frac{\epsilon}{\sigma k} \tag{16}$$

Next, we bound the probability (over $\mathcal{T} \sim \mathcal{D}^q$) of learning a hypothesis $h \in \mathcal{H}$ such that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \epsilon/\sigma$, that is, the probability of h predicting wrong next state of v is at least ϵ/σ. In particular, note that if $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \epsilon/\sigma$, then there must exist a layer $i \in [k]$ such that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, i, v, h)] \geq \epsilon/(\sigma k)$. By our aforementioned argument for Ineq (16), it follows that with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $k \cdot (1 - \epsilon/(\sigma k))^q$,

$$\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \frac{\epsilon}{\sigma} \tag{17}$$

Lastly, we consider the event where $h(\mathcal{C}) \neq h^*(\mathcal{C})$ for a configuration $\mathcal{C} \sim \mathcal{D}$, that is, the successor of $\mathcal{C}$ predicted by the learned hypothesis h is wrong. Note that if $\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] \geq \epsilon$, then there exists a vertex $v \in \mathcal{V}'$ such that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \epsilon/\sigma$, which happens with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $\sigma k \cdot (1 - \epsilon/(\sigma k))^q$. Setting $q = \lceil \frac{1}{\epsilon} \cdot \sigma k \cdot \log(\frac{\sigma k}{\delta}) \rceil$, one can verify that $\sigma k \cdot (1 - \epsilon/(\sigma k))^q \leq \delta$. Overall, when $q = \lceil \frac{1}{\epsilon} \cdot \sigma k \cdot \log(\frac{\sigma k}{\delta}) \rceil$, with probability at least $1 - \delta$ over the choices of q examples $\mathcal{T} \sim \mathcal{D}^q$, the learned $h \in \mathcal{H}$ satisfies the condition

$$\Pr_{\mathcal{C} \sim \mathcal{D}}[h(\mathcal{C}) \neq h^*(\mathcal{C})] < \epsilon. \tag{18}$$

This completes the proof. ■

Proofs in Section 3.3

> **Theorem 3.4.** *For any given $\epsilon, \delta, \beta \in (0, 1)$, with a training set $\mathcal{T}$ of size $q = \lceil 1/\epsilon \cdot 1/\beta \cdot k \cdot \log(\sigma k/\delta) \rceil$, the proposed algorithm learns an $h \in \mathcal{H}$, such that with probability at least $1 - \delta$ over $\mathcal{T} \sim \mathcal{D}^q$, h satisfies that*
>
> $$\Pr_{\mathcal{C} \sim \mathcal{D}}[W(h(\mathcal{C}), h^*(\mathcal{C})) \geq \beta\sigma] \ \leq \ \epsilon$$

Proof. We follow the analysis in Theorem 3.3. Let $h \in \mathcal{H}$ be the hypothesis learned by the algorithm. Recall that $A(\mathcal{C}, v, h)$ is the “bad” event where $h(\mathcal{C})(v) \neq h^*(\mathcal{C})(v)$ for a vertex $v \in \mathcal{V}'$ and $\mathcal{C} \sim \mathcal{D}$. Using Lemma 3.2 where we set $\alpha = (\epsilon\beta)/k$, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $k \cdot (1 - (\epsilon\beta)/k)^q$, we have $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq (\epsilon\beta)$ for any vertex $v \in \mathcal{V}'$. Thus, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at most $\sigma k \cdot (1 - (\epsilon\beta)/k)^q$, there exists a vertex $v \in \mathcal{V}'$ such that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] \geq \ \epsilon\beta$.

Equivalently, with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at least $1 - \sigma k \cdot (1 - (\epsilon\beta)/k)^q$, it holds that $\Pr_{\mathcal{C} \sim \mathcal{D}}[A(\mathcal{C}, v, h)] <$

$(\epsilon\beta)$ for all $v \in \mathcal{V}'$. Then by the linearity of expectation,

$$\mathbb{E}_{\mathcal{C}\sim\mathcal{D}}[W(h(\mathcal{C}),h^*(\mathcal{C}))] < \epsilon\beta \cdot \sigma \tag{19}$$

Using Markov Inequality, it follows that with probability (over $\mathcal{T} \sim \mathcal{D}^q$) at least $1 - \sigma k \cdot (1-(\epsilon\beta)/k)^q$, the learned $h \in \mathcal{H}$ satisfies

$$\Pr{}_{\mathcal{C}\sim\mathcal{D}}[W(h(\mathcal{C}),h^*(\mathcal{C})) \geq \beta\sigma] \leq \epsilon \tag{20}$$

Setting $q = \lceil 1/\epsilon \cdot 1/\beta \cdot k \cdot \log{(\sigma k/\delta)} \rceil$, we have $1 - \sigma k \cdot (1-(\epsilon\beta)/k)^q \;\geq\; 1-\delta$. This completes the proof. ■

6.3 Additional Material for Section 4

We first revisit some definitions.

Definition 6.1 (Shattering). Given a hypothesis class $\mathcal{H}$, a set $\mathcal{R} \subseteq \mathcal{X}$ is **shattered** by $\mathcal{H}$ if there exist two functions $g_1, g_2 : \mathcal{R} \to \mathcal{X}$ that satisfy both of the following conditions:

- Condition 1: For every $\mathcal{C} \in \mathcal{R}$, $g_1(\mathcal{C}) \neq g_2(\mathcal{C})$.
- Condition 2: For every subset $\mathcal{R}' \subseteq \mathcal{R}$, there exists $h \in \mathcal{H}$ such that $\forall \mathcal{C} \in \mathcal{R}'$, $h(\mathcal{C}) = f(\mathcal{C})$ and $\forall \mathcal{C} \in \mathcal{R} \setminus \mathcal{R}'$, $h(\mathcal{C}) = g(\mathcal{C})$.

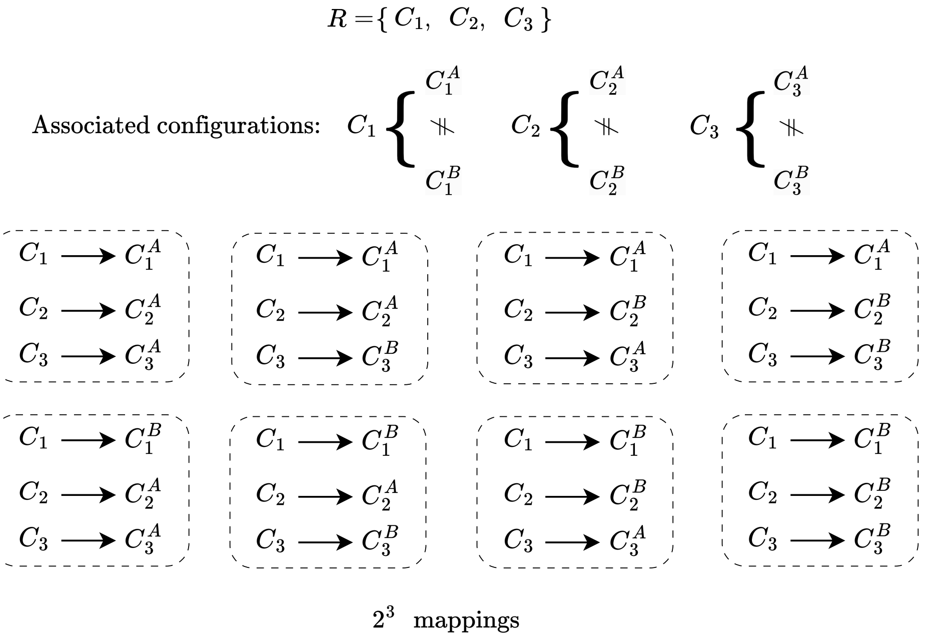

$R = \{ C_1,\ C_2,\ C_3 \}$
Associated configurations:
C_1 C_1^A $\not\Vdash$ C_1^B
C_2 C_2^A $\not\Vdash$ C_2^B
C_3 C_3^A $\not\Vdash$ C_3^B
$C_1 \to C_1^A$, $C_2 \to C_2^A$, $C_3 \to C_3^A$
$C_1 \to C_1^A$, $C_2 \to C_2^A$, $C_3 \to C_3^B$
$C_1 \to C_1^A$, $C_2 \to C_2^B$, $C_3 \to C_3^A$
$C_1 \to C_1^A$, $C_2 \to C_2^B$, $C_3 \to C_3^B$
$C_1 \to C_1^B$, $C_2 \to C_2^A$, $C_3 \to C_3^A$
$C_1 \to C_1^B$, $C_2 \to C_2^A$, $C_3 \to C_3^B$
$C_1 \to C_1^B$, $C_2 \to C_2^B$, $C_3 \to C_3^A$
$C_1 \to C_1^B$, $C_2 \to C_2^B$, $C_3 \to C_3^B$
2^3 mappings

Figure 4: An alternative interpretation of shattering: associated configurations and $2^{|\mathcal{R}|}$ mappings for a set $\mathcal{R}$.

An alternative interpretation of shattering. In our context, equivalent definitions of the two conditions are as follows:

- Condition 1: Each $\mathcal{C} \in \mathcal{R}$ is associated with two configurations, denoted by $\mathcal{C}^A$ and $\mathcal{C}^B$, where $\mathcal{C}^A \neq \mathcal{C}^B$ (i.e., $\mathcal{C}^A = g_1(\mathcal{C})$ and $\mathcal{C}^B = g_2(\mathcal{C})$).
- Condition 2: Consider the $2^{|\mathcal{R}|}$ possible mappings from $\mathcal{R}$ to the associated configurations, such that in each mapping Φ, every $\mathcal{C} \in \mathcal{R}$ is mapped to one of its associated configuration (i.e., $\Phi(\mathcal{C}) = \mathcal{C}^A$ or $\Phi(\mathcal{C}) = \mathcal{C}^B$). For each such mapping Φ, there exists a system (hypothesis) $h_\Phi \in \mathcal{H}$ that produces Φ. That is, $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$ for all $\mathcal{C} \in \mathcal{R}$.

Definition 6.2 (**Contested Vertices).** We call a vertex v **contested** for a $\mathcal{C} \in \mathcal{R}$ if $\mathcal{C}^A(v) \neq \mathcal{C}^B(v)$.

We use the above definition of shattering in all the proofs. An example of condition 1 and the $2^{|\mathcal{R}|}$ mappings of condition 2 for a set $\mathcal{R}$ with three configurations are shown in Fig 4.

Definition 6.3 (**Landmark Vertices).** Suppose the underlying network has a single layer. Given a set $\mathcal{R} \subseteq \mathcal{X}$, a vertex $v \in \mathcal{V}'$ is a ***landmark*** vertex for a configuration $\mathcal{C} \in \mathcal{R}$ if $\Gamma[\mathcal{C}, v] \neq \Gamma[\hat{\mathcal{C}}, v]$ for all $\hat{\mathcal{C}} \in \mathcal{R} \setminus \{\mathcal{C}\}$.

Let $\mathcal{W}(\mathcal{R}) \subseteq \mathcal{V}'$ be the set vertices that are landmarks for at least one configuration in $\mathcal{R}$.

Definition 6.4 (**Canonical Sets).** Suppose the underlying network has a single layer. A set $\mathcal{R} \subseteq \mathcal{X}$ is ***canonical*** w.r.t. $\mathcal{H}$ if there exists an injective mapping from $\mathcal{R}$ to $\mathcal{W}(\mathcal{R})$ s.t. each $\mathcal{C} \in \mathcal{R}$ is mapped to a landmark vertex of $\mathcal{C}$.

Detailed Proofs in Section 4.1

Lemma 4.4. *When the underlying network has a single-layer, a set $\mathcal{R} \subseteq \mathcal{X}$ can be shattered by $\mathcal{H}$* ***if and only if*** *$\mathcal{R}$ is canonical w.r.t. $\mathcal{H}$*

Proof. ($\Rightarrow$) Suppose $\mathcal{H}$ shatters $\mathcal{R}$. We want to show that $\mathcal{R}$ is canonical. For each configuration $\mathcal{C} \in \mathcal{R}$, let $\mathcal{C}^A$ and $\mathcal{C}^B$ be the two associated configurations, where $\mathcal{C}^A \neq \mathcal{C}^B$ (i.e., they disagree on the state of at least one vertex). Consider the $2^{|\mathcal{R}|}$ possible mappings from $\mathcal{R}$ to the associated configurations, where in each mapping Φ, each $\mathcal{C} \in \mathcal{R}$ is mapped to one of its associated configuration (i.e., $\Phi(\mathcal{C}) = \mathcal{C}^A$ or $\Phi(\mathcal{C}) = \mathcal{C}^B$). The *second condition* of shattering implies that for each of the mapping Φ defined above, there exists a system $h_\Phi \in \mathcal{H}$ such that $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$ for all $\mathcal{C} \in \mathcal{R}$.

Recall that a vertex v **contested** for a configuration $\mathcal{C} \in \mathcal{R}$ if the state of v in $\mathcal{C}^A$ is different from its state in $\mathcal{C}^B$. An example of a contested vertex is given in Fig 5. Since $\mathcal{C}^A \neq \mathcal{C}^B$, each $\mathcal{C} \in \mathcal{R}$ has at least one contested vertex. We argue that contexted vertices can only be in $\mathcal{V}'$.

Claim 6.4.1. If $\mathcal{H}$ shatters $\mathcal{R}$, then contested vertices can only be in the set $\mathcal{V}'$; that is, only vertices with unknown thresholds can be contested.

For purposes of contradiction, suppose there exists a vertex $v \in \mathcal{V} \setminus \mathcal{V}'$ whose threshold is known, and v is contested for a configuration $\mathcal{C} \in \mathcal{R}$. The second condition of shattering implies that there exist two systems

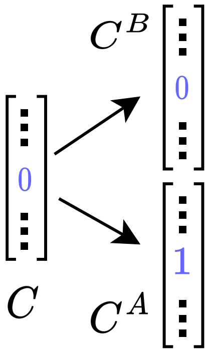


Figure 5: An example of a contested vertex v for a configuration $\mathcal{C}$. In particular, $\mathcal{C}^A$ and $\mathcal{C}^B$ are the two associated configurations of $\mathcal{C}$. The state of v is highlighted in blue.

$h, h' \in \mathcal{H}$ such that the state of v is 1 in $h(\mathcal{C})$, and is 0 under $h'(\mathcal{C})$. However, since the threshold of v is fixed, for the same configuration $\mathcal{C}$, the state of v is always the same in the successor of $\mathcal{C}$ regardless of the underlying system in $\mathcal{H}$. Therefore, such $h, h' \in \mathcal{H}$ cannot coexist, which contradicts the fact that $\mathcal{H}$ shatters $\mathcal{R}$. This establishes the claim.

Our argument of $\mathcal{R}$ being canonical is developed based on this notion of contested vertices. Overall, we want to show the following two claims: (i) configurations in $\mathcal{R}$ do not share contested vertices, and (ii) a contested vertex for a $\mathcal{C} \in \mathcal{R}$ is also a landmark vertex for $\mathcal{C}$. Then since each $\mathcal{C} \in \mathcal{R}$ has at least one contested vertex, it immediately follows that there exists an injective (i.e., one-to-one) mapping from $\mathcal{R}$ to $\mathcal{W}(\mathcal{R})$ where $\mathcal{W}(\mathcal{R}) \subseteq \mathcal{V}'$ is the set of vertices that are landmarks for at least one configuration in $\mathcal{R}$. Then by definition, $\mathcal{R}$ is canonical.

We now establish the above two claims. Recall that for a configuration $\mathcal{C}$ and a vertex v, $\Gamma[\mathcal{C}, v]$ is the **score** of v in $\mathcal{C}$, that is, the number of 1's in the input provided by $\mathcal{C}$ to the interaction function.

Claim 6.4.2. If $\mathcal{H}$ shatters $\mathcal{R}$, then no two configurations in $\mathcal{R}$ can have any common contested vertices.

For purposes of contradiction, suppose $v \in \mathcal{V}'$ is a contested vertex for at least two configurations in $\mathcal{R}$; let $\mathcal{C}_a$ and $\mathcal{C}_b$ be two such configurations. We now show that $\mathcal{H}$ cannot shatter $\mathcal{R}$. Recall that $h(\mathcal{C}_a)(v)$ is the state of v in the successor $h(\mathcal{C}_a)$ of $\mathcal{C}_a$ under a system $h \in \mathcal{H}$. By the second condition of shattering, there exists a $h \in \mathcal{H}$ such that $h(\mathcal{C}_a)(v) \neq h(\mathcal{C}_b)(v)$ (i.e., $h(\mathcal{C}_a)(v) = 1$ and $h(\mathcal{C}_b)(v) = 0$, or $h(\mathcal{C}_a)(v) = 0$ and $h(\mathcal{C}_b)(v) = 1$).

If $\Gamma[\mathcal{C}_a, v] = \Gamma[\mathcal{C}_b, v]$, then there cannot exist such a system $h \in \mathcal{H}$ where $h(\mathcal{C}_a)(v) \neq h(\mathcal{C}_b)(v)$ since the threshold condition of v cannot be both satisfied and unsatisfied under the same input to the interaction function. This violates the second condition of shattering; thus, $\mathcal{H}$ fails to shatter $\mathcal{R}$ under this case. Now suppose $\Gamma[\mathcal{C}_a, v] < \Gamma[\mathcal{C}_b, v]$. Then there cannot exist an $h \in \mathcal{H}$ such that $h(\mathcal{C}_a)(v) = 1$ but $h(\mathcal{C}_b)(v) = 0$ since if the threshold condition is satisfied under the smaller score (i.e., $\Gamma[\mathcal{C}_a, v]$), it must also be satisfied under the larger score (i.e., $\Gamma[\mathcal{C}_b, v]$). Thus, $\mathcal{H}$ fails to shatter $\mathcal{R}$. The argument for the case where $\Gamma[\mathcal{C}_a, v] > \Gamma[\mathcal{C}_b, v]$ follows analogously. This concludes the Claim 6.4.2.

Claim 6.4.3. If $\mathcal{H}$ shatters $\mathcal{R}$, then a contested vertex for a $\mathcal{C} \in \mathcal{R}$ is also a landmark vertex for $\mathcal{C}$.

We want to show that if $v \in \mathcal{V}'$ is contested for $\mathcal{C} \in \mathcal{R}$, then $\Gamma[\mathcal{C}, v] \neq \Gamma[\hat{\mathcal{C}}, v]$ for all $\hat{\mathcal{C}} \in \mathcal{R} \setminus \{\mathcal{C}\}$. Suppose there exists such a $\hat{\mathcal{C}} \in \mathcal{R}$ where $\Gamma[\mathcal{C}, v] = \Gamma[\hat{\mathcal{C}}, v]$. Claim 6.4.2 implies that v cannot be contested for $\hat{\mathcal{C}}$, that is, the state of v is the same in the two associated configurations of $\hat{\mathcal{C}}$; let s_v denote this state value. Given that v is contested for $\mathcal{C}$, the state of v in one of $\mathcal{C}$'s associated configurations must be different from s_v. Since $\Gamma[\mathcal{C}_a, v] = \Gamma[\mathcal{C}_b, v]$, however, there cannot exist an $h \in \mathcal{H}$ where $h(\mathcal{C}_a)(v) \neq s_v$ because the threshold condition of v cannot be both satisfied and unsatisfied under the same input to the interaction function, contradicting the second condition of shattering. This concludes the proof of Claim 6.4.3.

Overall, we have shown that configurations in $\mathcal{R}$ do not share common contested vertices, and that every contested vertex for a configuration is also a landmark vertex. it follows that there exists an injective mapping where each $\mathcal{C} \in \mathcal{R}$ is mapped to a landmark vertex of $\mathcal{C}$. Then by definition, $\mathcal{R}$ is canonical.

($\Leftarrow$) Suppose that $\mathcal{R} \subseteq \mathcal{X}$ is canonical w.r.t $\mathcal{H}$. To show that $\mathcal{H}$ shatters $\mathcal{R}$, we first discuss how the two associated configurations, $\mathcal{C}^A$ and $\mathcal{C}^B$, of each $\mathcal{C} \in \mathcal{R}$ should be chosen. We then establish that for each of the $2^{|\mathcal{R}|}$ possible mappings from $\mathcal{R}$ to the associated configurations (where $\mathcal{C} \in \mathcal{R}$ is mapped to one of its associated configurations), there exists a system in $\mathcal{H}$ that produces this mapping.

Given that $\mathcal{R}$ is canonical, let $\Upsilon : \mathcal{R} \to \mathcal{W}(\mathcal{R})$ be a corresponding *injective* mapping from $\mathcal{R}$ to the set $\mathcal{W}(\mathcal{R})$ such that each $\mathcal{C} \in \mathcal{R}$ is mapped to a landmark vertex of $\mathcal{C}$. For each $\mathcal{C} \in \mathcal{R}$, we construct two associated configuration $\mathcal{C}^A$ and $\mathcal{C}^B$ by specifying states of each vertex $v \in \mathcal{V}$ as follows:

- **Case 1**: Suppose $v \in \mathcal{V} \setminus \mathcal{V}'$, that is, the threshold of v, denoted by $\tau^{h^*}(v)$, is known. Then the state of v in $\mathcal{C}^A$ and $\mathcal{C}^B$ is the same, which is determined by $\tau^{h^*}(v)$ and $\Gamma[v, \mathcal{C}]$. That is, $\mathcal{C}^A(v) = \mathcal{C}^B(v) = 1$ if $\Gamma[v, \mathcal{C}] \geq \tau^{h^*}(v)$, and $\mathcal{C}^A(v) = \mathcal{C}^B(v) = 0$ otherwise.
- **Case 2**: $v \in \mathcal{V}'$.
  - Subcase 2.1: Suppose $v = \Upsilon(\mathcal{C})$. Then we set $\mathcal{C}^A(v) = 0$ and $\mathcal{C}^B(v) = 1$. That is, v is contested for $\mathcal{C}$.
  - Subcase 2.2: Suppose $v \neq \Upsilon(\mathcal{C})$, and $v = \Upsilon(\hat{\mathcal{C}})$ for some other $\hat{\mathcal{C}} \in \mathcal{R}$. Note that the case where $\Gamma[\mathcal{C}, v] = \Gamma[\hat{\mathcal{C}}, v]$ cannot arise since v is a landmark vertex for $\hat{\mathcal{C}}$. If $\Gamma[\mathcal{C}, v] < \Gamma[\hat{\mathcal{C}}, v]$, then $\mathcal{C}^A(v) = \mathcal{C}^B(v) = 0$. On the other hand, if $\Gamma[\mathcal{C}, v] > \Gamma[\hat{\mathcal{C}}, v]$, then $\mathcal{C}^A(v) = \mathcal{C}^B(v) = 1$.
  - Subcase 2.3: Suppose $v \neq \Upsilon(\mathcal{C})$, and also $v \neq \Upsilon(\hat{\mathcal{C}})$ for any other $\hat{\mathcal{C}} \in \mathcal{R}$, then $\mathcal{C}^A(v) = \mathcal{C}^B(v) = 1$.

This completes the construction of the two associated configurations $\mathcal{C}^A$ and $\mathcal{C}^B$ for each $\mathcal{C} \in \mathcal{R}$. We now show that $\mathcal{H}$ shatters $\mathcal{R}$ under the defined associations.

To begin with, observe that $\mathcal{C}^A \neq \mathcal{C}^B$ for all $\mathcal{C} \in \mathcal{R}$, as the states of $\Upsilon(\mathcal{C})$ are different in $\mathcal{C}^A$ and $\mathcal{C}^B$. Thus, the first condition of shattering is satisfied.

Now consider the $2^{|\mathcal{R}|}$ possible mappings from $\mathcal{R}$ to the associated configurations, where in each mapping Φ, each $\mathcal{C} \in \mathcal{R}$ is mapped to one of its associated configuration (i.e., $\Phi(\mathcal{C}) = \mathcal{C}^A$ or $\Phi(\mathcal{C}) = \mathcal{C}^B$). To prove the second condition of shattering, we want to show that for each Φ defined above, there exists a system $h_\Phi \in \mathcal{H}$ such that $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$ for all $\mathcal{C} \in \mathcal{R}$. Given a mapping Φ, we characterize h_Φ by presenting how the threshold of each vertex is determined:

- **Case 1**: Suppose $v \in \mathcal{V} \setminus \mathcal{V}'$, then its threshold is already known.

- **Case 2**: Suppose $v \in \mathcal{V}'$.
  - Subcase 2.1: If $v \neq \Upsilon(\mathcal{C})$ for any $\mathcal{C} \in \mathcal{R}$, then we set v's threshold to be 0.
  - Subcase 2.2: If $v = \Upsilon(\mathcal{C})$ for a $\mathcal{C} \in \mathcal{R}$. Then set v's threshold to be $\Gamma[\mathcal{C}, v]$ if $\Phi(\mathcal{C})(v) = 1$. On the other hand, $\Phi(\mathcal{C})(v) = 0$, then set v's threshold to be $\Gamma[\mathcal{C}, v] + 1$.

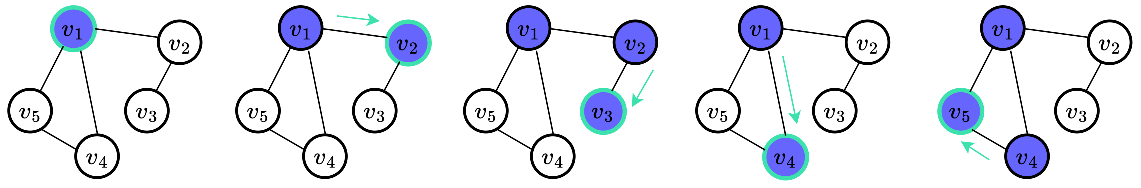
This completes the specification of h_Φ. One can easily verify that $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$ for all $\mathcal{C} \in \mathcal{R}$; that is, the second condition of shattering is satisfied. Overall, we have shown that $\mathcal{H}$ shatters $\mathcal{R}$. This concludes the proof. ■

> **Theorem 4.5.** *When the underlying network has a single layer, a shatterable set of size σ can be constructed. Thus, we have Ndim$(\mathcal{H}) = \sigma$.*

Proof. We show how a canonical set of size σ can be constructed. Then the theorem follows from the equivalence between a canonical set and a shatterable set. In particular, given *any* underlying single-layer network $\mathcal{G}$, we present an algorithm to construct a canonical set $\mathcal{R} \subset \mathcal{X}$ that consists of σ configurations.

Let $\mathcal{G}' = \mathcal{G}[\mathcal{V}']$ be the subgraph induced on $\mathcal{V}'$; $\mathcal{G}'$ could be disconnected. The algorithm involves a depth-first traversal over $\mathcal{G}'$, starting from any initial vertex. During the traversal, when a vertex $v \in \mathcal{V}'$ is visited for the first time, a configuration $\mathcal{C}_v$ is constructed. In particular, our algorithm enforces v to be the landmark vertex to which $\mathcal{C}_v$ is mapped under the injective mapping defined for a canonical set.

We now describe the algorithm. The set $\mathcal{R}$ is initially empty. Starting from any vertex $v_1 \in \mathcal{V}'$, we proceed with a depth-first traversal on $\mathcal{G}'$, while maintaining a stack $\mathcal{K} \subseteq \mathcal{V}'$ of vertices that are currently being visited. Let v_i, $i \in [\sigma]$, denote the ith vertex that is visited for the first time in the traversal. When v_i is visited for the first time, a configuration $\mathcal{C}_{v_i}$ is constructed and added to the set $\mathcal{R}$ where $\mathcal{C}_{v_i}(v) = 1$ if $v \in \mathcal{K}$ and $\mathcal{C}_{v_i}(v) = 0$ otherwise. Note that the states of vertices in $\mathcal{V} \setminus \mathcal{V}'$ are always 0 in $\mathcal{C}_{v_i}$. The algorithm terminates when all vertices in $\mathcal{V}'$ are visited (and thus $\mathcal{K}$ is empty), and returns $\mathcal{R}$. A pictorial example of the algorithm is given in Fig 6.



$$R = \left\{ C_{v_1} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix} \quad C_{v_2} = \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} \quad C_{v_3} = \begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix} \quad C_{v_4} = \begin{bmatrix}1\\0\\0\\1\\0\end{bmatrix} \quad C_{v_5} = \begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix} \right\}$$

Figure 6: A pictorial example of the algorithm running on a graph of 5 vertices. Vertices in the stack $\mathcal{K}$ are highlighted in blue.

Given the resulting set $\mathcal{R}$, Since $\mathcal{G}'$ has σ vertices, $|\mathcal{R}| = \sigma$. We now show that each $v_i \in \mathcal{V}'$ is a landmark vertex of $\mathcal{C}_{v_i} \in \mathcal{R}$, thereby establishing that $\mathcal{R}$ is canonical. For a $v_i \in \mathcal{V}'$, recall that the score v_i in a configuration $\mathcal{C}$, denoted by $\Gamma[\mathcal{C}, v_i]$, is the number of state-1 vertices in v's closed neighborhood in $\mathcal{G}$.

The proof of v_i being a landmark vertex for $\mathcal{C}_{v_i}$ proceeds in two steps. First, consider the subset $\mathcal{R}_1 = \{C_{v_1}, ..., \mathcal{C}_{v_{i-1}}\} \subset \mathcal{R}$ of configurations constructed by the algorithm *before* v_i was visited for the first time. (If $v_i = v_1$, then $\mathcal{R}_1$ is empty.) We argue that the score of v_i in any of the configurations in $\mathcal{R}_1$ is different from the score of v_i in $\mathcal{C}_{v_i}$. That is, $\Gamma[\mathcal{C}, v_i] \neq \Gamma[\mathcal{C}_{v_i}, v_i]$, $\forall \mathcal{C} \in \mathcal{R}_1$. Next, consider the subset $\mathcal{R}_2 = \{C_{v_i+1}, ..., \mathcal{C}_{v_\sigma}\} \subset \mathcal{R}$ of configurations constructed by the algorithm *after* v_i was visited for the first time; if $v_i = v_\sigma$, then $\mathcal{R}_2$ is empty. Similarly, we argue that the scores of v_i in configurations in $\mathcal{R}_2$ are different from the score of v_i in $\mathcal{C}_{v_i}$. We start with the first claim:

Claim 6.4.4. For each i, $1 \leq i \leq \sigma$, $\Gamma[\mathcal{C}, v_i] \neq \Gamma[\mathcal{C}_{v_i}, v_i]$, $\forall \mathcal{C} \in \mathcal{R}_1$ where $\mathcal{R}_1 = \{C_{v_1}, ..., \mathcal{C}_{v_{i-1}}\} \subset \mathcal{R}$.

The claim is trivially true if $v_i = v_1$ since $\mathcal{R}_1$ is empty. Suppose $i > 1$. Observe that when v_i is visited for the first time, v_i gets added to $\mathcal{K}$, and thus $\Gamma[\mathcal{C}_{v_i}, v_i] = \Gamma[\mathcal{C}_{v_{i-1}}, v_i] + 1$. We now show that before the algorithm visits v_i for the first time, the scores of v_i in the sequence of constructed configurations in $\mathcal{R}_1$ are non-decreasing. Note that when the algorithm traverses connected components that do **not** contain v_i, the score of v_i is always 0 in the resulting configurations. Now focus on the connected component containing v_i. Recall that in a depth-first traversal, a vertex remains on the stack if it has at least one unvisited neighbor. It follows that all of v_i's neighbors who were visited before v_i will remain on the stack $\mathcal{K}$ before v_i is visited. Since only vertices on the stack have state-1 in each configuration, the score of v_i is non-decreasing in $(C_{v_1}, ..., \mathcal{C}_{v_{i-1}})$, that is $\Gamma[\mathcal{C}_{v_1}, v_i] \leq ... \leq \Gamma[\mathcal{C}_{v_{i-1}}, v_i]$. Since $\Gamma[\mathcal{C}_{v_i}, v_i] = \Gamma[\mathcal{C}_{v_{i-1}}, v_i] + 1$, it follows that $\Gamma[\mathcal{C}, v_i] \neq \Gamma[\mathcal{C}_{v_i}, v_i]$, $\forall \mathcal{C} \in \mathcal{R}_1$. This concludes Claim 6.4.4.

Now, we establish the second claim:

Claim 6.4.5. For each i, $1 \leq i \leq \sigma$, $\Gamma[\mathcal{C}, v_i] \neq \Gamma[\mathcal{C}_{v_i}, v_i]$, $\forall \mathcal{C} \in \mathcal{R}_2$ where $\mathcal{R}_2 = \{C_{v_{i+1}}, ..., \mathcal{C}_{v_\sigma}\} \subset \mathcal{R}$.

The claim is trivially true if $v_i = v_\sigma$ since $\mathcal{R}_2$ is then empty. Suppose $i < \sigma$. We show that when a vertex $v_j \in \mathcal{V}'$, $i < j \leq \sigma$, is visited for the first time (and $\mathcal{C}_{v_j} \in \mathcal{R}_2$ is constructed), if v_i is on the stack (i.e., $v_i \in \mathcal{K}$), then $\Gamma[\mathcal{C}_{v_j}, v_i] > \Gamma[\mathcal{C}_{v_i}, v_i]$; if v_i is not on the stack, then $\Gamma[\mathcal{C}_{v_j}, v_i] < \Gamma[\mathcal{C}_{v_i}, v_i]$. Let $\mathcal{N}_{\text{bef}}(v)$ and $\mathcal{N}_{\text{aft}}(v)$ be the set of neighbors that were visited before and after v_i, respectively. Suppose $v_i \in \mathcal{K}$ when v_j is visited. Note that all vertices in $\mathcal{N}_{\text{bef}}(v)$ must also be on the stack $\mathcal{K}$. Further, at least one of v's neighbors in $\mathcal{N}_{\text{aft}}(v)$ must be on the stack. It follows that $\Gamma[\mathcal{C}_{v_j}, v_i] \geq \Gamma[\mathcal{C}_{v_i}, v_i] + 1 > \Gamma[\mathcal{C}_{v_i}, v_i]$. Now suppose $v_i \notin \mathcal{K}$ when v_j is visited. This means that no neighbors in $\mathcal{N}_{\text{aft}}(v)$ are on the stack. It follows that $\Gamma[\mathcal{C}_{v_j}, v_i] \leq \Gamma[\mathcal{C}_{v_i}, v_i] - 1 < \Gamma[\mathcal{C}_{v_i}, v_i]$. Consequently, $\Gamma[\mathcal{C}, v_i] \neq \Gamma[\mathcal{C}_{v_i}, v_i]$, $\forall \mathcal{C} \in \mathcal{R}_2$. This establishes the claim.

With Claims 6.4.4 and 6.4.5, we have shown that for any $\mathcal{C}_{v_i} \in \mathcal{R}$, it holds that

$$\Gamma[\mathcal{C}_{v_i}, v_i] \neq \Gamma[\mathcal{C}, v_i], \ \forall \mathcal{C} \in \mathcal{R}, \mathcal{C} \neq \mathcal{C}_{v_i} \tag{21}$$

That is, v_i is a landmark vertex of $\mathcal{C}_{v_i}$. This immediately implies the existence of an injective mapping where each $\mathcal{C}_{v_i} \in \mathcal{R}$ is mapped to v_i, $i \in [\sigma]$. Then by definition, $\mathcal{R}$ is canonical. Given the equivalence between a canonical set and a shatterable set shown in Lemma 4.4, it follows that $\mathcal{R}$ is also shatterable by $\mathcal{H}$. This concludes the proof. ■

Detailed Proofs in Section 4.2

Lemma 4.6. *Suppose the underlying network has $k \geq 2$ layers. Then the size of any shatterable set is at most $k\sigma$.*

Proof. We first show that for any shatterable set $\mathcal{R}$, each vertex $v \in \mathcal{V}'$ is contested for at most k configurations in $\mathcal{R}$. Recall that a vertex v is **contested** for a configuration $\mathcal{C} \in \mathcal{R}$ if $\mathcal{C}^A(v) \neq \mathcal{C}^B(v)$, where $\mathcal{C}^A$ and $\mathcal{C}^B$ are the two associated configurations of $\mathcal{C}$ defined by shattering (i.e., $\mathcal{C}^A = g_1(\mathcal{C})$ and $\mathcal{C}^B = g_2(\mathcal{C})$). It is easy to see that Claim 6.4.1 in Theorem 4.5 carries over to the multilayer case. That is, contested vertices can only be in the set $\mathcal{V}'$.

For a $v \in \mathcal{V}'$, let $\mathcal{R}_v \subseteq \mathcal{R}$ be the subset of configurations with v being (one of) their contested vertices. W.l.o.g., suppose $\mathcal{R}_v \neq \varnothing$. We establish the claim:

Claim 6.4.6. For each $\mathcal{C} \in \mathcal{R}_v$, $\exists\, i \in [k]$ such that $\Gamma_i(\mathcal{C}, v) > \Gamma_i(\hat{\mathcal{C}}, v),\ \forall \hat{\mathcal{C}} \in \mathcal{R}_v \smallsetminus \{\mathcal{C}\}$.

For contradiction, suppose there exists a $\mathcal{C} \in \mathcal{R}_v$ such that for all layers $i \in [k]$, $\Gamma_i(\mathcal{C}, v) \leq \Gamma_i(\hat{\mathcal{C}}, v)$ for at least one $\hat{\mathcal{C}} \neq \mathcal{C}$, $\hat{\mathcal{C}} \in \mathcal{R}_v$. We now argue that $\mathcal{H}$ cannot shatter $\mathcal{R}_v$ (and thus, cannot shatter $\mathcal{R}$). In particular, consider a mapping Φ (among the $2^{|\mathcal{R}|}$ possible mappings from $\mathcal{R}$ to the associated configurations) such that $\Phi(\mathcal{C})(v) = 1$, and $\Phi(\hat{\mathcal{C}})(v) = 0$ for all other $\hat{\mathcal{C}} \in \mathcal{R}_v \smallsetminus \{\mathcal{C}\}$. Suppose there exists a $h_\Phi \in \mathcal{H}$ that is consistent with such a mapping Φ, where $h_\Phi(\mathcal{C})(v) = 1$ and $h_\Phi(\hat{\mathcal{C}})(v) = 0$ for all $\hat{\mathcal{C}} \in \mathcal{R}_v \smallsetminus \{\mathcal{C}\}$. Let $\tau_i^{h_\Phi}(v)$ denote the threshold of v in the ith layer under such an h_Φ. Since $h_\Phi(\hat{\mathcal{C}})(v) = 0$ for all $\hat{\mathcal{C}} \neq \mathcal{C}$, we have

$$\tau_i^{h_\Phi}(v) > \max_{\hat{\mathcal{C}} \in \mathcal{R}_v \smallsetminus \{\mathcal{C}\}} \Gamma_i(\hat{\mathcal{C}}, v) \geq \Gamma_i(\mathcal{C}, v), \forall i \in [k] \tag{22}$$

However, the above inequality implies that the threshold condition of v is not satisfied on any of the k layers under $\mathcal{C}$, thereby contradicting the condition $h_\Phi(\mathcal{C})(v) = 1$. Thus, no such $h_\Phi \in \mathcal{H}$ exists, and $\mathcal{H}$ does not shatter $\mathcal{R}_v$. This establishes the claim. Overall, Claim 6.4.6 implies that for each $v \in \mathcal{V}'$, the size of $\mathcal{R}_v$ is at most k. It immediately follows that $|\mathcal{R}| \leq k\sigma$ for any shatterable set $\mathcal{R}$. This concludes the proof. ■

Lemma 4.7. *Suppose h^* is an MSyDS whose underlying network has $k \geq 2$ layers. Let $\hat{h}^*$ be a single-layer system obtained from h^* by using the network in any layer $i \in [k]$. If a set $\mathcal{R}$ is shatterable by the hypothesis class of $\hat{h}^*$, then it is also shatterable by the hypothesis class of h^*.*

Proof. Given the underlying multilayer network $\mathcal{M} = \{\mathcal{G}_i\}_{i=1}^k$ of the true system h^*, let $\hat{h}^*$ be a new system with a single-layer underlying network $\mathcal{G}_i \in \mathcal{M}$, for a layer $i \in [k]$. The vertices' thresholds on $\mathcal{G}_i$ are carried over from h^* to $\hat{h}^*$. For our learning context, the set $\mathcal{V}'$ of vertices with unknown thresholds remains the same between h^* and $\hat{h}^*$. Let $\hat{\mathcal{H}}$ be the corresponding hypothesis class of $\hat{h}^*$,.

Given a set $\mathcal{R}$ that is shatterable by $\hat{\mathcal{H}}$, for each $\mathcal{C} \in \mathcal{R}$, let $\hat{\mathcal{C}}^A$ and $\hat{\mathcal{C}}^B$ be the two associated configurations of $\mathcal{C}$. Further, for each mapping Φ from $\mathcal{R}$ to the associated configurations, let $\hat{h}_\Phi \in \hat{\mathcal{H}}$ be a system that produces Φ, that is, $\hat{h}_\Phi(\mathcal{C}) = \Phi(C)$, for all $\mathcal{C} \in \mathcal{R}$.

We show that $\mathcal{R}$ is also shatterable by $\mathcal{H}$, the hypothesis class of h^*. In particular, for each $\mathcal{C} \in \mathcal{R}$, let $\mathcal{C}^A$ and $\mathcal{C}^B$ be the two associated configurations under the shatterable condition for $\mathcal{H}$; we choose $\mathcal{C}^A = \hat{\mathcal{C}}^A$ and

$\mathcal{C}^B = \hat{\mathcal{C}}^B$. Now consider any of the $2^{|\mathcal{R}|}$ mappings from $\mathcal{R}$ to the associated configurations. We argue that for each of such a mapping Φ, there exists a system $h_\Phi \in \mathcal{H}$ where $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$ for all $\mathcal{C} \in \mathcal{R}$. Specifically, in h_Φ, the thresholds of each vertex $v \in \mathcal{V}'$ on each layer $j \in [k]$, denoted by, $\tau_j^{h_\Phi}(v)$, are assigned as follows. For each layer $j \in [k]$, if $j = i$ (i.e., the layer for which $\hat{h}^*$ is defined), then $\tau_j^{h_\Phi}(v)$ equals to the threshold of v in $\hat{h}_\Phi$. Otherwise, we set $\tau_j^{h_\Phi}(v) = deg_j(v) + 2$, where $\deg_j(v)$ is the degree of v in the jth layer. Note that setting $\tau_j^{h_\Phi}(v) = \deg_j(v) + 2$ makes v's interaction function on the jth layer to be the constant-0 function. One can easily verify that $h_\Phi(\mathcal{C}) = \Phi(\mathcal{C})$, for all $\mathcal{C} \in \mathcal{R}$ and thus $\mathcal{R}$ is shatterable by $\mathcal{H}$. ■

Detailed Proofs in Section 4.3

> **Lemma 4.9.** *Given a multilayer network $\mathcal{M}$ and a subset $\mathcal{V}'$ of vertices, for each set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$, there is a shatterable set of size $|\mathcal{Q}_{\mathcal{M},\mathcal{V}'}|$ for the corresponding hypothesis class over $\mathcal{M}$ where thresholds of vertices in $\mathcal{V}'$ are unknown.*

Proof. Given a k-layer network $\mathcal{M}$ with n vertices, let $\mathcal{V}'$ be any subset of vertices in $\mathcal{M}$. Let $\mathcal{H}$ be the threshold dynamical system over $\mathcal{M}$ where the threshold functions of vertices in $\mathcal{V}'$ are unknown. For a subset $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ of vertex-layer pairs, we present a method to construct a shatterable set $\mathcal{R}$ of size $|\mathcal{Q}_{\mathcal{M},\mathcal{V}'}|$. In particular, for each $(v, i) \in \mathcal{Q}_\mathcal{M}$, there is a corresponding configuration $\mathcal{C}_{(v,i)} \in \mathcal{R}$, defined as follows:

$$\begin{cases} \mathcal{C}_{(v,i)}(v') = 1 & \text{If } v' \in N[v, i] \\ \mathcal{C}_{(v,i)}(v') = 0 & \text{Otherwise} \end{cases}$$

where $N[v, i]$ is the closed neighborhood of v on the ith layer in $\mathcal{M}$.

It is easy to see that $|\mathcal{R}| = |\mathcal{Q}_{\mathcal{M},\mathcal{V}'}|$. We now show that the resulting set $\mathcal{R}$ is shatterable by $\mathcal{H}$. Recall that $\Gamma_i[\mathcal{C}, v]$ is the score (i.e., the number of state-1 vertices in v's closed neighborhood) of v in the ith layer under $\mathcal{C}$. We first observe the following:

Observation 6.5. $\Gamma_i[\mathcal{C}_{(v,i)}, v] = deg(v, i) + 1$, for all $(v, i) \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}$.

Observation 6.6. $\Gamma_{i'}[\mathcal{C}_{(v,i)}, v'] < deg(v', i') + 1$, for all $(v, i), (v', i') \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}$, $(v, i) \neq (v', i')$.

The first observation is held by the construction of $\mathcal{C}_{(v,i)}$. To see the second observation, recall that $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ is defined where $N_\mathcal{M}[v', i'] \setminus N_\mathcal{M}[v, i] \neq \emptyset, \forall (v', i'), (v, i) \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}, (v', i') \neq (v, i)$. This implies that given a $\mathcal{C}_{(v,i)} \in \mathcal{R}$ and a pair $(v', i') \in \mathcal{Q}$, at least one vertex in $N[v', i']$ is in state 0 under $\mathcal{C}_{(v,i)}$. The second observation follows immediately.

The key conclusion from the above two observations is that:

$$\Gamma_i[\mathcal{C}_{(v,i)}, v] > \Gamma_i[\mathcal{C}_{(v,i')}, v], \forall i' \neq i, i' \in [k] \tag{23}$$

This allows us to choose v as the contested vertex for $\mathcal{C}_{(v,i)}$, $i \in [k]$, under the shattering of $\mathcal{R}$. For each $(v, i) \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}$, we now discuss how the two associated configurations of $\mathcal{C}_{(v,i)}$, denoted by $\mathcal{C}^A_{(v,i)}$ and $\mathcal{C}^B_{(v,i)}$, can be chosen to satisfy the shattering conditions. In $\mathcal{C}^A_{(v,i)}$, the state of v is 1, where the states of all other vertices are 0. On the other hand, $\mathcal{C}^B_{(v,i)}$ is the zero vector. It is clear that $\mathcal{C}^A_{(v,i)} \neq \mathcal{C}^B_{(v,i)}$, that is, the first shattering condition is satisfied.

We now show that the second shattering condition also holds. In particular, for each mapping Φ from $\mathcal{R}$ to the associated configurations, by choosing the thresholds of vertices, we prove the existence of a system h_Φ that produces the mapping Φ. For each $\mathcal{C}_{(v,i)} \in \mathcal{R}$, if $\mathcal{C}^A_{(v,i)}(v) = 0$, then the threshold of v in the ith layer is set to $deg(v,i) + 2$ in h_Φ. On the other hand, if $\mathcal{C}^A_{(v,i)}(v) = 1$, then then the threshold of v in the ith layer is set to $deg(v,i) + 1$. By Ineq (23), one can easily verify that $h_\Phi(\mathcal{C}_{(v,i)}) = \Phi(\mathcal{C}_{(v,i)})$, for all $\mathcal{C}_{(v,i)} \in \mathcal{R}$. This concludes the proof. ■

Lemma 4.10. *Given $n \geq 1$, $k \geq 2$, and $\mathcal{V}' \subseteq [n]$, in the space of all k-layer graphs with n vertices, the proportion of graphs that admits a set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ of size $k\sigma$ is at least $1 - 4 \cdot (\sigma k)^2 \cdot (\frac{3}{4})^n$.*

Proof. Recall that $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ is a set of vertex-layer pairs $(v,i), v \in \mathcal{V}', i \in [k]$, such that every $(v,i) \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ satisfies:

$$N_{\mathcal{M}}[v,i] \smallsetminus N_{\mathcal{M}}[v',i'] \neq \varnothing, \forall (v',i') \in \mathcal{Q}_{\mathcal{M},\mathcal{V}'}, (v',i') \neq (v,i) \tag{24}$$

where $N_{\mathcal{M}}[v,i]$ is the closed neighborhood of v in the ith layer in $\mathcal{M}$.

We use $\mathcal{G}_{n,k,1/2}$ to denote the space of k-layer graphs with n vertices. Let $\mathcal{M}$ be a graph chosen uniformly at random from $\mathcal{G}_{n,k,1/2}$, denoted by $\mathcal{M} \sim \mathcal{G}_{n,k,1/2}$. Equivalently, $\mathcal{M}$ is a random k-layer graph with n vertices where *each edge in each layer is realized with probability* $p = 1/2$. We will use this equivalent definition in the proof.

Fix two vertex-layer pairs, (v,i) and (v',i'), $v' \in \mathcal{V}'$, $i \in [k]$, $(v,i) \neq (v',i')$. Let $A \subseteq \mathcal{V}'$, $\{v,v'\} \subseteq A$, be a subset of vertices that includes v and v'. Let $d = A$. Recall that $N[v,i]$ denotes the closed neighborhood of v on the ith layer in $\mathcal{M}$. Suppose $v \neq v'$. The probability (over $\mathcal{M} \sim \mathcal{G}_{n,k,1/2}$) that $N[v,i] = A$ and $A \subseteq N[v',i']$ (i.e., condition (24) is violated) is of the form

$$\begin{aligned}
&\Pr{}_{\mathcal{M}\sim\mathcal{G}_{n,k,1/2}}[N[v,i] = A \text{ and } A \subseteq N[v',i']] \\
&= \underbrace{\frac{1}{2}}_{\text{Edge}(v,v')} \cdot \underbrace{(\frac{1}{2})^{d-2} \cdot (\frac{1}{2})^{n-d}}_{\text{Other neighbors and non-neighbors of } v} \cdot \underbrace{(\frac{1}{2})^{d-2} \cdot \left(\sum_{j=0}^{n-d} \binom{n-d}{j} (\frac{1}{2})^{j} (\frac{1}{2})^{n-d-j}\right)}_{\text{Other neighbors and non-neighbors of } v'} \\
&= (\frac{1}{2})^{n+d-3}
\end{aligned}$$

If $v = v'$, then one can verify that $\Pr_{\mathcal{M}\sim\mathcal{G}_{n,k,1/2}}[N[v,i] = A \text{ and } A \subseteq N[v',i']] = (1/2)^{n+d-2}$.

Extending the argument to any such subset A, we then have

$$\begin{aligned}
&\Pr{}_{\mathcal{M}\sim\mathcal{G}_{n,k,1/2}}[N[v,i] \subseteq N[v',i']] \leq \sum_{d=1}^{n} \binom{n}{d} (\frac{1}{2})^{n+d-2} \\
&< (\frac{1}{2})^{n-2} \cdot (\frac{3}{2})^{n} \\
&= 4 \cdot (\frac{3}{4})^{n}
\end{aligned}$$

Combining all pairs, the probability (over $\mathcal{M} \sim \mathcal{G}_{n,k,1/2}$) that there exists a (v,i) and (v',i'), $v \in \mathcal{V}', i \in [k]$ such that $N[v,i] \subseteq N[v',i']$ is at most:

$$8 \cdot \binom{\sigma k}{2} \cdot (\frac{3}{4})^{n} \leq 4 \cdot (\sigma k)^{2} \cdot (\frac{3}{4})^{n}$$

Lastly, with probability at least $1 - 4 \cdot (\sigma k)^{2} \cdot (\frac{3}{4})^{n}$, condition (24) holds for all pairs $(v,i), v \in \mathcal{V}', i \in [k]$, that is, there exists a set $\mathcal{Q}_{\mathcal{M},\mathcal{V}'}$ of size $k\sigma$. This concludes the proof. ■

6.4 Additional Experiments

Resources. All experiments were performed on Intel Xeon(R) Linux machines with 64GB of RAM. Our source code (in C++ and Python), documentation, and selected datasets are available in the code appendix.

Additional Results on the Nararajan Dimension

Lemma 4.9 in Section 4.3 can be used to estimate the Natarajan dimension of a given graph in the following manner. Two vertex-layer pairs (v,i) and (v',i') satisfy the *pairwise non-nested neighborhood* (PNN) property if $N[v,i] \nsubseteq N[v',i']$ and $N[v',i'] \nsubseteq N[v,i]$. We recall that the Natarajan dimension is lower bounded by the cardinality of any set of vertex-layer pairs satisfying the PNN property. Therefore, our objective here is to find a large set of such pairs. To this end, we construct a graph over vertex-layer pairs, called the PNN graph. We draw an edge between two pairs if they violate the PNN property, i.e., if one of the closed neighborhoods is a subset of another. We apply a greedy vertex coloring algorithm and then choose the largest subset of vertex-layer pairs assigned the same color. By definition of vertex coloring, the chosen vertex-layer pairs form an independent set in the PNN graph, which in turn implies that any vertex-layer pairs in this set satisfy the PNN property. Hence, the cardinality of this set is a lower bound on the Natarajan dimension.

The results are shown in Figure 7. Recall that the theoretical results show that with very high probability, every vertex-layer pair satisfies the PNN property, and therefore, the Natarajan dimension is $k\sigma$ w.h.p, even for the case where $\sigma = n$ for suitably large n. Our experiments suggest that this holds for even smaller graphs, say of size $n = 1000$. Secondly, our results indicate that the Natarajan dimension is close to $k\sigma$ for a large range of edge densities. In the left panel of Figure 7, we note that only for very small or very large values of edge probabilities, the set size reduces. For the extreme cases where the graph is an independent set or is a complete graph in all layers, it can be easily shown that the Natarajan dimension is σ. We observe the same behavior for increasing k. In the right panel of Figure 7, we plot separately for very small edge probabilities to observe the evolution of the lower bound from n to nk. We note that the standard deviation across replicates is very low as well (< 50).

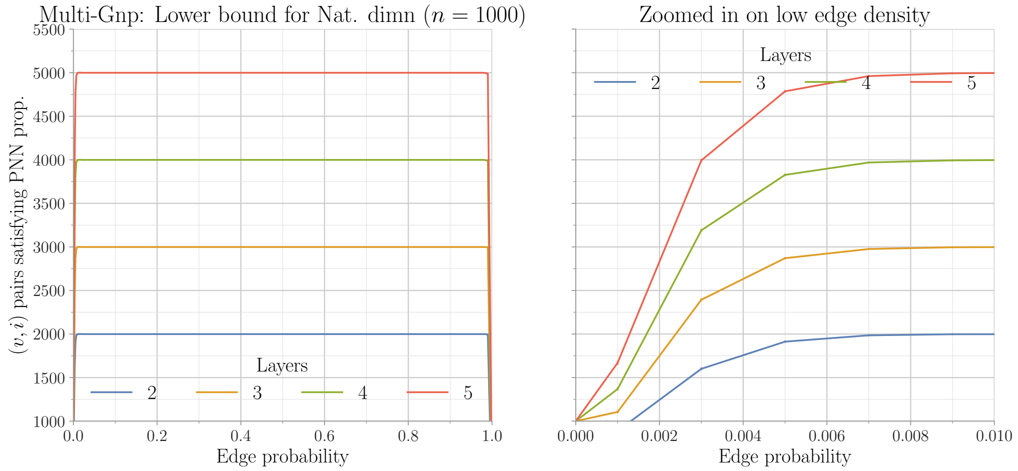


Figure 7: Experimental estimates for Natarajan dimension for Multi-Gnp graphs with a varying number of layers k and edge probability. Each graph has 1000 vertices. For each value of k and edge probability, 100 replicates were used. The maximum standard deviation across replicates is less than 50.

6.5 Additional Remarks and Discussions of the Results

Remark on the optimal sample complexity

First, let's recall the following sample complexity bounds from the paper: lower bounded by $\Omega(1/\epsilon \cdot \sigma + 1/\epsilon \cdot \log(1/\delta))$ upper bounded by $O(1/\epsilon \cdot \sigma k \cdot \log(\sigma k/\delta))$, where σ is the number of vertices with unknown interaction functions, and k is the number of layers in the graph. We now elaborate on the intuitions about the optimal sample complexity in both theory and practice.

- **Practice:** The number of layers k in real-world multilayer networks is usually a constant [24]. In that case, the lower and upper bounds differ only by a factor $O(\log(\sigma))$ (Please also see the remark in the main paper for this discussion). This suggests that for the realistic networks that we encounter, the minimum number of samples needed to learn the system will usually be at most a factor $O(\log(\sigma))$ smaller than our upper bound above.
- **Theory:** From a theoretical perspective, without additional assumptions on the sampling distribution, we believe that the factor $k\sigma$ in the sample complexity is inevitable. To see this, note that there are $k\sigma$ unknown interaction functions to be learned. An adversary may choose a distribution (unknown to the learner) such that each data point only reveals useful information to infer at most one such unknown function. Consequently, one needs at least $k\sigma$ samples to infer the full system. Based on this intuition, the optimal sample complexity would be at most a factor $O(\log(k\sigma))$ smaller than our upper bound. In general, we note that the issue of determining the optimal sample complexity of multiclass learning is a well-known open problem [17].

Remark on the difference between learning problems for single-layer vs multi-layer systems

In multilayer dynamics, a state change of a vertex can be caused by the change in the behavior of interaction functions on any of the layers; however, information regarding exactly which of the interaction functions on the layers triggered the change is **not** contained in the training set. This issue does not occur when the network has only a single layer as there will be **no** ambiguity.

To explain this difference in detail, we provide the following concrete example. The figure of this example is shown in Fig 8

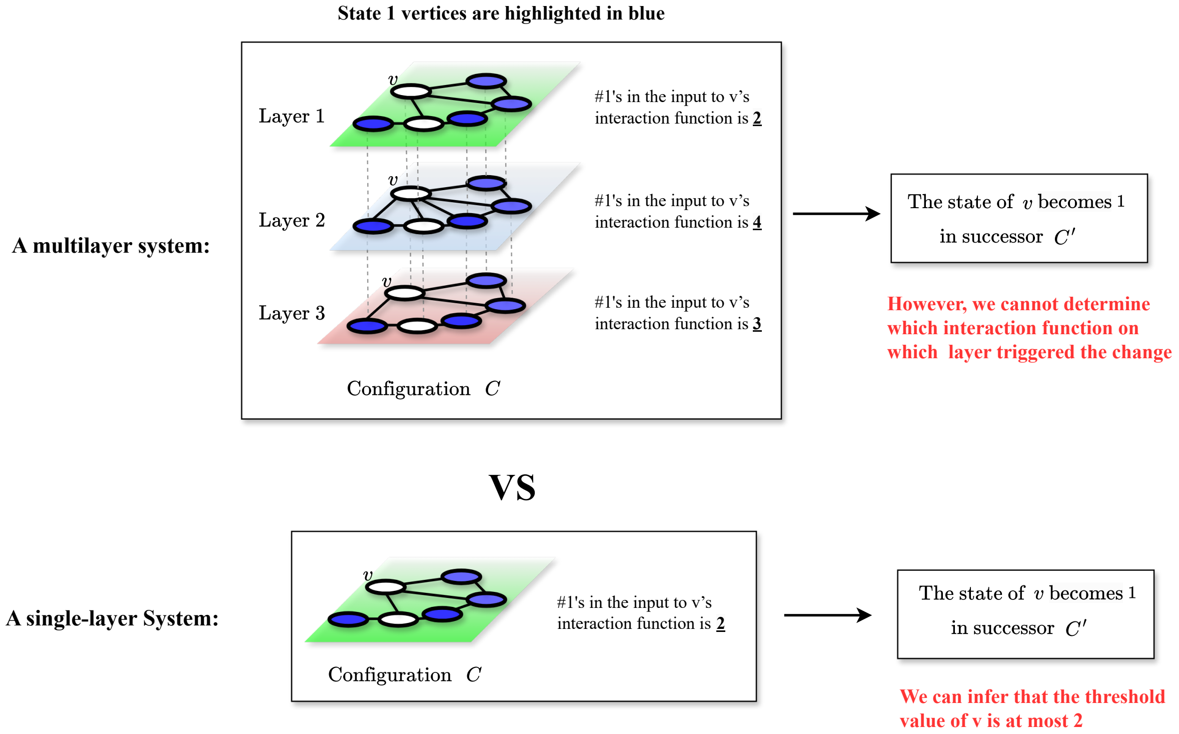

State 1 vertices are highlighted in blue
A multilayer system:
Layer 1
v
#1's in the input to v's interaction function is 2
Layer 2
v
#1's in the input to v's interaction function is 4
Layer 3
v
#1's in the input to v's interaction function is 3
Configuration C
The state of v becomes 1 in successor C'
However, we cannot determine which interaction function on which layer triggered the change
VS
A single-layer System:
v
#1's in the input to v's interaction function is 2
Configuration C
The state of v becomes 1 in successor C'
We can infer that the threshold value of v is at most 2

Figure 8: An example of learning a single-layer system vs learing multilayer-system

Consider a single-layer graph G and a multilayer graph M with 3 layers. In both graphs, we focus on a vertex v whose interaction function(s) are to be learned. Given a data point (C, C') in the training set, please recall that C' is the successor of C.

Single-layer case: For a system over the single-layer graph G, suppose that under C:

- The number of 1's in the input to v's interaction function is 2
- The state of v changes from 0 (under C) to 1 (under C').

Without ambiguity, this change of state is caused by the input value 2 to v's function; there is only one such function because there is only one layer. So, from this data point (C, C') alone, one can infer that the threshold value of v's function is at most 2. This significantly simplifies the learning process.

Multilayer case: Consider a system over the 3-layer graph M. In general, v's interaction functions (which are unknown) can be different on different layers. In this example, suppose that under C, the number of 1's in the input to v's interaction function on each layer is as follows:

- 1st layer: the number of 1's is 2

- 2nd layer: the number of 1's is 4

- 3rd layer: the number of 1's is 3

Suppose the state of v changes from 0 (under C) to 1 (under C'). We remark that all we observe from the data point is the final state of v in C', which is jointly affected by the outputs of the three unknown interaction functions. However, we do not have the actual output values of these (unknown) functions from the training set. So, the difficulty is that we do not know which of the interaction functions on the three layers triggered the change in v's state. Thus, one cannot draw a concrete conclusion on these three unknown interaction functions based only on the entry (C, C'). It requires a strategic cross-examination of samples in the entire training set. The issue becomes even more complex as the number of layers k increases.