

Supplementary information for: Limits on the computational expressivity of non-equilibrium biophysical processes

Carlos Floyd,^{1,2,*} Aaron R. Dinner,^{1,2,3} Arvind Murugan,^{2,4} and Suriyanarayanan Vaikuntanathan^{1,2,3,†}

¹*The Chicago Center for Theoretical Chemistry, The University of Chicago, Chicago, Illinois 60637, USA*

²*The James Franck Institute, The University of Chicago, Chicago, Illinois 60637, USA*

³*Department of Chemistry, The University of Chicago, Chicago, Illinois 60637, USA*

⁴*Department of Physics, The University of Chicago, Chicago, Illinois 60637, USA*

(Dated: July 2, 2025)

CONTENTS

I. Training Markov networks	2
A. Training algorithm	2
B. Variational quantities of Markov steady states	2
C. Computing the update derivatives	3
D. Nudging	4
II. Visualization of trained networks	4
III. Estimating the VC dimension for $D = 1$	4
IV. Reduced degrees of freedom from equality constraints	7
A. Equality constraint among directed tree weights	7
B. Degrees of freedom for $M = 1, D = 1$	8
C. Degrees of freedom for $M = 1, D = 2$	9
V. Sharpness of the decision boundary	14
A. Derivation of Equation 8 in the main text	14
B. Analysis of trees in the extended push-pull networks	16
C. Analysis of trees in the extended ladder networks	17
VI. Low dimensionality of trained networks	18
VII. Softmax-like form of the classification function	21
A. Connection to transformers and modern Hopfield networks	21
B. Derivation of Equation 3 in the main text	22
C. Creased landscape of $\ln Z$	23
VIII. How non-equilibrium driving allows expressivity	24
A. Expressivity from the parameters E_j , B_{ij} , and F_{ij}	24
B. Three examples	27
IX. Mutual information formulation of the computation	29
A. Defining the mutual information	29
B. Theory for optimality of one-hot encoding	30
X. Incorporating bimolecular reactions	31
Supplementary references	33

* csfloyd@uchicago.edu

† svaikunt@uchicago.edu

I. TRAINING MARKOV NETWORKS

A. Training algorithm

The results in this paper are obtained by training Markov networks using an algorithm based on Refs. 1–4 which approximates gradient descent on a loss landscape. Broadly, this approach involves presenting a labeled example \mathbf{F} to the system, obtaining its free output $\boldsymbol{\pi}(\boldsymbol{\theta})$ under its current set of learnable parameters $\boldsymbol{\theta}$, nudging the system toward the desired output $\boldsymbol{\pi}'$ for that label, and then updating $\boldsymbol{\theta}$ according the different outputs in the free and nudged configurations. Throughout this section we suppress the dependence of quantities on \mathbf{F} . The update rule

$$\Delta\boldsymbol{\theta} = \frac{\partial\mathcal{L}[\boldsymbol{\pi}(\boldsymbol{\theta});\boldsymbol{\theta}]}{\partial\boldsymbol{\theta}} - \frac{\partial\mathcal{L}[\boldsymbol{\pi}';\boldsymbol{\theta}]}{\partial\boldsymbol{\theta}} \quad (1)$$

involves a variational quantity $\mathcal{L}[\mathbf{p};\boldsymbol{\theta}]$ which is minimized at the steady state $\lim_{t\rightarrow\infty}\mathbf{p}(t) = \boldsymbol{\pi}$ of the network. This algorithm has recently been used to train physical systems to act like neural networks [1, 2]. Typical physical systems at or near equilibrium minimize a variational quantity, such as an energy function (like the elastic energy in a spring networks) or dissipation function (like the total dissipated power in resistor networks). A useful feature of these variational quantities is that their derivatives with respect to $\boldsymbol{\theta}$ sometimes decompose into functions which are local to nodes in the network, meaning that Equation 1 can be applied without global information; each node only requires information in from its immediate neighborhood in the network to update its value.

We find below that while variational quantities \mathcal{L} for the steady states of far-from-equilibrium Markov networks can be defined and used in Equation 1, their derivatives with respect to $\boldsymbol{\theta}$ do not decompose exactly into local quantities. As a result, using this approximate gradient descent technique to train Markov networks currently has no inherent advantage over exact gradient descent; both approaches require numerically estimating similar derivatives. Despite this, we describe the set-up for using this algorithm here because it may be possible in future work to obtain approximations to the derivatives $\partial_{\boldsymbol{\theta}}\mathcal{L}$ in terms of easily computable local quantities, which would allow for a highly scalable training approach conducive to large system sizes. We note that other training approaches may be possible, such as genetic algorithms [5, 6] or automatic differentiation [7]. Rather than compare each approach, our focus in this paper is on the underlying limitations of computation by the trained networks.

B. Variational quantities of Markov steady states

We describe two variational quantities $\mathcal{L}[\mathbf{p};\boldsymbol{\theta}]$ which are minimized by the steady state $\mathbf{p} = \boldsymbol{\pi}(\boldsymbol{\theta})$. The first is the Kullback-Leibler (KL) divergence between \mathbf{p} and $\boldsymbol{\pi}$:

$$\mathcal{L}^{\text{KL}}[\mathbf{p};\boldsymbol{\theta}] \equiv \sum_k p_k \ln \frac{p_k}{\pi_k(\boldsymbol{\theta})}. \quad (2)$$

It can be shown that \mathcal{L}^{KL} acts as a Lyapunov function under the dynamics $\dot{\mathbf{p}} = \mathbf{W}\mathbf{p}$, such that $\partial_t\mathcal{L}[\mathbf{p}(t);\boldsymbol{\theta}] \leq 0$ along any approach toward steady state [8]. At steady state, \mathcal{L}^{KL} is minimized to zero.

While the variational nature of \mathcal{L}^{KL} can be established through its connection to a dynamical Lyapunov functional, it is of interest to consider an alternative route to a variational quantity through analogy to a thermodynamic functional, namely, the Helmholtz free energy. To make this analogy, we first note that Equation 1 of the main text can be expressed in the Boltzmann-like form

$$\pi_i(\boldsymbol{\theta}) = \frac{e^{-\Phi_i(\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})} \quad (3)$$

where

$$\Phi_i(\boldsymbol{\theta}) = -\ln \sum_{T^\alpha \in \mathcal{T}} w(T_i^\alpha; \boldsymbol{\theta}) \quad (4)$$

is the “non-equilibrium potential” and

$$Z(\boldsymbol{\theta}) = \sum_{k=1}^{N_n} e^{-\Phi_k(\boldsymbol{\theta})} \quad (5)$$

is the “partition function.” As the Boltzmann distribution minimizes the Helmholtz free energy, i.e., it maximizes the entropy subject to the constraint of normalization and an average energy, we can write a variational “free energy” which Equation 3 minimizes:

$$\mathcal{L}^{\text{FE}}[\mathbf{p}; \boldsymbol{\theta}] = \sum_k p_k \ln p_k + \sum_k \Phi_k(\boldsymbol{\theta}) p_k + \lambda(\boldsymbol{\theta}) \left(\sum_k p_k - 1 \right). \quad (6)$$

The Lagrange multiplier $\lambda(\boldsymbol{\theta})$ enforces the normalization of p_k . The derivative of \mathcal{L}^{FE} with respect to p_k is

$$\frac{\partial \mathcal{L}^{\text{FE}}}{\partial p_k} = \ln p_k + 1 + \lambda(\boldsymbol{\theta}) + \Phi_k(\boldsymbol{\theta}). \quad (7)$$

The minimizer $\boldsymbol{\pi}(\boldsymbol{\theta}) = \operatorname{argmin}_{\mathbf{p}} \mathcal{L}^{\text{FE}}[\mathbf{p}; \boldsymbol{\theta}]$ will satisfy $\frac{\partial \mathcal{L}}{\partial p_k} = 0$, and it can be shown that the solution of this is Equation 3, with the interpretation that $\lambda(\boldsymbol{\theta}) = \ln Z(\boldsymbol{\theta}) - 1$. Inserting this, we can rewrite \mathcal{L} as

$$\begin{aligned} \mathcal{L}^{\text{FE}} &= \sum_k p_k \ln p_k + \sum_k p_k (\Phi_k(\boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta})) - \sum_k p_k + 1 - \ln Z(\boldsymbol{\theta}) \\ &= \sum_k p_k \ln p_k + \sum_k p_k \ln \pi_k(\boldsymbol{\theta}) - \sum_k p_k + 1 - \ln Z(\boldsymbol{\theta}) \\ &= \sum_k p_k \ln \frac{p_k}{\pi_k(\boldsymbol{\theta})} - \sum_k p_k + 1 - \ln Z(\boldsymbol{\theta}). \end{aligned} \quad (8)$$

We see that

$$\mathcal{L}^{\text{FE}} = \mathcal{L}^{\text{KL}} - \sum_k p_k + 1 - \ln Z(\boldsymbol{\theta}), \quad (9)$$

which means that $\partial_{\boldsymbol{\theta}} (\mathcal{L}^{\text{FE}} - \mathcal{L}^{\text{KL}})$ depends only on $\boldsymbol{\theta}$, not \mathbf{p} . As a result, the update rule in Equation 1 leads to the same result for both of these variational quantities:

$$\Delta \boldsymbol{\theta} = - \sum_k \frac{\pi_k(\boldsymbol{\theta})}{\pi_k(\boldsymbol{\theta})} \frac{\partial \pi_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \sum_k \frac{\pi'_k(\boldsymbol{\theta})}{\pi_k(\boldsymbol{\theta})} \frac{\pi_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_k \frac{\pi'_k(\boldsymbol{\theta})}{\pi_k(\boldsymbol{\theta})} \frac{\partial \pi_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad (10)$$

where the first term vanishes due to normalization of $\pi_k(\boldsymbol{\theta})$.

C. Computing the update derivatives

The derivatives $\partial_{\boldsymbol{\theta}} \pi_k(\boldsymbol{\theta})$ have received recent attention [9–12]. For $\theta_m \in \{E_j\}_{j=1}^{N_n}$, the derivative is simply [9]

$$\frac{\partial \pi_k(\boldsymbol{\theta})}{\partial E_j} = \begin{cases} -\pi_k(1 - \pi_k) & \text{if } j = k \\ \pi_k \pi_j & \text{if } j \neq k. \end{cases} \quad (11)$$

Inserting this into Equation 10 gives, after simplification, the local update rule

$$\Delta E_j = \pi_j(\boldsymbol{\theta}) - \pi'_j. \quad (12)$$

For parameters $\theta_m \in \{B_{ij}\}_{ij \in \mathcal{E}}$ and $\theta_m \in \{F_{ij}\}_{ij \in \mathcal{E}}$, the derivatives do not simplify as cleanly. Recent work has shown how the derivatives $\partial_{\boldsymbol{\theta}} \pi_k(\boldsymbol{\theta})$ with respect to these parameters can be bounded in magnitude [9, 10], but they do not apparently admit simple expressions (but see Ref. 11 for expressions in terms of matrix minors of \mathbf{W}). As a result, we resort here to numerical approximation of these derivatives using finite differences, by evaluating

$$\frac{\partial \pi_k(\boldsymbol{\theta})}{\partial \theta_m} \approx \frac{\pi_k(\theta_m + \delta) - \pi_k(\theta_m - \delta)}{2\delta} \quad (13)$$

with $\delta = 10^{-3}$ and the remaining parameters θ_n , $n \neq m$, fixed. We numerically compute $\boldsymbol{\pi}$ as the normalized leading eigenvector of \mathbf{W} .

D. Nudging

To apply nudging, we first present an example of an input pattern \mathbf{F} and find the steady state $\boldsymbol{\pi}(\mathbf{F}; \boldsymbol{\theta})$. If we want this input to produce high probability at node j , and low probability at node k , for example, then we nudge $E'_j \leftarrow E_j - \epsilon$ (which will have the effect of raising π_j) and $E'_k \leftarrow E_k + \epsilon$ with $\epsilon \sim 1$. We then recompute the nudged steady state $\boldsymbol{\pi}' = \boldsymbol{\pi}(\mathbf{F}; \boldsymbol{\theta}')$ under these new parameters. Note that we are not clamping the steady-state values through this nudging procedure, only indirectly encouraging it to be higher or lower by adjusting the $\{E_j\}_{j=1}^{N_n}$ parameters. To solve the classification problem, we encourage node i to be high and other nodes $j \in \mathcal{O}$ in the output set to be low when an example from class i is presented. After presenting one example during the n^{th} training iteration, we nudge and then apply the update

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n + \eta \Delta \boldsymbol{\theta}^n \quad (14)$$

with a scalar learning rate $\eta \sim 1$. We train for 10^3 iterations or until convergence.

II. VISUALIZATION OF TRAINED NETWORKS

Here we illustrate the training process and resulting network parameters for two graphs trained to solve a binary classification problem. In Figure 1 we show a fully-connected network with six nodes. The training data and learned decision boundaries are shown in Figure 1A, and the convergence during training is shown in Figure 1B. Figure 1C shows the assigned input edges and output nodes, and it further depicts the learned parameters $\boldsymbol{\theta}$ that resulted from training. One can see by inspection how the learned edge weights allow the network to shunt probability toward node 1 when a force \mathbf{F}^1 from in class 1 is presented, and how it directs probability toward node 2 when a force \mathbf{F}^2 from in class 2 is presented. Figure 1D depicts the steady state of the network in each of these two cases, confirming that the classification task is successfully solved. In Figure 2 we show the same information for a random network.

III. ESTIMATING THE VC DIMENSION FOR $D = 1$

In previous work we showed that if all other parameters are held fixed, then the partial derivative $\partial\pi_k/\partial F_{ij}$ has a fixed sign across the entire range of F_{ij} [12]. This imposes a limitation on the expressivity of the possible decision boundaries. To overcome this limitation, we can consider a new parameterization in which the input variables affect multiple edges simultaneously. We denote by F_a a component of the input vector \mathbf{F} and write

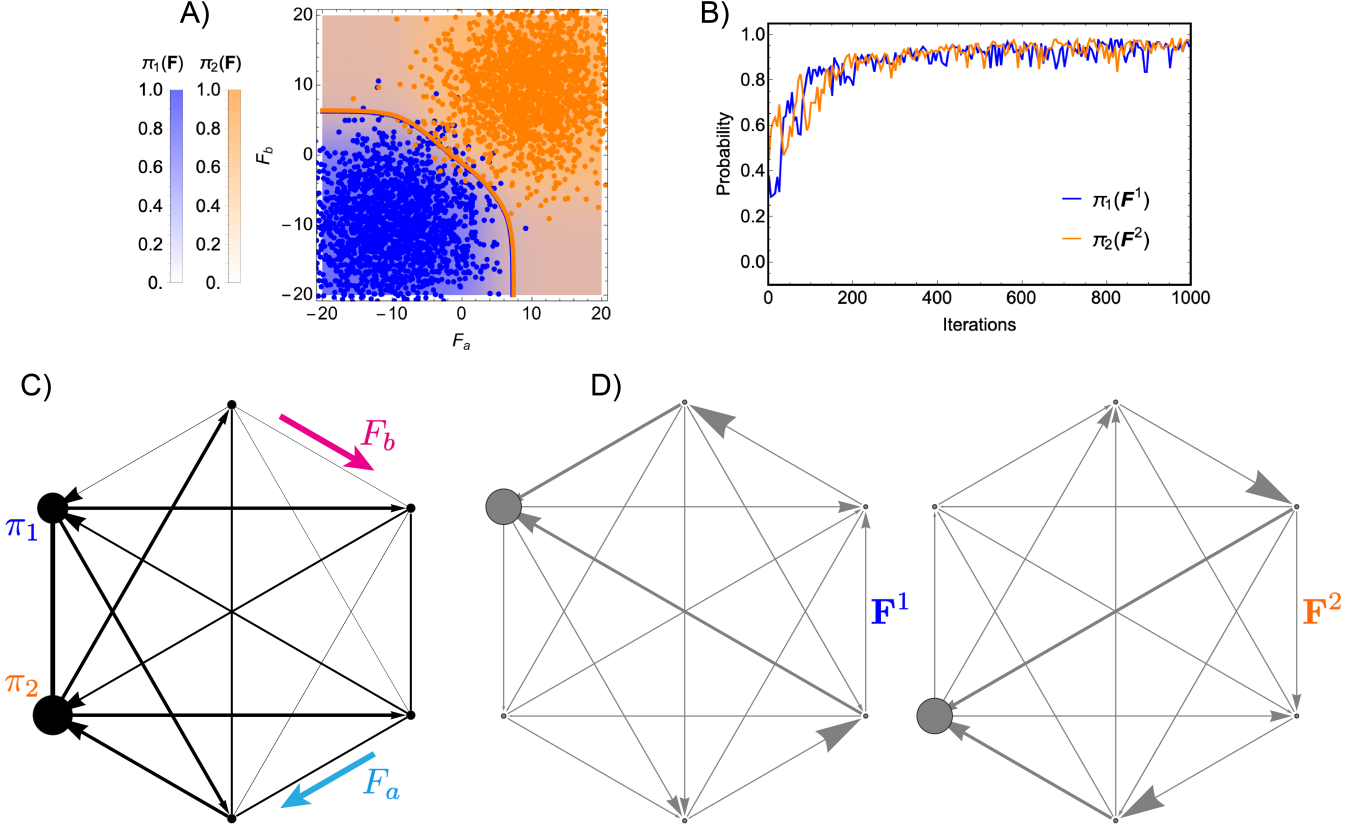
$$\frac{\partial\pi_k}{\partial F_a} = \sum_{ij \in \mathcal{E}_a} \frac{\partial\pi_k}{\partial \tilde{F}_{ij}} \frac{\partial \tilde{F}_{ij}}{\partial F_a} \quad (15)$$

where \mathcal{E}_a denotes the set of edges affected by the input variable x_a . We assume that each of the D components of the input vector affect the same number M of edges, i.e., $|\mathcal{E}_a| = M$, $a \in \mathcal{A}$, so that the total number of affected edges is MD . We model the input F_a as directly adding to the learned value of F_{ij} so that $\tilde{F}_{ij} = F_{ij} + F_a$ and $\partial \tilde{F}_{ij} / \partial F_a = 1$. Unlike the partial derivative $\partial\pi_k/\partial F_{ij}$, the partial derivative $\partial\pi_k/\partial F_a$ is not constrained to have a fixed sign because it is the sum of several terms whose signs can differ, and, furthermore, each term $\partial\pi_k/\partial \tilde{F}_{ij}$ can change sign as F_a changes because \tilde{F}_{ij} is not the only parameter being varied.

How many times can the sign of $\partial\pi_k/\partial F_a$ change as a function of M ? We consider here the case $D = 1$, meaning that there is one input variable F which affects M edges. If $M = 1$, then we can write π_k as (cf. Equation 80 below)

$$\pi_k = \frac{\zeta_1^k y + \zeta_{-1}^k y^{-1} + \zeta_0^k}{\zeta_1 y + \zeta_{-1} y^{-1} + \zeta_0}. \quad (16)$$

This is a rational polynomial in $y = e^{F/2} > 0$. For general M , the numerator and denominator can contain tree factors whose dependence on F ranges from $e^{-MF/2}$, $e^{-(M-1)F/2}$, \dots , $e^{(M-1)F/2}$, $e^{MF/2}$ depending on if a tree contains all of the affected edges running with (incurring $e^{MF/2}$) or against (incurring a factor $e^{-MF/2}$) the positive orientation, or if there is a mixture. Multiplying so that the lowest order term is proportional to $y^0 = 1$, the numerator and



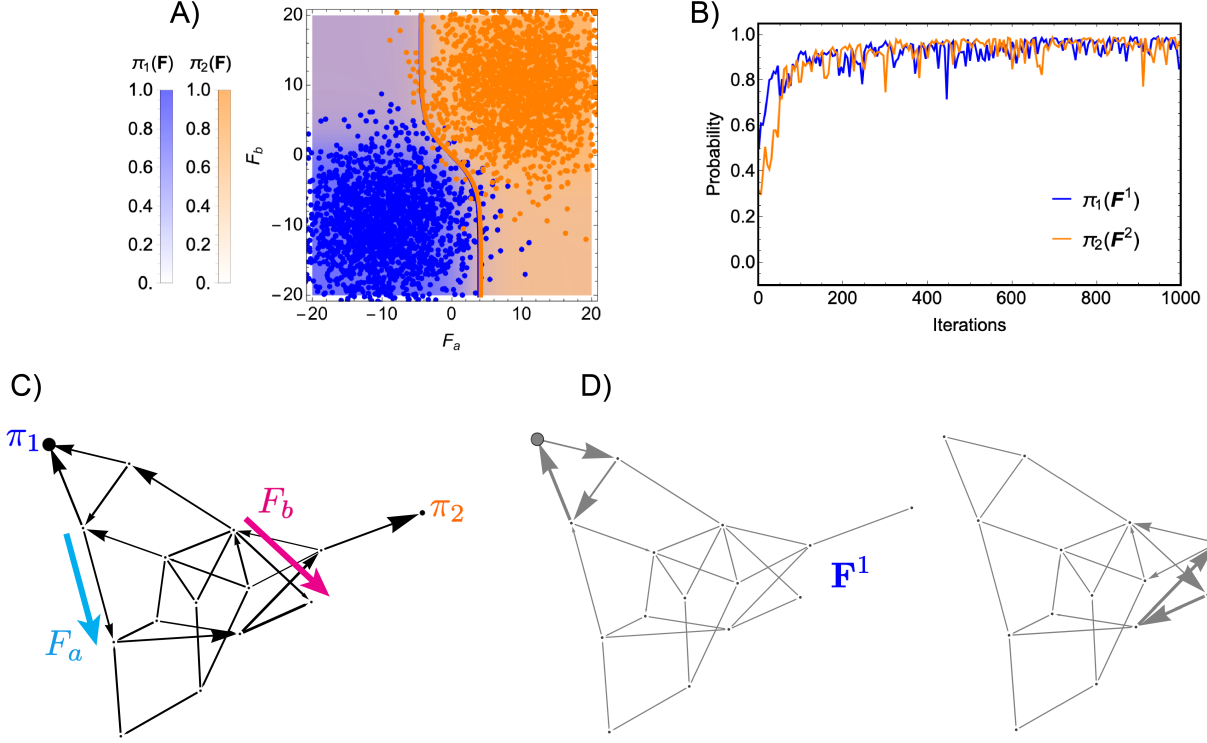
Supplementary Figure 1. Visualization of training a fully-connected 6 node Markov network for binary classification. A) Plot of the learned classification functions $\pi_1(\mathbf{F})$ and $\pi_2(\mathbf{F})$ shown as colored density plots over the input force space. On top of this, scatter plots show the data set, colored by assigned class, which was used to train the network. Solid lines show the contour $\pi_1(\mathbf{F}) = 1/2$ and $\pi_2(\mathbf{F}) = 1/2$. B) Plot illustrating convergence during training. Convergence is measured as the steady-state probability $\pi_\rho(\mathbf{F}^\rho)$ of node ρ when presented with examples \mathbf{F}^ρ from its assigned class. C) Visualization of the learned parameters in the network. The output nodes and input edges are labeled as in Figure 1E of the main text. The node sizes are a linear function of e^{-E_j} . The edge widths are a linear function of B_{ij} , so that smaller values of B_{ij} (faster edge rates) are thicker. The arrowhead directions depend on the sign of F_{ij} and their sizes are a linear function of $|F_{ij}|$. D) Visualization of the steady states for this network when presented with inputs from class 1 (left) and from class 2 (right). The node sizes are a linear function of π_i . The edge widths are a linear function of the frenesy $\pi_j W_{ij} + \pi_i W_{ji}$. The arrowhead directions depend on the sign of the probability flux $\pi_j W_{ij} - \pi_i W_{ji}$ and their sizes are a linear function of the flux magnitude.

denominator of π_k for general M will be polynomials in y of degree up to $2M$. We write this as

$$\pi_k = \frac{\sum_{m=0}^{2M} \zeta_m^k y^m}{\sum_{m=0}^{2M} \bar{\zeta}_m y^m} \quad (17)$$

where all coefficients ζ_m^k and $\bar{\zeta}_m$ are non-negative.

In Ref. 13, it is proven that a rational function $f(y)/g(y)$, where $f(y)$ and $g(y)$ are polynomials of degrees n and m , respectively, and all coefficients are non-negative, can have at most $\min(n, m)$ positive ($y > 0$) critical points if $n \neq m$, and at most $m - 1$ if $n = m$. The proof involves analyzing the Wronskian $W(f, g) = fg' - gf'$, which appears in the numerator of the derivative $(f/g)'$. Using Rolle's theorem, the authors establish that between any two positive roots of $W(f, g)$ there must be a positive root of $W(f', g')$. They then apply Descartes' rule of signs to argue that the number of positive roots of f , g , and their derivatives is constrained by the signs of their coefficients. By iteratively applying this reasoning to successive derivatives of f and g , they show that the process must eventually terminate, yielding the stated bounds on the number of critical points. Note that this result for rational functions with positive coefficients is non-trivial, since the degree of $W(f, g)$ is $n + m - 1$ and one naively expects this larger number of roots. Incidentally, they also show that the number of positive inflection points of such a rational function cannot exceed



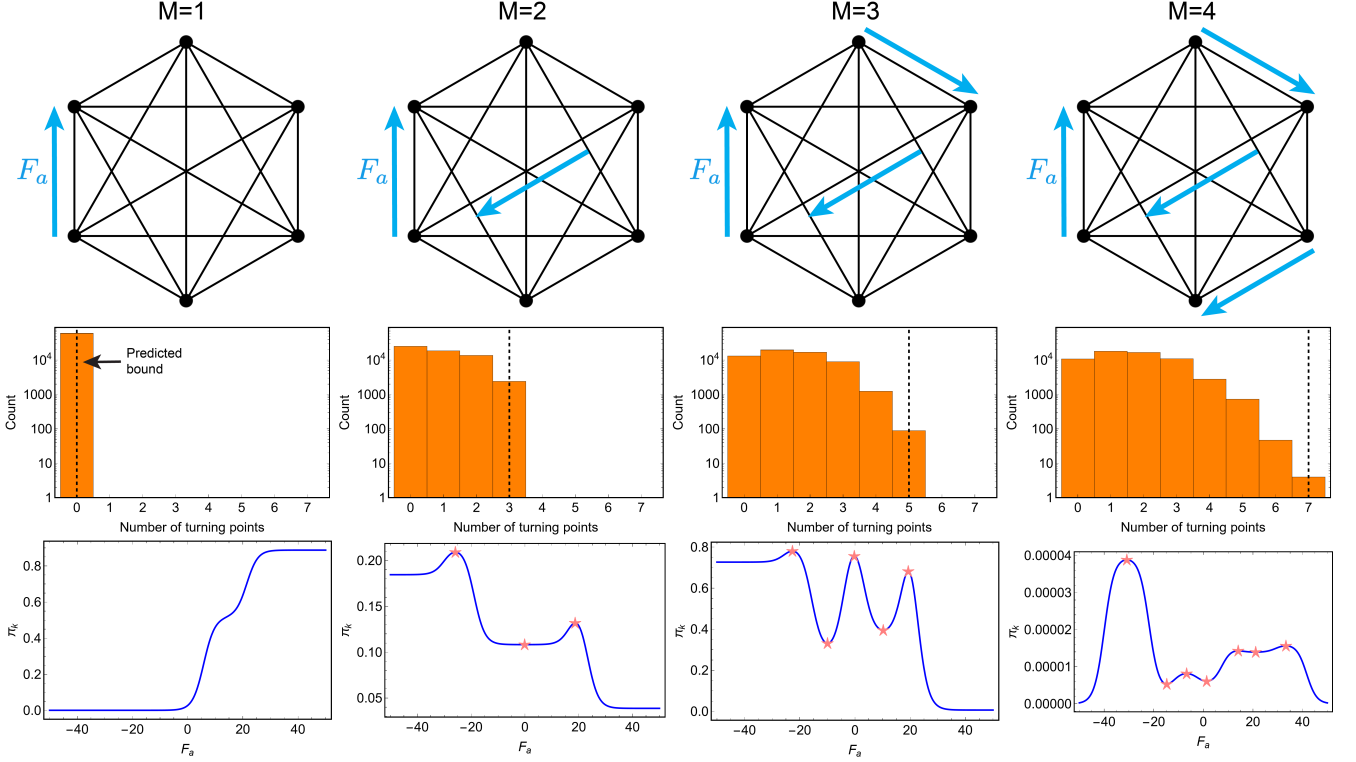
Supplementary Figure 2. Visualization of training a random Markov network for binary classification. This graph is the same as used in Figures 1E, 3, 4, and 5 in the main text. See the caption for Figure 1 above for a description of the plots in this figure.

$2m - 1$ when $n = m$, with other bounds derived for the case $n \neq m$. In our case, we have $n = m = 2M$, so the number of positive critical points is at most $2M - 1$, and the number of positive inflection points is at most $4M - 1$.

For the case $M = 1$, our monotonicity constraint imposes that there are no roots of $\partial\pi_k/\partial F$, which falls below the allowed limit of $2M - 1 = 1$ for general rational polynomials of degree $2M$. As detailed in Ref. 12, this fact for $M = 1$ is due to a non-trivial set of relations satisfied by the terms $\zeta_1^k/\bar{\zeta}_1$, $\zeta_{-1}^k/\bar{\zeta}_{-1}$, and $\zeta_0^k/\bar{\zeta}_0$ in Equation 16. For $M > 1$, however, there is no monotonicity constraint and the number of sign changes may saturate the algebraic bound $2M - 1$.

In SI Figure 3 we numerically verify the predicted scaling of the maximal number of turning points of π_k with M . To do this we consider a fully connected graph with six nodes and randomly sample parameters E_i , B_{ij} , F_{ij} in the range $[-10, 10]$. We draw 10,000 such samples for values of M up to $M = 4$, fixing the chosen input edges for each M . We then compute the response $\pi_k(F_a)$ for each node in the graph, giving 60,000 curves for each value of M , and we find the number of critical points for each curve. Histograms of this quantity for each M illustrate saturation of the predicted bound as well as a general increase of the typical number of critical points as M increases. For larger M , parameters saturating the bound $2M - 1$ become rarer and harder to find through random search, but we expect the bound continues to be saturated. In summary, for $D = 1$, the number of times the sign of $\partial\pi_k/\partial F$ can change as a function of F is 0 for $M = 1$ and at most $2M - 1$ for $M > 1$.

This result informs about the Vapnik-Chervonenkis (VC) dimension of the classifier $\pi_k(F)$. The VC dimension is the largest number N_{VC} of points that the classifier can shatter for any of the $2^{N_{VC}}$ binary assignments of labels to the points. For $D = 1$, N_{VC} can be estimated by determining the number of points which can be separated in the most challenging of the $2^{N_{VC}}$ label assignments. The most challenging arrangement is the one in which each point's label differs from those of its nearest neighbors. To separate such a set of labeled points, $\partial\pi_k/\partial F$ must be able to change signs at least $N_{VC} - 2$ times. For the classifier $\pi_k(F)$, then, the VC dimension is 2 for $M = 1$ and $2M + 1$ for $M > 1$. This is consistent with the more general estimate for any D based on the theorem by Dudley discussed in the main text.



Supplementary Figure 3. Numerical evidence that the predicted bound on the number of turning points is saturated for a fully connected graph with six nodes. For each value of M we apply the input F_a along the edges indicated in the top row. We then sample 10,000 random values of E_i , B_{ij} , F_{ij} and determine the number of tuning points in the curve $\pi_k(F_a)$ for each node k . Histograms of this quantity are in the middle row, where the black dashed line indicates the predicted bound. The bottom row shows representative examples of the curves $\pi_k(F_a)$ for samples saturating the bound, with turning points labeled as pink stars.

IV. REDUCED DEGREES OF FREEDOM FROM EQUALITY CONSTRAINTS

A. Equality constraint among directed tree weights

In this section we present analytical and numerical arguments based on counting degrees of freedom with constraints which illustrate why certain multi-class classification tasks cannot be solved by networks with $M = 1$. The main result we rely on is an equality we derived in previous work [12] that relates the products of directed tree weights conditioned on containing edge $i \leftarrow j$, $j \leftarrow i$, or neither of these, for trees rooted on nodes i , j , and an arbitrary node p . We first describe this equality and then use it to show that two-class classification is possible for $D = 1$ but, somewhat surprisingly, four-class classification is not possible for $D = 2$. We leave a general treatment for arbitrary M and D to future work.

We let ζ_1^p denote the sum over all directed tree weights for directed trees rooted at node p and which include the directed edge $i \leftarrow j$, ζ_{-1}^p denote the same for directed trees which include the directed edge $j \leftarrow i$, and ζ_0^p denote the same for directed trees which do not include either $i \leftarrow j$ or $j \leftarrow i$. Previous work of ours shows that [12]

$$\frac{\zeta_1^p}{W_{ij}} \zeta_0^i + \frac{\zeta_{-1}^p}{W_{ji}} \zeta_0^j = \frac{\zeta_1^i}{W_{ij}} \zeta_0^p = \frac{\zeta_{-1}^j}{W_{ji}} \zeta_0^p. \quad (18)$$

Applying this for $p = i$ or $p = j$ yields the equations

$$\frac{\zeta_1^i}{W_{ij}} = \frac{\zeta_{-1}^j}{W_{ji}} \quad (19)$$

and

$$\zeta_{-1}^i = \zeta_1^j = 0. \quad (20)$$

In Ref. 12 we proved Equation 18 by establishing a one-to-one correspondence between the product of tree weights for each pair of trees in the products of sums on the left hand and right hand sides of the equality. We established the correspondence using a recent theoretical tool called tree surgery, introduced in Ref. 9. The special case Equation 19 implies that the sum of all tree weights for directed trees rooted on node i and which pass through the edge $i \leftarrow j$ is the same as that for directed trees rooted on node j and which pass through edge $j \leftarrow i$, after dividing out W_{ij} and W_{ji} . Equation 20 simply says that no directed tree can be rooted on node i and pass through the edge $j \leftarrow i$; similarly no tree can be rooted on node j and pass through the edge $i \leftarrow j$.

B. Degrees of freedom for $M = 1, D = 1$

1. Counting degrees of freedom

We first explicitly write out Equation 2 in the main text for $M = 1$ and $D = 1$,

$$\pi_p = \frac{\zeta_1^p e^{F_a/2} + \zeta_{-1}^p e^{-F_a/2} + \zeta_0^p}{\zeta_1 e^{F_a/2} + \zeta_{-1} e^{-F_a/2} + \zeta_0}, \quad (21)$$

and ask how many degrees of freedom are accessible among the $3N_n$ parameters in the set $\mathcal{S}^{\text{orig}} \equiv \{\{\zeta_s^p\}_{s \in \{-1,0,1\}}\}_{p=1}^{N_n}$.

First, by Equation 20 we know that $\zeta_{-1}^i = \zeta_1^j = 0$, where i and j are the nodes adjacent to the input edge. This reduces our degrees of freedom by 2. Next, we note that whether an equation counts as a constraint on the number of degrees of freedom in a set depends on whether the only functions appearing in the equation come from this set. If they do, then the equation acts as a constraint and reduces the number of degrees of freedom in the set, but if there are functions not found in the set which appear in the equation, then these unaccounted for functions may be independently tuned and prevent the equation from constraining the number of degrees of freedom in the set.

The edge rates W_{ij} and W_{ji} appear in Equations 18 and 19 but are not contained in $\mathcal{S}^{\text{orig}}$. To obtain constraints, we rearrange Equation 19 as

$$\frac{W_{ij}}{W_{ji}} = \frac{\zeta_1^i}{\zeta_{-1}^j} \quad (22)$$

and, noticing that the dependence on W_{ij} and W_{ji} in Equation 18 occurs only through the ratio W_{ij}/W_{ji} , we substitute Equation 22 into each instance of Equation 18:

$$\zeta_1^p \zeta_0^i + \frac{\zeta_1^i}{\zeta_{-1}^j} \zeta_{-1}^p \zeta_0^j = \zeta_1^i \zeta_0^p \quad \forall p \neq i, j. \quad (23)$$

There are $N_n - 2$ instances of this equation for each $p \neq i, j$, and each instance represents a constraint on the degrees of freedom in $\mathcal{S}^{\text{orig}}$. Accounting for the 2 functions which are zero, there are thus $3N_n - 2 - (N_n - 2) = 2N_n$ degrees of freedom in $\mathcal{S}^{\text{orig}}$.

2. Classification of two classes

We now ask how this type of counting argument can be used to assess whether it is possible to classify two classes for the case $M = 1, D = 1$. We assume here that class 1 lies in the region $F_a > 0$ and class 2 in the region $F_a < 0$. We can place a condition for solving this classification task on the limiting values of π_p in each of these spaces. Specifically, we require that

$$\lim_{F_a \rightarrow \infty} \pi_1 > \lim_{F_a \rightarrow \infty} \pi_p \quad \forall p \neq 1 \quad (24)$$

$$\lim_{F_a \rightarrow -\infty} \pi_2 > \lim_{F_a \rightarrow -\infty} \pi_p \quad \forall p \neq 2. \quad (25)$$

It is straightforward to show that these inequalities in turn place the following conditions on the functions

$$\zeta_1^1 > \zeta_1^p \forall p \neq 1 \quad (26)$$

$$\zeta_{-1}^2 > \zeta_{-1}^p \forall p \neq 2. \quad (27)$$

These are $2N_n - 2$ inequalities which must be satisfied by the $2N_n$ functions in $\mathcal{S}^{\text{ineq}} \equiv \{\{\zeta_s^p\}_{s \in \{-1,1\}}\}_{p=1}^{N_n} \subset \mathcal{S}^{\text{orig}}$. At first glance, because $2N_n > 2N_n - 2$, we appear to have enough degrees of freedom to satisfy all of the inequalities. It will turn out that we do have enough, but the situation is slightly more complicated because we have not yet accounted for the constraints which exist among the functions in $\mathcal{S}^{\text{ineq}}$.

First, the functions ζ_{-1}^i and ζ_1^j are zero, which trivially satisfies the inequalities in which they appear, assuming that $i \neq 2$ and $j \neq 1$. We can thus subtract 2 from the number of non-trivial inequalities which remain, bringing the number to $2N_n - 4$, and which need to be satisfied by the remaining $2N_n - 2$ non-zero functions.

Additionally, the inequalities in Equations 26 and 27 cannot depend on W_{ij} or W_{ji} . This is because Equation 26 cannot include W_{ji} as a factor (since all trees pass through $i \leftarrow j$), and both sides contain W_{ij} as a common factor so that it divides out. Similar reasoning holds for Equation 27. We may thus treat W_{ij} and W_{ji} as arbitrary constants when counting remaining degrees of freedom, and we no longer need to use Equation 19 to eliminate W_{ij} or W_{ji} where they appear.

Considering the case $N_n = 2$, there are 2 non-zero functions in $\mathcal{S}^{\text{ineq}} = \{\zeta_1^i, \zeta_{-1}^j, 0, 0\}$, and there is only 1 degree of freedom among these because of Equation 19. This 1 degree of freedom is sufficient to satisfy the only requirement for two-class classification, which is that ζ_1^i and ζ_{-1}^j are both non-zero. Both non-zero functions in the set $\mathcal{S}^\circ \equiv \mathcal{S}^{\text{orig}} \setminus \mathcal{S}^{\text{ineq}} = \{\zeta_0^i, \zeta_0^j\}$ are degrees of freedom because no equalities constrain them.

For $N_n > 2$, every additional node p introduces 2 new functions $\{\zeta_{\pm 1}^p\}$ in $\mathcal{S}^{\text{ineq}}$ and 1 new function $\{\zeta_0^p\}$ in \mathcal{S}° . Equation 18 can be used to solve for ζ_0^p and eliminate it as a degree of freedom. Although this equation also involves the 2 elements $\{\zeta_0^i, \zeta_0^j\} \subset \mathcal{S}^\circ$, the values of these functions do not affect the inequality conditions and we may use the 2 degrees of freedom which they represent to set their values arbitrarily. Then Equation 18 can be used to fix ζ_0^p so that the 2 functions $\{\zeta_{\pm 1}^p\}$ remain as degrees of freedom. We therefore have 1 degree of freedom in $\mathcal{S}^{\text{ineq}}$ for $N_n = 2$ and 2 degrees of freedom for every node beyond this, so that the total number of degrees of freedom in this set (having treated W_{ij} and W_{ji} as fixed) is $1 + 2(N_n - 2) = 2N_n - 3$. Because this is greater than the number of non-trivial inequalities to satisfy, two-class classification is possible for any N_n .

3. Numerical verification

To numerically verify these counting arguments, we form the Jacobian matrices $J^{\mathcal{S}^{\text{orig}}}$, whose elements are the derivatives of functions in $\mathcal{S}^{\text{orig}}$ with respect to all edge rates, and $J^{\mathcal{S}^{\text{ineq}}}$, whose elements are the derivatives of functions in $\mathcal{S}^{\text{ineq}}$ with respect to all edge rates except W_{ij} or W_{ji} . The rank of these matrices indicates the corresponding number of degrees of freedom in the arguments presented above. Numerical evaluation for fully connected graphs with increasing numbers of nodes confirms that $\text{rank}(J^{\mathcal{S}^{\text{orig}}}) = 2N_n$ and $\text{rank}(J^{\mathcal{S}^{\text{ineq}}}) = 2N_n - 3$.

C. Degrees of freedom for $M = 1$, $D = 2$

1. Counting degrees of freedom

We next add an additional input edge kl and explicitly write out Equation 2 in the main text for $M = 1$ and $D = 2$:

$$\pi_p = \frac{\zeta_{1,1}^p e^{F_a/2} e^{F_b/2} + \zeta_{-1,1}^p e^{-F_a/2} e^{F_b/2} + \zeta_{-1,-1}^p e^{-F_a/2} e^{-F_b/2} + \zeta_{1,-1}^p e^{F_a/2} e^{-F_b/2} + \dots}{\bar{\zeta}_{1,1} e^{F_a/2} e^{F_b/2} + \bar{\zeta}_{-1,1} e^{-F_a/2} e^{F_b/2} + \bar{\zeta}_{-1,-1} e^{-F_a/2} e^{-F_b/2} + \bar{\zeta}_{1,-1} e^{F_a/2} e^{-F_b/2} + \dots}, \quad (28)$$

where the ellipses represent terms that are less than second order in $e^{F_a/2}$ and $e^{F_b/2}$. The coefficient $\zeta_{s,t}^p$ represents the sum of all directed tree weights for trees rooted at node p and which contain the input F_a (along edge $i \leftarrow j$) in its $s \in \{-1, 0, 1\}$ orientation and the input F_b (along edge $k \leftarrow l$) in its $t \in \{-1, 0, 1\}$ orientation. We ask how many degrees of freedom are accessible among the $9N_n$ in functions in the set $\mathcal{S}^{\text{orig}} \equiv \{\{\zeta_{s,t}^p\}_{s,t \in \{-1,0,1\}}\}_{p=1}^{N_n}$ by varying all the edge rates.

To understand how many constraints we can write down, we consider Equation 18 for node p and input edge ij . A

direct application yields

$$\frac{\sum_{n \in \{-1, 0, 1\}} \zeta_{1,n}^p}{W_{ij}} \left(\sum_{n' \in \{-1, 0, 1\}} \zeta_{0,n'}^i \right) + \frac{\sum_{n \in \{-1, 0, 1\}} \zeta_{-1,n}^p}{W_{ij}} \left(\sum_{n' \in \{-1, 0, 1\}} \zeta_{0,n'}^j \right) = \frac{\sum_{n \in \{-1, 0, 1\}} \zeta_{1,n}^i}{W_{ij}} \left(\sum_{n' \in \{-1, 0, 1\}} \zeta_{0,n'}^p \right). \quad (29)$$

Here we have summed over the possible occurrences at edge kl . Each product of the form $\sum_n f_n \sum_{n'} g_{n'}$ involves 9 terms, and Equation 29 involves summing over every term. In fact, we can obtain more equations by conditioning, rather than summing, over the possible occurrences at edge kl . For example, we can condition on $n, n' = 1$ to write

$$\frac{\zeta_{1,1}^p}{W_{ij}} \zeta_{0,1}^i + \frac{\zeta_{-1,1}^p}{W_{ji}} \zeta_{0,1}^j = \frac{\zeta_{1,1}^i}{W_{ij}} \zeta_{0,1}^p. \quad (30)$$

We can similarly write ($n, n' = 0$)

$$\frac{\zeta_{1,0}^p}{W_{ij}} \zeta_{0,0}^i + \frac{\zeta_{-1,0}^p}{W_{ji}} \zeta_{0,0}^j = \frac{\zeta_{1,0}^i}{W_{ij}} \zeta_{0,0}^p \quad (31)$$

and ($n, n' = -1$)

$$\frac{\zeta_{1,-1}^p}{W_{ij}} \zeta_{0,-1}^i + \frac{\zeta_{-1,-1}^p}{W_{ji}} \zeta_{0,-1}^j = \frac{\zeta_{1,-1}^i}{W_{ij}} \zeta_{0,-1}^p. \quad (32)$$

It can be shown that these diagonal terms of the double sum $\sum_n f_n \sum_{n'} g_{n'}$ are supplemented by symmetrized off-diagonal combinations. We therefore also have ($n = 1, n' = -1 \cup n = -1, n' = 1$)

$$\frac{\zeta_{1,1}^p}{W_{ij}} \zeta_{0,-1}^i + \frac{\zeta_{-1,1}^p}{W_{ji}} \zeta_{0,-1}^j + \frac{\zeta_{1,-1}^p}{W_{ij}} \zeta_{0,1}^i + \frac{\zeta_{-1,-1}^p}{W_{ji}} \zeta_{0,1}^j = \frac{\zeta_{1,1}^i}{W_{ij}} \zeta_{0,-1}^p + \frac{\zeta_{1,-1}^i}{W_{ij}} \zeta_{0,1}^p \quad (33)$$

and ($n = 1, n' = 0 \cup n = 0, n' = 1$)

$$\frac{\zeta_{1,1}^p}{W_{ij}} \zeta_{0,0}^i + \frac{\zeta_{-1,1}^p}{W_{ji}} \zeta_{0,0}^j + \frac{\zeta_{1,0}^p}{W_{ij}} \zeta_{0,1}^i + \frac{\zeta_{-1,0}^p}{W_{ji}} \zeta_{0,1}^j = \frac{\zeta_{1,1}^i}{W_{ij}} \zeta_{0,0}^p + \frac{\zeta_{1,0}^i}{W_{ij}} \zeta_{0,1}^p \quad (34)$$

and ($n = -1, n' = 0 \cup n = 0, n' = -1$)

$$\frac{\zeta_{1,-1}^p}{W_{ij}} \zeta_{0,0}^i + \frac{\zeta_{-1,-1}^p}{W_{ji}} \zeta_{0,0}^j + \frac{\zeta_{1,0}^p}{W_{ij}} \zeta_{0,-1}^i + \frac{\zeta_{-1,0}^p}{W_{ji}} \zeta_{0,-1}^j = \frac{\zeta_{1,-1}^i}{W_{ij}} \zeta_{0,0}^p + \frac{\zeta_{1,0}^i}{W_{ij}} \zeta_{0,-1}^p. \quad (35)$$

In addition to these 6 equations for node p based on edge ij , we have 6 equations for node p based on edge kl , which are

$$\frac{\zeta_{1,1}^p}{W_{kl}} \zeta_{1,0}^k + \frac{\zeta_{1,-1}^p}{W_{lk}} \zeta_{1,0}^l = \frac{\zeta_{1,1}^k}{W_{kl}} \zeta_{1,0}^p \quad (36)$$

$$\frac{\zeta_{0,1}^p}{W_{kl}} \zeta_{0,0}^k + \frac{\zeta_{0,-1}^p}{W_{lk}} \zeta_{0,0}^l = \frac{\zeta_{0,1}^k}{W_{kl}} \zeta_{0,0}^p \quad (37)$$

$$\frac{\zeta_{-1,1}^p}{W_{kl}} \zeta_{-1,0}^k + \frac{\zeta_{-1,-1}^p}{W_{lk}} \zeta_{-1,0}^l = \frac{\zeta_{-1,1}^k}{W_{kl}} \zeta_{-1,0}^p \quad (38)$$

$$\frac{\zeta_{1,1}^p}{W_{kl}} \zeta_{-1,0}^k + \frac{\zeta_{1,-1}^p}{W_{lk}} \zeta_{-1,0}^l + \frac{\zeta_{-1,1}^p}{W_{kl}} \zeta_{1,0}^k + \frac{\zeta_{-1,-1}^p}{W_{lk}} \zeta_{1,0}^l = \frac{\zeta_{1,1}^k}{W_{kl}} \zeta_{-1,0}^p + \frac{\zeta_{-1,1}^k}{W_{kl}} \zeta_{1,0}^p \quad (39)$$

$$\frac{\zeta_{1,1}^p}{W_{kl}} \zeta_{0,0}^k + \frac{\zeta_{1,-1}^p}{W_{lk}} \zeta_{0,0}^l + \frac{\zeta_{0,1}^p}{W_{kl}} \zeta_{1,0}^k + \frac{\zeta_{0,-1}^p}{W_{lk}} \zeta_{1,0}^l = \frac{\zeta_{1,1}^k}{W_{kl}} \zeta_{0,0}^p + \frac{\zeta_{0,1}^k}{W_{kl}} \zeta_{1,0}^p \quad (40)$$

$$\frac{\zeta_{-1,1}^p}{W_{kl}} \zeta_{0,0}^k + \frac{\zeta_{-1,-1}^p}{W_{lk}} \zeta_{0,0}^l + \frac{\zeta_{0,1}^p}{W_{kl}} \zeta_{-1,0}^k + \frac{\zeta_{0,-1}^p}{W_{lk}} \zeta_{-1,0}^l = \frac{\zeta_{-1,1}^k}{W_{kl}} \zeta_{0,0}^p + \frac{\zeta_{0,-1}^k}{W_{kl}} \zeta_{-1,0}^p. \quad (41)$$

If we consider $p = i$ or $p = j$ at edge ij we are led to the equations (cf. Equation 19)

$$\frac{\zeta_{1,1}^i}{W_{ij}} = \frac{\zeta_{-1,1}^j}{W_{ij}} \quad (42)$$

$$\frac{\zeta_{1,0}^i}{W_{ij}} = \frac{\zeta_{-1,0}^j}{W_{ij}} \quad (43)$$

$$\frac{\zeta_{1,-1}^i}{W_{ij}} = \frac{\zeta_{-1,-1}^j}{W_{ij}} \quad (44)$$

and (cf. Equation 20)

$$\zeta_{-1,1}^i = 0 \quad (45)$$

$$\zeta_{-1,0}^i = 0 \quad (46)$$

$$\zeta_{-1,-1}^i = 0 \quad (47)$$

$$\zeta_{1,1}^j = 0 \quad (48)$$

$$\zeta_{1,0}^j = 0 \quad (49)$$

$$\zeta_{1,-1}^j = 0. \quad (50)$$

Similarly, considering $p = k$ or $p = l$ at edge kl leads to

$$\frac{\zeta_{1,1}^k}{W_{kl}} = \frac{\zeta_{1,-1}^l}{W_{lk}} \quad (51)$$

$$\frac{\zeta_{0,1}^k}{W_{kl}} = \frac{\zeta_{0,-1}^l}{W_{lk}} \quad (52)$$

$$\frac{\zeta_{-1,1}^k}{W_{kl}} = \frac{\zeta_{-1,-1}^l}{W_{lk}} \quad (53)$$

and

$$\zeta_{1,-1}^k = 0 \quad (54)$$

$$\zeta_{0,-1}^k = 0 \quad (55)$$

$$\zeta_{-1,-1}^k = 0 \quad (56)$$

$$\zeta_{1,1}^l = 0 \quad (57)$$

$$\zeta_{0,1}^l = 0 \quad (58)$$

$$\zeta_{-1,1}^l = 0. \quad (59)$$

We have listed the various equations derived from Equation 18 which may reduce the number of degrees of freedom in $\mathcal{S}^{\text{orig}}$. Equations 45-50 and 54-59 constrain 12 functions to be zero. We use Equations 42 and 51 to substitute W_{ij}/W_{ji} and W_{kl}/W_{lk} where they appear in the remaining equations so that they are closed in the functions in $\mathcal{S}^{\text{orig}}$.

An additional complication remains, however, which is that these equations are not all independent of each other. We first count how many equations we can write down and then consider how to find the number of independent constraints among these:

- Considering $p = i$ or $p = j$ at edge ij yields 2 equations (not counting Equation 42 which allowed eliminating W_{ij}/W_{ji} , and not counting Equations 45-50 which allowed zeroing out 6 functions)
- Considering $p = k$ or $p = l$ at edge kl yields 2 equations (not counting Equation 51 which allowed eliminating W_{kl}/W_{lk} , and not counting Equations 54-59 which allowed zeroing out 6 functions)
- Considering $p = i$ at edge kl yields 6 equations
- Considering $p = j$ at edge kl yields 6 equations
- Considering $p = k$ at edge ij yields 6 equations

- Considering $p = l$ at edge ij yields 6 equations
- Considering arbitrary $p \neq i, j, k, l$ at edge ij yields 6 equations for each p
- Considering arbitrary $p \neq i, j, k, l$ at edge kl yields 6 equations for each p

We thus have $28 + 12(N_n - 4)$ equalities for $N_n \geq 4$. We collect these equations, of the form $C(\mathcal{S}^{\text{orig}}) = 0$, in a set \mathcal{C} . To determine how many of these equations are independent, we form the Jacobian matrix $J^{\mathcal{C}}$ whose elements are the derivatives of the equations with respect to the elements of $\mathcal{S}^{\text{orig}}$.

To understand the scaling with N_n , we first ignore node $p \neq i, j, k, l$ and consider the submatrix of $J^{\mathcal{C}}$ formed by excluding the equations that involve node p and excluding the functions $\zeta_{s,t}^p$ from $\mathcal{S}^{\text{orig}}$; this is a closed set of equations on functions defined at nodes i, j, k , and l . We find that the rank of this matrix is 12, far fewer than the 28 equations involving these functions listed above. These 28 equations thus represent 12 independent constraints on the $36 - 12 = 24$ non-zero functions in $\mathcal{S}^{\text{orig}} \setminus \{\zeta_{s,t}^p\}_{s,t \in \{-1,0,1\}}$, leaving 12 degrees of freedom.

We now include the equations and functions for node p . Note that node p will only be coupled via the equations to nodes i, j, k , and l , not to any other node p' . We may thus view the degrees of freedom per extra node as adding linearly. The rank of the matrix $J^{\mathcal{C}}$ including the fifth node p is 18, representing 6 new independent constraints. Every node beyond the fourth thus introduces 9 new functions and 6 new constraints, leaving 3 degrees of freedom per node. This means that the total number of degrees of freedom $\mathcal{S}^{\text{orig}}$ is $12 + 3(N_n - 4) = 3N_n$. Numerical evaluation of the rank of the Jacobian matrix $J^{\mathcal{S}^{\text{orig}}}$ for fully connected graphs with increasing N_n confirms this predicted scaling: $\text{rank}(J^{\mathcal{S}^{\text{orig}}}) = 3N_n$. This shows that equations derived from Equation 18, after accounting for dependencies among these equations, correctly predicts how many degrees of freedom are available in the set of polynomial coefficients $\mathcal{S}^{\text{orig}}$.

2. Prevented classification of four classes

We now assess whether it is possible to classify two classes for the case $M = 1, D = 2$. We assume that we have a problem with class 1 in quadrant I ($F_a, F_b > 0$), class 2 in quadrant II ($F_a < 0, F_b > 0$), class 3 in quadrant III ($F_a, F_b < 0$), and class 4 in quadrant IV ($F_a > 0, F_b < 0$). We can place a condition for solving this classification task on the limiting values of π_p in each of these quadrants. Specifically, we require that

$$\lim_{F_a \rightarrow \infty, F_b \rightarrow \infty} \pi_1 > \lim_{F_a \rightarrow \infty, F_b \rightarrow \infty} \pi_p \quad \forall p \neq 1 \quad (60)$$

$$\lim_{F_a \rightarrow -\infty, F_b \rightarrow \infty} \pi_2 > \lim_{F_a \rightarrow -\infty, F_b \rightarrow \infty} \pi_p \quad \forall p \neq 2 \quad (61)$$

$$\lim_{F_a \rightarrow -\infty, F_b \rightarrow -\infty} \pi_3 > \lim_{F_a \rightarrow -\infty, F_b \rightarrow -\infty} \pi_p \quad \forall p \neq 3 \quad (62)$$

$$\lim_{F_a \rightarrow \infty, F_b \rightarrow -\infty} \pi_4 > \lim_{F_a \rightarrow \infty, F_b \rightarrow -\infty} \pi_p \quad \forall p \neq 4. \quad (63)$$

These inequalities in turn place the following conditions on the functions

$$\zeta_{1,1}^1 > \zeta_{1,1}^p \quad \forall p \neq 1 \quad (64)$$

$$\zeta_{-1,1}^2 > \zeta_{-1,1}^p \quad \forall p \neq 2 \quad (65)$$

$$\zeta_{-1,-1}^3 > \zeta_{-1,-1}^p \quad \forall p \neq 3 \quad (66)$$

$$\zeta_{1,-1}^4 > \zeta_{1,-1}^p \quad \forall p \neq 4. \quad (67)$$

These are $4N_n - 4$ inequalities which must be satisfied by the $4N_n$ functions in $\mathcal{S}^{\text{ineq}} \equiv \{\{\zeta_{s,t}^p\}_{s,t \in \{-1,1\}}\}_{p=1}^{N_n} \subset \mathcal{S}^{\text{orig}}$. From Equations 45-50 and 54-59, 8 of these functions are zero and trivially satisfy the inequalities in which they appear, assuming that $i \neq 2, j \neq 1, k \neq 4$, and $l \neq 3$. We note that inequalities in Equations 64-67 cannot depend on W_{ij}, W_{ji}, W_{kl} or W_{lk} for similar reasons as described in the previous section (either both sides of an inequality contain no dependence on an edge rate or are proportional to it). We are thus led to ask how many non-zero degrees of freedom are available in $\mathcal{S}^{\text{ineq}}$ through variation in all edges rates except W_{ij}, W_{ji}, W_{kl} and W_{lk} .

Considering the case $N_n = 4$, there are 8 non-zero functions in $\mathcal{S}^{\text{ineq}}$. There are now 14 independent equations among all the function in $\mathcal{S}^{\text{orig}}$ (because we no longer use Equations 42 and 51 to eliminate W_{ij}/W_{ji} and W_{kl}/W_{lk}), and 4 of these equations (Equations 42, 44, 51, and 53) constrain elements in $\mathcal{S}^{\text{ineq}}$, leaving 4 degrees of freedom in this set. There are thus exactly as many degrees of freedom in $\mathcal{S}^{\text{ineq}}$ as there are inequality conditions to satisfy for $N_n = 4$, but we show in the next section that four-class classification is still impossible because these inequalities

are mutually contradictory. There are 16 non-zero functions in $\mathcal{S}^\circ \equiv \mathcal{S}^{\text{orig}} \setminus \mathcal{S}^{\text{ineq}}$, and the remaining 10 constraints reduce the number of degrees of freedom in this set to 6.

For $N_n > 4$, every additional node p introduces 4 new functions $\{\zeta_{\pm 1, \pm 1}^p\}$ in $\mathcal{S}^{\text{ineq}}$ and 5 new functions in \mathcal{S}° . As shown in the previous section we also introduce 6 new constraints among all 9 functions for this node and the functions for nodes i, j, k , and l . Although these constraints also involve the 6 degrees of freedom in \mathcal{S}° for nodes i, j, k , and l , the values of the functions in \mathcal{S}° do not affect the inequality conditions, and we may use up their degrees of freedom to set their values arbitrarily. We may then use the 6 independent constraints for node p to solve for all 5 new functions in \mathcal{S}° and for 1 new function in $\mathcal{S}^{\text{ineq}}$. This leaves only 3 new degrees of freedom in $\mathcal{S}^{\text{ineq}}$ for every node p . The total number of degrees of freedom in this set (having treated W_{ij} , W_{ji} , W_{kl} , and W_{lk} as fixed) is thus $4 + 3(N_n - 4) = 3N_n - 8$. We numerically confirm this by evaluating $\text{rank}(J^{\mathcal{S}^{\text{ineq}}}) = 3N_n - 8$ for fully connected graphs with increasing N_n (again not including Jacobian columns for derivatives with respect to W_{ij} , W_{ji} , W_{kl} , or W_{lk}). Comparing $3N_n - 8$ to the number of non-trivial inequalities, $4N_n - 12$, we conclude that there will be insufficient degrees of freedom to achieve four-class classification of freedom for $N_n > 4$. The case $N_n = 4$ is discussed next. We note that to achieve three-class classification there are only $3N_n - 8$ non-trivial inequalities to satisfy, so this is indeed possible for $M = 1$, $D = 2$.

3. Frustrated inequalities

An interesting situation arises when considering classification with $M = 1$, $D = 2$, and $N_n = 4$. Assuming that the input edges are not co-incident on any node, it is necessary that the four output nodes be adjacent to the input edges since there are no other nodes in the network. The situation we will describe in fact happens whenever the output nodes and input edges are adjacent, so can consider a more general setting (with arbitrary N_n) in which class 1 is at node i , class 2 at node j , class 3 at node k , and class 4 at node l . Accounting for the fact that $\zeta_{-1,1}^1 = \zeta_{-1,-1}^1 = \zeta_{1,1}^2 = \zeta_{1,-1}^2 = \zeta_{1,-1}^3 = \zeta_{-1,-1}^3 = \zeta_{1,1}^4 = \zeta_{-1,1}^4 = 0$, the four inequalities which must be satisfied to solve the four-class classification problem are

$$\zeta_{1,1}^1 > \zeta_{1,1}^2 \quad (68)$$

$$\zeta_{-1,1}^2 > \zeta_{-1,1}^3 \quad (69)$$

$$\zeta_{-1,-1}^3 > \zeta_{-1,-1}^4 \quad (70)$$

$$\zeta_{1,-1}^4 > \zeta_{1,-1}^1. \quad (71)$$

We may divide each of these inequalities by the product of edge rates which factor out on both sides, so that

$$\frac{\zeta_{1,1}^1}{W_{ij}W_{kl}} > \frac{\zeta_{1,1}^2}{W_{ij}W_{kl}} \quad (72)$$

$$\frac{\zeta_{-1,1}^2}{W_{ji}W_{kl}} > \frac{\zeta_{-1,1}^3}{W_{ji}W_{kl}} \quad (73)$$

$$\frac{\zeta_{-1,-1}^3}{W_{ji}W_{lk}} > \frac{\zeta_{-1,-1}^4}{W_{ji}W_{lk}} \quad (74)$$

$$\frac{\zeta_{1,-1}^4}{W_{ij}W_{lk}} > \frac{\zeta_{1,-1}^1}{W_{ij}W_{lk}}. \quad (75)$$

Now we use the fact that since the output nodes are adjacent to the input edges they must obey Equations 42, 44, 51, and 53. Substituting from these equations, we arrive at

$$\frac{\zeta_{-1,1}^3}{W_{ji}W_{kl}} > \frac{\zeta_{1,1}^2}{W_{ij}W_{kl}} \quad (76)$$

$$\frac{\zeta_{-1,1}^2}{W_{ji}W_{kl}} > \frac{\zeta_{-1,1}^3}{W_{ji}W_{kl}} \quad (77)$$

$$\frac{\zeta_{-1,-1}^3}{W_{ji}W_{lk}} > \frac{\zeta_{-1,1}^2}{W_{ji}W_{kl}} \quad (78)$$

$$\frac{\zeta_{1,1}^2}{W_{ij}W_{kl}} > \frac{\zeta_{-1,-1}^3}{W_{ji}W_{lk}}. \quad (79)$$

One can see that this set of inequalities is contradictory. As a result, four-class classification is impossible for $N_n = 4$, or more generally when the output nodes are adjacent to the input edges.

V. SHARPNESS OF THE DECISION BOUNDARY

A. Derivation of Equation 8 in the main text

1. Maximizing $\partial\pi_S(F; \boldsymbol{\theta})/\partial F$

In the case $D = 1$, we can simplify Equation 2 of the main text applied at the substrate node S as

$$\pi_S(F; \boldsymbol{\theta}) = \frac{\sum_{m=M_{\min}}^{M_{\max}} \zeta_m^S(\boldsymbol{\theta}) e^{mF/2}}{\sum_{m=M_{\min}}^{M_{\max}} \bar{\zeta}_m(\boldsymbol{\theta}) e^{mF/2}}. \quad (80)$$

Here, M_{\min} refers to the smallest power of $e^{F/2}$ which occurs in the numerator or denominator and M_{\max} refers to the greatest such power. We multiply the numerator and denominator of Equation 80 by $e^{-M_{\min}F/2}$, define $M_R \equiv M_{\max} - M_{\min}$, and re-define the index as $m \leftarrow m - M_{\min}$, so that

$$\pi_S(F; \boldsymbol{\theta}) = \frac{\sum_{m=0}^{M_R} \zeta_m^S(\boldsymbol{\theta}) e^{mF/2}}{\sum_{m=0}^{M_R} \bar{\zeta}_m(\boldsymbol{\theta}) e^{mF/2}}. \quad (81)$$

We are interested in the magnitude of the gradient $\partial\pi_S/\partial F$ evaluated at the decision boundary $F = 0$, which is (suppressing the $\boldsymbol{\theta}$ dependence of the coefficients)

$$\frac{\partial\pi_S(F; \boldsymbol{\theta})}{\partial F} = \frac{\sum_{m=0}^{M_R} \sum_{m'=0}^{M_R} (m - m') \zeta_m^S \bar{\zeta}_{m'} e^{(m+m')F/2}}{2 \left(\sum_{m=0}^{M_R} \bar{\zeta}_m e^{mF/2} \right)^2}. \quad (82)$$

An offset of the decision boundary location to F_0 could be absorbed into the coefficients by redefining $F \leftarrow F - F_0$. At $F = 0$ we have

$$\frac{\partial\pi_S(0; \boldsymbol{\theta})}{\partial F} = \frac{\sum_{m=0}^{M_R} \sum_{m'=0}^{M_R} (m - m') \zeta_m^S \bar{\zeta}_{m'}}{2 \left(\sum_{m=0}^{M_R} \bar{\zeta}_m \right)^2}. \quad (83)$$

We can rewrite the numerator of this expression as

$$\begin{aligned} \sum_{m=0}^{M_R} \sum_{m'=0}^{M_R} (m - m') \zeta_m^S \bar{\zeta}_{m'} &= \left(\sum_{m'=0}^{M_R} \bar{\zeta}_{m'} \right) \sum_{m=0}^{M_R} m \zeta_m^S - \left(\sum_{m=0}^{M_R} \zeta_m^S \right) \sum_{m'=0}^{M_R} m' \bar{\zeta}_{m'} \\ &\equiv \bar{\zeta}_{\text{tot}} \sum_{m=0}^{M_R} m \zeta_m^S - \zeta_{\text{tot}}^S \sum_{m'=0}^{M_R} m' \bar{\zeta}_{m'} \end{aligned} \quad (84)$$

and the denominator as $2\bar{\zeta}_{\text{tot}}^2$. We thus have

$$\frac{\partial\pi_S(0; \boldsymbol{\theta})}{\partial F} = \frac{\zeta_{\text{tot}}^S}{2\bar{\zeta}_{\text{tot}}} \left(\frac{\sum_{m=0}^{M_R} m \zeta_m^S}{\zeta_{\text{tot}}^S} - \frac{\sum_{m=0}^{M_R} m \bar{\zeta}_m}{\bar{\zeta}_{\text{tot}}} \right). \quad (85)$$

We next aim to maximize this function to understand its dependence on M_R , the range of powers of $e^{F/2}$. We first maximize with respect to the functions $\{\zeta_m^S\}_{m=0}^{M_R}$. It is straightforward to show that

$$\frac{\sum_{m=0}^{M_R} m \zeta_m^S}{\zeta_{\text{tot}}^S} \leq M_R \quad (86)$$

with equality when $\zeta_{M_R}^S = \zeta_{\text{tot}}^S$ and the other ζ_m^S are zero. Thus,

$$\frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} \leq \frac{\zeta_{\text{tot}}^S}{2\bar{\zeta}_{\text{tot}}} \left(M_R - \frac{\sum_{m=0}^{M_R} m \bar{\zeta}_m}{\bar{\zeta}_{\text{tot}}} \right). \quad (87)$$

We next maximize this with respect to the quantities $\{\bar{\zeta}_m\}_{m=0}^{M_R}$. To encode the fact that $\bar{\zeta}_m \geq \zeta_m^S \forall m$, we write $\bar{\zeta}_m = \zeta_m^S + \tilde{\zeta}_m^S$ and $\bar{\zeta}_{\text{tot}} = \zeta_{\text{tot}}^S + \tilde{\zeta}_{\text{tot}}^S$, with $\tilde{\zeta}_m^S \geq 0$ and $\tilde{\zeta}_{\text{tot}}^S \geq 0$. We have

$$\frac{\zeta_{\text{tot}}^S}{2\bar{\zeta}_{\text{tot}}} \left(M_R - \frac{\sum_{m=0}^{M_R} m \bar{\zeta}_m}{\bar{\zeta}_{\text{tot}}} \right) = \frac{\zeta_{\text{tot}}^S}{2(\zeta_{\text{tot}}^S + \tilde{\zeta}_{\text{tot}}^S)} \left(M_R - \frac{M_R \zeta_{\text{tot}}^S + \sigma}{\zeta_{\text{tot}}^S + \tilde{\zeta}_{\text{tot}}^S} \right) \quad (88)$$

where $\sigma \equiv \sum_{m=0}^{M_R} m \tilde{\zeta}_m^S - M_R \zeta_{\text{tot}}^S$. Standard calculus techniques allow showing that this expression is optimized with respect to $\tilde{\zeta}_{\text{tot}}^S$ when

$$\tilde{\zeta}_{\text{tot}}^S = \zeta_{\text{tot}}^S + \frac{2\sigma}{M_R}, \quad (89)$$

i.e., when

$$\bar{\zeta}_{\text{tot}} = 2 \left(\zeta_{\text{tot}}^S + \frac{\sigma}{M_R} \right). \quad (90)$$

Inserting the solution into Equation 87 gives

$$\frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} \leq \frac{\zeta_{\text{tot}}^S M_R^2}{8(\zeta_{\text{tot}}^S M_R + \sigma)}. \quad (91)$$

We finally aim to maximize this expression with respect to the $\{\bar{\zeta}_m\}_{m=0}^{M_R}$ that enter the variable σ , subject to Equation 90. Equation 91 will be maximized when $\sigma = 0$, when Equation 90 imposes $\bar{\zeta}_{\text{tot}} = 2\zeta_{\text{tot}}^S$. Recalling that maximization with respect to the quantities $\{\zeta_m^S\}_{m=0}^{M_R}$ yielded $\zeta_{\text{tot}}^S = \zeta_{M_R}^S$, we see that these conditions will be achieved if $\bar{\zeta}_0 = \bar{\zeta}_{M_R} = \zeta_{M_R}^S$ and all other $\bar{\zeta}_m = 0$. At this point, we have the desired result

$$\max_{\{\zeta_m^S\}, \{\bar{\zeta}_m\}} \frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} = \frac{M_R}{8}. \quad (92)$$

Thus, the bound of the derivative's magnitude is proportional to the range of exponential powers. We verified this expression numerically using conjugate gradient optimization under the assumption that all functions ζ_m^S and $\tilde{\zeta}_m^S$ are free and independent.

2. Minimizing $\partial \pi_S(F; \boldsymbol{\theta}) / \partial F$

Let us now minimize Equation 85. We have that

$$\frac{\sum_{m=0}^{M_R} m \zeta_m^S}{\zeta_{\text{tot}}^S} \geq 0 \quad (93)$$

with equality when $\zeta_0^S = \zeta_{\text{tot}}^S$ and the other ζ_m^S are zero. This implies

$$\frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} \geq -\frac{\zeta_{\text{tot}}^S}{2\bar{\zeta}_{\text{tot}}} \frac{\sum_{m=0}^{M_R} m \bar{\zeta}_m}{\bar{\zeta}_{\text{tot}}} = -\frac{\zeta_{\text{tot}}^S}{2(\zeta_{\text{tot}}^S + \tilde{\zeta}_{\text{tot}}^S)^2} \sum_{m=0}^{M_R} m \tilde{\zeta}_m^S, \quad (94)$$

where the last equality results from the condition that only ζ_0^S is non-zero when writing $\bar{\zeta}_m = \zeta_m^S + \tilde{\zeta}_m^S$. Equation 94 is minimized when $\tilde{\zeta}_{M_R}^S = \tilde{\zeta}_{\text{tot}}^S$ and when $\tilde{\zeta}_{\text{tot}}^S = \zeta_{\text{tot}}^S$, i.e. when $\bar{\zeta}_0 = \bar{\zeta}_{M_R} = \zeta_0^S$. Inserting these values into Equation

85 yields

$$\min_{\{\zeta_m^S\}, \{\bar{\zeta}_m\}} \frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} = -\frac{M_R}{8}. \quad (95)$$

We again verified this result numerically using conjugate gradient optimization.

To summarize, the magnitude of the derivative is bounded as

$$\left| \frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} \right| \leq \frac{M_R}{8}. \quad (96)$$

Returning to the original definition of the index m which runs from M_{\min} to M_{\max} , the lower bound is saturated when $\bar{\zeta}_{M_{\min}} = \bar{\zeta}_{M_{\max}} = \zeta_{M_{\min}}^S$, and the upper bound is saturated when $\bar{\zeta}_{M_{\min}} = \bar{\zeta}_{M_{\max}} = \zeta_{M_{\max}}^S$.

3. Tightening the bound

The exponential range M_R in Equation 96 refers to the difference between the maximum and minimum powers of $e^{F/2}$ in the terms of Equation 80 whose coefficients $\bar{\zeta}_m$ are non-zero. These functions $\bar{\zeta}_m = \sum_i \zeta_m^i$ sum over all nodes in the network, but in order for the output function to saturate near $\pi_i \approx 1$ in its assigned input region for class i the terms $\zeta_m^j e^{mF/2}$ for $j \neq i$ should be small in this region. The trained network will thus tend to have large values of ζ_m^i only for nodes i in the set of designated output nodes \mathcal{O} . If the non-output nodes had large values then they would compete with the output nodes and prevent their probabilities from saturating in their designated input regions. As a result, we can approximate $\pi_S(F)$ near the decision boundary as (cf. Equation 80)

$$\pi_S(F; \boldsymbol{\theta}) \approx \frac{\sum_{m=M_{\min}^{\mathcal{O}}}^{M_{\max}^{\mathcal{O}}} \zeta_m^S(\boldsymbol{\theta}) e^{mF/2}}{\sum_{m=M_{\min}^{\mathcal{O}}}^{M_{\max}^{\mathcal{O}}} \bar{\zeta}_m^{\mathcal{O}}(\boldsymbol{\theta}) e^{mF/2}} \quad (97)$$

where $\bar{\zeta}_m^{\mathcal{O}}(\boldsymbol{\theta}) \equiv \sum_{i \in \mathcal{O}} \zeta_m^i(\boldsymbol{\theta})$ and $M_{\min}^{\mathcal{O}}$ (resp. $M_{\max}^{\mathcal{O}}$) is the minimum (resp. maximum) power of $e^{F/2}$ whose coefficient $\bar{\zeta}_m^{\mathcal{O}}(\boldsymbol{\theta})$ is non-zero. From this, we can effectively tighten the bound in Equation 96 as

$$\left| \frac{\partial \pi_S(0; \boldsymbol{\theta})}{\partial F} \right| \lesssim \frac{M_R^{\mathcal{O}}}{8} \leq \frac{M_R}{8} \quad (98)$$

where $M_R^{\mathcal{O}} \equiv M_{\max}^{\mathcal{O}} - M_{\min}^{\mathcal{O}} \leq M_R$. There will be at least as large a range M_R of powers among all nodes as the range $M_R^{\mathcal{O}}$ among the output nodes. Because of this, at extremal values of $F \rightarrow \pm\infty$ the highest and lowest powers of $e^{F/2}$ will dominate, which may not correspond to the large probabilities at the output nodes. Near the decision boundary, however, and for the finite values of F in the data on which the network was trained, the approximation $\bar{\zeta}_m \approx \bar{\zeta}_m^{\mathcal{O}}$ should hold due to the training goal of saturating $\pi_i \approx 1$ for $i \in \mathcal{O}$ in these regions of input space.

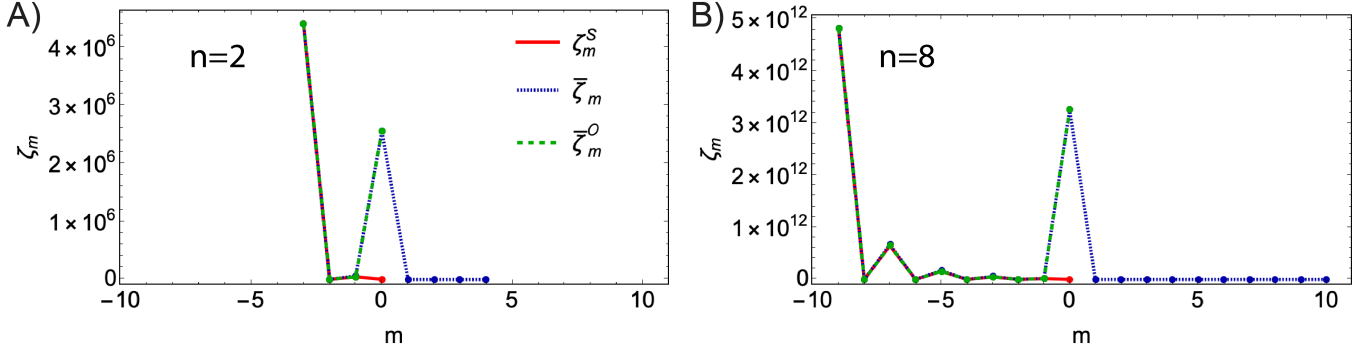
B. Analysis of trees in the extended push-pull networks

How does this analysis help to understand the difference between the parallel and series extensions of the push-pull network in Figure 7 of the main text? We first consider the parallel extension. The range in exponential powers $M_R^{\mathcal{O}}$ depends on the set of directed spanning trees. A maximum power of the exponent will occur for a tree rooted on node S or S^* in which the path connecting S and S^* occurs among the un-driven edges (black edges in Figure 7). The directed edges for these rooted spanning trees will be oriented in the direction of $F > 0$ for $n+1$ of the driven edges, contributing a term of $e^{(n+1)F/2}$. A minimum power of the exponent will occur for trees rooted on node S or S^* in which the path connecting S and S^* occurs among the driven edges. The directed edges for these rooted spanning trees will have two edges oriented in opposite directions relative to $F > 0$, cancelling, and n edges oriented in the direction of $F > 0$. These trees thus contribute a term of $e^{nF/2}$. The range $M_R^{\mathcal{O}} = 1$ for the parallel extension does not grow with n , explaining why the decision boundary cannot get sharper as n increases.

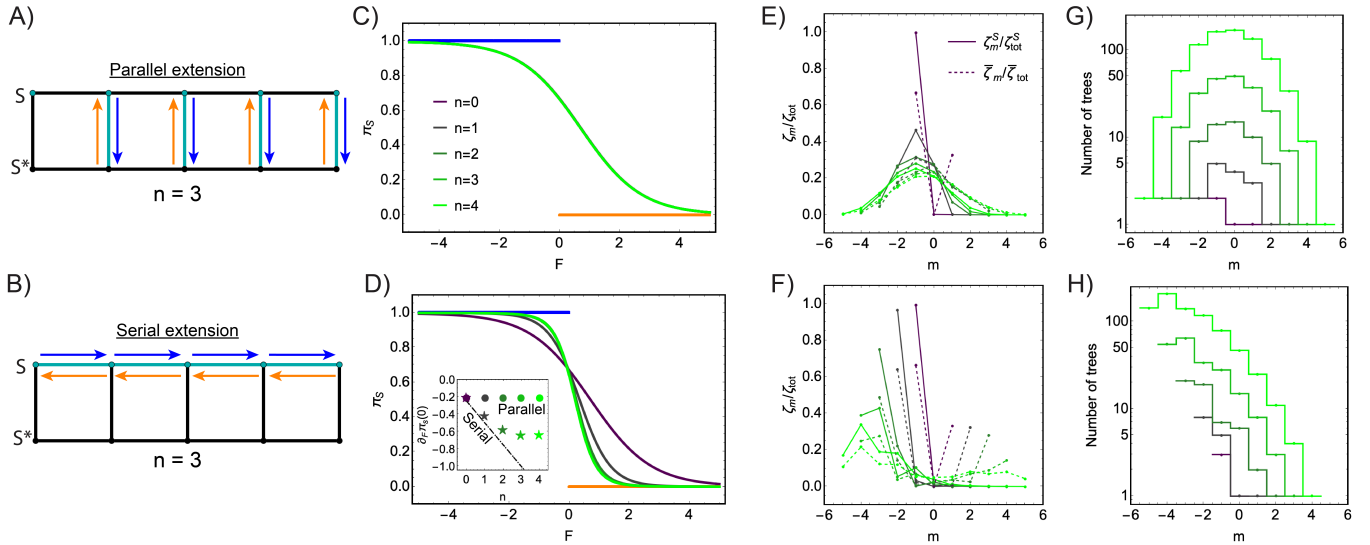
By contrast, $M_R^{\mathcal{O}}$ can grow with n for the serial extension of the push-pull network. A maximum power of the exponent will occur for trees rooted on node S or S^* in which the removed edge adjacent to the central top node. This will contribute a term $e^{(n+1)F/2}$. A minimum power will occur for trees in which the removed edge is taken from among the un-driven edges. These trees will contribute a term e^0 , implying that $M_R^{\mathcal{O}} = n+1$, which grows with n as

desired. Note that $M_R^O = 1$ for $n = 0$ for both serial and parallel.

In Figure 4 we plot the various functions ζ_m^S , ζ_m^O , and $\bar{\zeta}_m$ which have been learned during training of the extended push-pull networks. These numerical results validate the approximation $\zeta_m^O \approx \bar{\zeta}_m$ and the strategy of trying to set $\bar{\zeta}_{M_{\min}^O} = \bar{\zeta}_{M_{\max}^O} = \zeta_{M_{\min}^O}^S$ to minimize $\partial_S \pi_S(0, F)$.



Supplementary Figure 4. Plots of the learned values $\{\zeta_m^S\}_{m=M_{\min}^O}^{M_{\max}^O}$, $\{\bar{\zeta}_m\}_{m=M_{\min}^O}^{M_{\max}^O}$, and $\{\zeta_m^O\}_{m=M_{\min}^O}^{M_{\max}^O}$ for the serially extended push-pull networks in Figure 7 of the main text.



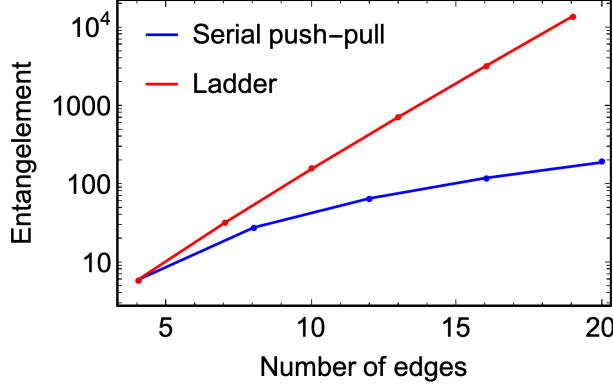
Supplementary Figure 5. Sharpening decision boundaries for common biochemical motifs. A,B) Illustrations of a ladder network with edges driven in parallel (A) or in series (B). C,D) Plots of the trained $\pi_S(F; \theta)$ curves for increasing numbers of nodes in the parallel (C) or serial (D) extensions of the ladder network. The inset is the same as in Figure 7C of the main text; here the dashed and dotted lines overlap. E,F) Plots of the trained functions for the parallel (E) or serial (F) extensions $\{\zeta_m^S\}_{m=M_{\min}^O}^{M_{\max}^O}$ and $\{\bar{\zeta}_m\}_{m=M_{\min}^O}^{M_{\max}^O}$ as fractions of their summations over m . G,H) Histograms of trees in the parallel (G) or serial (H) extensions rooted at node S which have a net number m of contributions from the driven edges.

C. Analysis of trees in the extended ladder networks

We consider another classic biochemical motif, a ladder network, which is often used to describe sequential conformational states proteins such as the flagellar motor [14, 15]. We consider extending these motifs as before in a parallel and a serial fashion (Figures 5A,B).

Training these networks for increasing n , we find as before that the parallel strategy does not allow sharpening the boundary whatsoever (Figures 5C). In contrast to the previous example, however, the gain in sharpness from

adding edges in a serial manner begins to diminish after $n \sim 3$ (Figures 5D). As for the push-pull networks, one can analyze the spanning trees in the ladder networks to show that for both the parallel and serial extensions, $M_R^O = 2(n+1)$. To understand why this bound is not tight, recall how saturating Equation 96 requires the concentrating all the weight of the ζ_m^S functions to $\zeta_{M_{\min}^O}^S$ and the weight of the $\bar{\zeta}_m$ functions evenly to $\bar{\zeta}_{M_{\min}^O}$ and $\bar{\zeta}_{M_{\max}^O}$.



Supplementary Figure 6. Entanglement, defined as the average number of directed trees which a given directed edge is a part of, as a function of network size for the serial push-pull and ladder networks.

sharpness. For these ladder networks there are many entanglements of the degrees of freedom, and the sharpness optimization point likely lies far from the reachable subspace spanned by the trainable degrees of freedom W_{ij} .

A simple metric supports this picture. For a given graph we count how many directed trees a given directed edge is a part of, and we average this number over all directed edges in the graph. This gives a rough estimate of entanglement (corresponding to the leftmost two layers of the schematic in Figure 4B of the main text). In Figure 6 we show this metric defined for increasing sizes of the serial push-pull network and the ladder networks. For the ladder networks, there are orders of magnitude more trees which a single degree of freedom W_{ij} is typically a part of, making the optimization task more difficult to solve and preventing saturation of the bound in Equation 98.

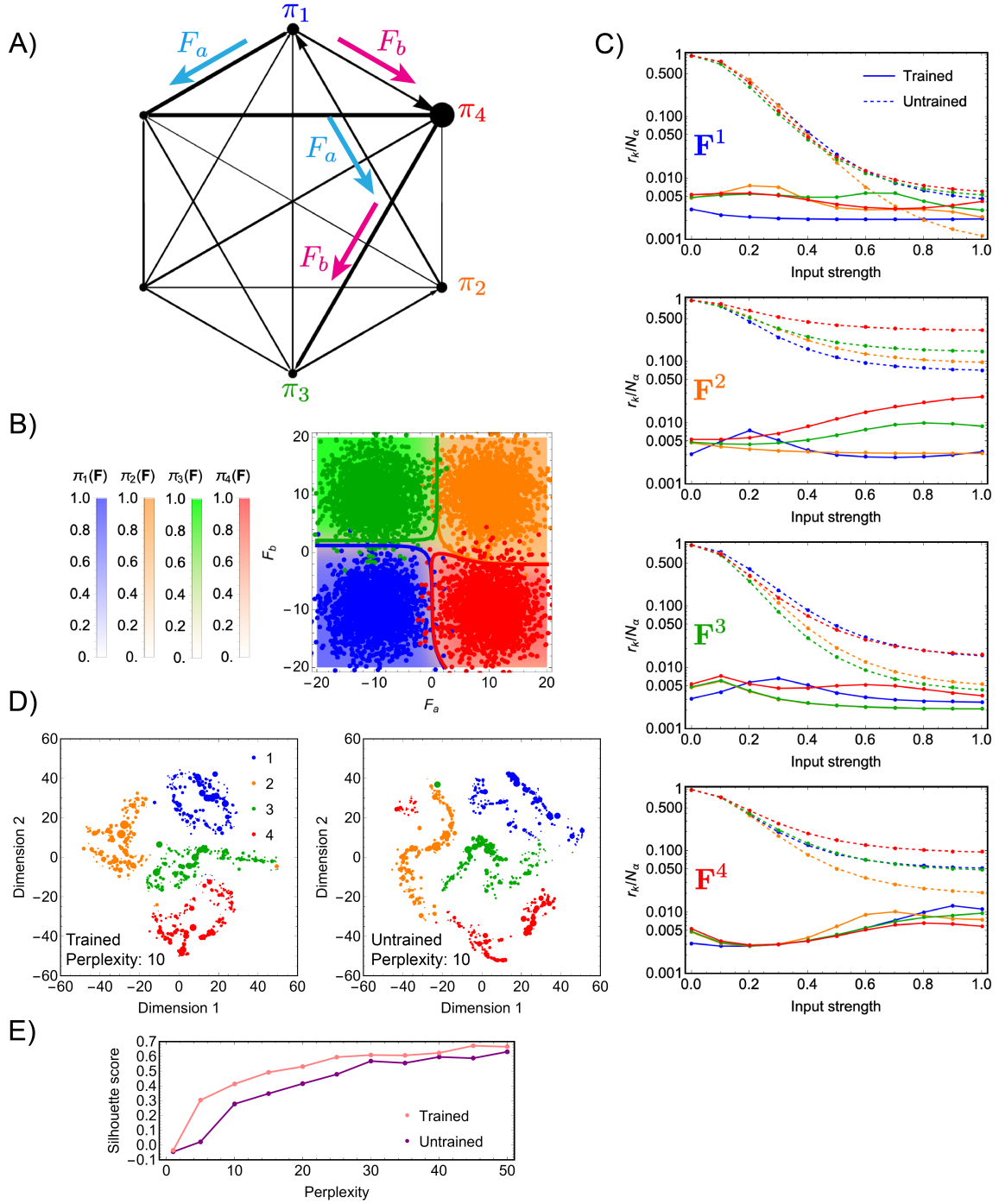
VI. LOW DIMENSIONALITY OF TRAINED NETWORKS

Here we ask how networks trained to perform classification differ from networks with uniform weights. It is known that neural networks trained to perform classification have a subset of weights which are important for accurate performance and a different, less important subset which can actually be pruned from the network without significantly decreasing the accuracy [16]. Relatedly, physical networks trained to perform classification have heterogeneous weights and more localized normal modes [17]. To explore this in the context of Markov networks, we consider a standard measure of mode delocalization called the inverse participation ratio (IPR). For an eigenbasis $\{\mathbf{v}_k\}_{k=1}^{N_n}$, where each vector \mathbf{v}_k has components $v_{k,\alpha}$ the IPR is defined as

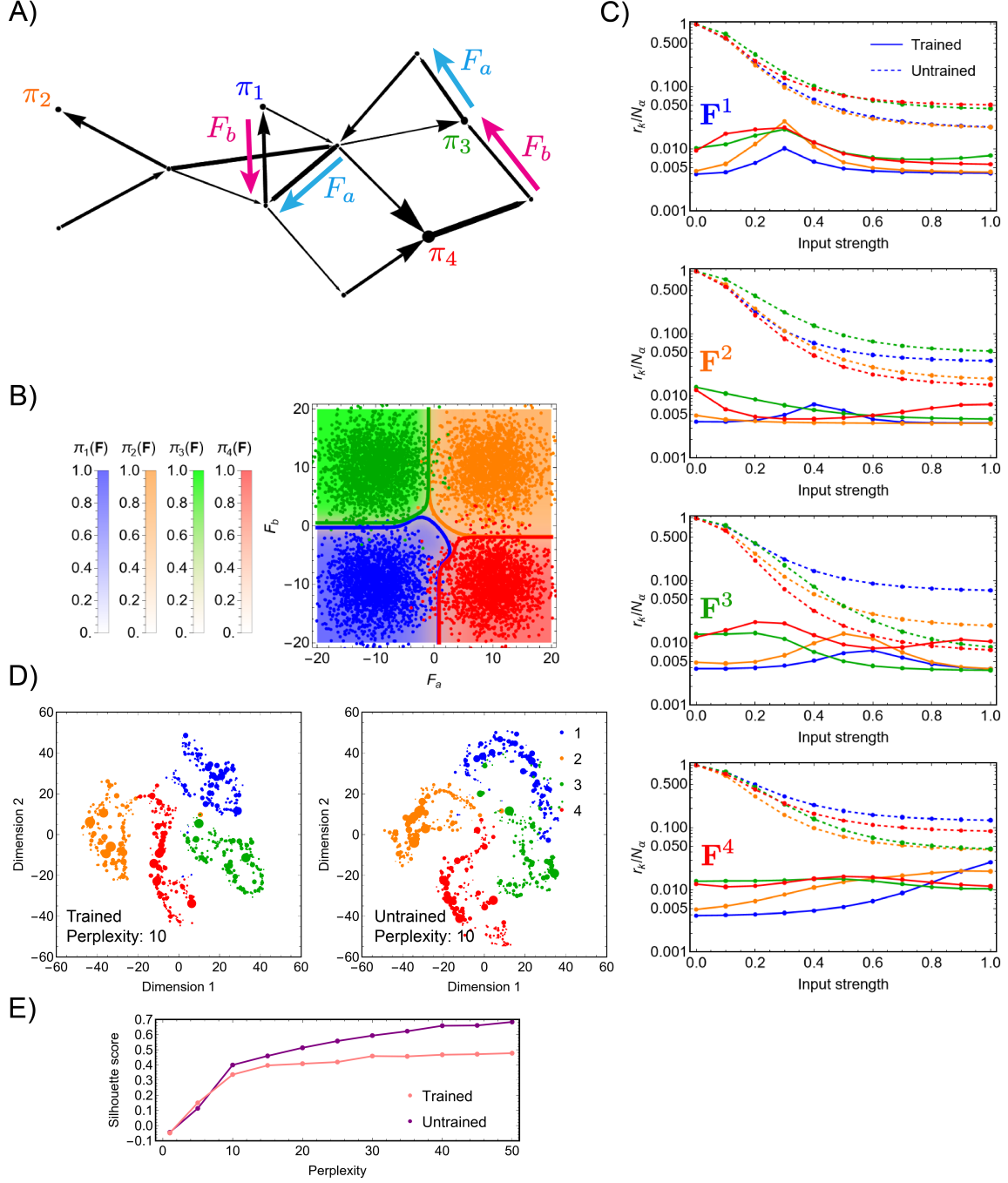
$$r_k^H = \frac{\left(\sum_{\alpha} v_{k,\alpha}^2\right)^2}{\sum_{\alpha} v_{k,\alpha}^4} = v_k^4 \left(\sum_{\alpha} v_{k,\alpha}^4\right)^{-1} \quad (99)$$

where v_k is the vector norm of \mathbf{v}_k . It is straightforward to show that if the eigenmode is spread evenly over its N_{α} components (i.e., $v_{k,\alpha} \propto 1/N_{\alpha}$), then $r_k = N_{\alpha}$; if it is fully localized on one component, then $r_k = 1$. Analogously, we define for node k of the Markov network

$$r_k = \frac{\left(\sum_{\alpha} w(T_k^{\alpha})\right)^2}{\sum_{\alpha} w(T_k^{\alpha})^2} = w_k^2 \left(\sum_{\alpha} w(T_k^{\alpha})^2\right)^{-1}. \quad (100)$$



Supplementary Figure 7. Low dimensionality of a trained full-connected network. A) Illustration of the problem definition and the learned parameters (cf. Figure 1 above). B) Plot of the learned decision boundaries which solve the four-class classification problem. C) Plots of the fractional inverse participation ratio $r_k/N_\alpha \in [1/N_\alpha, 1]$ for $k = 1, 2, 3, 4$ in trained and untrained graphs, as input forces $f\mathbf{F}^\rho$ from patterns $\rho = 1, 2, 3, 4$ are presented for increasing strengths $f \in [0, 1]$. D) A visualization showing clustering of the high dimensional $\mathbf{A}(\mathbf{F})$ which record the frenergy for each tree in the tree space, projected into two reduced dimensions using t-SNE. 200 samples of \mathbf{F}^ρ from each class $\rho = 1, 2, 3, 4$ (the data clouds in panel B) are drawn, and the projection $\mathbf{A}(\mathbf{F}^\rho)$ for each input is a point in the graph colored by ρ . The points are additionally sized so that the projected vectors $\mathbf{A}(\mathbf{F})$ is larger if \mathbf{F}^ρ is close to the center of the class ρ . On the left we show the t-SNE plot for the trained graph using a perplexity of 10, and on the right we show the same for the untrained graph. E) Plot of the silhouette score, which measures the quality of clustering in the t-SNE space, as the t-SNE perplexity is increased.



Supplementary Figure 8. Low dimensionality of a trained random network. See SI Figure 7 for a description.

To justify this definition, we note that $\pi_k \propto \sum_{\alpha} w(T_k^{\alpha}) \equiv w_k$. If only one directed tree, say β , contributes to this sum, then $w(T_k^{\alpha}) = w_k \delta_{\alpha\beta}$, and $r_k = 1$. If, instead, every tree contributes evenly, then $w(T_k^{\alpha}) = 1/w_k$, and $r_k = N_{\alpha}$. This definition thus has the desired properties to indicate how delocalized the contribution of spanning tree weights are to the steady-state probability π_k . We use the exponent 2 instead of 4 because the vector components $w(T_k^{\alpha})$ are all non-negative and their natural norm is $w_k = \sum_{\alpha} w(T_k^{\alpha})$, not $\sum_{\alpha} w(T_k^{\alpha})^2$. Mathematically, though, using the exponent 4 instead would also have the desired property that r_k ranges from 1 to N_{α} .

We next study the IPR in untrained ($\theta = \mathbf{0}$) and trained ($\theta = \theta^*$) graphs. The graph shown in Figure 7A was trained to solve a four-class classification task, with the results shown in Figure 7B. We find that the trained network

with $\mathbf{F} = \mathbf{0}$ has a significantly lower value of r_k than the untrained network for each of the output nodes, r_1, r_2, r_3, r_4 (Figure 7C). Additionally, these values depend on which input pattern is presented to the network and the strength at which it is presented. As the input forcing \mathbf{F} is presented more strongly the values of r_k generally decrease in the untrained graph, because certain trees aligned with the input forcing begin to dominate over others. However, in the trained graph, the delocalization r_i does not necessarily decrease when input \mathbf{F}^j is presented because this input is exciting trees other than those which have been trained to contribute to node i . As during development of neural pathways in the brain, training of these networks for a classification task graphs has, via updates to $\boldsymbol{\theta}$, winnowed certain trees from contributing to π_i and reinforced others that help to solve the task [18, 19]. We note that these results are broadly consistent with those described for neural networks and resistor and spring networks in Refs. 16, 17.

To explore which subsets of trees contribute to the networks response, we define a measure of undirected tree activation in terms of the total frenesy $A(T_\alpha) = \sum_{ij \in T_\alpha} (\pi_j W_{ij} + \pi_i W_{ji})$ along the edges of the tree T_α at steady state. We computed the high dimensional vector over tree space $\mathbf{A}(\mathbf{F})$ for hundreds input forces drawn from the four input classes in Figure 7B. We then use t-SNE [20] to reduce the dimensionality of these vectors from N_α to 2 (Figure 7D). This allows easy visualization of the grouping of vectors $\mathbf{A}(\mathbf{F}^\rho)$ according to class ρ . We see in both trained and untrained networks that the tree activation profiles are clustered in this high-dimensional space according to which class the input force is drawn from. The input driving alone accounts for some of this clustering, but we see that the clustering in the trained network is somewhat tighter as measured using the silhouette score in the t-SNE space across a range of t-SNE perplexity values [21].

We repeat this analysis for a different graph which has a random, rather than fully-connected, topology (SI Figure 8). We find again that training reduces the inverse participation ratio, but the clustering of trees according to input driving force is evidently tighter in the untrained graph than in the trained one. This may result from the smaller number of trees (284) in this sparse random graph than in the fully connected six-node graph with the same number of edges (1,296) in SI Figure 7.

VII. SOFTMAX-LIKE FORM OF THE CLASSIFICATION FUNCTION

A. Connection to transformers and modern Hopfield networks

Here we explore connections between classification using Markov networks as treated in this paper and recent developments in machine learning that link modern Hopfield models of associative memory to transformer architectures, which are commonly used in technologies like ChatGPT. We first summarize this link, and then discuss how our work may fit into this emerging picture.

In the classic Hopfield model of associative memory [22], one considers a trained Hamiltonian of spin degrees of freedom \mathbf{x} which involves a sum over patterns $\boldsymbol{\xi}^\rho$ vectors:

$$\mathcal{F}(\mathbf{x}; \{\boldsymbol{\xi}^\rho\}) = - \sum_{\rho} (\boldsymbol{\xi}^\rho \cdot \mathbf{x})^2. \quad (101)$$

This expression is equivalent to the spin-glass energy

$$\mathcal{F}(\mathbf{x}; \{\boldsymbol{\xi}^\rho\}) = - \sum_{\alpha, \beta} x_\alpha J_{\alpha\beta} x_\beta \quad (102)$$

where

$$J_{\alpha\beta} = \sum_{\rho} \xi_\alpha^\rho \xi_\beta^\rho \quad (103)$$

is given by the Hebbian rule. Equation 101 has recently been generalized into a so-called modern Hopfield model as [23, 24]

$$\mathcal{F}(\mathbf{x}; \{\boldsymbol{\xi}^\rho\}) = - \ln \sum_{\rho} F(\boldsymbol{\xi}^\rho \cdot \mathbf{x}) \quad (104)$$

for some non-linear function F . Taking $F(x) = e^x$, which can be viewed as including all orders of a polynomial expansion of $\boldsymbol{\xi}^\rho \cdot \mathbf{x}$, has been shown to provide a significantly larger capacity for pattern memory than in classical counterpart [24]. Additionally, this form leads to an energy gradient-based dynamical update rule for \mathbf{x} which is

equivalent to a softmax formula that compares \mathbf{x} to the input patterns ξ^ρ [24]. This softmax update rule introduces an interesting connection between models of associative memory and transformers, which commonly use softmax functions to compare a query vector \mathbf{x} to a set of key vectors ξ^ρ .

To connect to our work on classification tasks performed by Markov networks, we note that transformers can in some cases be made more computationally efficient using an approximation called linear attention:

$$\text{Softmax}(\mathbf{x}, \xi^\rho) = \frac{e^{\xi^\rho \cdot \mathbf{x}}}{\sum_{\rho'} e^{\xi^{\rho'} \cdot \mathbf{x}}} \approx \frac{\phi_\xi(\xi^\rho) \cdot \phi_x(\mathbf{x})}{\sum_{\rho'} \phi_\xi(\xi^{\rho'}) \cdot \phi_x(\mathbf{x})}, \quad (105)$$

where ϕ_ξ and ϕ_x are non-linear vector functions which featurize the input data and patterns [25]. Interestingly, linear attention has recently been shown to appear naturally in a biophysical model for the computational role played by astrocytes in the brain [26]. In Ref. 26 a linear attention form is described for the dependence of a neural network's terminal layer's activation on those of previous layers, with the astrocyte processes implementing a softmax-like comparison between neural activations and learnable feature weights.

Comparison of Equation 105 and Equation 3 of the main text reveals the way in which classification using Markov networks implements a type of softmax-based attention on the input data \mathbf{F} using the non-linear feature vectors $\psi(i; \theta)$ and $\chi(i, \mathbf{F})$. We note, however, that our feature vectors functions are different functions for each node, i.e., $\psi(i; \theta) \neq \psi(j; \theta)$ and similarly $\chi(j; \mathbf{F}) \neq \chi(i; \mathbf{F})$. In contrast, in the typical linear attention architecture the same non-linear feature functions π are used for each class, and are typically chosen to make the approximation in Equation 105 close [25, 26], so our comparison to linear attention architectures is not exact. We note that qualitative comparisons to other existing machine learning architectures such as kernel-based classifiers [27] may also be possible. We next show how Equation 3 in the main text is derived from the matrix tree theorem.

B. Derivation of Equation 3 in the main text

Starting from Equations 1 and 3 of the main text, we first identify

$$\psi(T_i^\alpha; \theta) \chi(T_i^\alpha, \mathbf{F}) = w(T_i^\alpha, \mathbf{F}; \theta) \quad (106)$$

and can hence interpret the sum over α as a dot product among the vector elements $\psi(T_i^\alpha; \theta)$ and $\chi(T_i^\alpha, \mathbf{F})$. This decomposition is possible because the directed tree weight $w(T_i, \mathbf{F}; \theta)$ depends exponentially on sums involving the parameters θ and input forces \mathbf{F} . Specifically, we have

$$w(T_i^\alpha, \mathbf{F}; \theta) = \exp \left(\sum_k s^E(k, T_i^\alpha) E_k - \sum_{jk} s^B(jk, T_i^\alpha) B_{jk} + \sum_{jk} s^F(jk, T_i^\alpha) F_{jk} \right) \exp \left(\sum_a s^F(a, T_i^\alpha) F_a \right) \quad (107)$$

where $s^E(k, T_i^\alpha) \in \{0, 1\}$ is a structural indicator of whether E_k appears in the product edge rates for the directed tree T_i^α , $s^B(jk, T_i^\alpha) \in \{0, 1\}$ similarly encodes the structural dependence B_{jk} , $s^F(jk, T_i^\alpha) \in \{-1/2, 0, 1/2\}$ does so for the learned parameter F_{ij} , and $s^F(a, T_i^\alpha) \in \{-1/2, 0, 1/2\}$ does so for the input force F_a . We can write this all more compactly as in Equation 106 with

$$\psi(T_i^\alpha; \theta) = \exp(\mathbf{u}_i^\alpha \cdot \theta) \quad (108)$$

and

$$\chi(T_i^\alpha; \mathbf{F}) = \exp(\mathbf{v}_i^\alpha \cdot \mathbf{F}), \quad (109)$$

Here, \mathbf{u}_i^α is a vector with the same dimension as the total number of parameters which contains the structural factors s^E , s^B , s^F for directed tree T_i^α . Similarly, \mathbf{v}_i^α contains the structural factors s^F for T_i^α . A dot product over trees $*$ can be defined in terms of the vector elements $\psi(T_i^\alpha; \theta)$ and $\chi(T_i^\alpha, \mathbf{F})$, and we can write

$$\psi(i; \theta) * \chi(i, \mathbf{F}) = \sum_\alpha \psi(T_i^\alpha; \theta) \chi(T_i^\alpha, \mathbf{F}). \quad (110)$$

Since the elements of $\psi(i; \theta)$ and $\chi(i, \mathbf{F})$ are non-negative, the dot product is as well. With these definitions, we can rewrite the steady-state occupation as in Equation 3 of the main text.

C. Creased landscape of $\ln Z$

Dynamics in modern Hopfield networks are governed by relaxation down the gradient of the energy function (Equation 104), which is a quantity of central interest. By analogy, we are led to consider the quantity

$$\mathcal{F}(\mathbf{F}; \boldsymbol{\theta}) = -\ln Z(\mathbf{F}; \boldsymbol{\theta}) = -\ln \sum_k \psi(k; \boldsymbol{\theta}) * \chi(k, \mathbf{F}). \quad (111)$$

Here, we characterize \mathcal{F} and explore how it differs between trained and untrained networks.

We train the graph shown in Figure 9A to solve a four-class classification problem, and it learns the solution shown in Figure 9B; note that these data clouds are rotated in input space relative to other four-class classification tasks presented in this paper. In Figure 9C we plot the entropy

$$S[\boldsymbol{\pi}] = -\sum_k \pi_k \ln \pi_k \quad (112)$$

of the steady-state distribution over input space. Regions of high entropy tend to localize along the learned decision boundaries, and away from these regions the entropy is generally low because the output nodes dominate the steady state.

In Figures 9D and E we plot $\mathcal{F}(\mathbf{F}; \boldsymbol{\theta})$ for the untrained ($\boldsymbol{\theta} = \mathbf{0}$) and trained ($\boldsymbol{\theta} = \boldsymbol{\theta}^*$) networks. We see that in both cases it is characterized by flat sloping regions that are creased along strips of high curvature $\nabla_{\mathbf{F}}^2 \mathcal{F}(\mathbf{F}; \boldsymbol{\theta})$. In the trained network, some of these creases localize to the learned decision boundaries (Figure 9F). Zooming out reveals that training has not eliminated, but merely shifted, the creases found in the untrained network (Figure 9G). The zoomed-out structure of the gradient $\nabla_{\mathbf{F}} \mathcal{F}(\mathbf{F})$ for both networks is similar; away from where the region of input space where the network was trained, the vector field $\nabla_{\mathbf{F}} \mathcal{F}(\mathbf{F})$ has the same behavior at limiting regions of the domain (9H and I). In particular, both networks contain a diagonal strip of mismatch between the vector fields approaching the top right quadrant of the domain.

To understand this structure, we compute the gradient of \mathcal{F} using the multi-index notation of Equation 2 in the main text

$$-\nabla_{\mathbf{F}} \ln Z = -\frac{\nabla_{\mathbf{F}} Z}{Z} = -\frac{\sum_{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) \nabla_{\mathbf{F}} y^{\mu}(\mathbf{F})}{\sum_{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})} \quad (113)$$

The gradient $\nabla_{\mathbf{F}} y^{\mu}(\mathbf{F})$ can be written as

$$\nabla_{\mathbf{F}} y^{\mu}(\mathbf{F}) = \nabla_{\mathbf{F}} \prod_{a \in \mathcal{A}} e^{\mu_a F_a / 2} \quad (114)$$

$$= \frac{\boldsymbol{\mu}}{2} \prod_{a \in \mathcal{A}} e^{\mu_a F_a / 2} \quad (115)$$

$$= \frac{\boldsymbol{\mu}}{2} y^{\mu}(\mathbf{F}) \quad (116)$$

where $\boldsymbol{\mu}$ is a vector containing the the components of the multi-index μ . We have

$$-\nabla_{\mathbf{F}} \ln Z = -\frac{1}{2} \frac{\sum_{\mu} \boldsymbol{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})}{\sum_{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})} \quad (117)$$

$$= -\frac{1}{2} \sum_k \frac{\sum_{\mu} \boldsymbol{\mu} \zeta_{\mu}^k(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})}{\sum_{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})} \quad (118)$$

$$\equiv -\frac{1}{2} \sum_k \langle \boldsymbol{\mu} \rangle_k. \quad (119)$$

The quantity

$$\langle \boldsymbol{\mu} \rangle_k = \frac{\sum_{\mu} \boldsymbol{\mu} \zeta_{\mu}^k(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})}{\sum_{\mu} \bar{\zeta}_{\mu}(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})} \quad (120)$$

records for node k a direction in input space weighted by the relative contribution of the term $\zeta_{\mu}^k(\boldsymbol{\theta}) y^{\mu}(\mathbf{F})$ to the sum

$\bar{\zeta}_\mu(\boldsymbol{\theta})y^\mu(\mathbf{F})$. The total gradient is proportional to a sum of $\langle \boldsymbol{\mu} \rangle_k$ over all nodes.

In a region of input space for which $\pi_k(\mathbf{F}; \boldsymbol{\theta}) \approx 1$, then $-\nabla_{\mathbf{F}} \ln Z \approx -(1/2)\langle \boldsymbol{\mu} \rangle_k$. At decision boundaries, where the steady-state occupation transitions to, say, $\pi_{k'}(\mathbf{F}; \boldsymbol{\theta}) \approx 1$, we will generally expect a region of high curvature in \mathcal{F} if $\langle \boldsymbol{\mu} \rangle_k \neq \langle \boldsymbol{\mu} \rangle_{k'}$ because the gradient is transitioning from $(1/2)\langle \boldsymbol{\mu} \rangle_k$ to $(1/2)\langle \boldsymbol{\mu} \rangle_{k'}$. Thus, a decision boundary will tend to co-localize to a strip of high curvature in \mathcal{F} .

While some creases can thus be created through training, by drawing new decision boundaries in input space, some creases are more fixed and result from the topological properties of the network. These creases do not necessarily co-localize with decision boundaries. For example, the horizontal and vertical creases parallel to the axes as well as the diagonal crease going off too the point $F_a \rightarrow \infty$, $F_b \rightarrow \infty$ appear in both the untrained and trained networks in Figures 9H and I. These result from mismatched limiting behavior of $\nabla_{\mathbf{F}} \mathcal{F}$ at the boundaries of the domain. For example, in Figure 9I, the mismatch between the limits at $\mathbf{F} = (-\infty, \infty)$ and $\mathbf{F} = (-\infty, -\infty)$ produces the crease separating the blue and magenta regions. Additionally, the order of the limit matters. In the top right quadrant, near $\mathbf{F} = (\infty, \infty)$, it can be shown that

$$\lim_{F_a \rightarrow \infty} \lim_{F_b \rightarrow \infty} \nabla_{\mathbf{F}} \mathcal{F} \neq \lim_{F_b \rightarrow \infty} \lim_{F_a \rightarrow \infty} \nabla_{\mathbf{F}} \mathcal{F}. \quad (121)$$

As a result, there is a different vector field when approaching $\mathbf{F} = (\infty, \infty)$ along F_a and then F_b than in the other order. This produces a crease along the diagonal. These creases which encode the behavior at the boundaries cannot be removed through training, because we assume training cannot set coefficients exactly to zero, it can only make them small. For large enough \mathbf{F} , these terms will still eventually dominate and yield the same limiting behavior as in the untrained network.

To summarize, the quantity \mathcal{F} , the analogous quantity to the energy in modern Hopfield networks, encodes both global, topological information about the Markov network through creases which result from mismatched behavior at the boundaries, as well as local information which results from training that produces new decision boundaries in the input space. We leave to future work further exploration of possible connections between computations using Markov networks and other machine learning models.

VIII. HOW NON-EQUILIBRIUM DRIVING ALLOWS EXPRESSIVITY

A. Expressivity from the parameters E_j , B_{ij} , and F_{ij}

In the absence of any non-equilibrium driving, either through the learned parameters F_{ij} or the input variables F_a , the steady-state distribution is a Boltzmann form $\pi_i \propto e^{-E_i}$ and does not depend on the B_{ij} parameters. Beating this restrictive functional form and achieving non-trivial classification expressivity thus requires non-equilibrium driving. From the representation of the linear attention-like form of the matrix-tree expression in Equation 3 of the main text, it is apparent that the key to successfully learning the classification task is the freedom to move the vectors $\boldsymbol{\psi}(i; \boldsymbol{\theta})$ around, so that for the output nodes these vectors have large dot products with the input force vectors $\boldsymbol{\chi}(i, \mathbf{F})$ when \mathbf{F} is in the corresponding region of input space. We thus ask how each of the three types of parameters E_j , B_{ij} , and F_{ij} affect this freedom.

Let us first see how to transform a vector $\boldsymbol{\psi}(i; \boldsymbol{\theta})$ rooted at node i into a vector $\boldsymbol{\psi}(j; \boldsymbol{\theta})$ rooted at node j . To re-root the directed tree at node j into that rooted at node i will require flipping the edges $k \leftarrow l$ which connect nodes i and j in the tree T_α^i . This implies dividing the corresponding rates W_{kl} from the product in the vector element $\psi(T_\alpha^i; \boldsymbol{\theta})$ and multiplying by W_{lk} . We define the product of these ratios for all flipped edges as $g^\alpha(i, j; \boldsymbol{\theta})$. We then have

$$\psi(T_j^\alpha; \boldsymbol{\theta}) = g^\alpha(i, j; \boldsymbol{\theta}) \psi(T_i^\alpha; \boldsymbol{\theta}) \quad (122)$$

and note that $g^\alpha(j, i; \boldsymbol{\theta}) = 1/g^\alpha(i, j; \boldsymbol{\theta})$. Now, the product of these ratios W_{lk}/W_{kl} can only depend on F_{kl} and the energies E_i and E_j , and we can write

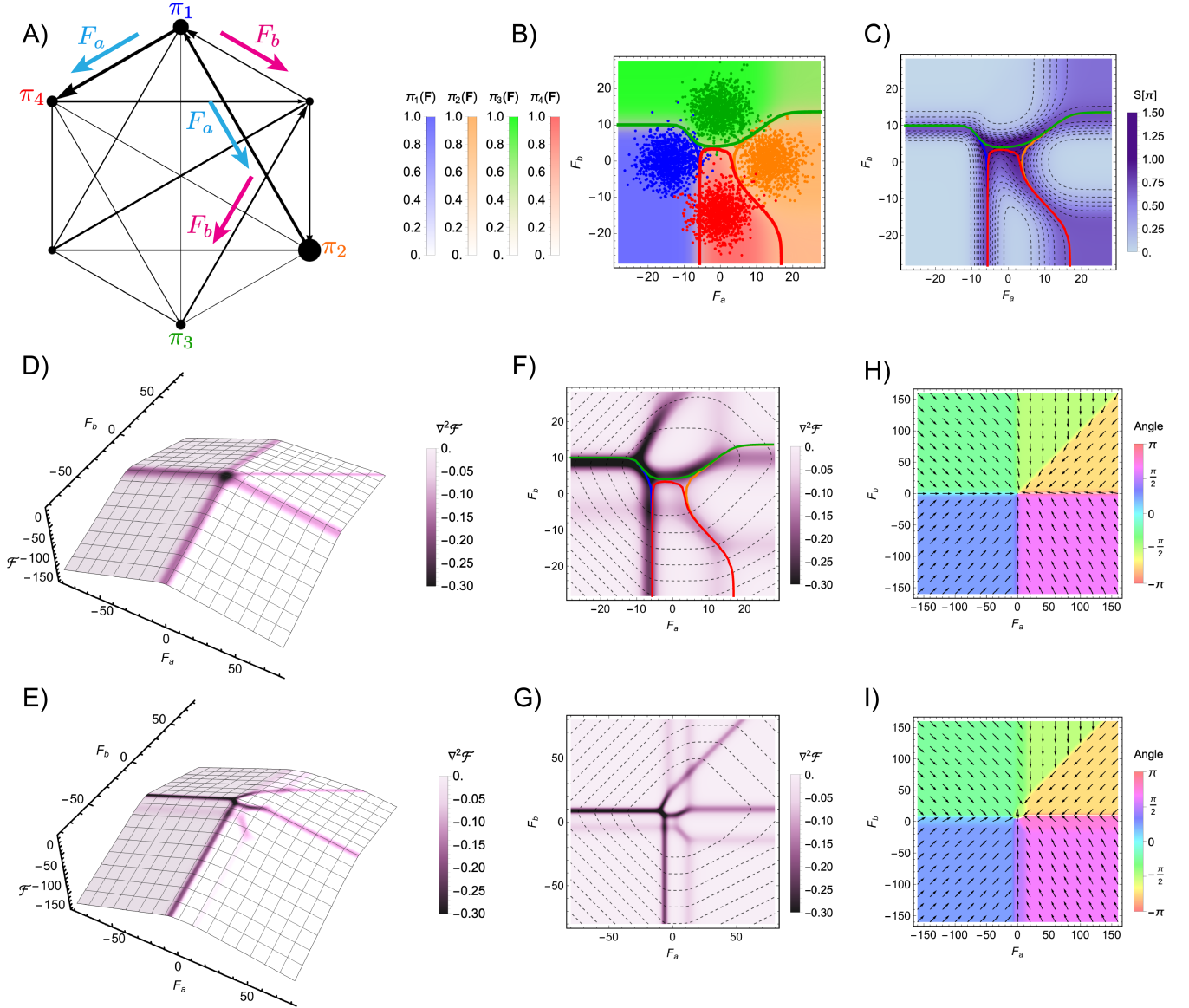
$$g^\alpha(i, j; \boldsymbol{\theta}) = g_E^\alpha(i, j; \boldsymbol{\theta}_E) g_F^\alpha(i, j; \boldsymbol{\theta}_F) \quad (123)$$

where

$$g_E^\alpha(i, j; \boldsymbol{\theta}_E) = e^{E_i - E_j} \quad (124)$$

and

$$g_F^\alpha(i, j; \boldsymbol{\theta}_F) = e^{-\sum_{kl} F_{kl}}. \quad (125)$$

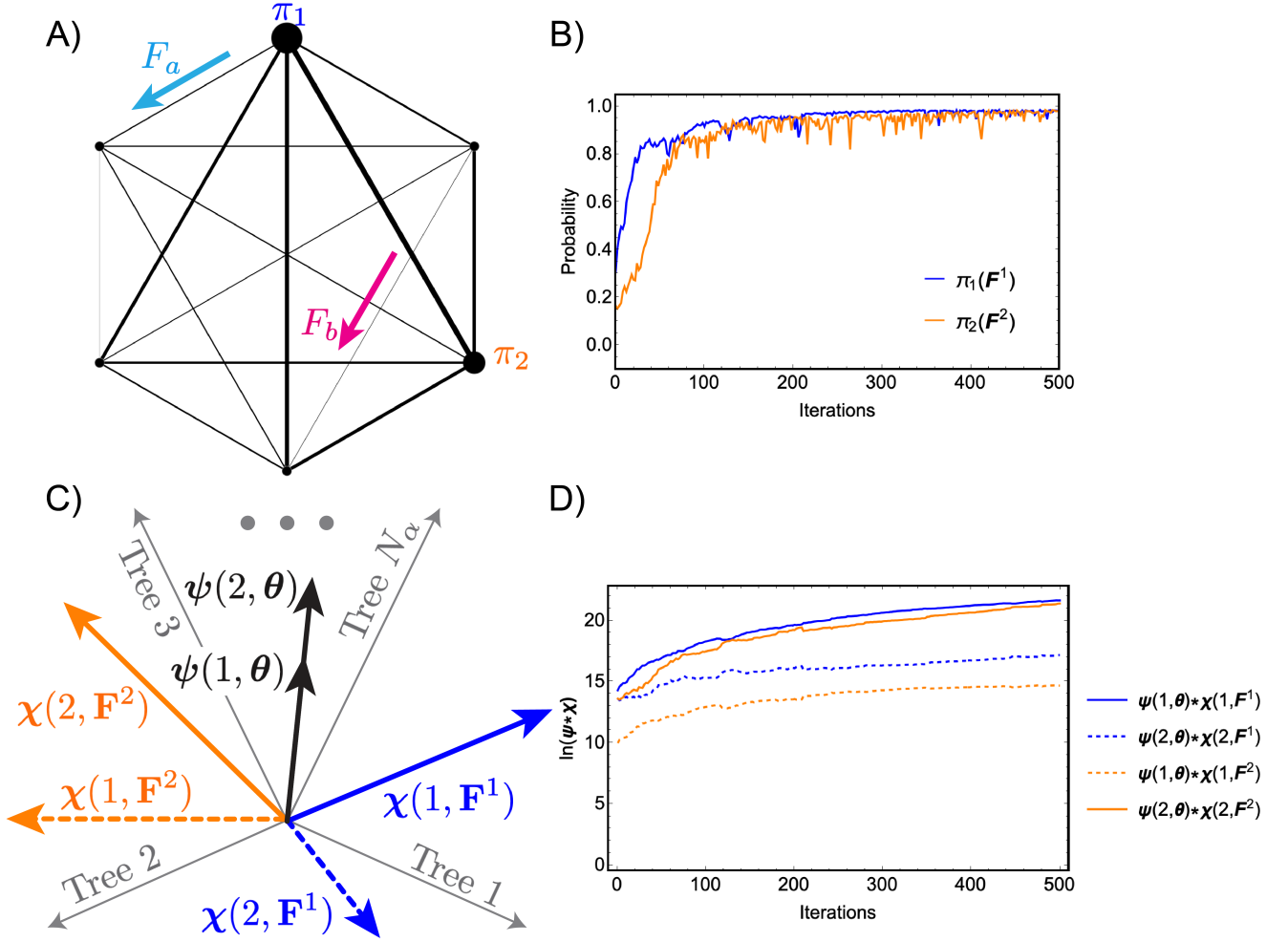


Supplementary Figure 9. Creased landscape of \mathcal{F} . A) Illustration of the problem definition and the learned parameters (cf. Figure 1 above). B) Plot of the learned decision boundaries which solve the four-class classification problem. C) Combination density and contour plot of the entropy $S[\pi]$ in for the trained network. D) Plot of \mathcal{F} for the untrained network (with $\theta = 0$). E) Plot of \mathcal{F} for the trained network. F) Density plot of $\nabla_{\mathbf{F}}^2 \mathcal{F}$ for the trained network. Contours are drawn at fixed values of \mathcal{F} . The decision of panel B boundaries are overlayed. G) Same as panel F, but zoomed out to show more of the large-scale structure. H) Vector plot of $\nabla_{\mathbf{F}} \mathcal{F}$ of the untrained graph. Arrows are not drawn according to vector magnitude, only to show direction. Coloring also indicates the direction of the vectors. I) Same as panel H, but for the trained graph.

The primed summation is only over edges flipped in the re-rooting process, and θ_E and θ_F contain only the E_j and F_{ij} parameters respectively. Collecting the factors $g^\alpha(j, i; \theta)$ for each α along the diagonal of a $N_\alpha \times N_\alpha$ matrix $\mathbf{G}(i, j; \theta)$, we can write in matrix-vector notation

$$\psi(i; \theta) = \mathbf{G}(i, j; \theta) * \psi(j; \theta) = \mathbf{G}_E(i, j; \theta_E) * \mathbf{G}_F(i, j; \theta_F) * \psi(j; \theta). \quad (126)$$

The matrix $\mathbf{G}_E(i, j; \theta_E) = e^{E_i - E_j} \mathbf{I}$ is proportional to the identity matrix, but the matrix $\mathbf{G}_F(i, j; \theta_E)$ can have differing elements along the diagonal which allows transforming $\psi(j; \theta)$ anisotropically.



Supplementary Figure 10. Successful learning in a fully connected 6 node graph where F_{ij} parameters are not learned but the inputs are presented through \mathbf{F} . A) Illustration of the problem definition and the learned parameters (cf. Figure 1 above). B) Plot showing training convergence during training. C) Schematic illustration of the tree vector space for this problem. D) Plot showing how different pairs of vector dot products evolve during training.

If we first assume that we only adjust E_j values during training (i.e., $B_{ij} = F_{ij} = 0$), then this analysis implies that

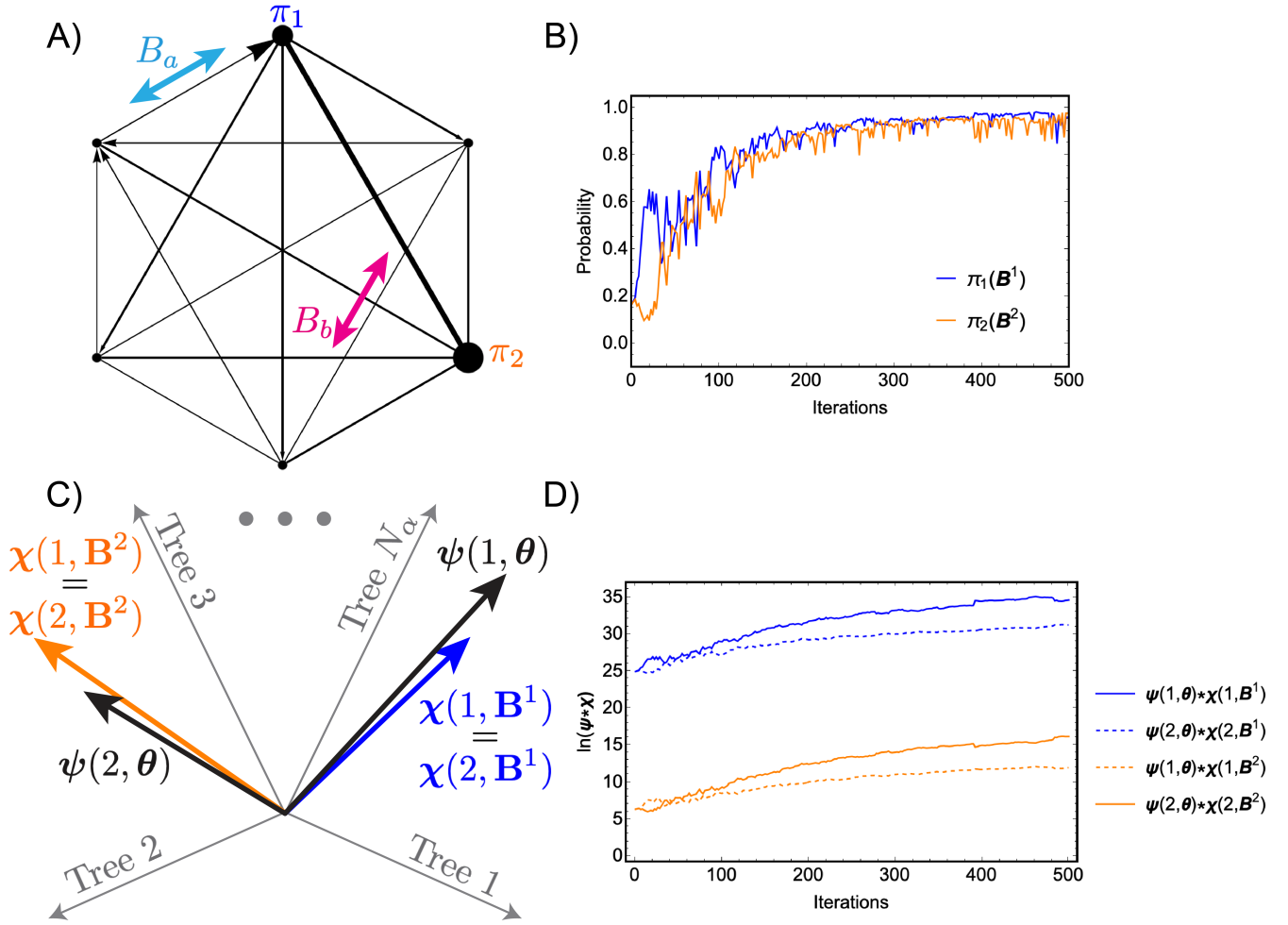
$$\psi(i; \theta) = e^{E_{\text{tot}} - E_i} \mathbf{1} \quad (127)$$

where $E_{\text{tot}} = \sum_k E_k$ and $\mathbf{1}$ is the vector of ones over tree space. This severely limits the available space the vectors $\psi(i; \theta)$ can occupy. If we additionally learn B_{ij} parameters, then

$$\psi(i; \theta) = e^{E_{\text{tot}} - E_i} \tilde{\psi}(\theta) \quad (128)$$

where $\tilde{\psi}(\theta)$ is a single learnable vector that is shared among all nodes in the network. If we learn F_{ij} 's, then the vectors $\psi(i; \theta)$ can differ from each other in an element-by-element way, representing a large gain in expressivity.

To illustrate the interpretation of the classification task in terms of the vectors $\psi(i, \theta)$ and $\chi(i, \mathbf{F})$, and how successful classification depends on non-equilibrium driving, we next consider three examples. We consider here two ways of presenting inputs to the network: through contributions \mathbf{F} to the anti-symmetric, non-equilibrium rate parameters, or through contributions \mathbf{B} to the symmetric, equilibrium rate parameters. If presenting inputs through \mathbf{F} , then for the reasons just outlined the vectors $\chi(i, \mathbf{F})$ and $\chi(j, \mathbf{F})$ can differ from each other for $i \neq j$. If presenting inputs through \mathbf{B} , however, then $\chi(i, \mathbf{F}) = \chi(j, \mathbf{F})$ for all i, j .



Supplementary Figure 11. Successful learning in a fully connected 6 node graph where F_{ij} parameters are learned and the inputs are presented through \mathbf{B} . A) Illustration of the problem definition and the learned parameters (cf. Figure 1 above). B) Plot showing training convergence during training. C) Schematic illustration of the tree vector space for this problem. D) Plot showing how different pairs of vector dot products evolve during training.

B. Three examples

1. Inputs through \mathbf{F} , not learning F_{ij}

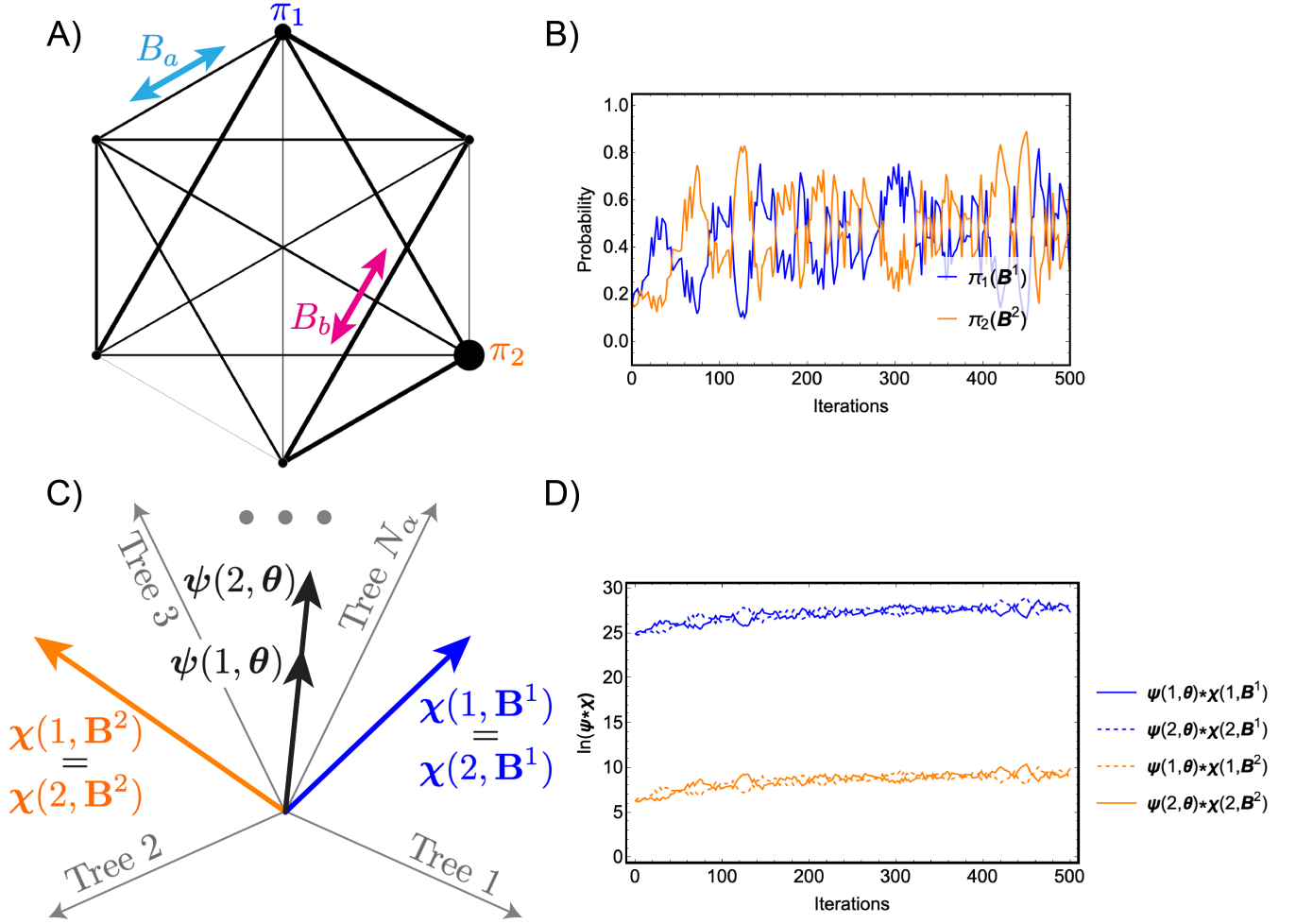
First, we consider an example in which we present inputs through \mathbf{F} but are not allowed to learn F_{ij} parameters. In this case learning is still generally possible, as illustrated in Figure 10. Assuming the output nodes are labeled 1 and 2, the goal of learning in this case is to position the shared learnable vector $\tilde{\psi}(\theta)$ so that

$$\psi(1; \theta) * \chi(1, \mathbf{F}^1) > \psi(2; \theta) * \chi(2, \mathbf{F}^1) \quad (129)$$

and

$$\psi(2; \theta) * \chi(2, \mathbf{F}^2) > \psi(2; \theta) * \chi(2, \mathbf{F}^1) \quad (130)$$

where \mathbf{F}^1 and \mathbf{F}^2 are typical inputs from classes 1 and 2. To achieve this the vector $\tilde{\psi}(\theta)$ should move near the subspace spanned by $\chi(1, \mathbf{F}^1)$ and $\chi(2, \mathbf{F}^2)$. It is apparent from studying how the dot products $\tilde{\psi}(\theta) * \chi(1, \mathbf{F}^1)$ and $\tilde{\psi}(\theta) * \chi(2, \mathbf{F}^2)$ change during training that something like this happens in Figure 10C.



Supplementary Figure 12. Unsuccessful learning in a fully connected 6 node graph where F_{ij} parameters are not learned and the inputs are presented through \mathbf{B} . A) Illustration of the problem definition and the learned parameters (cf. Figure 1 above). B) Plot showing training convergence during training. C) Schematic illustration of the tree vector space for this problem. D) Plot showing how different pairs of vector dot products evolve during training.

2. Inputs through \mathbf{B} , learning F_{ij}

Second, we consider an example in which we present inputs through \mathbf{B} but are allowed to learn F_{ij} parameters (Figure 11). In this case learning is also possible, because although the input vectors $\chi(i, \mathbf{B})$ are equal for every i and j , they differ as \mathbf{B} is varied. This makes it possible to move the independent learnable vectors $\psi(i; \theta)$ to satisfy Equations 129 and 130.

3. Inputs through \mathbf{B} , not learning F_{ij}

Finally, we consider an example in which we present inputs through \mathbf{B} and cannot learn F_{ij} parameters (Figure 12). In this case learning is not possible, because the vectorial part of $\psi(i, \theta)$ is shared for each i , and the input vectors $\chi(i, \mathbf{B})$ are also equal for each i . This prevents satisfying Equations 129 and 130, which in this case simplify to the contradictory requirements

$$e^{E_1} < e^{E_2} \text{ and } e^{E_2} < e^{E_1}. \quad (131)$$

If we were to gradually increase the threshold on the absolute values of the F_{ij} parameters, then the vectorial parts of $\psi(i, \boldsymbol{\theta})$ could start to split by from each other by increasing amounts, so that Figure 12C approaches Figure 11C. Figure 5 in the main text illustrates how this continuously allows for more successful classification accuracy.

IX. MUTUAL INFORMATION FORMULATION OF THE COMPUTATION

A. Defining the mutual information

We have a Markov process that computes a steady state $\boldsymbol{\pi}(\mathbf{F}; \boldsymbol{\theta}) \in \mathbb{R}^{N_n}$. We assume that we have defined an input distribution $p_{\mathcal{F}}(\mathbf{F})$ which encodes the environment, i.e., the probabilities of the inputs \mathbf{F} . The mutual information between two correlated distributions $p_X(x)$ and $p_Y(y)$ is defined in general as

$$I(X; Y) = \int dx dy p_{(X,Y)}(x, y) \ln \left(\frac{p_{(X,Y)}(x, y)}{p_X(x)p_Y(y)} \right). \quad (132)$$

Here $p_{(X,Y)}(x, y)$ is the joint distribution of the two variables, and $p_X(x)$ and $p_Y(y)$ are the two marginal distributions. For our system, we have

$$I(\mathcal{I}; \mathcal{F}) = \sum_{k=1}^{N_n} \int d\mathbf{F} p_{(\mathcal{I}, \mathcal{F})}(k, \mathbf{F}) \ln \left(\frac{p_{(\mathcal{I}, \mathcal{F})}(k, \mathbf{F})}{p_{\mathcal{I}}(k)p_{\mathcal{F}}(\mathbf{F})} \right) \quad (133)$$

where \mathcal{I} represents the node indices. We compute the joint distribution as

$$p_{(\mathcal{I}, \mathcal{F})}(k, \mathbf{F}) = p(k|\mathbf{F})p_{\mathcal{F}}(\mathbf{F}) = \pi_k(\mathbf{F})p_{\mathcal{F}}(\mathbf{F}) \quad (134)$$

where $\pi_k(\mathbf{F})$ is the steady-state probability at node k under input \mathbf{F} . We thus have

$$I(\mathcal{I}; \mathcal{F}) = \sum_{k=1}^{N_n} \int d\mathbf{F} \pi_k(\mathbf{F})p_{\mathcal{F}}(\mathbf{F}) \ln \left(\frac{\pi_k(\mathbf{F})}{p_{\mathcal{I}}(k)} \right). \quad (135)$$

We compute the marginal

$$p_{\mathcal{I}}(k) = \int d\mathbf{F} p(k, \mathbf{F}) = \int d\mathbf{F} \pi_k(\mathbf{F})p_{\mathcal{F}}(\mathbf{F}) \equiv \nu_k, \quad (136)$$

in terms of which we have

$$I(\mathcal{I}; \mathcal{F}) = S_{\nu} + \sum_{k=1}^{N_n} \int d\mathbf{F} \pi_k(\mathbf{F})p_{\mathcal{F}}(\mathbf{F}) \ln \pi_k(\mathbf{F}) \quad (137)$$

where

$$S_{\nu} = - \sum_k \nu_k \ln \nu_k. \quad (138)$$

Further identifying

$$S_{\pi}(\mathbf{F}) = - \sum_k \pi_k(\mathbf{F}) \ln \pi_k(\mathbf{F}) \quad (139)$$

we can write

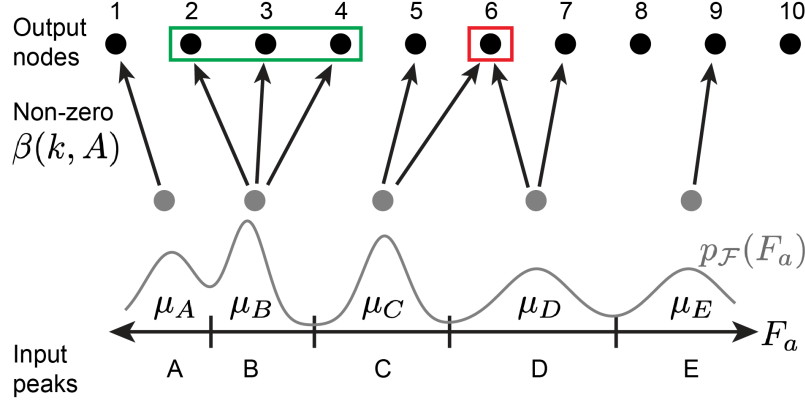
$$I(\mathcal{I}; \mathcal{F}) = S_{\nu} - \int d\mathbf{F} p_{\mathcal{F}}(\mathbf{F}) S_{\pi}(\mathbf{F}) \equiv S_{\mu} - \langle S_{\pi}(\mathbf{F}) \rangle_{p_{\mathcal{F}}}. \quad (140)$$

B. Theory for optimality of one-hot encoding

Here we consider ideal encodings for a peaked input distribution. As an approximation to how training may assign different nodes to be on for different regions of input space corresponding to different peaks, we ignore the transitions between these regions and use a piecewise constant function

$$\pi_k(\mathbf{F}) = \sum_A \mathbb{1}_{\mathcal{F}_A}(\mathbf{F})\beta(k, A). \quad (141)$$

Here, $\mathbb{1}_{\mathcal{F}_A}(\mathbf{F})$ is an indicator function for \mathbf{F} belonging to \mathcal{F}_A , the set of all values of \mathbf{F} assigned to peak A .



Supplementary Figure 13. Schematic illustration the set-up for our mutual information optimization theory, in which various regions of the input space (here 1D) are assigned to peaks and we approximate the output nodes as being uniform within these regions. Each peak has a measure μ_A given by integrating the density over support of the peak. Non-zero values of $\beta(k, A)$, which are the constant values assumed by the nodes when within each region of input space, are shown as black arrows. The green box indicates a scenario in which multiple output nodes have been assigned to the same peak. The box indicates an output node which has been assigned to more than one peak.

For a one-hot encoding, the term $\beta(k, A)$ takes values in $\{0, 1\}$ depending on if node k has been assigned to read out peak A . We then have

$$\nu_k = \int d\mathbf{F} p_{\mathcal{F}}(\mathbf{F}) \pi_k(\mathbf{F}) = \int_{\mathcal{F}_{A(k)}} d\mathbf{F} p_{\mathcal{F}}(\mathbf{F}) \equiv \mu_{A(k)}, \quad (142)$$

where $A(k)$ is the peak to which node k has been assigned, and μ_A is the measure of the set \mathcal{F}_A . For every value of \mathbf{F} we have $S_{\pi}(\mathbf{F}) = 0$ because only one node is on at a time, so that

$$I(\mathcal{I}; \mathcal{F}) = - \sum_k \mu_{A(k)} \ln \mu_{A(k)} \quad (143)$$

for a one-hot encoding. Because each peak is assigned to one node, we can rewrite this as

$$I(\mathcal{I}; \mathcal{F}) = - \sum_A \mu_A \ln \mu_A, \quad (144)$$

which is the entropy of the set of measures of each input peak: observing a one-hot encoded output reduces uncertainty by this amount of entropy.

Now we consider how \mathcal{I} degrades in the more general case when the node assignments $\beta(k, A)$ can take on any value in $[0, 1]$ and a node may be assigned to more than one peak. We again take a piecewise constant approximation of $\beta(k, A)$ on \mathcal{F}_A for simplicity. We must have that $\sum_k \beta(k, A) = 1$ for every A by normalization of the Markov state, but we otherwise these numbers are general. We have that

$$\nu_k = \sum_A \beta(k, A) \mu_A, \quad (145)$$

and

$$S_\nu = - \sum_k \sum_A \beta(k, A) \mu_A \ln \left(\sum_B \beta(k, B) \mu_B \right). \quad (146)$$

We also have

$$\begin{aligned} \langle S_\pi(\mathbf{F}) \rangle_{p_{\mathcal{F}}} &= - \sum_A \int_{\mathcal{F}_A} d\mathbf{F} p_{\mathcal{F}}(\mathbf{F}) \sum_k \beta(k, A) \ln(\beta(k, A)) \\ &= - \sum_A \mu_A \sum_k \beta(k, A) \ln \beta(k, A). \end{aligned} \quad (147)$$

The mutual information can be expressed as

$$\begin{aligned} I(\mathcal{I}; \mathcal{F}) &= \sum_k \sum_A \beta(k, A) \mu_A \ln \left(\frac{\beta(k, A)}{\sum_B \beta(k, B) \mu_B} \right) \\ &= \sum_k \sum_A \beta(k, A) \mu_A \ln \left(\frac{\beta(k, A) \mu_A}{\sum_B \beta(k, B) \mu_B} \right) \\ &\quad - \sum_A \mu_A \ln \mu_A. \end{aligned} \quad (148)$$

The last term is equal to the mutual information in the case of a one-hot encoding, Equation 144. The arguments of the logarithms in the first term can be at most unity, and hence all terms in the first sum are negative or zero and cannot increase the mutual information. To set them to zero, then for each A for which $\beta(k, A) \neq 0$ we must have

$$\beta(k, A) \mu_A = \sum_B \beta(k, B) \mu_B, \quad (149)$$

implying $\beta(k, B) = 0 \ \forall B \neq A$. In words, this means that if node k reads out peak A then it cannot read out any other peak, i.e., no output node should have an assignment to more than one peak (see red box in Figure 13). This makes sense, because if we observed this node being on we would not be certain in which role it is acting, i.e., which peak it is meant to indicate the presence of.

We can ask if this is true in the other direction, that is, does it degrade $I(\mathcal{I}; \mathcal{F})$ if more than one node is assigned to a certain peak, provided that these nodes are only assigned to this peak? We can see that this does not degrade $I(\mathcal{I}; \mathcal{F})$. Mathematically, this is the case because due to the unique assignment of peaks to nodes (as just described), $I(\mathcal{I}; \mathcal{F})$ reduces to the one-hot encoding limit, Equation 144, which does not depend on the specific values of $\beta(k, A)$. If every node reads out just one peak, then observing any of the nodes for this peak uniquely specifies it (see green box in Figure 13). This situation is depicted in Figure 13.

To summarize, for this analysis we assume that the Markov steady state takes on approximately piecewise constant sets of values over the input space. Interpreting these as classes is somewhat arbitrary; they are simply the regions over which the output nodes are assumed to take different constant values over a multiply peaked input distribution. In this picture, the most mutual information that can be obtained through a choice of different constant values $\beta(k, A)$ in each peak is the entropy of the set of measures of each class. This will be obtained for any choice of $\beta(k, A)$ values in which each node only reads out to one peak, and multiple nodes can simultaneously read out the same peak without decreasing the mutual information.

Finally, we may ask how easy it is to reach this bound as a function of the input multiplicity M . If $M = 1$, then due to monotonicity the steady-state values may be forced to be assigned to multiple classes, decreasing the mutual information. If the steady-state values are non-monotonic, then they can read out for a finite band of the input space and more easily read out individual peaks.

X. INCORPORATING BIMOLECULAR REACTIONS

Here, we preliminarily study the effects of introducing non-linearities into the reaction network scheme to investigate their impact on expressivity. Non-linearities can be introduced in various ways, and some types can be approximated under assumptions of timescale separation or large reservoir sizes, in some cases reducing to linear dynamics; see Refs. 28, 29 for examples. Our aim here is to illustrate, through a simple example, how a non-linearity in the

dynamics can yield non-monotonic response functions even with $M = 1$, serving to effectively increase the value of M .

We first consider the linear system illustrated in SI Figure 14A, with three species A , B , and C obeying the dynamics

$$\frac{\partial p_A}{\partial t} = W_{AB}p_B + W_{AC}p_C - W_{CAPA} - W_{BAPA}, \quad (150)$$

$$\frac{\partial p_B}{\partial t} = W_{BAPA} + W_{BC}e^{F_a/2}p_C - W_{AB}p_B - W_{CBe}^{-F_a/2}p_B, \quad (151)$$

$$\frac{\partial p_C}{\partial t} = W_{CAPA} + W_{CBe}^{-F_a/2}p_B - W_{AC}p_C - W_{BC}e^{F_a/2}p_C, \quad (152)$$

where we have indicated the input driving F_a along the edge $C \rightarrow B$. Using the matrix-tree theorem to solve for the steady-state occupancies as a function of F_a , we obtain the response curves shown in the bottom panel of SI Figure 14A. Because $M = 1$ for this linear reaction network, there can be no non-monotonicity for any of the responses at the three nodes.

We next introduce a mild form of non-linearity by letting the rate from $A \rightarrow B$ depend on the occupancy p_C in such a way that C is not consumed by the reaction (SI Figure 14B). Thus, the term $W_{AB}p_A$ in the dynamics for p_A and p_B is replaced with $W_{AB}p_A p_C$, while the dynamics for C remain unchanged:

$$\frac{\partial p_A}{\partial t} = W_{AB}p_B + W_{AC}p_C - W_{CAPA} - W_{BAPA}p_C, \quad (153)$$

$$\frac{\partial p_B}{\partial t} = W_{BAPA}p_C + W_{BC}e^{F_a/2}p_C - W_{AB}p_B - W_{CBe}^{-F_a/2}p_B, \quad (154)$$

$$\frac{\partial p_C}{\partial t} = W_{CAPA} + W_{CBe}^{-F_a/2}p_B - W_{AC}p_C - W_{BC}e^{F_a/2}p_C. \quad (155)$$

This system would reduce to a first-order reaction network if not for the factor of p_C multiplying the rate W_{BAPA} . Since the matrix-tree theorem does not directly apply, we algebraically reinterpret the system. We first replace p_C with a constant π_C , reducing the system to the linear scheme described above with an effective rate $W_{BA}\pi_C$ in place of W_{BA} . Applying the matrix-tree theorem then gives expressions of the form:

$$\pi_A = h_A(\pi_C, F_a), \quad (156)$$

$$\pi_B = h_B(\pi_C, F_a), \quad (157)$$

$$\pi_C = h_C(\pi_C, F_a). \quad (158)$$

We solve the last equation self-consistently for $\pi_C(F_a)$ and use this solution to determine $\pi_A(F_a)$ and $\pi_B(F_a)$. We verified the validity of this approach by numerically integrating the ODE system, and the resulting steady-state distribution is plotted in the bottom panel of SI Figure 14B. Notably, the occupancy at node B exhibits a non-monotonic dependence on F_a despite $M = 1$.

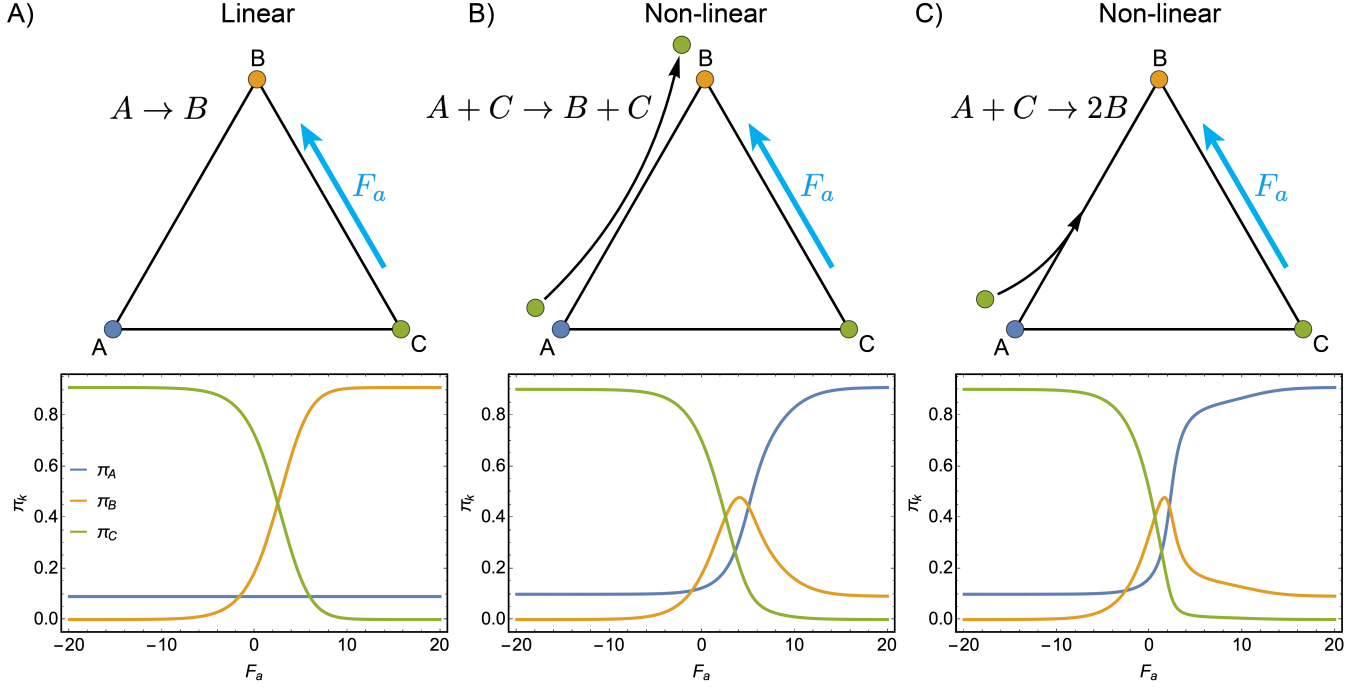
This approach provides a simple insight into how non-monotonicity in $\pi_B(F_a)$ arises. The function $h_C(\pi_C, F_a)$ is a rational polynomial of $e^{F_a/2}$ of order two, but solving $h_C(\pi_C, F_a) = \pi_C$ for π_C introduces a more complex algebraic dependence of this occupancy on F_a . Consequently, substituting this result into $h_B(\pi_C, F_a)$ results in $\pi_B(F_a)$ acquiring a similarly intricate dependence, which allows for non-monotonicity. The more complex algebraic dependence of the steady state on the input driving in these non-linear systems can be interpreted as “effectively” increasing the value of M , which controls the number of ways each input influences the steady-state response. One can also see that increasing M for non-linear reaction networks would qualitatively have the same effect as in linear networks by further increasing the algebraic complexity of the steady-state response, due to the order of the rational polynomial functions $h_k(F_a)$ increasing.

We leave a full exploration of how non-linear dynamics affect the number of critical points in steady-state occupancies to future work. However, we consider one last example with a stronger type of non-linearity in the dynamics. Instead of letting C chaperone the reaction $A \rightarrow B$, we now let it be consumed, preserving total concentration through the reaction $A + C \rightarrow 2B$ (SI Figure 14C). The modified dynamics are:

$$\frac{\partial p_A}{\partial t} = W_{AB}p_B + W_{AC}p_C - W_{CAPA} - W_{BAPA}p_C, \quad (159)$$

$$\frac{\partial p_B}{\partial t} = 2W_{BAPA}p_C + W_{BC}e^{F_a/2}p_C - W_{AB}p_B - W_{CBe}^{-F_a/2}p_B, \quad (160)$$

$$\frac{\partial p_C}{\partial t} = W_{CAPA} + W_{CBe}^{-F_a/2}p_B - W_{AC}p_C - W_{BC}e^{F_a/2}p_C - W_{BAPA}p_C. \quad (161)$$



Supplementary Figure 14. Illustration of how non-linear dynamics can yield non-monotonic response functions even when $M = 1$. A) A linear three-state network is illustrated with the driving force along rate $C \rightarrow B$ labeled. The steady-state occupancies at the three nodes as functions of F_a are plotted in the bottom panel. B) A non-linear three-state network in which C chaperones the transition $A \rightarrow B$ but is not consumed by it. C) A non-linear three-state network in which C is consumed in the reaction $A + C \rightarrow 2B$. For each of these plots we use the parameters $(W_{AB}, W_{AC}, W_{BC}, W_{BA}, W_{CA}, W_{CB}) = (1, 1, 0.05, 10, 0.1, 4)$.

To convert these equations into an effective linear form, we replace W_{BA} with $W_{BA}\pi_C$ and $W_{BC}e^{F_a/2}$ with $W_{BC}e^{F_a/2} + W_{BA}\pi_A$. This transformation produces a linear system to which the matrix-tree theorem applies, yielding:

$$\pi_A = h_A(\pi_A, \pi_C, F_a), \quad (162)$$

$$\pi_B = h_B(\pi_A, \pi_C, F_a), \quad (163)$$

$$\pi_C = h_C(\pi_A, \pi_C, F_a). \quad (164)$$

Solving these equations self-consistently gives the steady-state occupancies as functions of F_a , plotted in the bottom panel of SI Figure 14C. The algebraic expressions here are even more intricate than in the previous example, again exhibiting non-monotonic dependencies on F_a despite $M = 1$.

SUPPLEMENTARY REFERENCES

- [1] Menachem Stern, Daniel Hexner, Jason W Rocks, and Andrea J Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2):021045, 2021.
- [2] Menachem Stern and Arvind Murugan. Learning without neurons in physical systems. *Annual Review of Condensed Matter Physics*, 14:417–441, 2023.
- [3] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.
- [4] Benjamin Scellier and Yoshua Bengio. Equivalence of equilibrium propagation and recurrent backpropagation. *Neural computation*, 31(2):312–329, 2019.
- [5] Easun Arunachalam and Milo M Lin. Information gain limit of biomolecular computation. *Physical Review Letters*, 134(14):148401, 2025.
- [6] Jacob Parres-Gold, Matthew Levine, Benjamin Emert, Andrew Stuart, and Michael B Elowitz. Contextual computation by competitive protein dimerization networks. *Cell*, 2025.
- [7] Adip Jhaveri, Spencer Loggia, Yian Qian, and Margaret E Johnson. Discovering optimal kinetic pathways for self-assembly using automatic differentiation. *Proceedings of the National Academy of Sciences*, 121(19):e2403384121, 2024.

- [8] Jürgen Schnakenberg. Network theory of microscopic and macroscopic behavior of master equation systems. *Reviews of Modern physics*, 48(4):571, 1976.
- [9] Jeremy A Owen, Todd R Gingrich, and Jordan M Horowitz. Universal thermodynamic bounds on nonequilibrium response with biochemical applications. *Physical Review X*, 10(1):011066, 2020.
- [10] Gabriela Fernandes Martins and Jordan M Horowitz. Topologically constrained fluctuations and thermodynamics regulate nonequilibrium response. *Physical Review E*, 108(4):044113, 2023.
- [11] Timur Aslyamov and Massimiliano Esposito. Nonequilibrium response for markov jump processes: Exact results and tight bounds. *Physical Review Letters*, 132(3):037101, 2024.
- [12] Carlos Floyd, Aaron R Dinner, and Suriyanarayanan Vaikuntanathan. Learning to control non-equilibrium dynamics using local imperfect gradients. *arXiv preprint arXiv:2404.03798*, 2024.
- [13] WG Bardsley and RMW Wood. Critical points and sigmoidicity of positive rational functions. *The American Mathematical Monthly*, 92(1):37–48, 1985.
- [14] Yuhai Tu. The nonequilibrium mechanism for ultrasensitivity in a biological switch: sensing by maxwell’s demons. *Proceedings of the National Academy of Sciences*, 105(33):11737–11741, 2008.
- [15] Jeremy A Owen and Jordan M Horowitz. Size limits the sensitivity of kinetic schemes. *Nature Communications*, 14(1):1280, 2023.
- [16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.
- [17] Menachem Stern, Andrea J Liu, and Vijay Balasubramanian. Physical effects of learning. *Physical Review E*, 109(2):024311, 2024.
- [18] Dan H Sanes, Thomas A Reh, and William A Harris. *Development of the nervous system*. Academic press, 2011.
- [19] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.
- [20] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [21] Ketan Rajshekhar Shahapure and Charles Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, pages 747–748. IEEE, 2020.
- [22] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.
- [23] Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.
- [24] Carlo Lucibello and Marc Mézard. Exponential capacity of dense associative memories. *Physical Review Letters*, 132(7):077301, 2024.
- [25] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.
- [26] Leo Kozachkov, Ksenia V Kastanenko, and Dmitry Krotov. Building transformers from neurons and astrocytes. *Proceedings of the National Academy of Sciences*, 120(34):e2219150120, 2023.
- [27] Peter E Hart, David G Stork, Richard O Duda, et al. *Pattern classification*. Wiley Hoboken, 2000.
- [28] Jeremy Gunawardena. A linear framework for time-scale separation in nonlinear biochemical systems. *PloS one*, 7(5):e36321, 2012.
- [29] Kee-Myoung Nam, Rosa Martinez-Corral, and Jeremy Gunawardena. The linear framework: using graph theory to reveal the algebra and thermodynamics of biomolecular systems. *Interface Focus*, 12(4):20220013, 2022.