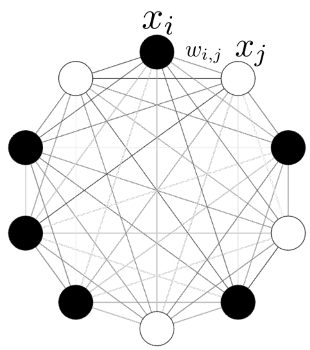
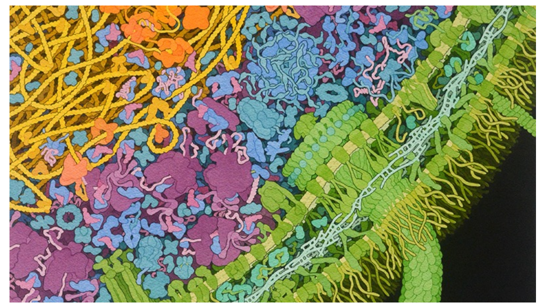
# Equilibrium is Inference: Lessons from a Model of Liquid-Liquid Phase Separation



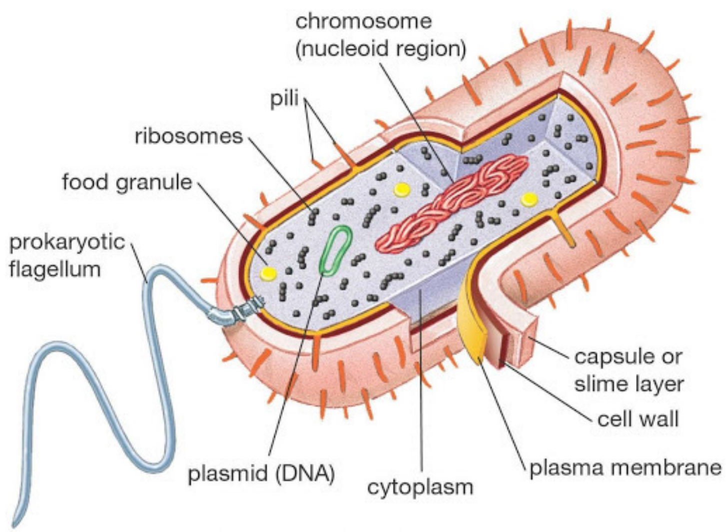
E. coli, by David Goodsell

Cameron Chalk, Salvador Buse, Krishna Shrinivas, Arvind Murugan, Erik Winfree

“Learning and Inference in a Lattice Model of Multicomponent Condensates” (2024)

# The Ancient Information Revolution



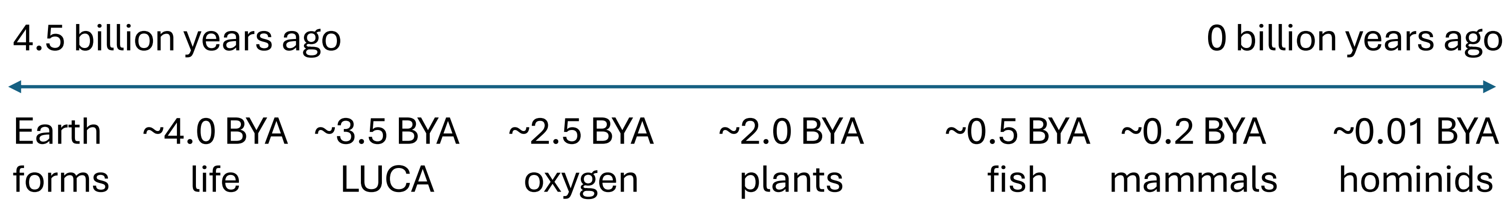

Biology: Life on Earth

DNA: stores information in a linear string

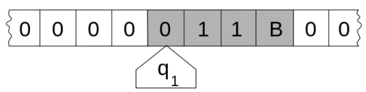
Central Dogma of Molecular Biology:
information controls processes
DNA → RNA → protein

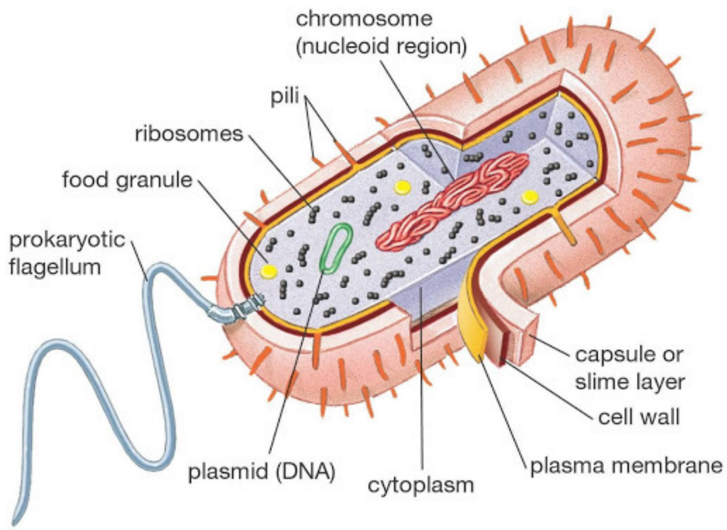
Evolution:
(blindly) change information
to program new tasks

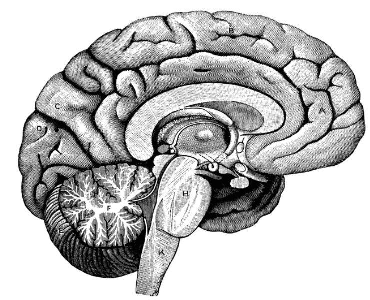
Life: information-based chemistry

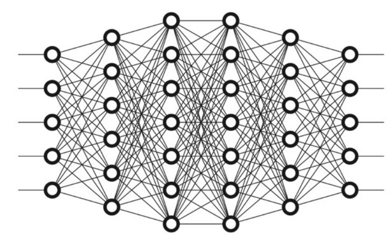


# Everything is code, but what kind of code is it?



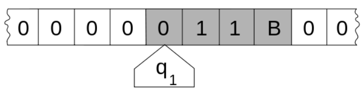



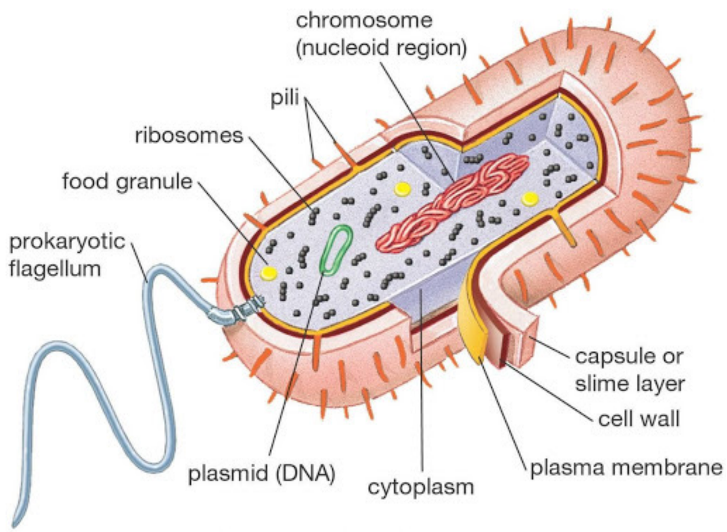


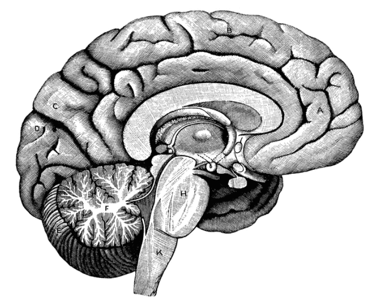
What are "natural" models for biomolecular algorithms?

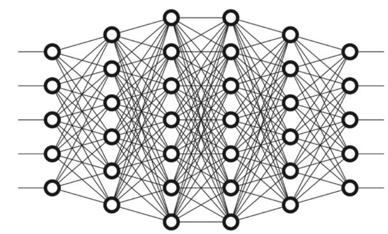
How do we look for them and where do we see them?

# Will we find natural algorithms in equilibrium or non-equilibrium processes?





Computation is possible at arbitrarily low energetic cost
(Charles Bennett, 1973, 1982, 1989)
Neural computation as random walks on energy landscapes
(John Hopfield, 1982; Geoff Hinton et al, 1985 etc)

# Equilibrium is Inference

Two problems with this thesis: (1) It's not true. (2) It's not new.

Equilibrium and inference are distinct concepts. But they can be related!

Stanislaw Ulam and/or John von Neumann (apocryphal?):
"A theory of non-equilibrium systems is like a theory of non-elephants!"
"A theory of non-linear systems is like a theory of non-elephants!"

Elephants can be both quite interesting and quite powerful.

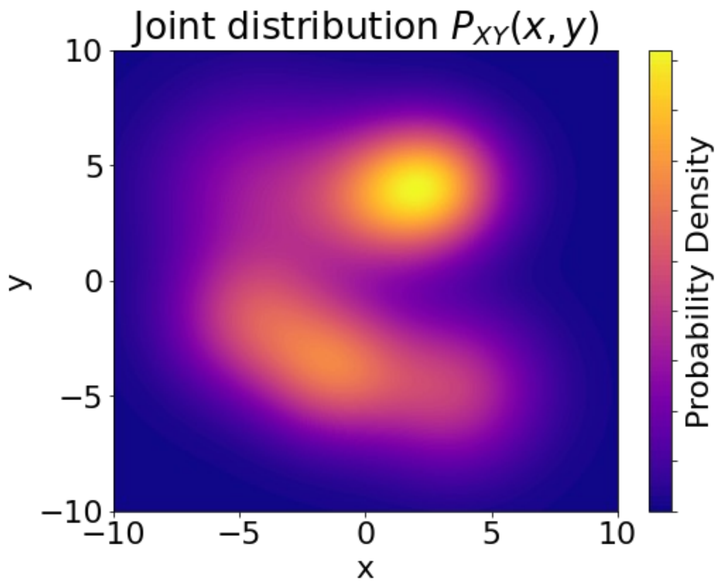
# Equilibrium is Inference

| | |
|---|---|
| Reachable state space: | $x \in X$ |
| Energy: | $E(x)$ |
| Neighbors: | $x \sim y$ |
| Detailed balance: | $k_{x \to y} \; e^{-E(x)} = k_{y \to x} \; e^{-E(y)}$ |
| Boltzmann distribution: | $P(x) = \frac{1}{Z} \; e^{-E(x)}$ |
| Partition function: | $Z = \sum_{x \in X} e^{-E(x)}$ |

# Equilibrium is Inference

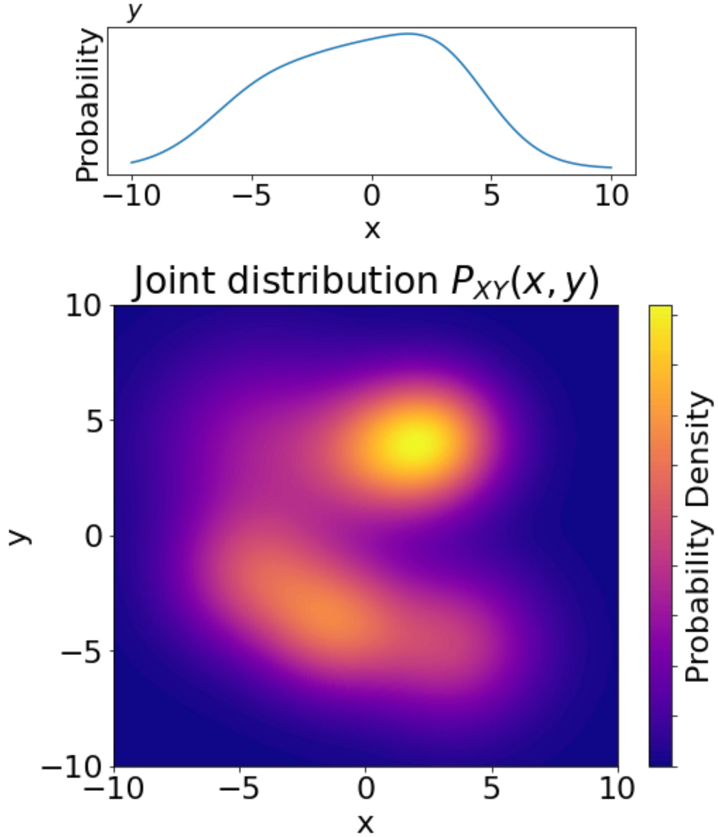
| | |
|---|---|
| State space: | $x \in X$ |
| A priori probability: | $P(x)$ |
| Subspace of events: | $y \in Y \subset X$ |
| A posteriori probability: | $P(x\|Y) = P(x)/P(Y)$ |
| Subset of variables: | $x = vh$ |
| Marginal distribution: | $P(v) = \sum_h P(vh)$ |
| A posteriori probability: | $P(h\|v) = P(vh)/P(v)$ |

# Key concepts of multidimensional probability distributions



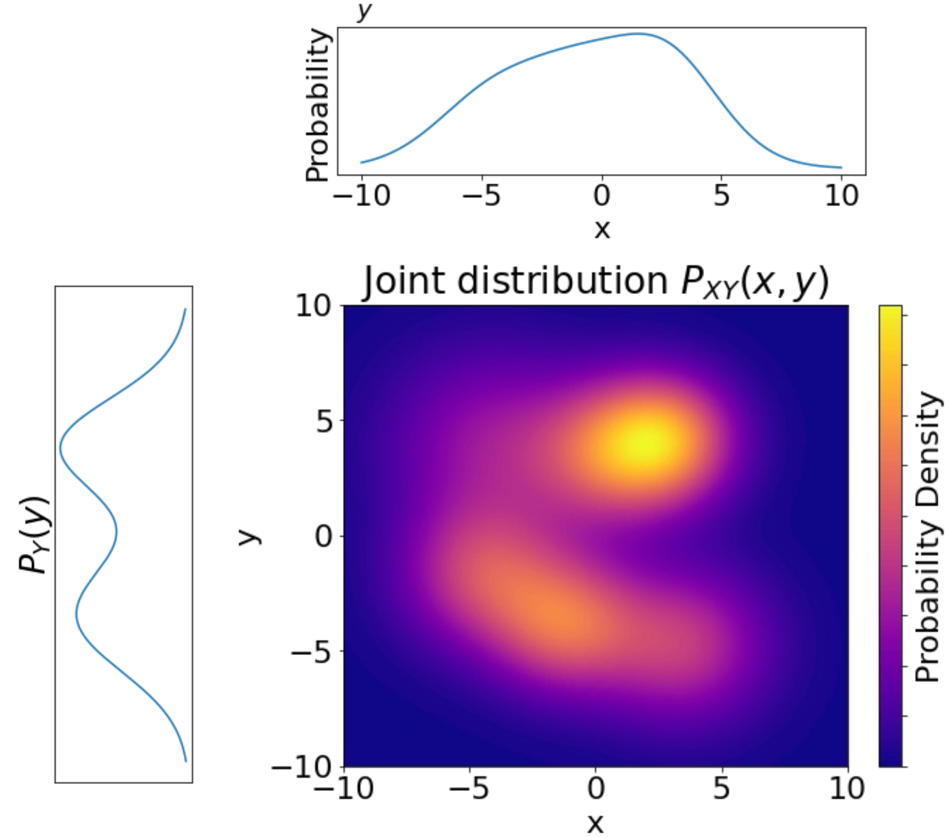
# Key concepts of multidimensional probability distributions

$P_X(x) = \sum_y P_{XY}(x, y)$: Marginal probability of $x$

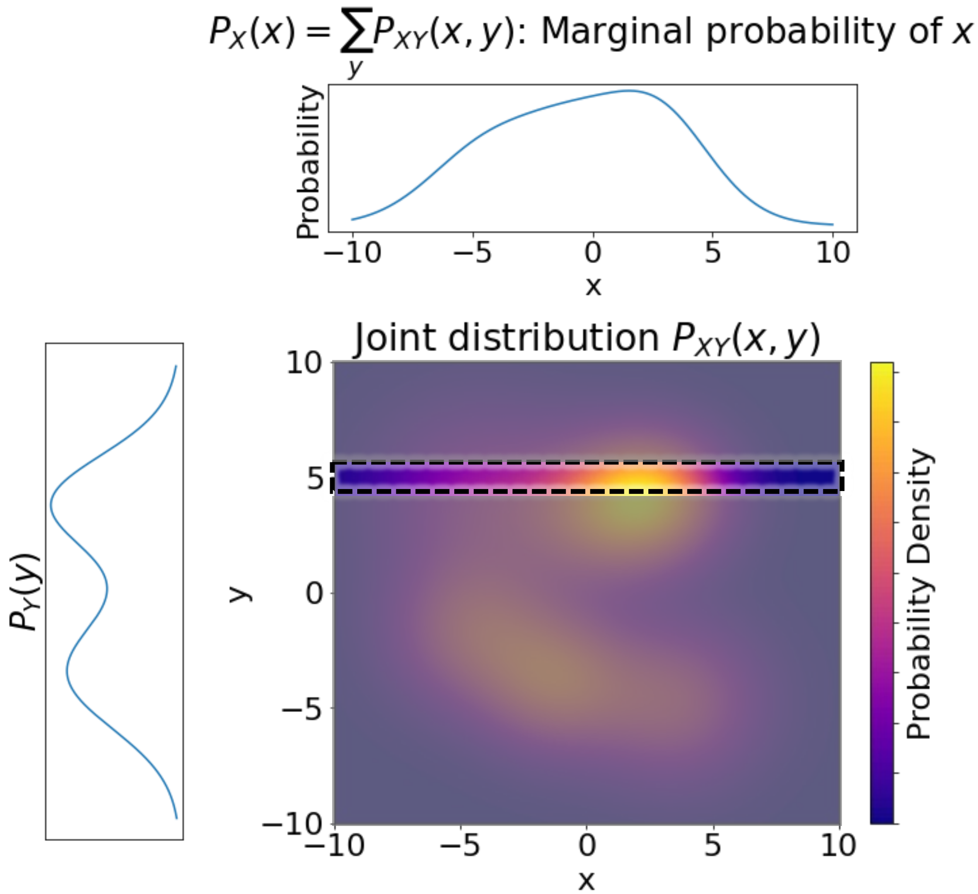


# Key concepts of multidimensional probability distributions

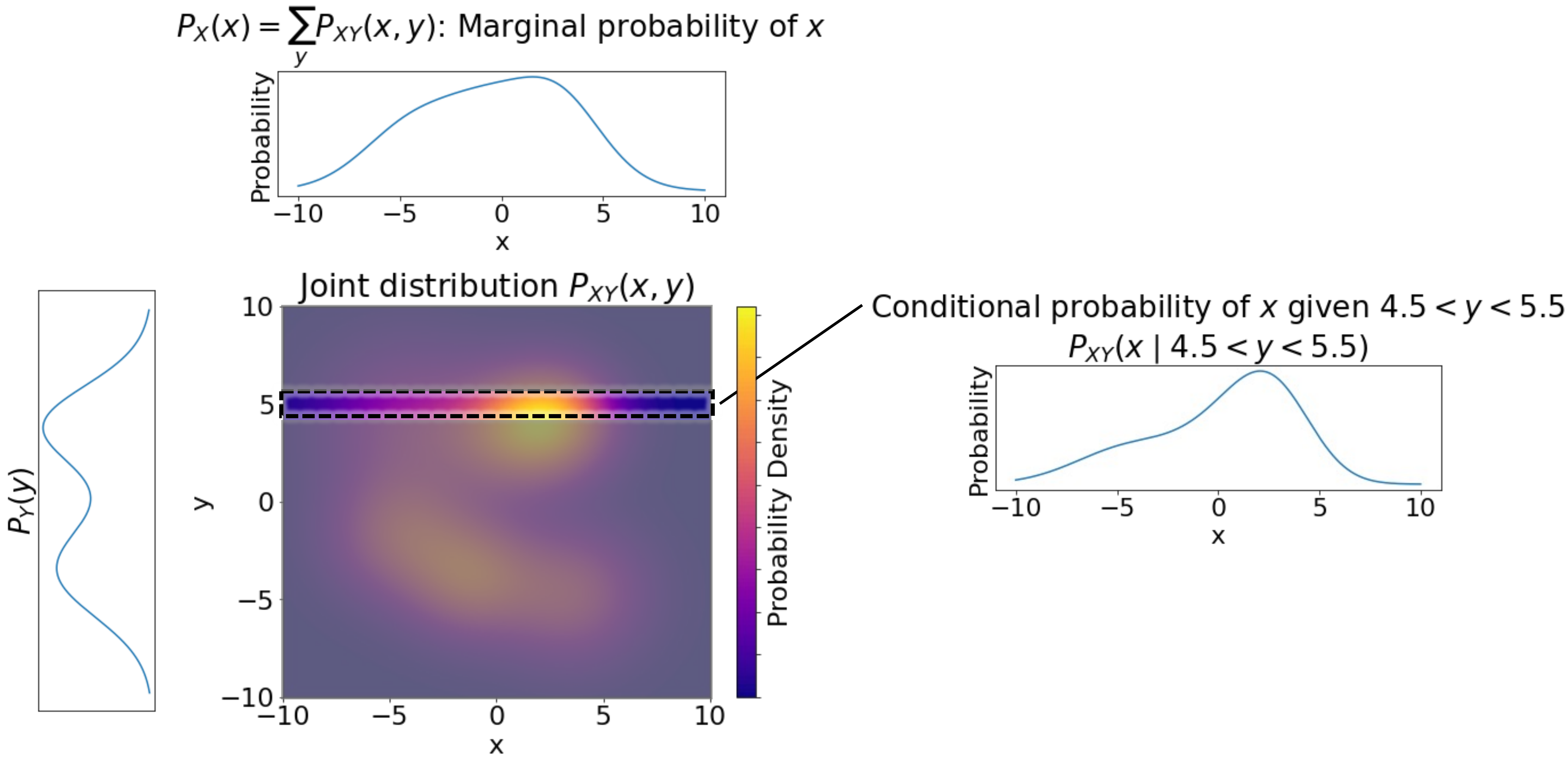
$P_X(x) = \sum_y P_{XY}(x, y)$: Marginal probability of $x$



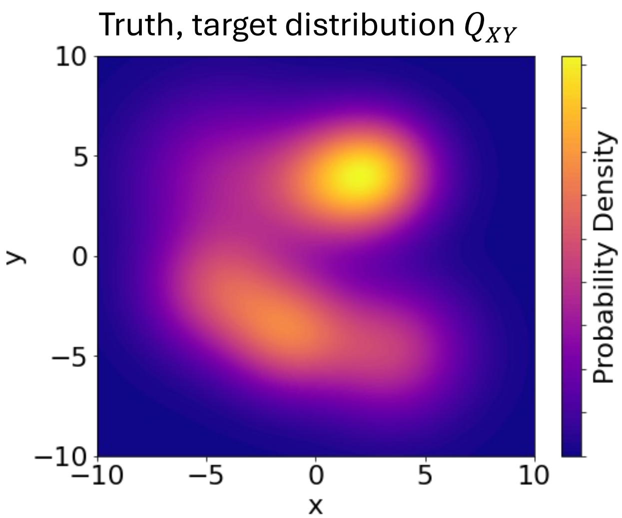
# Key concepts of multidimensional probability distributions



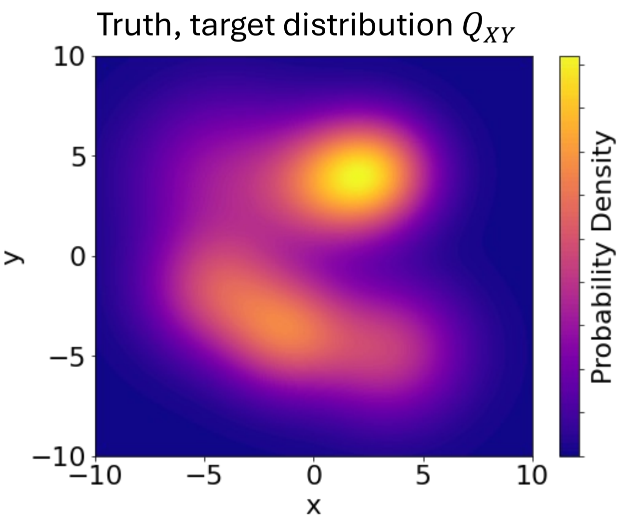
# Key concepts of multidimensional probability distributions

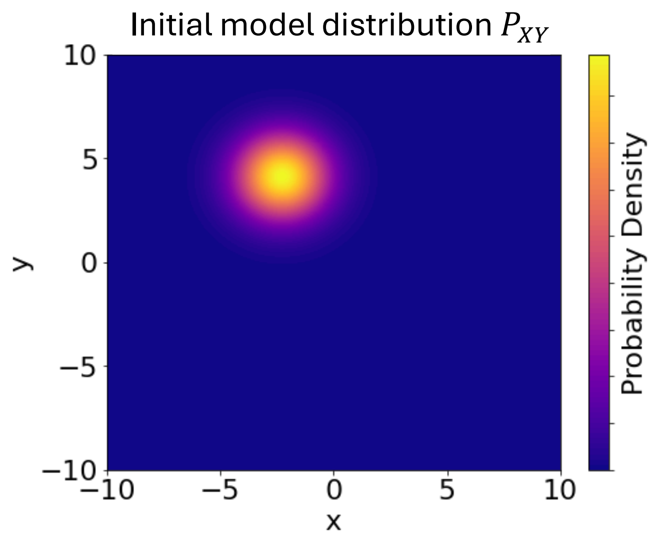


# The big picture of learning distributions

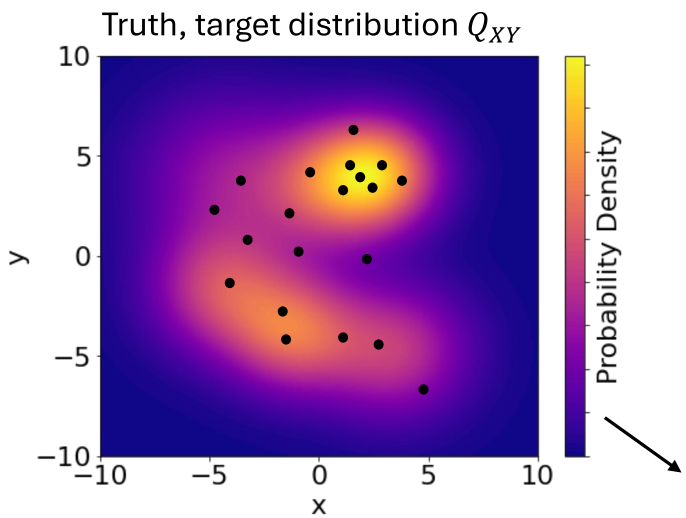


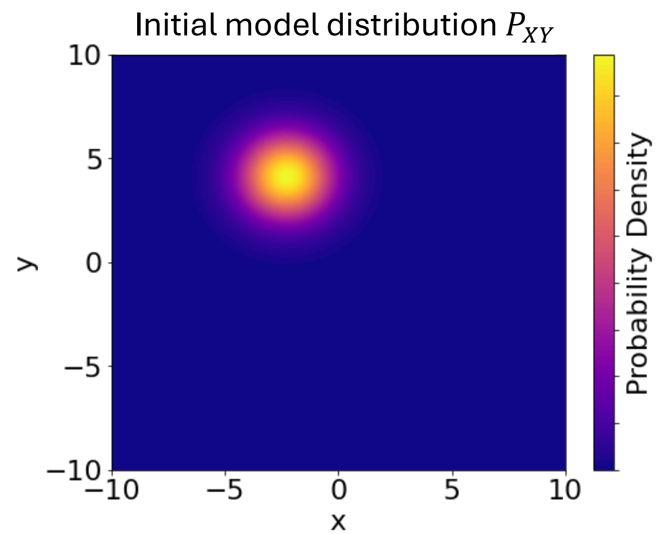
# The big picture of learning distributions



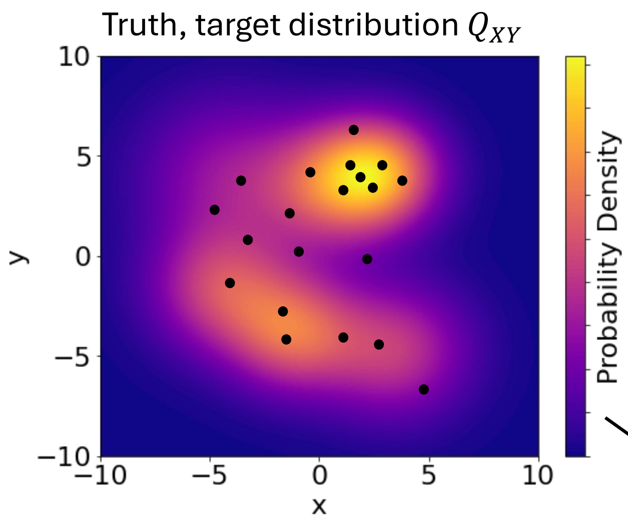
# The big picture of learning distributions

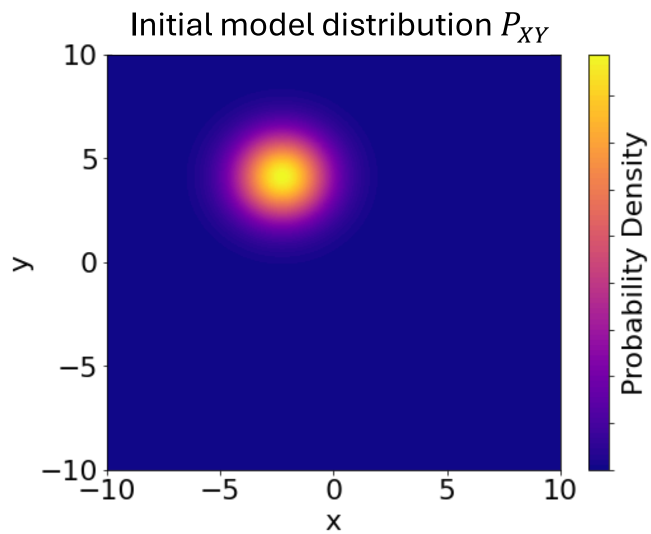



Draw samples from $Q_{XY}$

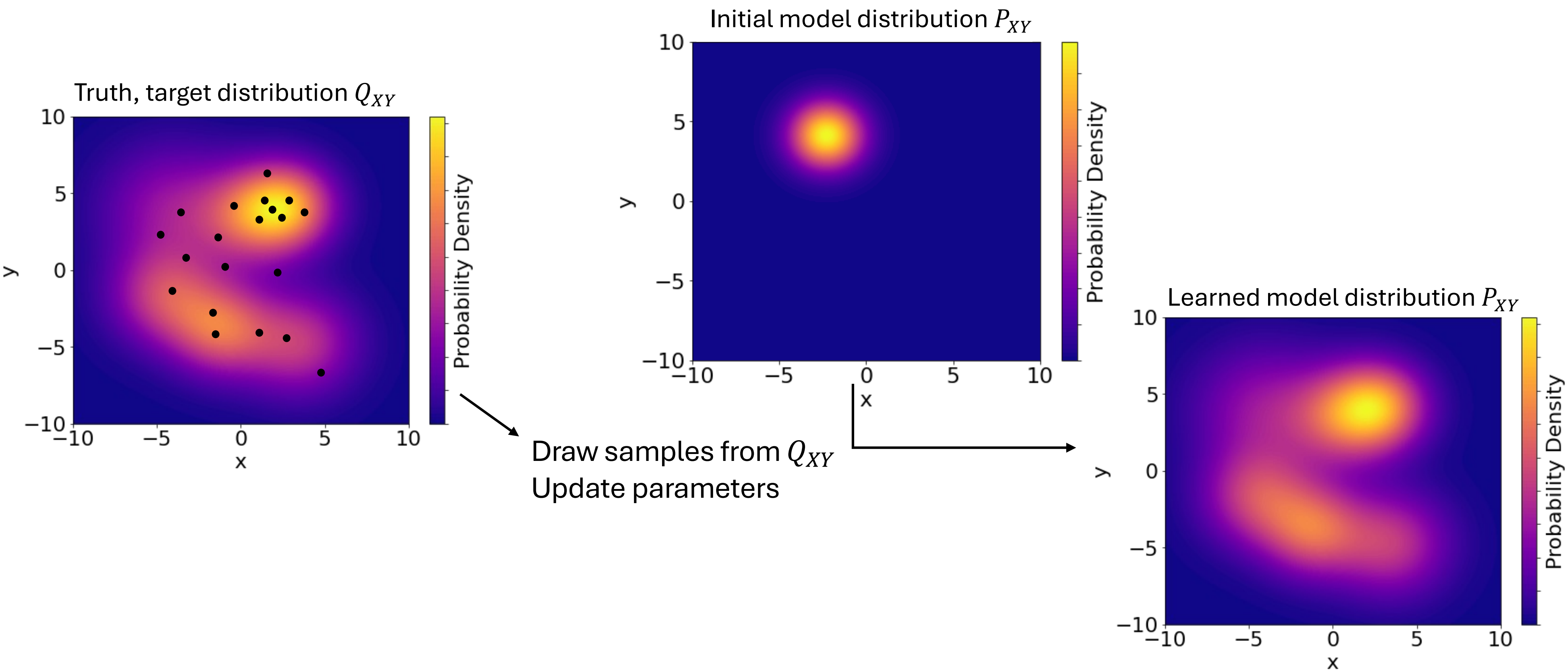
# The big picture of learning distributions




Draw samples from $Q_{XY}$
Update parameters

# The big picture of learning distributions



# Energy-based neural network models

# Hopfield Associative Memory

Hopfield (1982, 1984)

Asynchronous updates. +1 = true = on, -1 = false = off

$$x_i \leftarrow +1 \text{ if } \sum_j w_{ij} x_j + b_i > 0$$

Hebbian learning

$$w_{ij} = \sum_\alpha x_i^\alpha x_j^\alpha \qquad b_i = \sum_\alpha x_i^\alpha$$

Energy function

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} x_i x_j - \sum_i b_i x_i$$

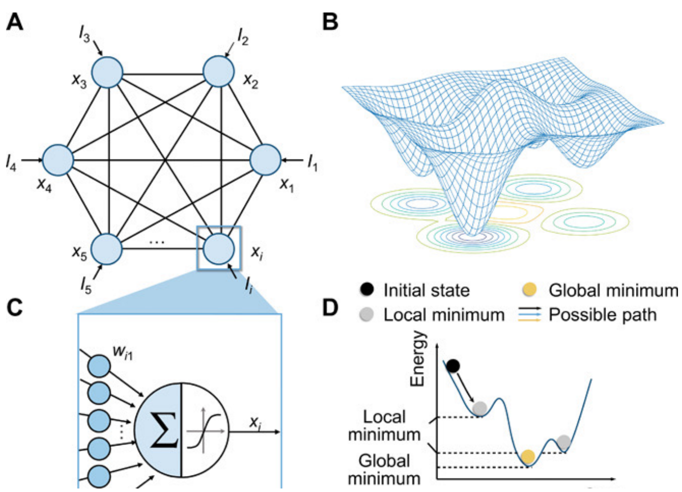



Illustration from Yang et al (2020)

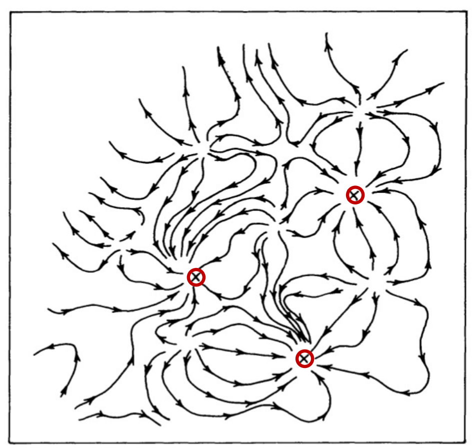


Illustration from Hopfield (1988)

Initial state Intermediate Final Recall

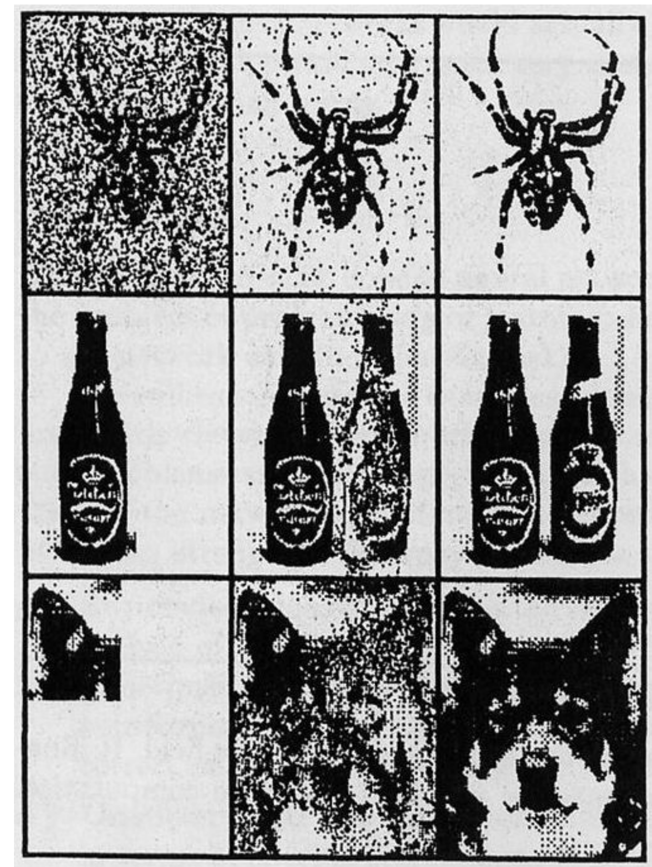


Illustration from Hertz, Krogh, Palmer (1991)

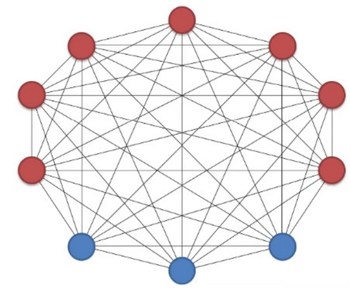
# Boltzmann Machines

Ackley, Hinton, Sejnowski (1985)

Energy function

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j - \sum_i b_i x_i$$

hidden units



visible units

Energy change

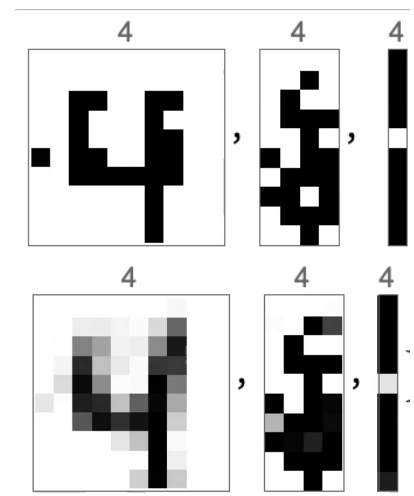
$$\Delta E_i = E(x_i \text{ on}) - E(x_i \text{ off}) \propto -\sum_j w_{ij}x_j + b_i$$

Stochastic detailed balance

$$x_i \leftarrow +1 \text{ with prob } \frac{1}{1+\exp(\Delta E_i/T)}$$

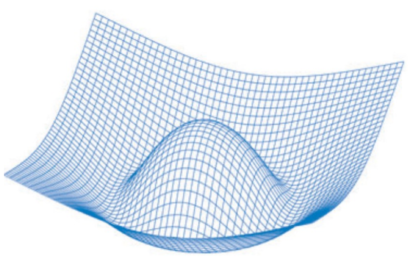
Equilibrium probability

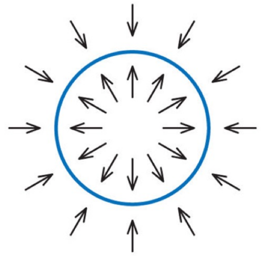
$$p(x) = \frac{1}{Z}\exp(-E(x)/T) \text{ with } Z = \sum_x \exp(-E(x)/T)$$




Hebbian/anti-Hebbian wake/sleep learning/unlearning

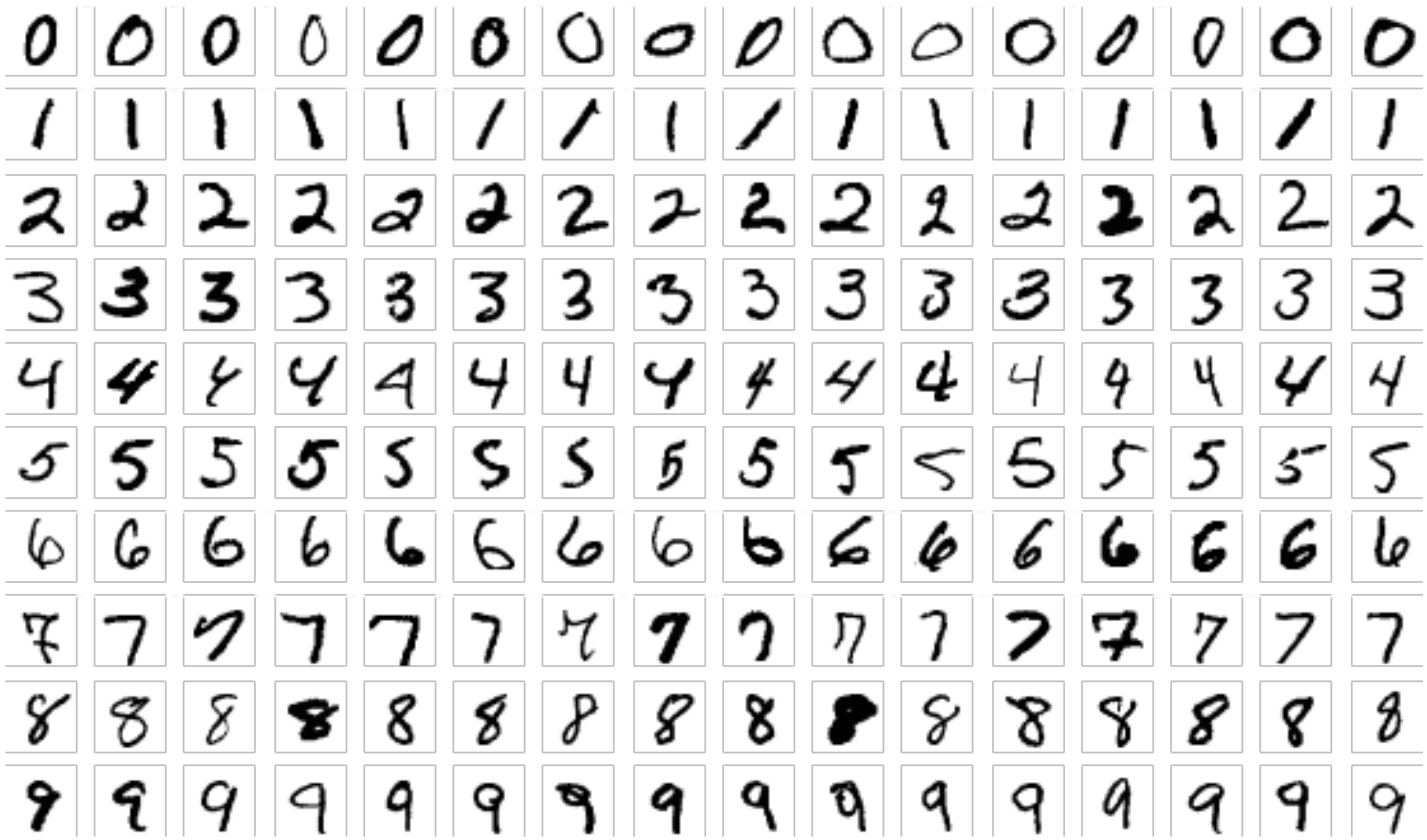
$$\Delta w_{ij} \propto \langle x_i x_j \rangle_{\text{wake}} - \langle x_i x_j \rangle_{\text{sleep}}$$





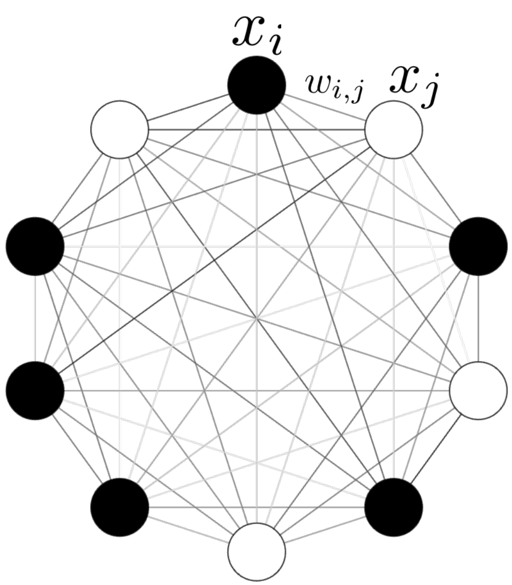
Illustrations from Knierim & Zhang (2012)

# High-dimensional probability distributions represent the world



# Stochastic neural networks represent distributions by minimizing energy

Boltzmann machine




State space: $x \in \{-1, 1\}^N$

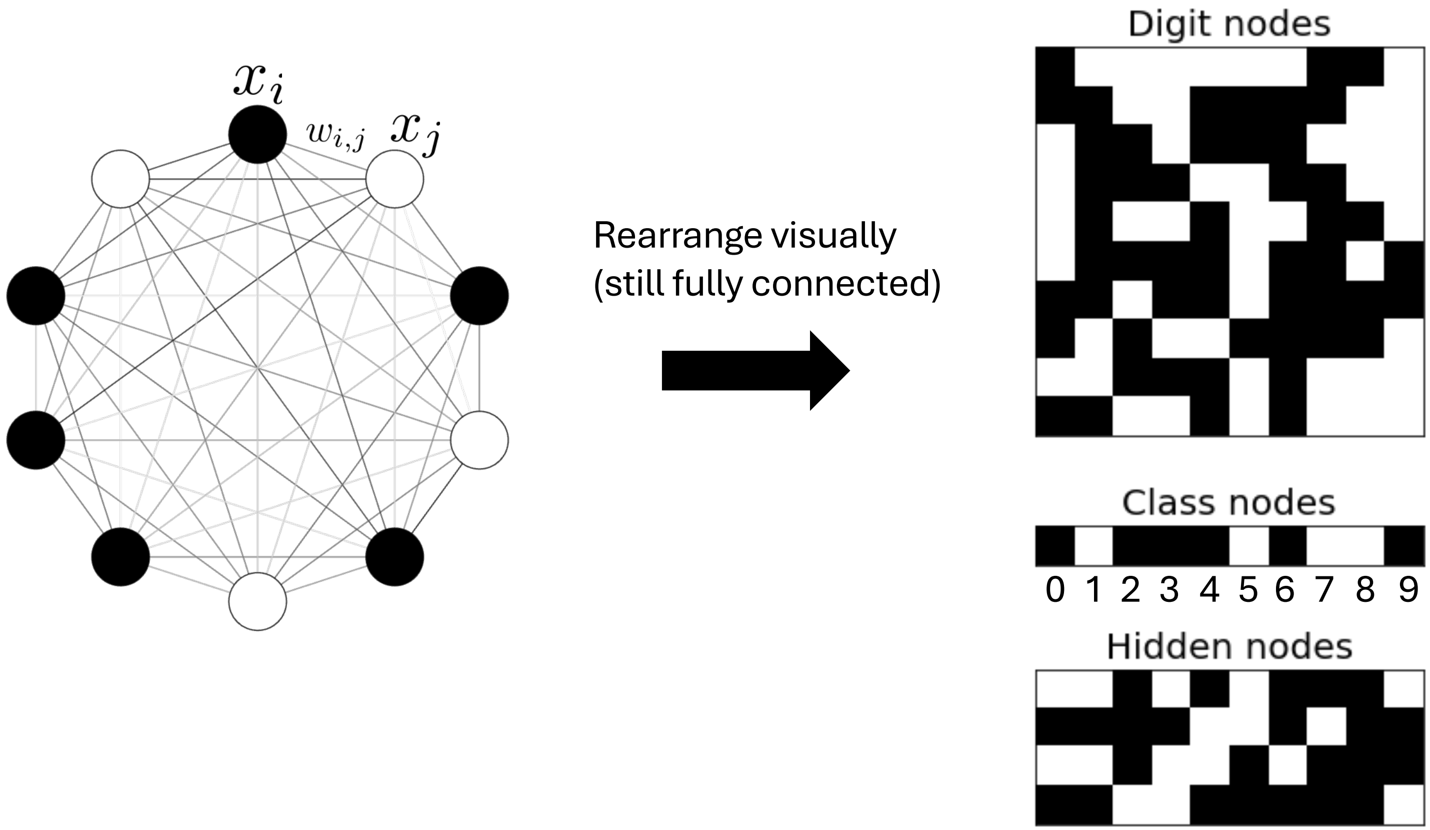
Energy: $$E(x) = -\frac{1}{2}\sum_i \sum_j w_{i,j} x_i x_j - \sum_i \theta_i x_i$$

Probability distribution (Boltzmann distribution):

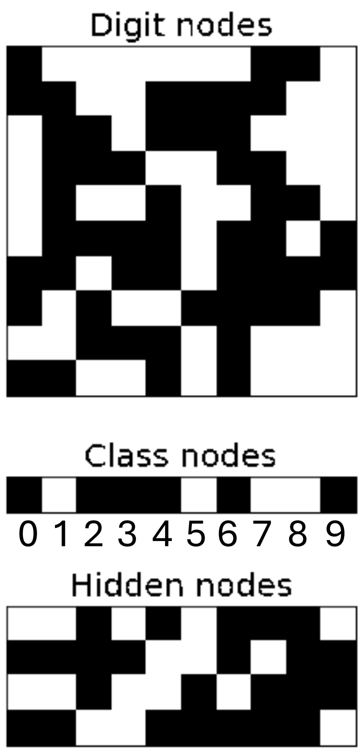
$$P(x) = \frac{1}{Z} e^{-E(x)/kT} \qquad Z = \sum_x e^{-E(x)/kT}$$

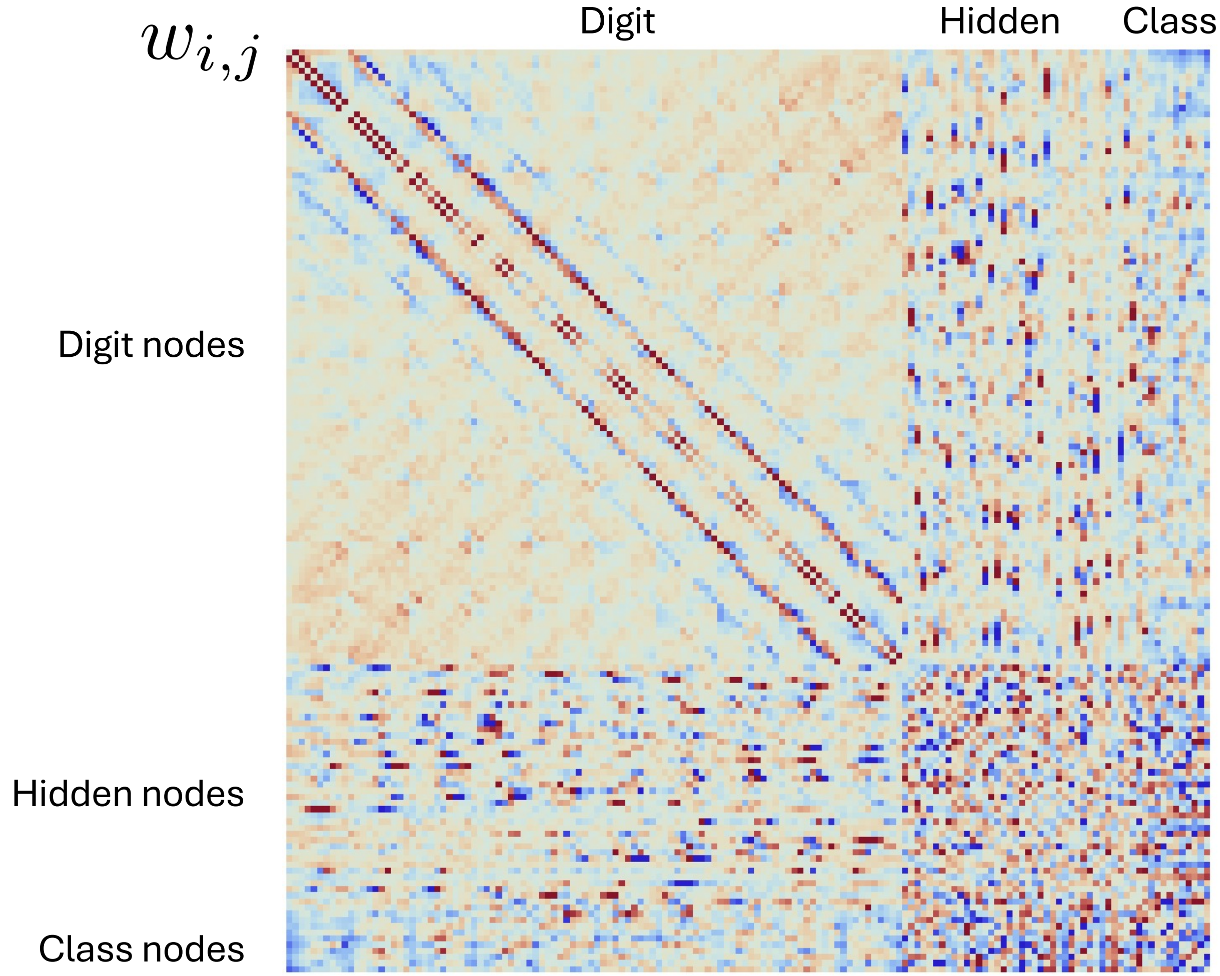
Ackley, Hinton, Sejnowski, 1985

# A Boltzmann machine representing handwritten digits

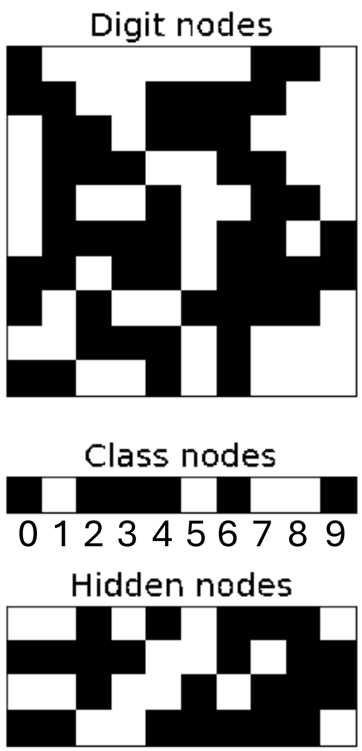


# A Boltzmann machine representing handwritten digits

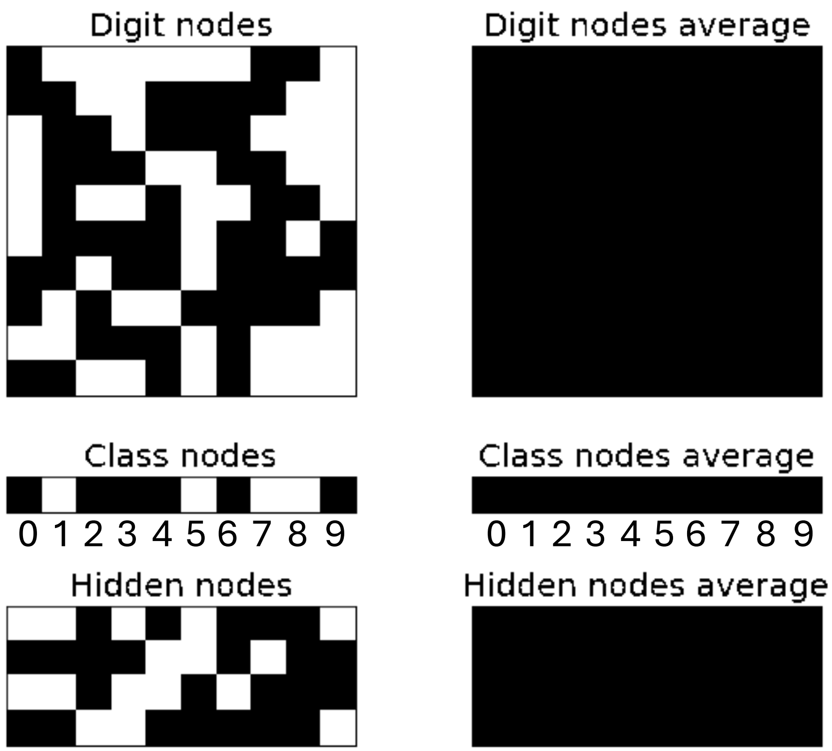


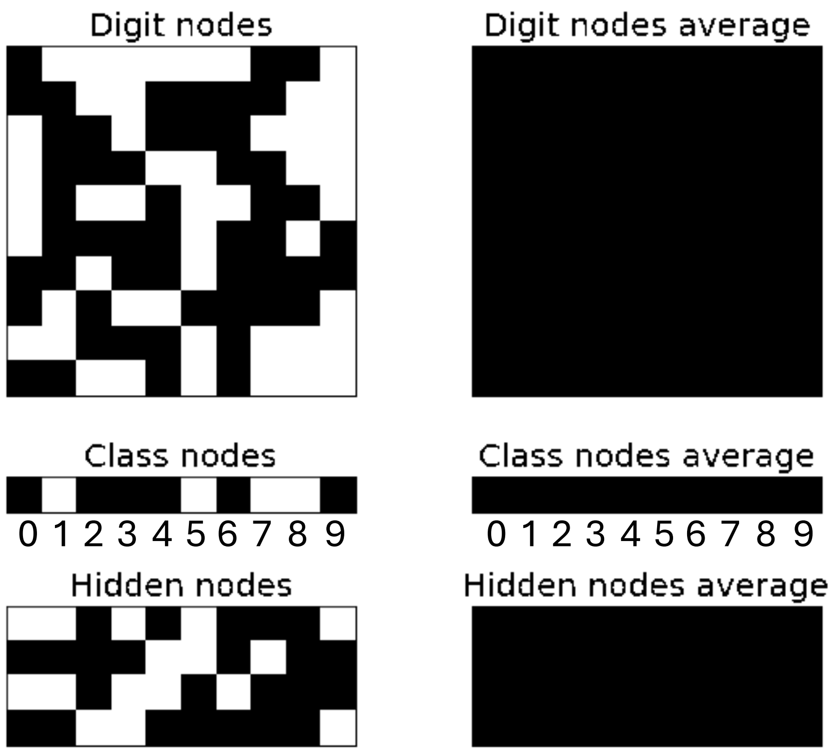
# A Boltzmann machine representing handwritten digits

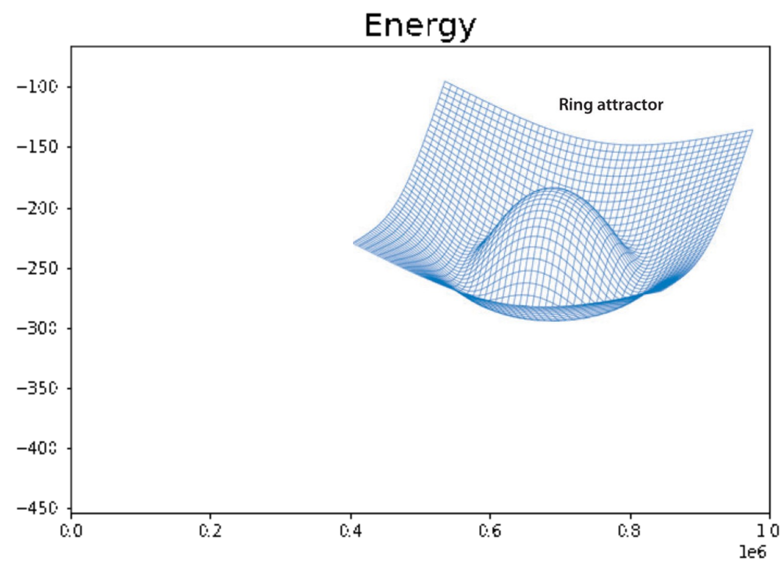


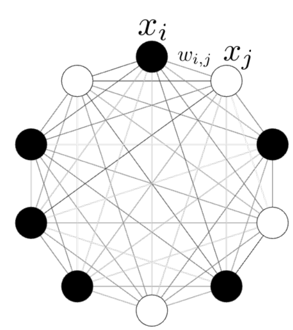
# A Boltzmann machine representing handwritten digits



# A Boltzmann machine representing handwritten digits

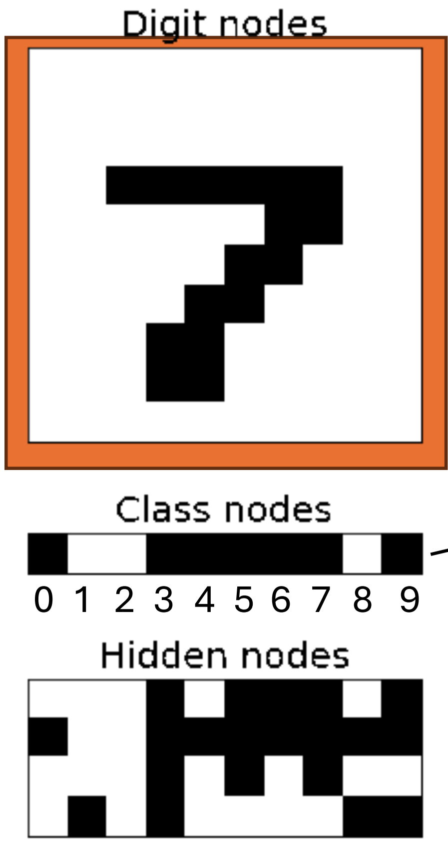



$$E(x) = -\frac{1}{2}\sum_i \sum_j w_{i,j} x_i x_j - \sum_i \theta_i x_i$$

# Inference: recognizing digits via conditional probability



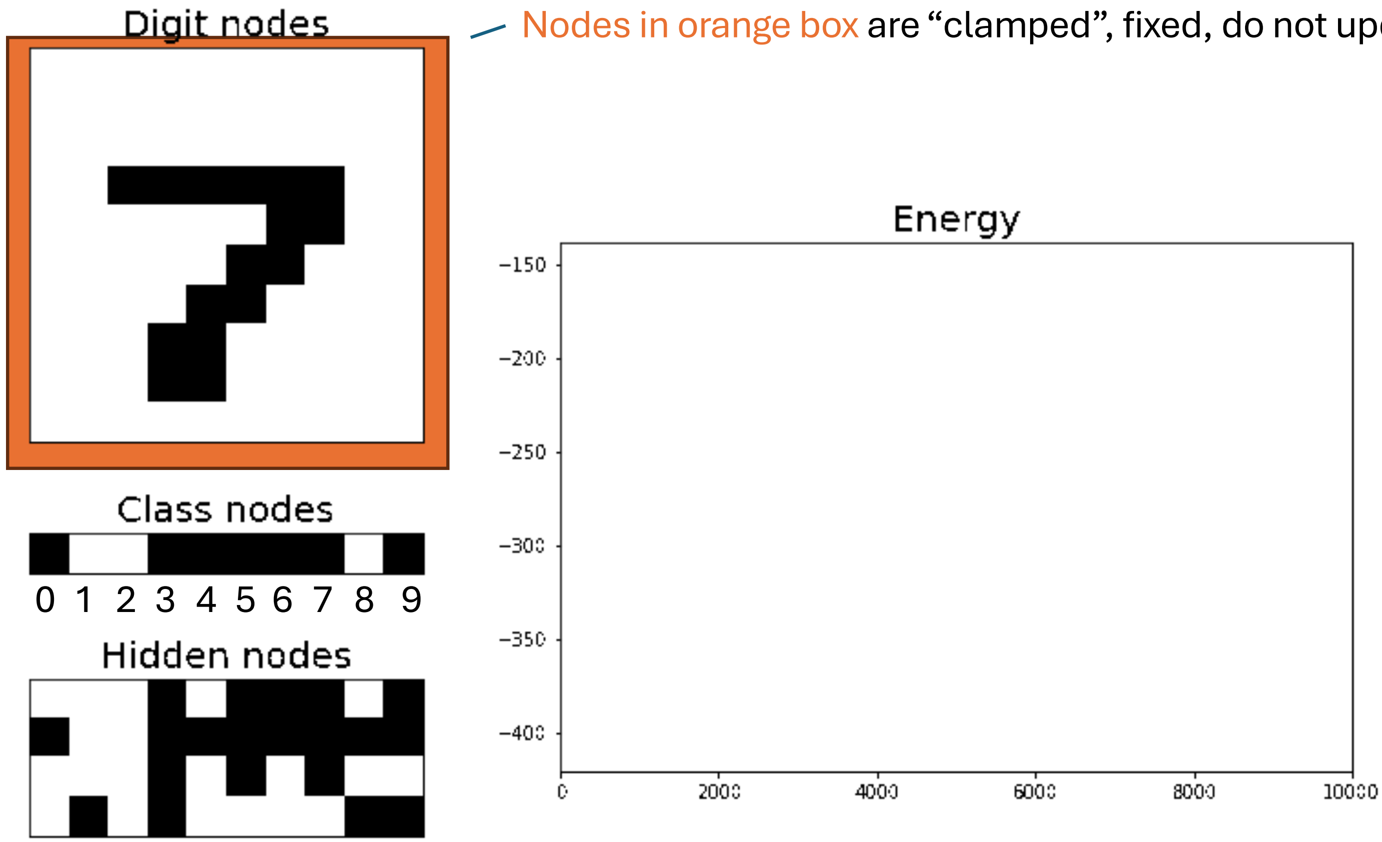

Nodes in orange box are "clamped", fixed, do not update

Unclamped nodes compute conditional distribution

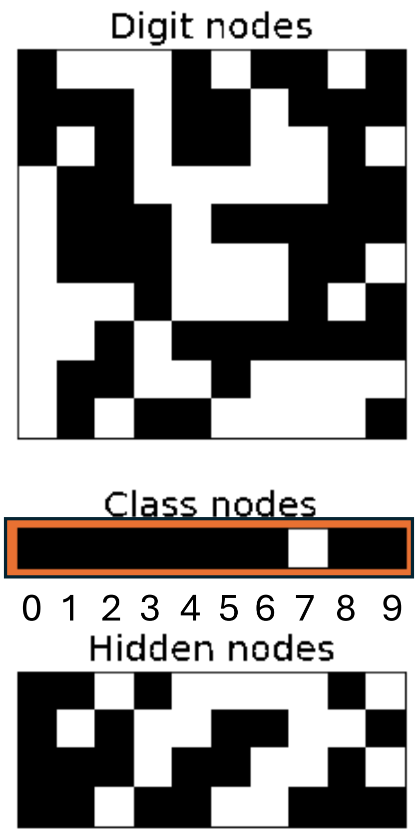
$$P(class\ nodes \mid digit\ nodes = this\ 7)$$

# Inference: recognizing digits via conditional probability

Nodes in orange box are “clamped”, fixed, do not update



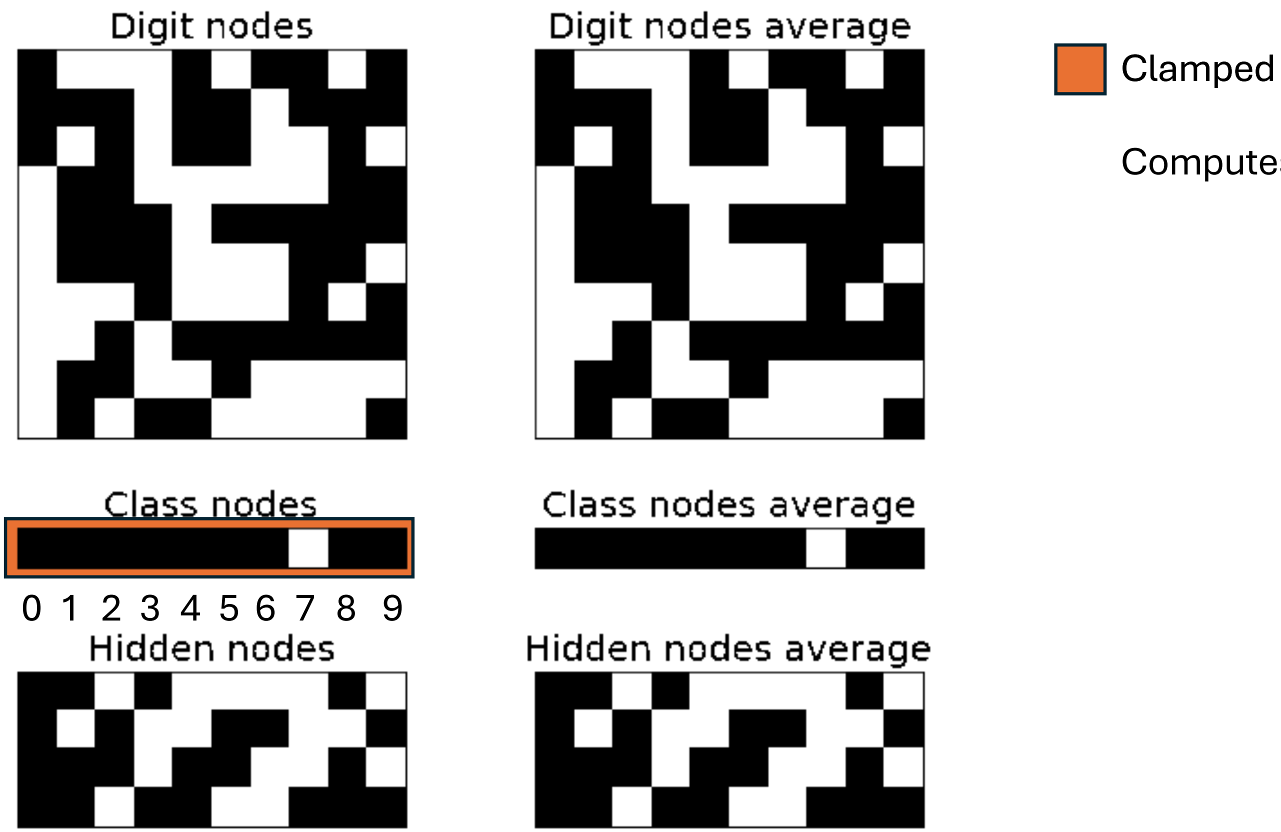
# Inference is "omnidirectional": generalizes input vs output




Clamped

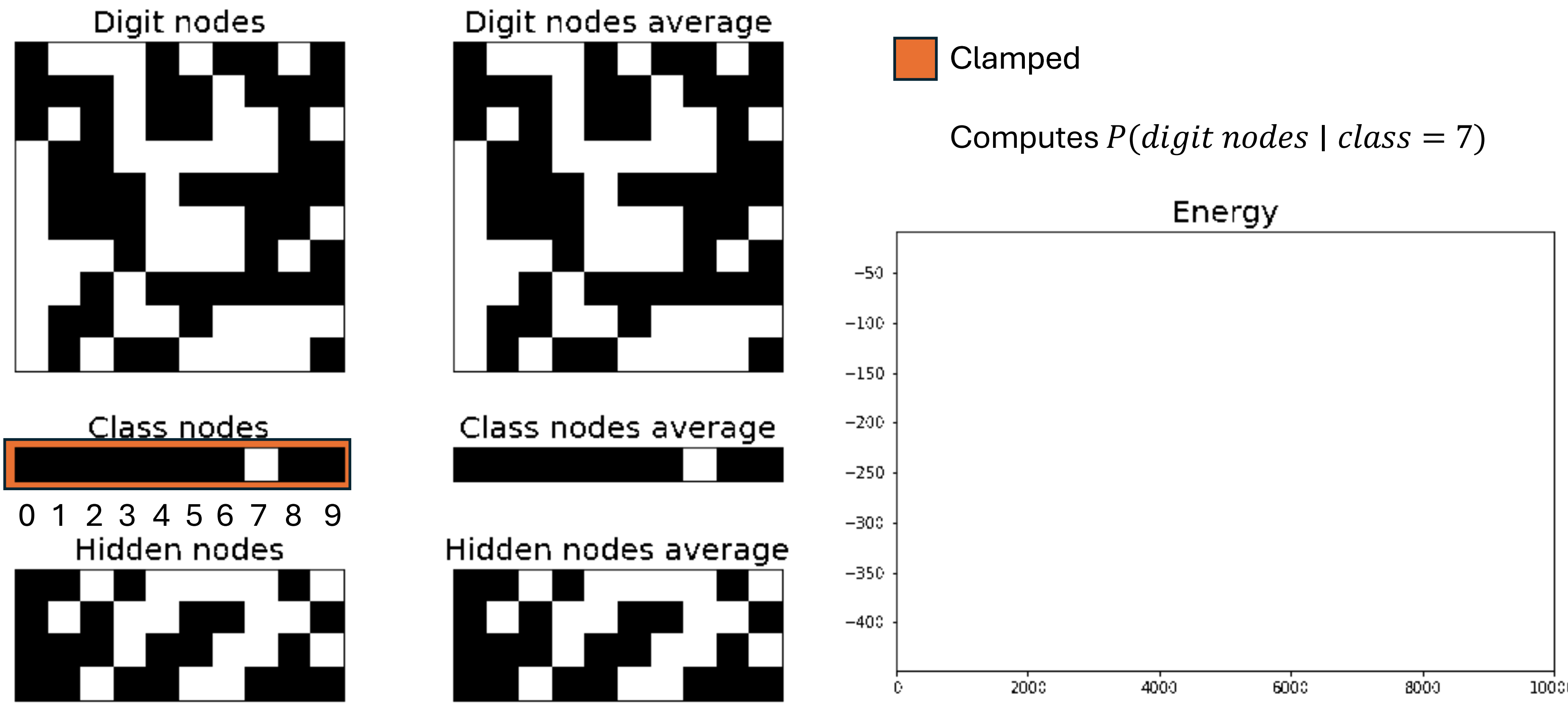
Computes $P(digit\ nodes \mid class = 7)$

# Inference is “omnidirectional”: generalizes input vs output



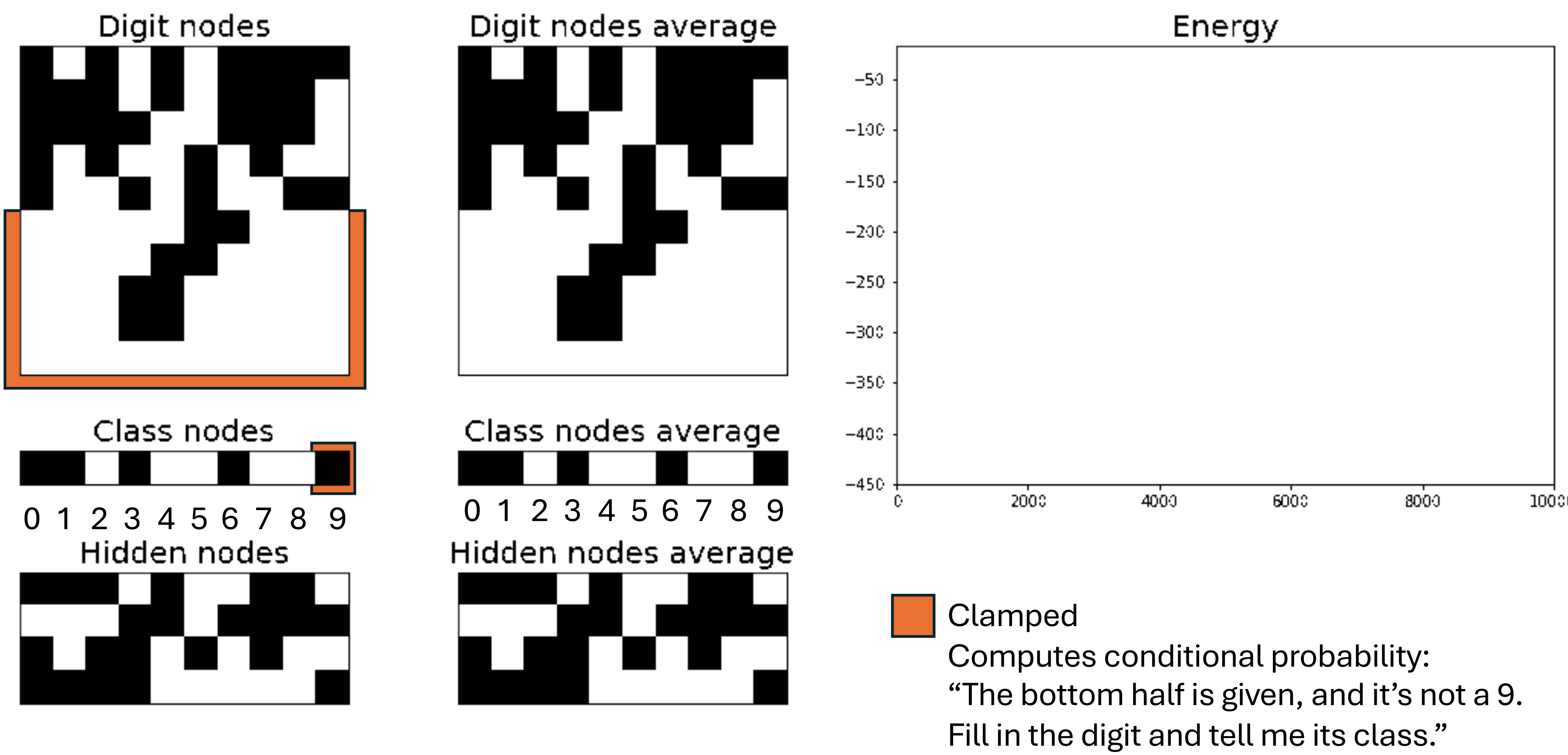

Computes $P(digit\ nodes \mid class = 7)$

# Inference is “omnidirectional”: generalizes input vs output

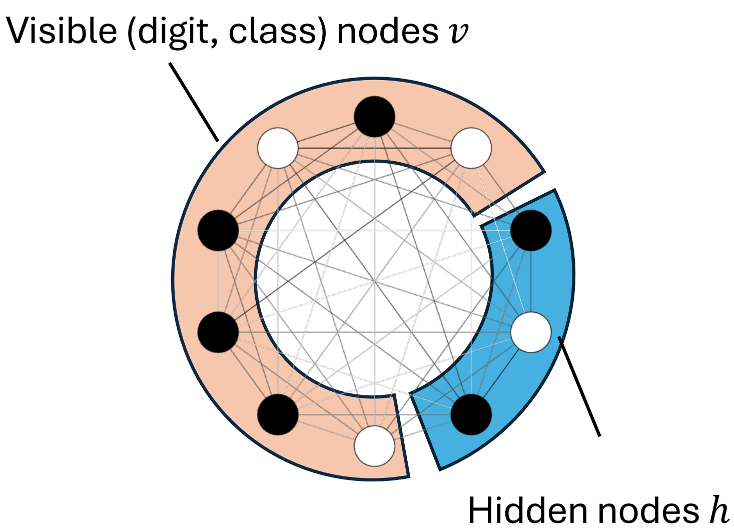



Computes $P(digit\ nodes \mid class = 7)$

# Inference is “omnidirectional”: generalizes input vs output

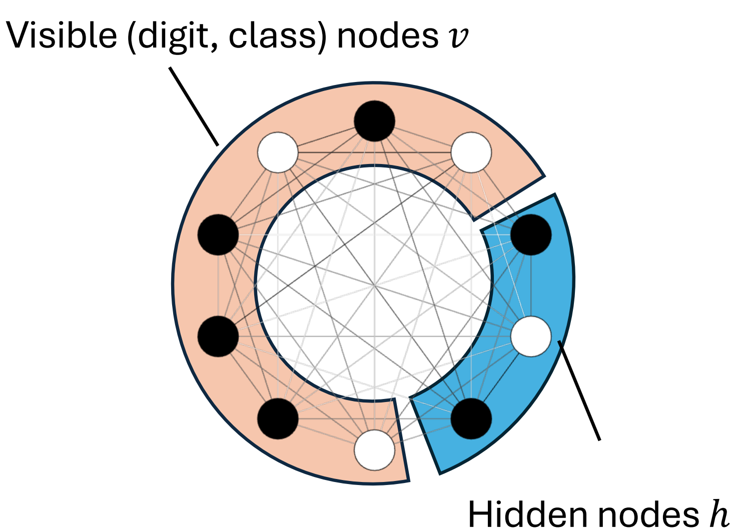


# How Boltzmann machines learn: wake-sleep




Ackley, Hinton, Sejnowski, 1985

# How Boltzmann machines learn: wake-sleep

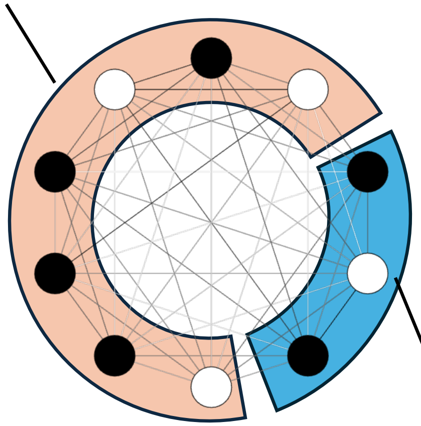



$P_v(v)$ : Marginal distribution over visible units for current $w_{i,j}$

$Q_v(v)$ : Target distribution over visible units

Ackley, Hinton, Sejnowski, 1985

# How Boltzmann machines learn: wake-sleep




Relative entropy ("distance") :

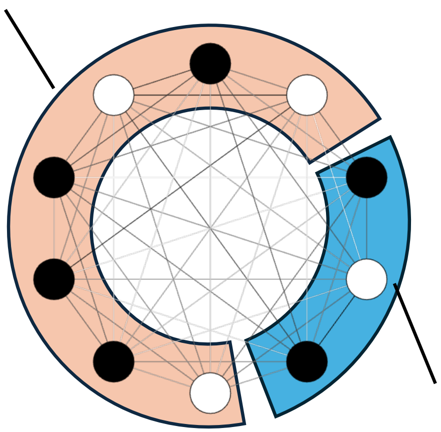
$$R(Q_v \,||\, P_v) = \sum_v Q_v(v) \ln \frac{Q_v(v)}{P_v(v)}$$

$P_v(v)$ : Marginal distribution over visible units for current $w_{i,j}$

$Q_v(v)$ : Target distribution over visible units

Ackley, Hinton, Sejnowski, 1985

# How Boltzmann machines learn: wake-sleep




Relative entropy ("distance") :

$$R(Q_v \,||\, P_v) = \sum_v Q_v(v) \ln \frac{Q_v(v)}{P_v(v)}$$

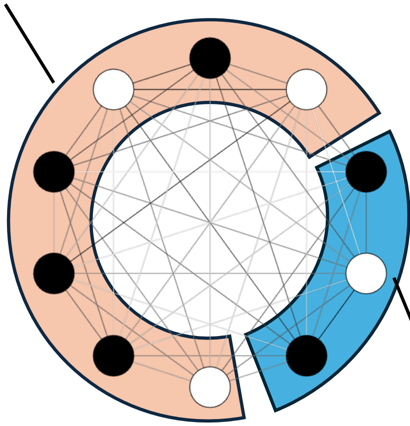
$$\frac{\partial R(Q_v \,||\, P_v)}{\partial w_{i,j}} = \langle x_i x_j \rangle_P - \langle x_i x_j \rangle_Q$$

$P_v(v)$ : Marginal distribution over visible units for current $w_{i,j}$

$Q_v(v)$ : Target distribution over visible units

Ackley, Hinton, Sejnowski, 1985

# How Boltzmann machines learn: wake-sleep




$P_v(v)$ : Marginal distribution over visible units for current $w_{i,j}$

$Q_v(v)$ : Target distribution over visible units

Relative entropy (“distance”) :

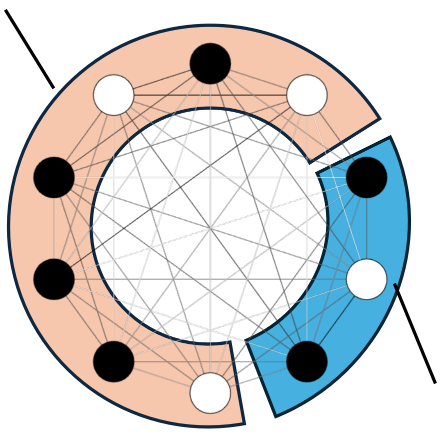
$$R(Q_v \,||\, P_v) = \sum_v Q_v(v) \ln \frac{Q_v(v)}{P_v(v)}$$

$$\frac{\partial R(Q_v \,||\, P_v)}{\partial w_{i,j}} = \langle x_i x_j \rangle_P - \langle x_i x_j \rangle_Q$$

$$\frac{\mathrm{d}w_{i,j}}{\mathrm{d}t} = \langle x_i x_j \rangle_Q - \langle x_i x_j \rangle_P$$

Ackley, Hinton, Sejnowski, 1985

# How Boltzmann machines learn: wake-sleep




$P_v(v)$ : Marginal distribution over visible units for current $w_{i,j}$

$Q_v(v)$ : Target distribution over visible units

Relative entropy ("distance") :

$$R(Q_v \,||\, P_v) = \sum_v Q_v(v) \ln \frac{Q_v(v)}{P_v(v)}$$

$$\frac{\partial R(Q_v \,||\, P_v)}{\partial w_{i,j}} = \langle x_i x_j \rangle_P - \langle x_i x_j \rangle_Q$$

$$\frac{\mathrm{d}w_{i,j}}{\mathrm{d}t} = \langle x_i x_j \rangle_Q - \langle x_i x_j \rangle_P$$

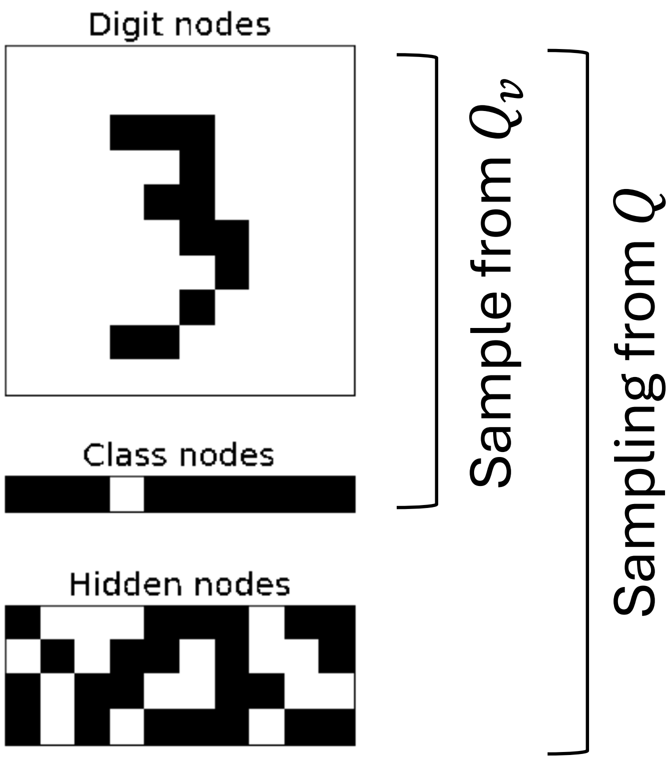
Clamp the machine to $Q_v$, run and collect collect $x_i x_j$ (wake phase)

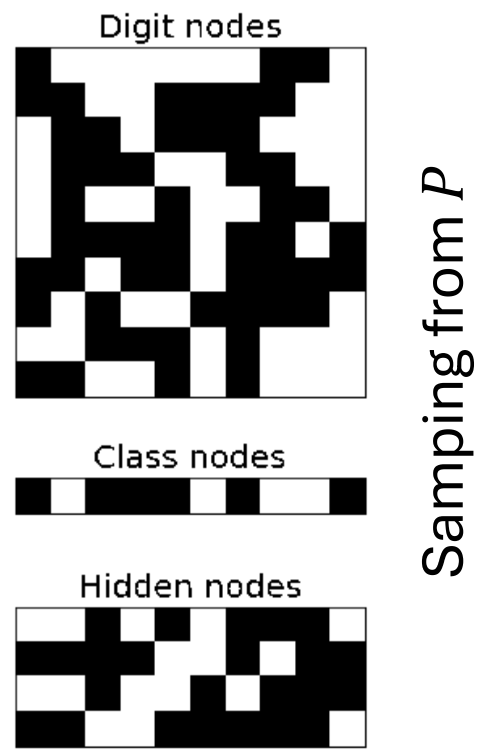
Run the machine and collect $x_i x_j$ to approximate average (sleep phase)

Ackley, Hinton, Sejnowski, 1985

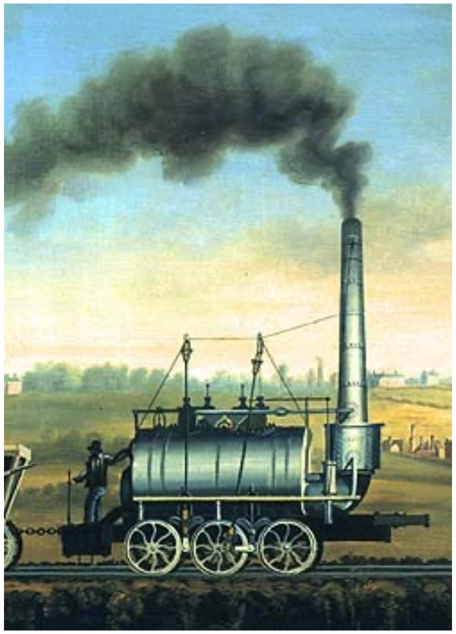
# Teaching handwritten digits to a Boltzmann machine

$$\frac{\mathrm{d}w_{i,j}}{\mathrm{d}t} = \langle x_i x_j \rangle_Q - \langle x_i x_j \rangle_P$$

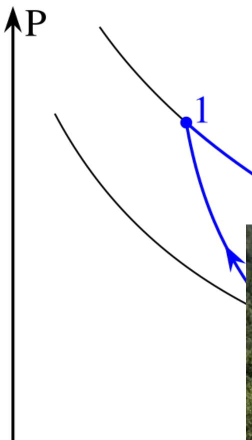


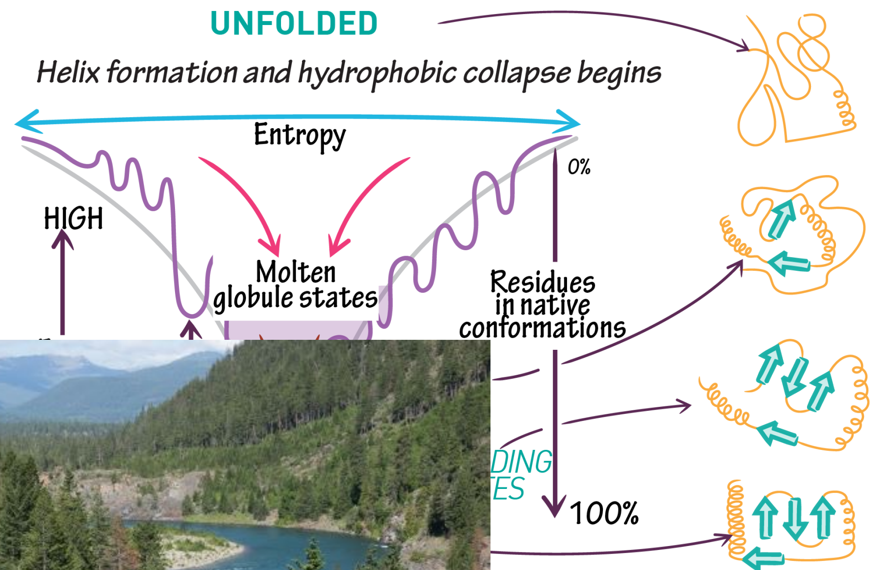
# From technology to science



Steam Elephant, c.a. 1815



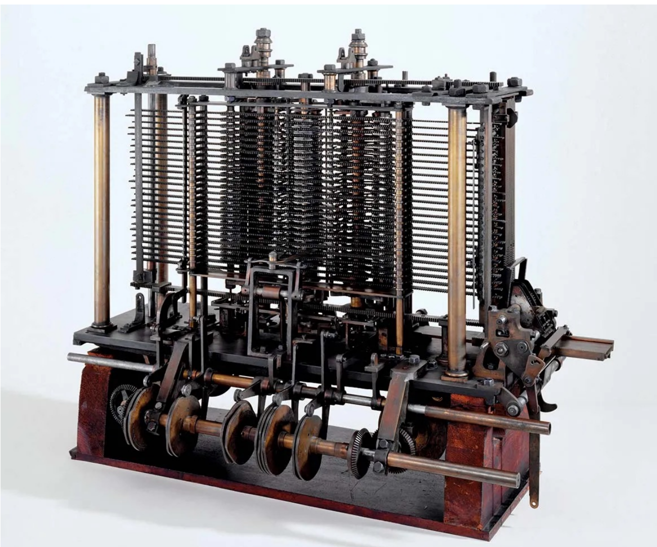

Carn



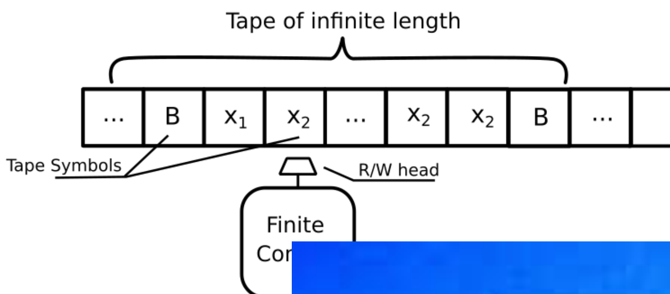

d statistical mechanics

YES, BUT NO!

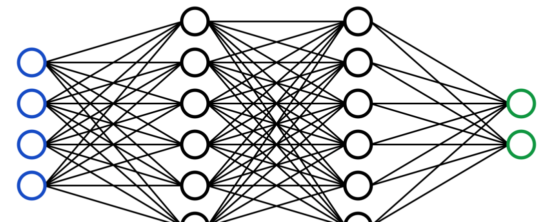
# From technology to science



Babbage’s Analytical Engine, 1837




Turin

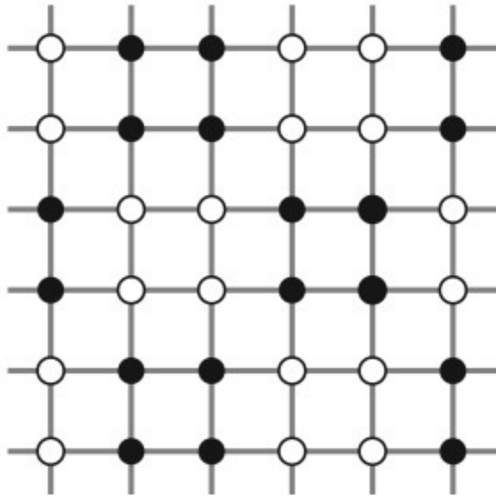


ation

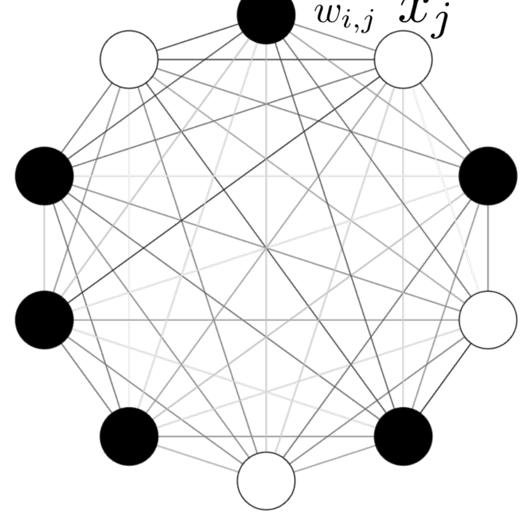

**YES, BUT NO!**

# Energy-based models and physical systems

Ising model of magnetism



Energy is physical




Hopfield model
of neural computation

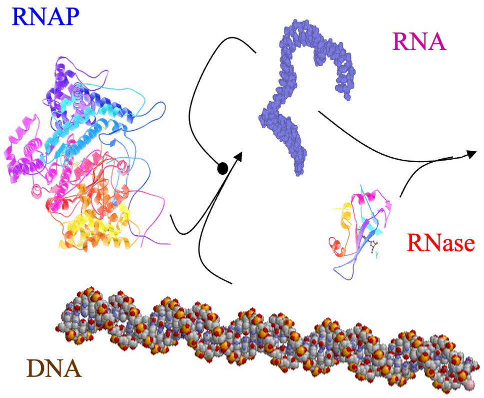
Energy is a mathematical metaphor

$$E(x) = \sum_{i \sim j} J \; x_i \; x_j + \sum_i H \; x_i$$

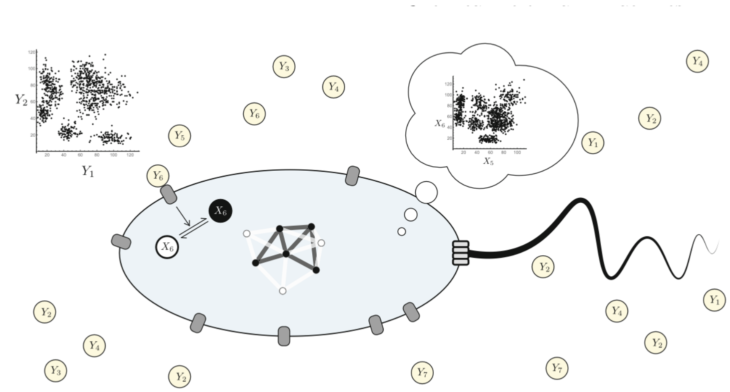
$$E(x) = -\frac{1}{2} \sum_i \sum_j w_{i,j} x_i x_j - \sum_i \theta_i x_i$$

More is different. More kinds is different.

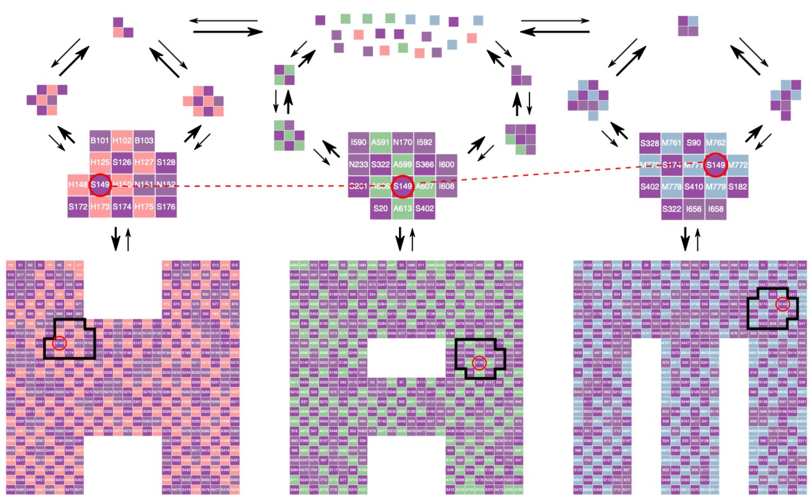
# How Hopfield and Hinton influenced my work over the years



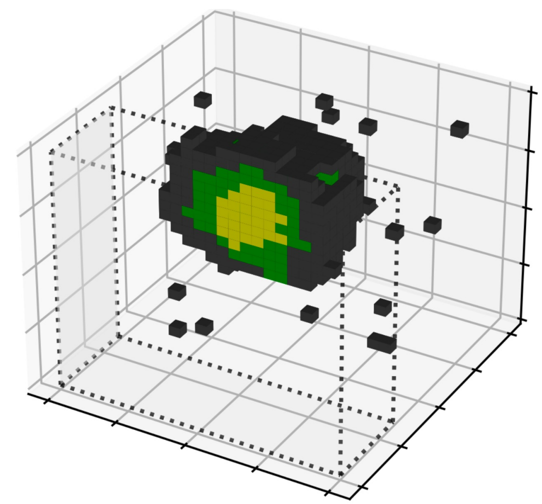

Kim, Hopfield, Winfree (NIPS, 2004); Kim, White, Winfree (MSB, 2006)



Poole et al (DNA23, 2017); Poole et al (arXiv, 2022)

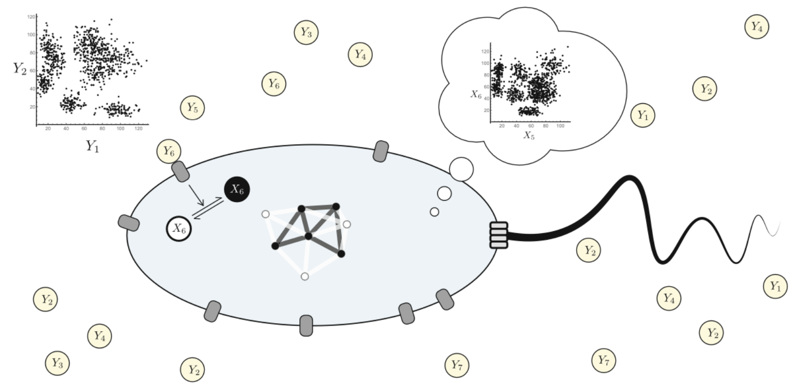


Evans, O'Brien, Winfree, Murugan (Nature, 2024)

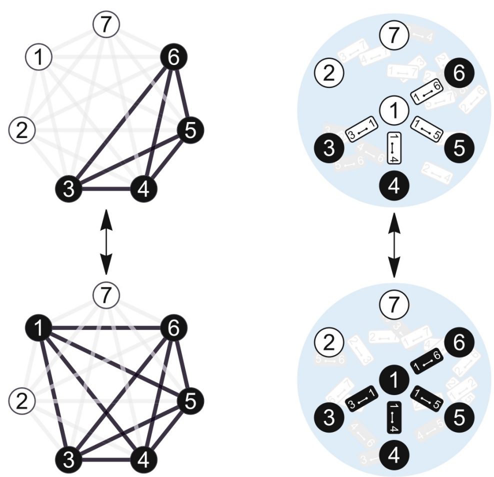


Chalk, Buse, Krishnan, Murugan, Winfree (DNA30, 2024)

# Stochastic chemical reaction networks can represent distributions



A0 + **B1** + WA0B1 $\rightleftharpoons$ **A1** + **B1** + **WA1B1**



Species $S_i$ with counts $s_i$ and energies $G[S_i] = g_i$

Full system free energy

$$\mathcal{G}(s) = \sum_{i=1}^{M} s_i G[S_i] + \log(s_i!)$$

Equilibrium Boltzmann distribution

$$\pi(s) = \frac{1}{Z} e^{-\mathcal{G}(s)} = \frac{1}{Z} \prod_{i=1}^{M} \frac{e^{-s_i G[S_i]}}{s_i!}$$

$$Z = \sum_{s' \in \Omega_{s_0}} e^{-\mathcal{G}(s')}$$

Learning rule $$\frac{dg_i}{dt} = -\frac{\partial D_{KL}}{\partial g_i} = \langle s_i \rangle_Q - \langle s_i \rangle_\pi$$

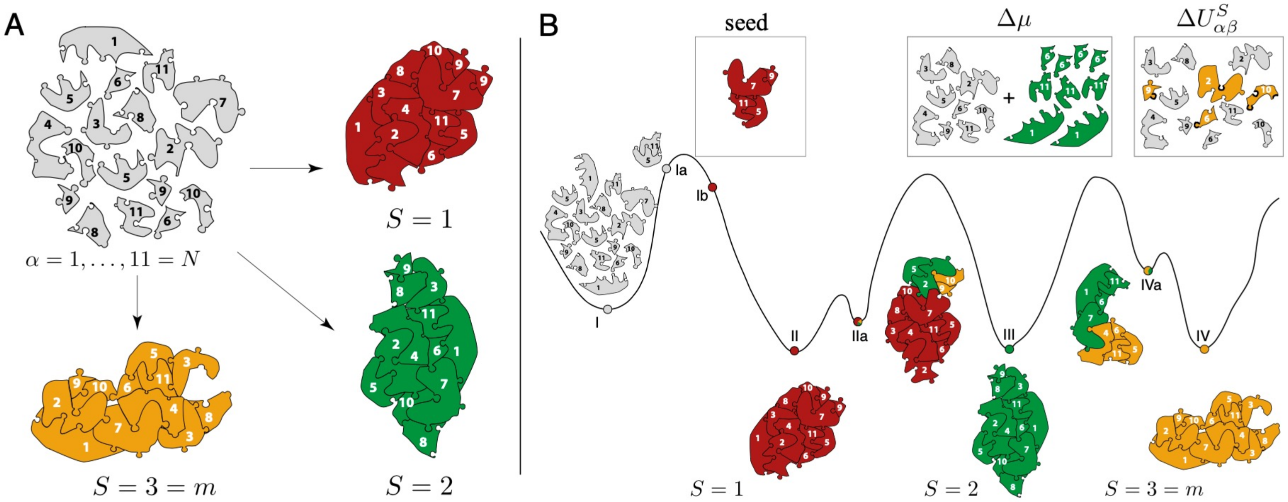
"Chemical Boltzmann Machines" (DNA23, 2017)
Poole, Ortiz-Muñoz, Behera, Jones, Ouldridge, Winfree, Gopalkrishnan
"Detailed balance chemical reaction networks as generalized Boltzmann machines"
Poole, Ouldridge, Gopalkrishnan, Winfree (arXiv, 2022)

# Self-assembly with memory...

## Multifarious assembly mixtures: Systems allowing retrieval of diverse stored structures

Arvind Murugan[a,b,1,2], Zorana Zeravcic[a,b,1,2], Michael P. Brenner[a,b], and Stanislas Leibler[c,d]

PNAS, 2015




$$J^S_{\alpha\beta} = \begin{cases} \delta & \text{if } \alpha, \beta \text{ adjacent in } S \\ 0 & \text{otherwise} \end{cases}$$

$$G(\text{subassembly of } S) = -\frac{1}{2}\sum_{\alpha,\beta} J^S_{\alpha\beta} x_\alpha x_\beta - \sum_\alpha \Theta_\alpha x_\alpha$$

$$\Theta_\alpha = RT \ln[\text{tile } \alpha]/u_0$$

$$x_\alpha = 1 \text{ if tile } \alpha \text{ present, else } 0$$

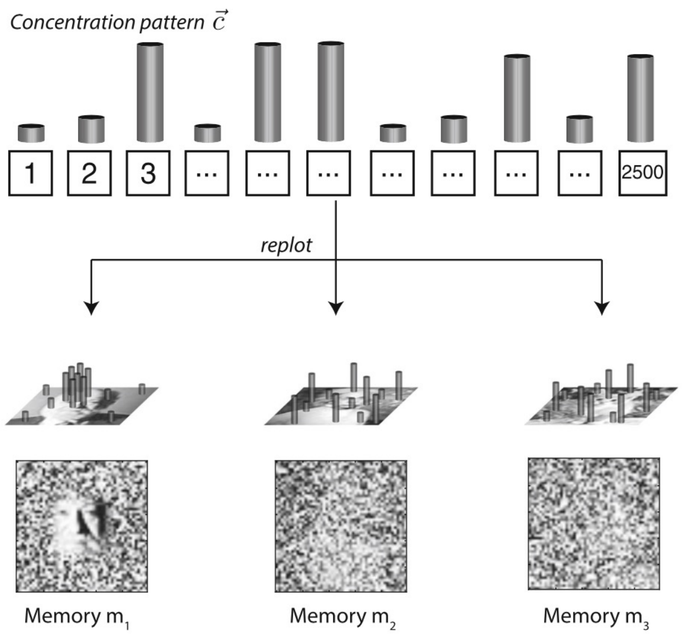
$$\text{attach } \alpha \text{ if } \sum_\beta J^S_{\alpha\beta} x_\beta + \Theta_\alpha > 0$$

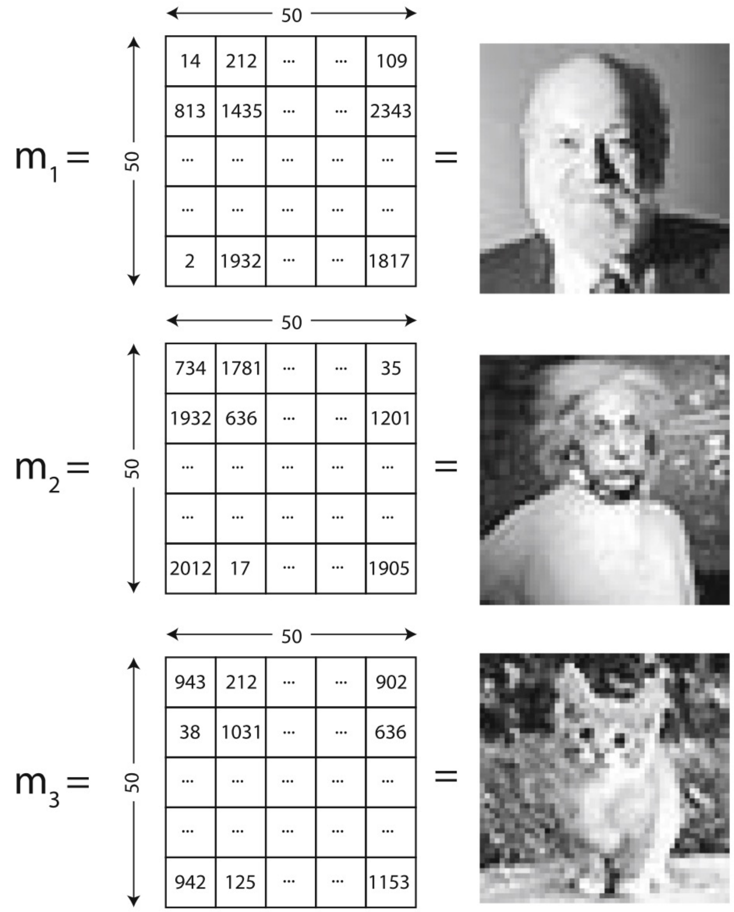
# ...can do pattern recognition...

J Stat Phys (2017) 167:806–826
DOI 10.1007/s10955-017-1774-2

## Associative Pattern Recognition Through Macro-molecular Self-Assembly

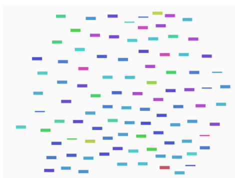
**Weishun Zhong**[1] · **David J. Schwab**[2] · **Arvind Murugan**[1]



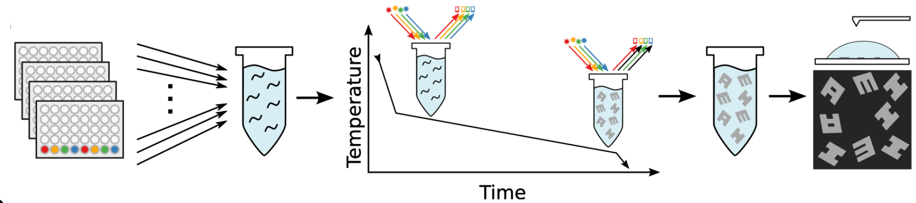
# Pattern Recognition in the Nucleation Kinetics of Non-Equilibrium Self-Assembly

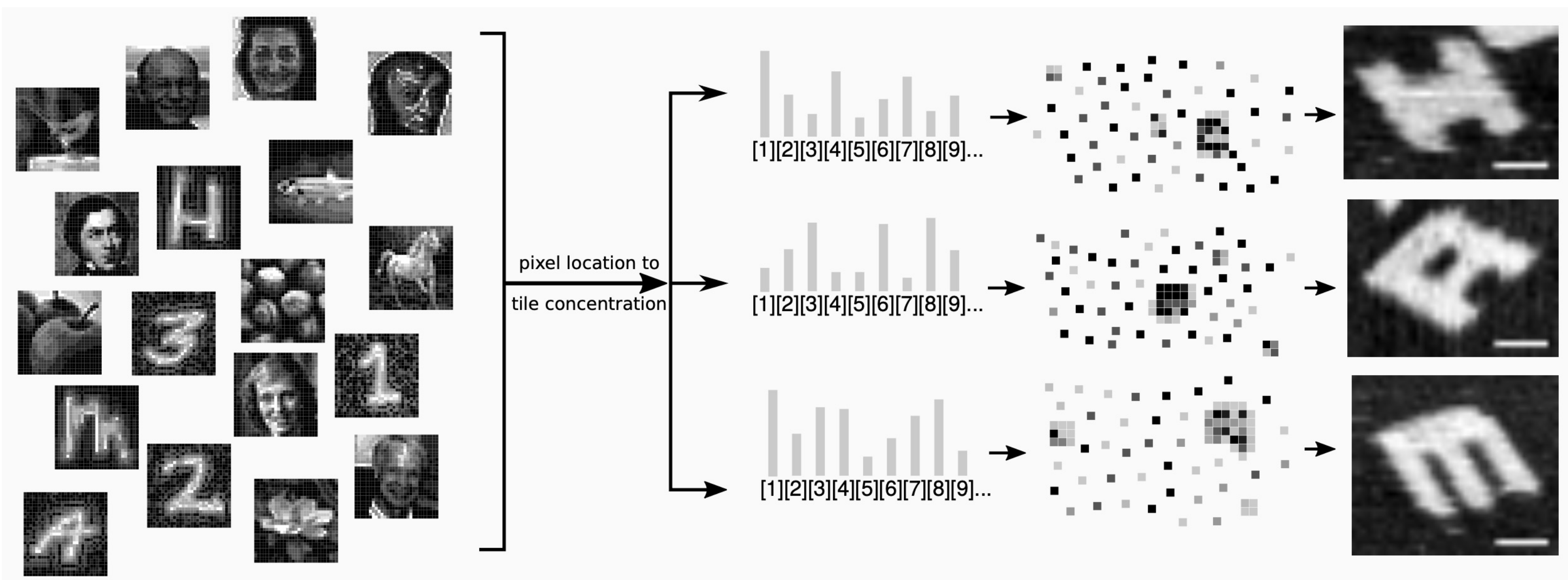
Constantine G. Evans, Jackson O'Brien, Erik Winfree, Arvind Murugan (Nature, 2024)



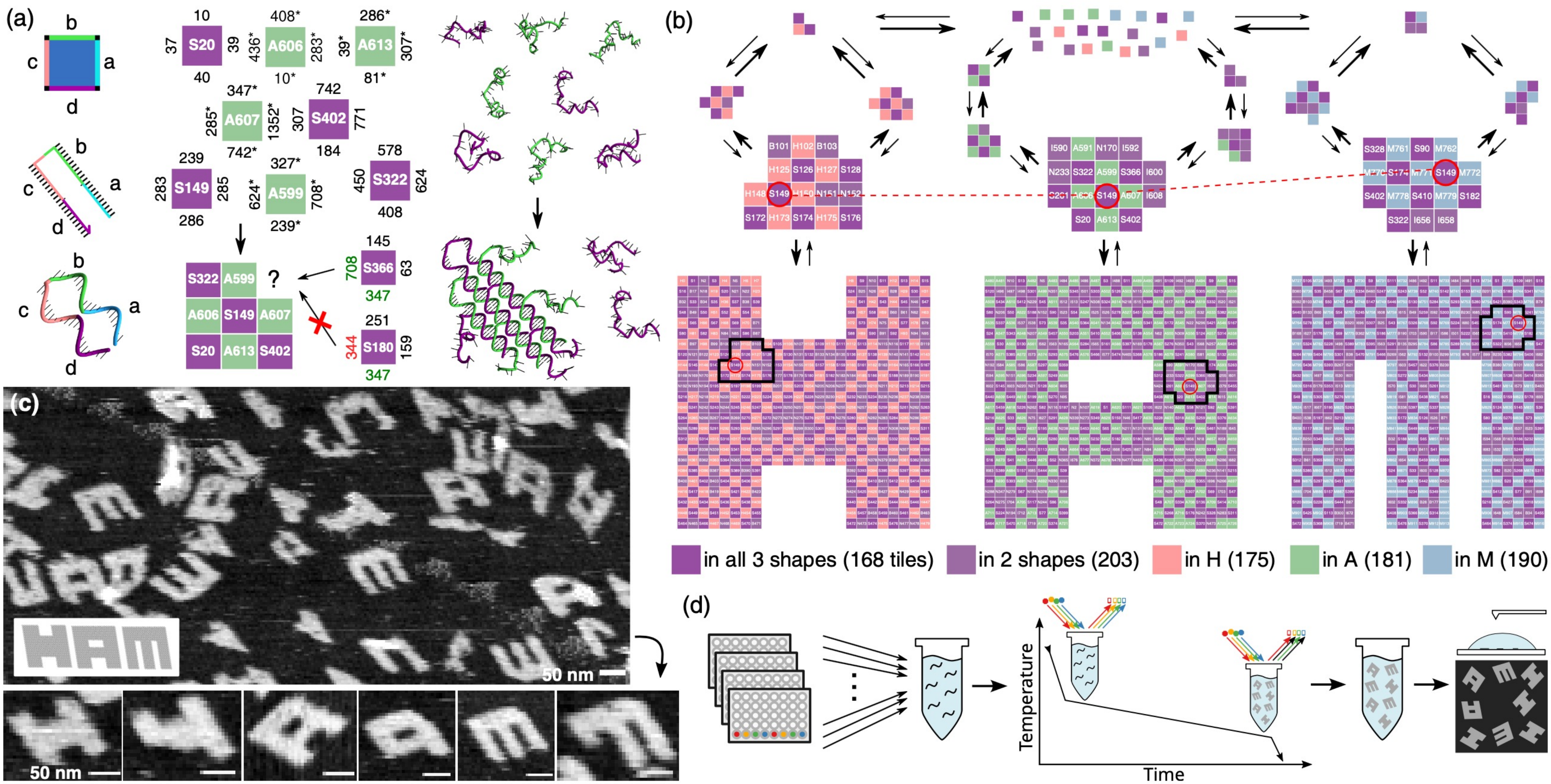
917 species

single-stranded tile



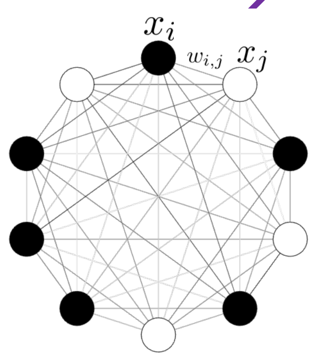
# Multifarious self-assembly in real life...

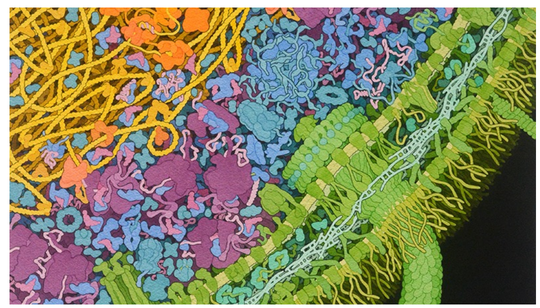


# Learning and Inference in a Lattice Model of Multicomponent Condensates

Cameron Chalk, Salvador Buse, Krishna Shrinivas, Arvind Murugan, Erik Winfree (DNA30, 2024)

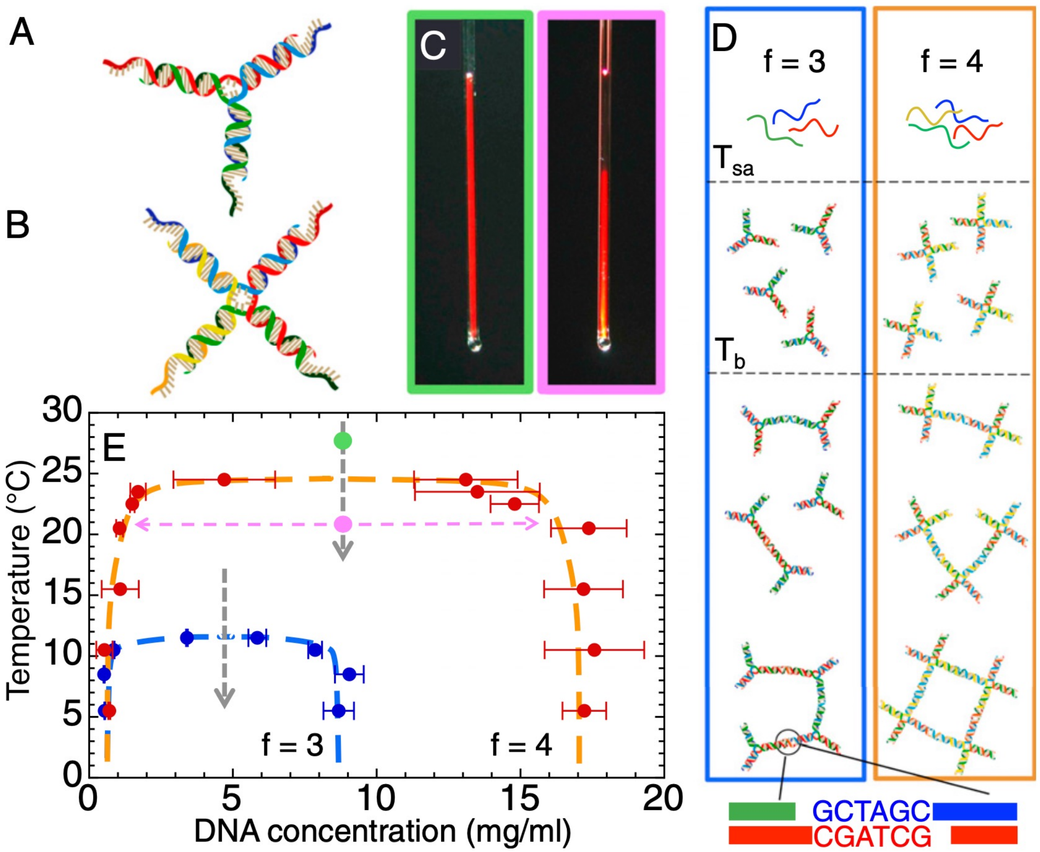
→ "Boltzmann liquids" ←



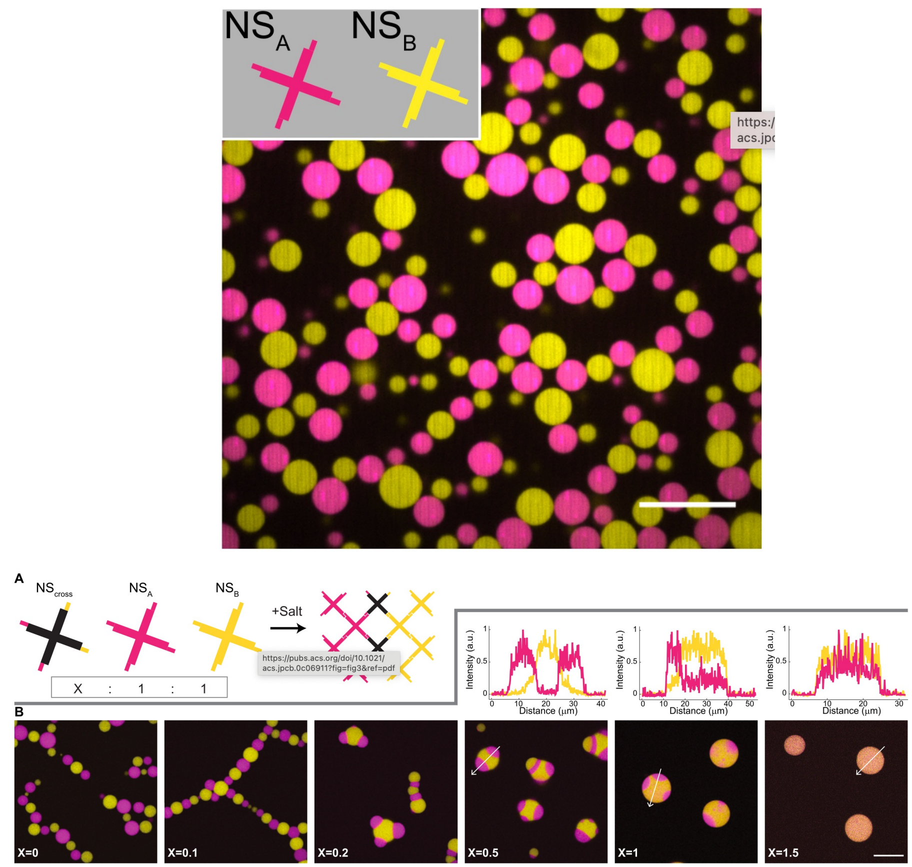



E. coli, by David Goodsell

# A toy system: programmable DNA nanostars



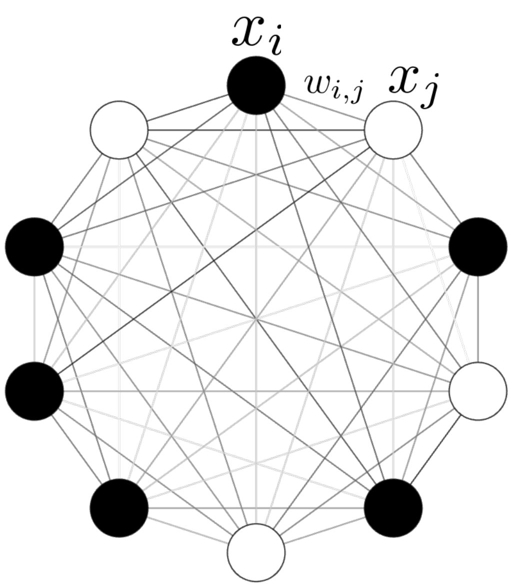

Biffi et al (PNAS, 2013)

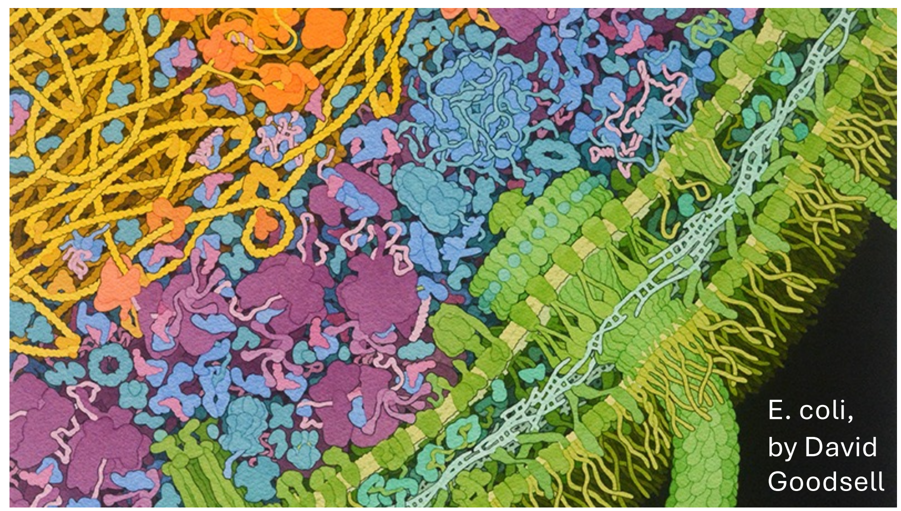



Jeon, Nguyen, Saleh (J Phys Chem B, 2020)

# Biomolecular systems are much like neural networks

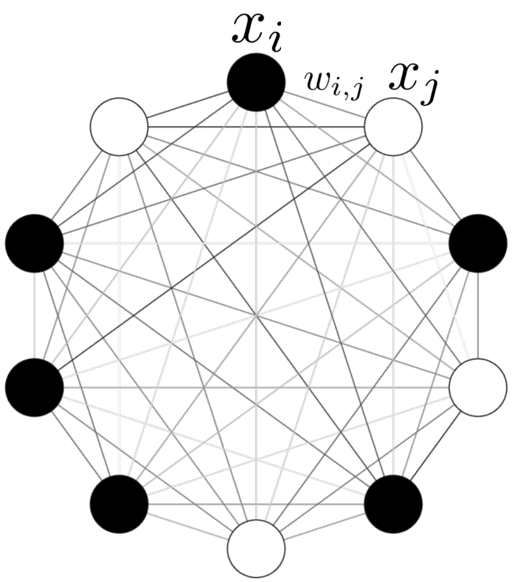
(1) Their parameters encode probability distributions
(2) They do inference: compute conditional distributions
(3) Their parameters can be learned
(4) Learned parameters generalize beyond their training set



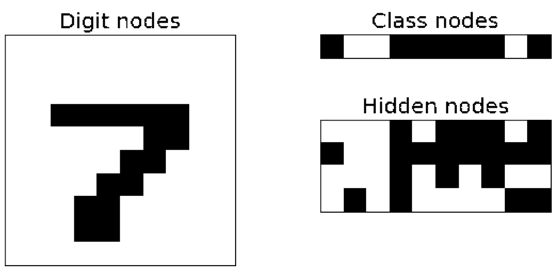



# Biomolecular systems are much like neural networks

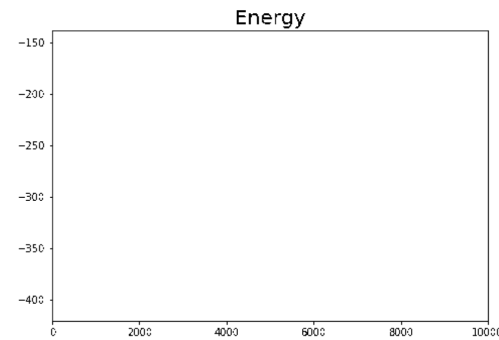
**(1) Their parameters encode probability distributions**
(2) They do inference: compute conditional distributions
(3) Their parameters can be learned
(4) Learned parameters generalize beyond their training set

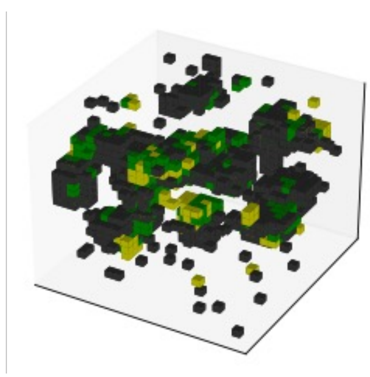



# Probabilistic computation can help us understand condensates

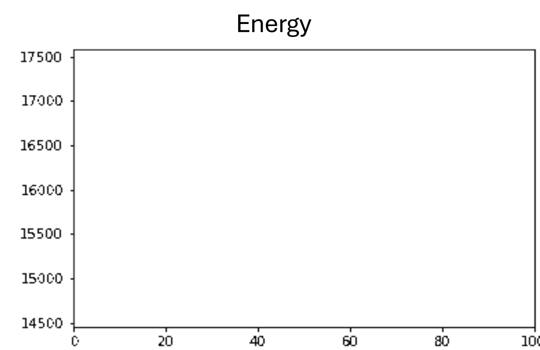



$$P(x) = \frac{1}{Z} e^{-E(x)/kT}$$



$$P(s) = \frac{1}{Z} e^{-G(s)/kT}$$

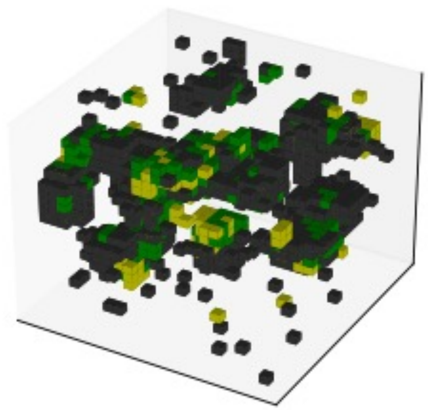


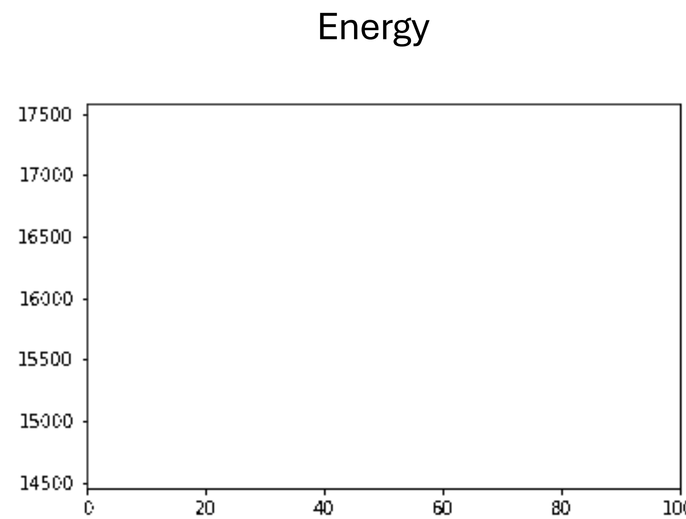
# Equilibrium lattice model of biomolecular condensates

**Boltzmann liquids:**
Isotropic molecules in a discrete lattice

Canonical (closed box) or
Grand Canonical (open box)

Metropolis-Hasting dynamics
sample equilibrium in the limit of time



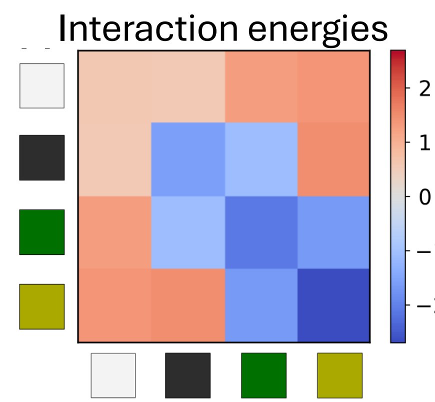



Energy:

$$G(s) = \frac{1}{2} \sum_{x \in L} \sum_{y \in \mathcal{N}(x)} G_{s(x),s(y)} + \sum_{x \in L} G_{s(x)}$$

Probability distribution (Boltzmann distribution):

$$P(s) = \frac{1}{Z} e^{-G(s)/kT} \qquad Z = \sum_{s} e^{-G(s)/kT}$$



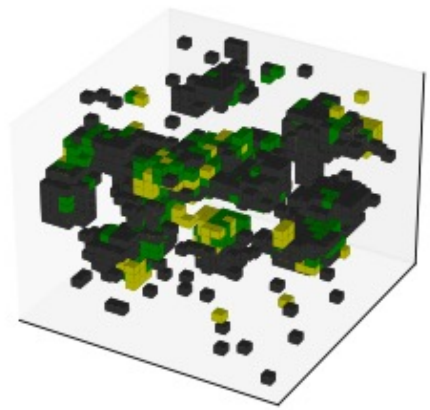

Same model as Jacobs, Frenkel, 2013, 2017

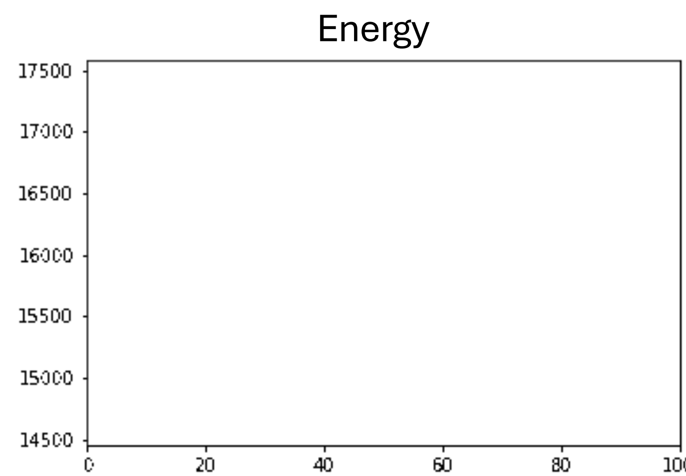
# Equilibrium lattice model of biomolecular condensates

**Boltzmann liquids:**
Isotropic molecules in a discrete lattice

Canonical (closed box) or
Grand Canonical (open box)

Metropolis-Hasting dynamics
sample equilibrium in the limit of time





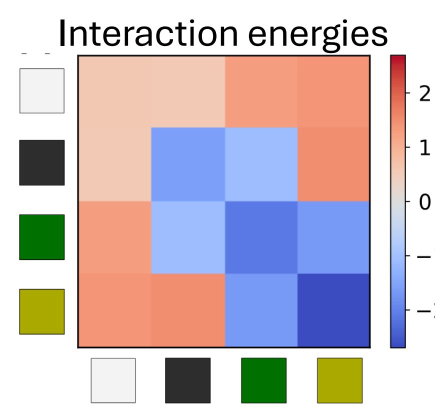

Energy:

Neighbor-neighbor molecular interactions

$$G(s) = \boxed{\frac{1}{2}\sum_{x\in L}\sum_{y\in\mathcal{N}(x)} G_{s(x),s(y)}} + \sum_{x\in L} G_{s(x)}$$

Probability distribution (Boltzmann distribution):

$$P(s) = \frac{1}{Z}e^{-G(s)/kT} \qquad Z = \sum_{s} e^{-G(s)/kT}$$



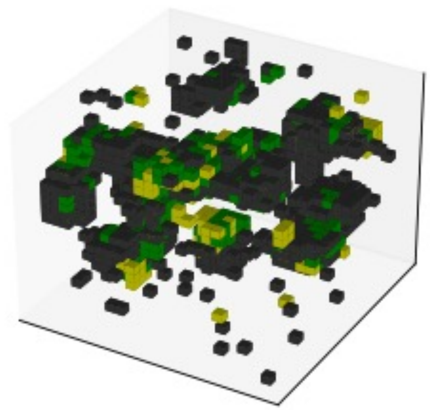

Same model as Jacobs, Frenkel, 2013, 2017

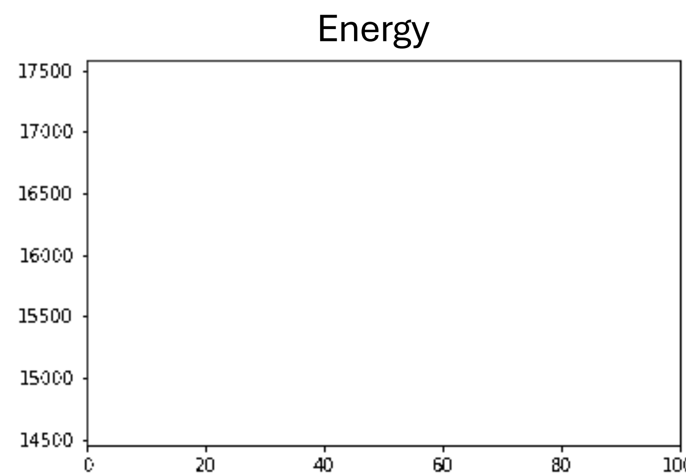
# Equilibrium lattice model of biomolecular condensates

**Boltzmann liquids:**
Isotropic molecules in a discrete lattice

Canonical (closed box) or
Grand Canonical (open box)

Metropolis-Hasting dynamics
sample equilibrium in the limit of time






Energy:

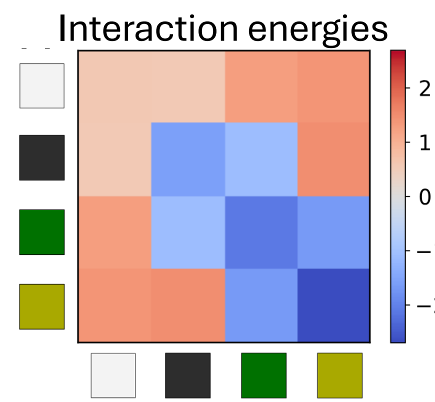
Neighbor-neighbor molecular interactions

$$G(s) = \boxed{\frac{1}{2}\sum_{x\in L}\sum_{y\in\mathcal{N}(x)} G_{s(x),s(y)}} + \boxed{\sum_{x\in L} G_{s(x)}}$$

Per-molecule energy contribution

Probability distribution (Boltzmann distribution):

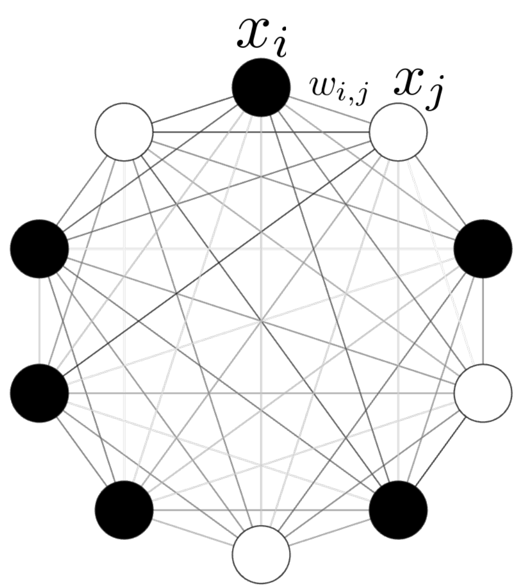
$$P(s) = \frac{1}{Z}e^{-G(s)/kT} \qquad Z = \sum_{s} e^{-G(s)/kT}$$




Same model as Jacobs, Frenkel, 2013, 2017

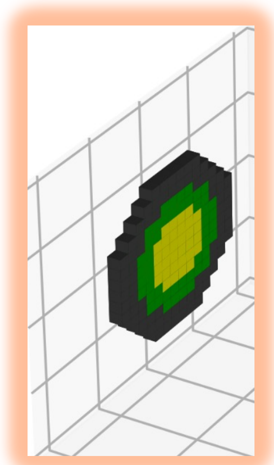
# Biomolecular systems are much like neural networks

(1) Their parameters encode probability distributions
**(2) They do inference: compute conditional distributions**
(3) Their parameters can be learned
(4) Learned parameters generalize beyond their training set

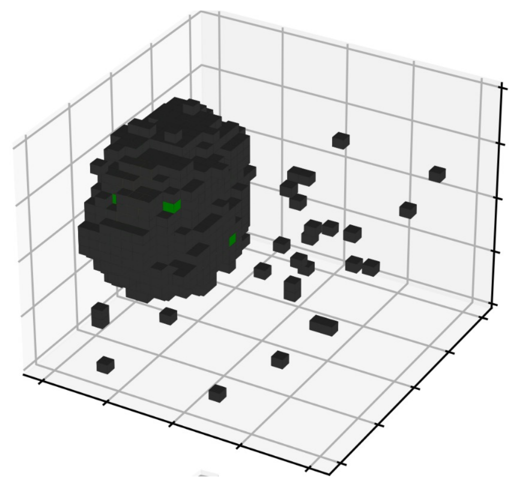



# Clamping a surface localizes condensation

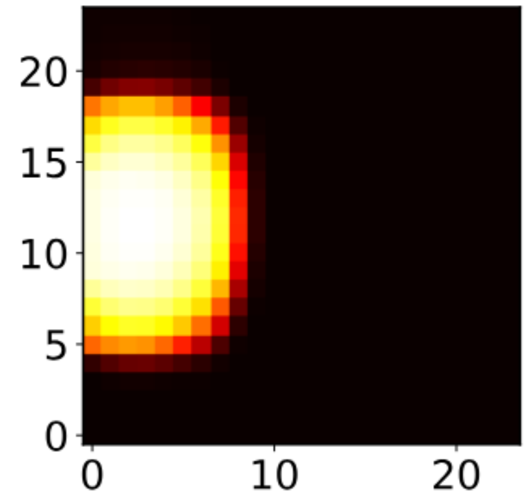
“Surface clamp”



Simulation snapshot

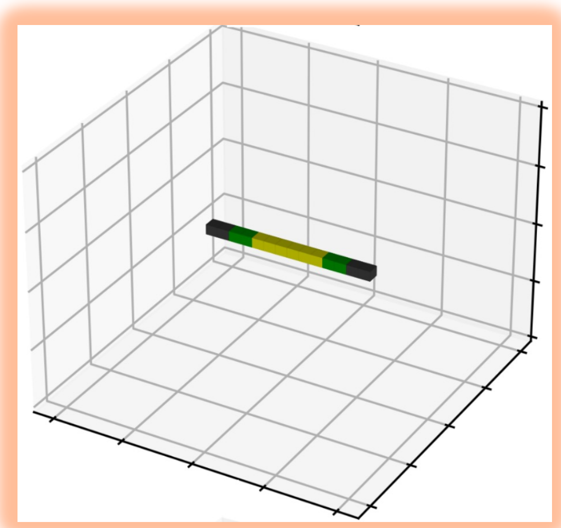


Nonsolvent species heatmap
10,000 independent trials

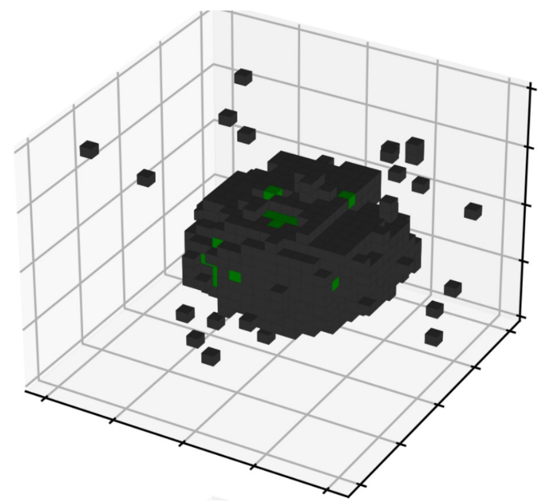


# Clamping a polymer localizes condensation around the polymer

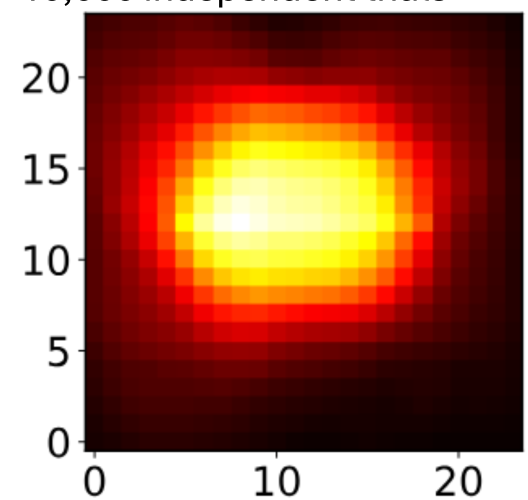
“Polymer clamp”



Simulation snapshot

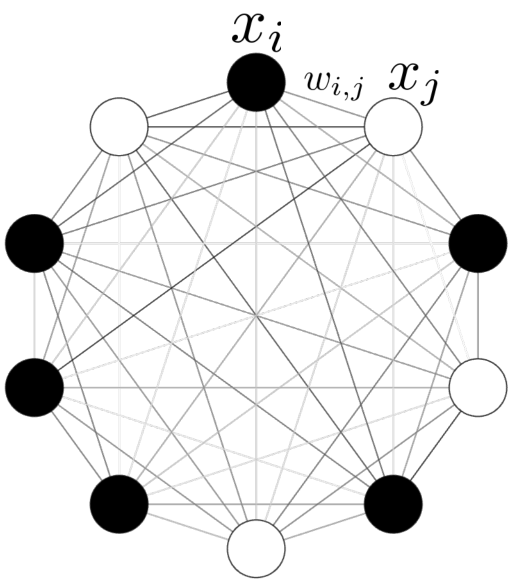


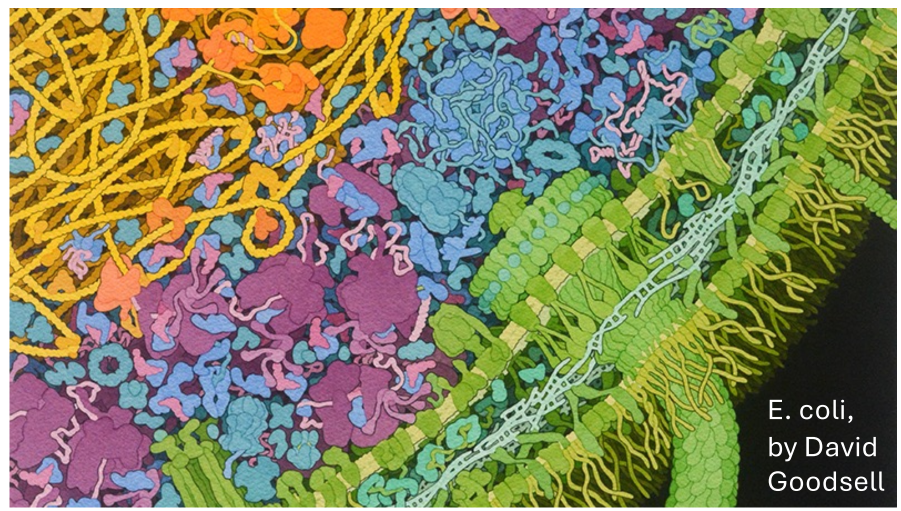
Nonsolvent species heatmap
10,000 independent trials



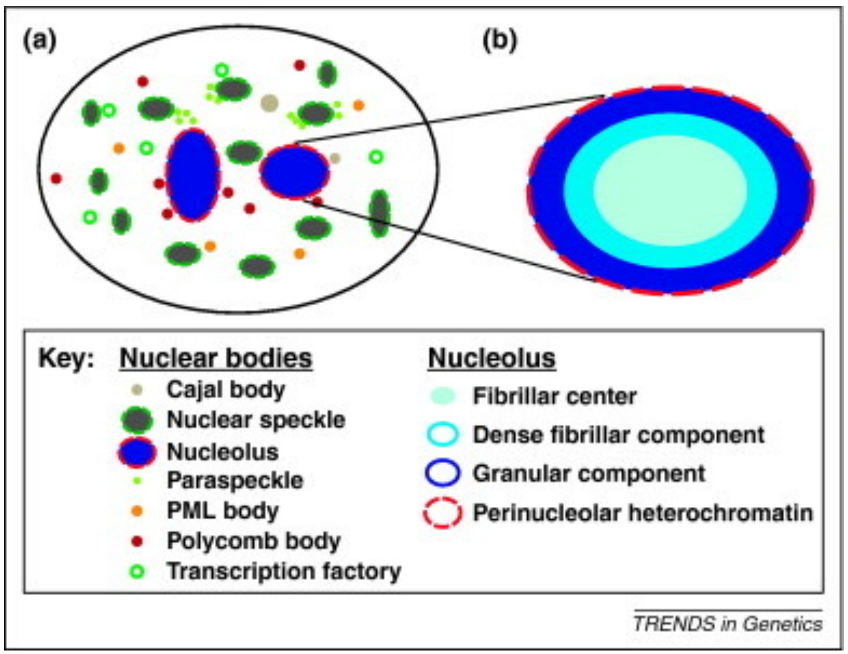
# Biomolecular systems are much like neural networks

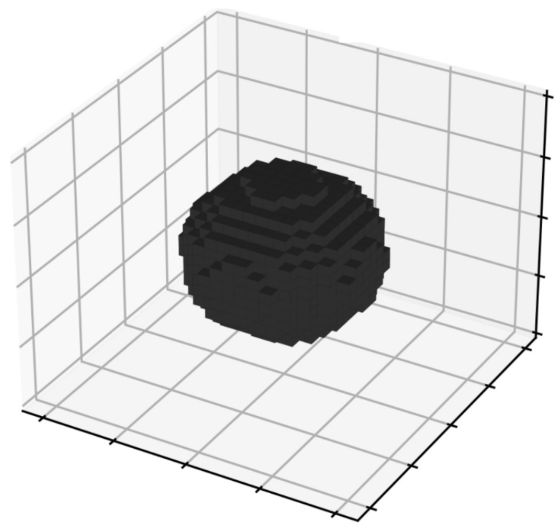
(1) Their parameters encode probability distributions
(2) They do inference: compute conditional distributions
**(3) Their parameters can be learned**
(4) Learned parameters generalize beyond their training set

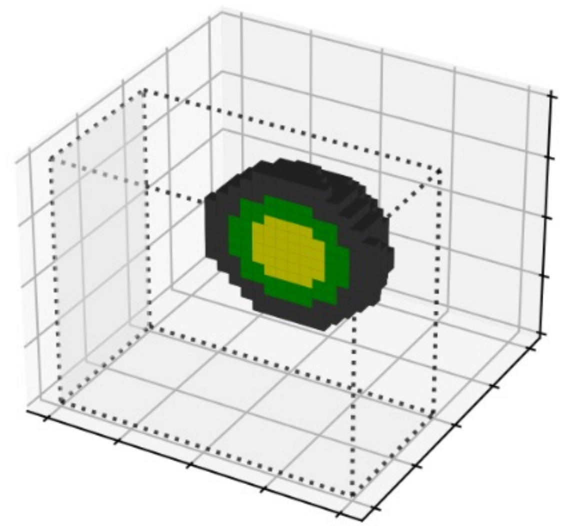


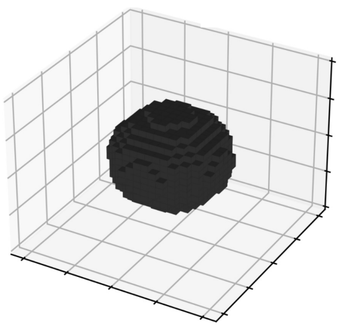
# Task 1: Learning “Avocado” morphology



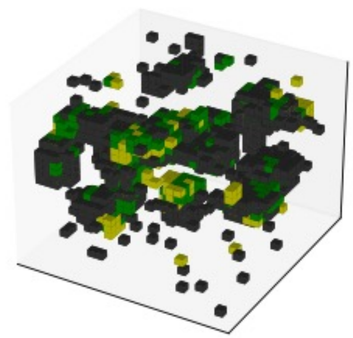





# Condensation parameters can be learned



Q



P

$$R(Q_v \mid\mid P_v) = \sum_v Q_v(v) \ln \frac{Q_v(v)}{P_v(v)}$$

$\langle n_{i,j} \rangle_D$ : Expected number of interfaces of molecule types *i* and *j* in dist. *D*

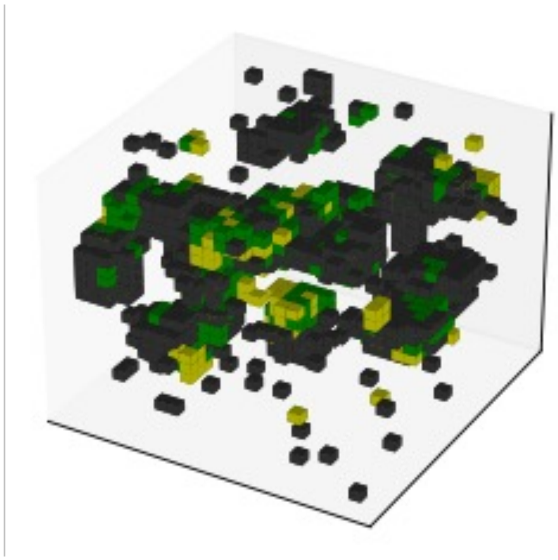
$P_v(v)$ : Marginal dist. over visible positions for current $G_{i,j}$

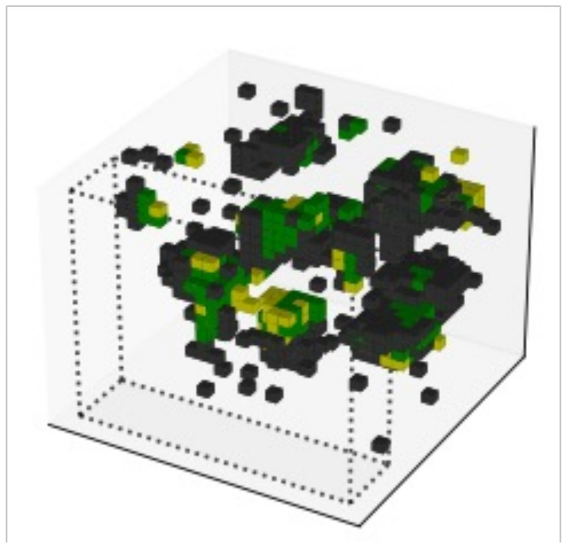
$Q_v(v)$ : Target distribution over visible positions

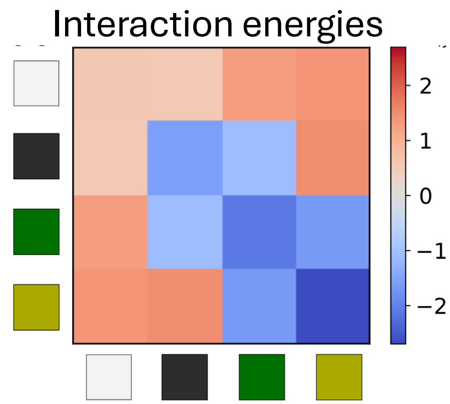
$$\frac{\partial R(Q_v \mid\mid P_v)}{\partial G_{i,j}} = \langle n_{i,j} \rangle_Q - \langle n_{i,j} \rangle_P$$

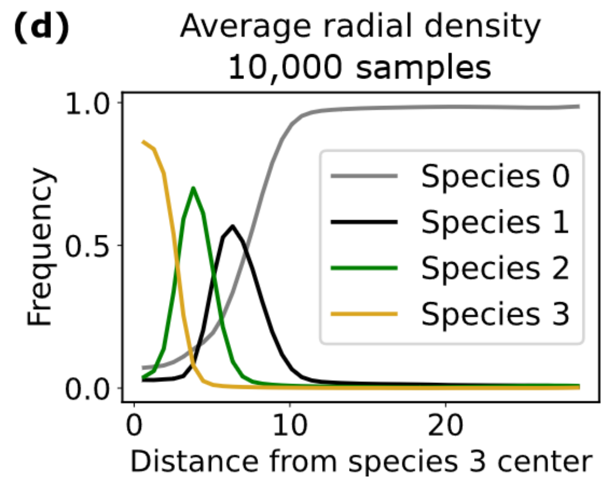
$$\frac{\mathrm{d}G_{i,j}}{\mathrm{d}t} = \langle n_{i,j} \rangle_P - \langle n_{i,j} \rangle_Q$$

# Learned parameters form correct morphology



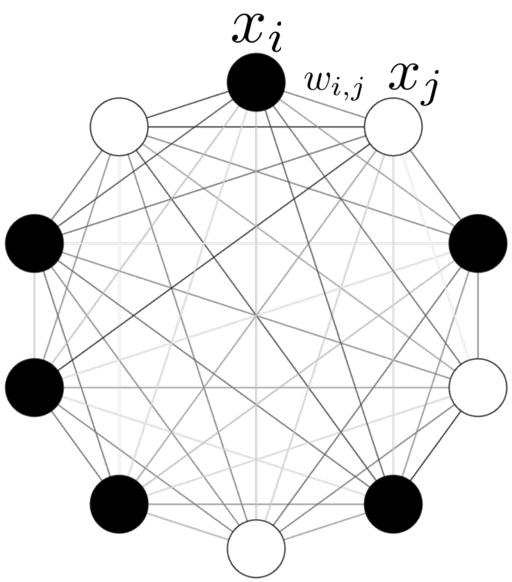




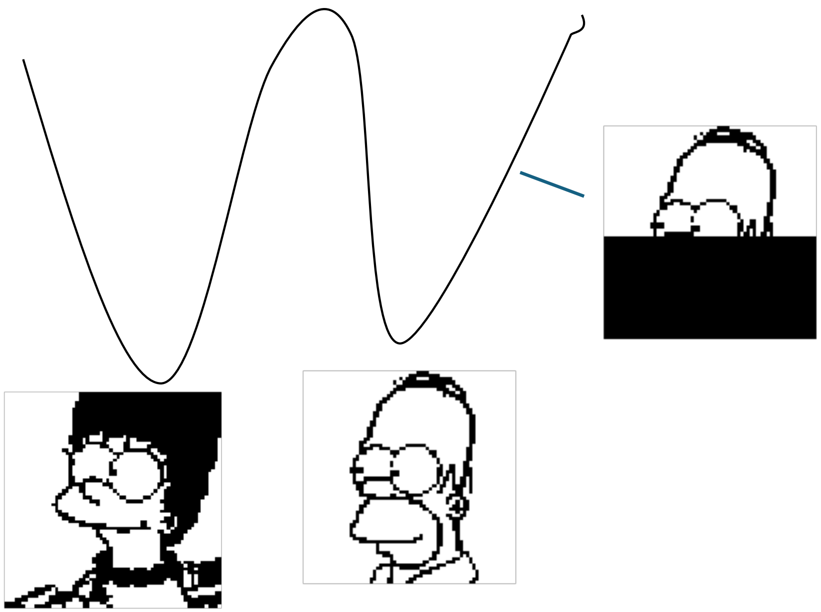
# Biomolecular systems are much like neural networks

(1) Their parameters encode probability distributions
(2) They do inference: compute conditional distributions
(3) Their parameters can be learned
**(4) Learned parameters generalize beyond their training set**



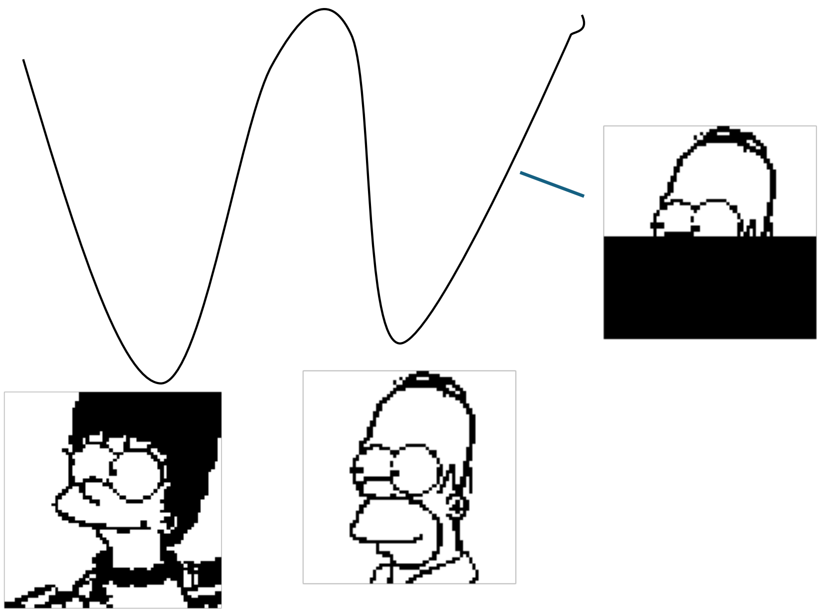

# Task 2: "Associative recall" of nonorthogonal compositions

Hopfield network "memories"

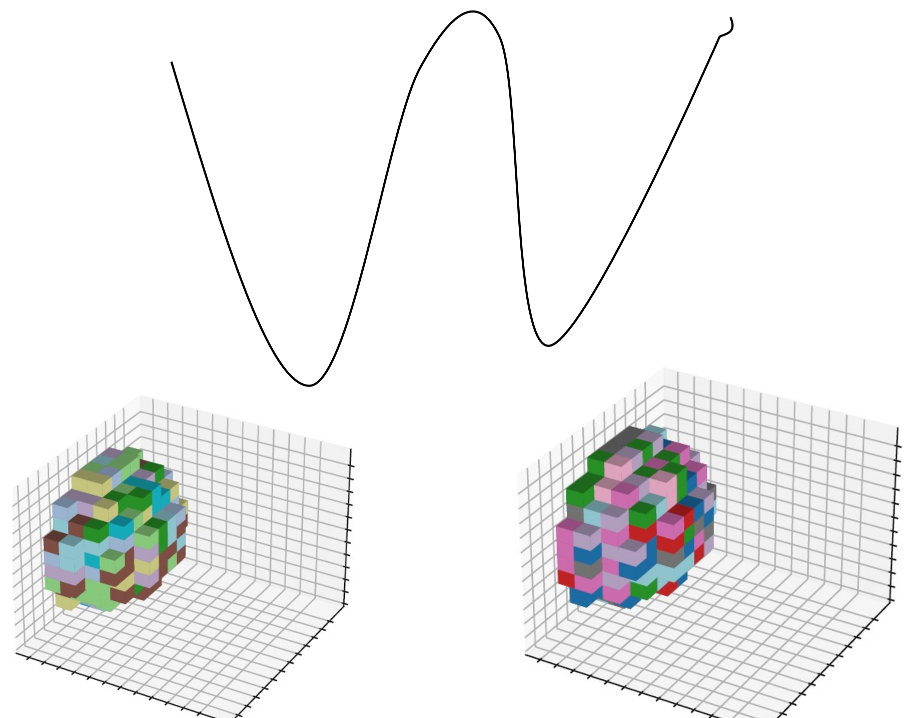


# Task 2: “Associative recall” of nonorthogonal compositions

## Hopfield network “memories”

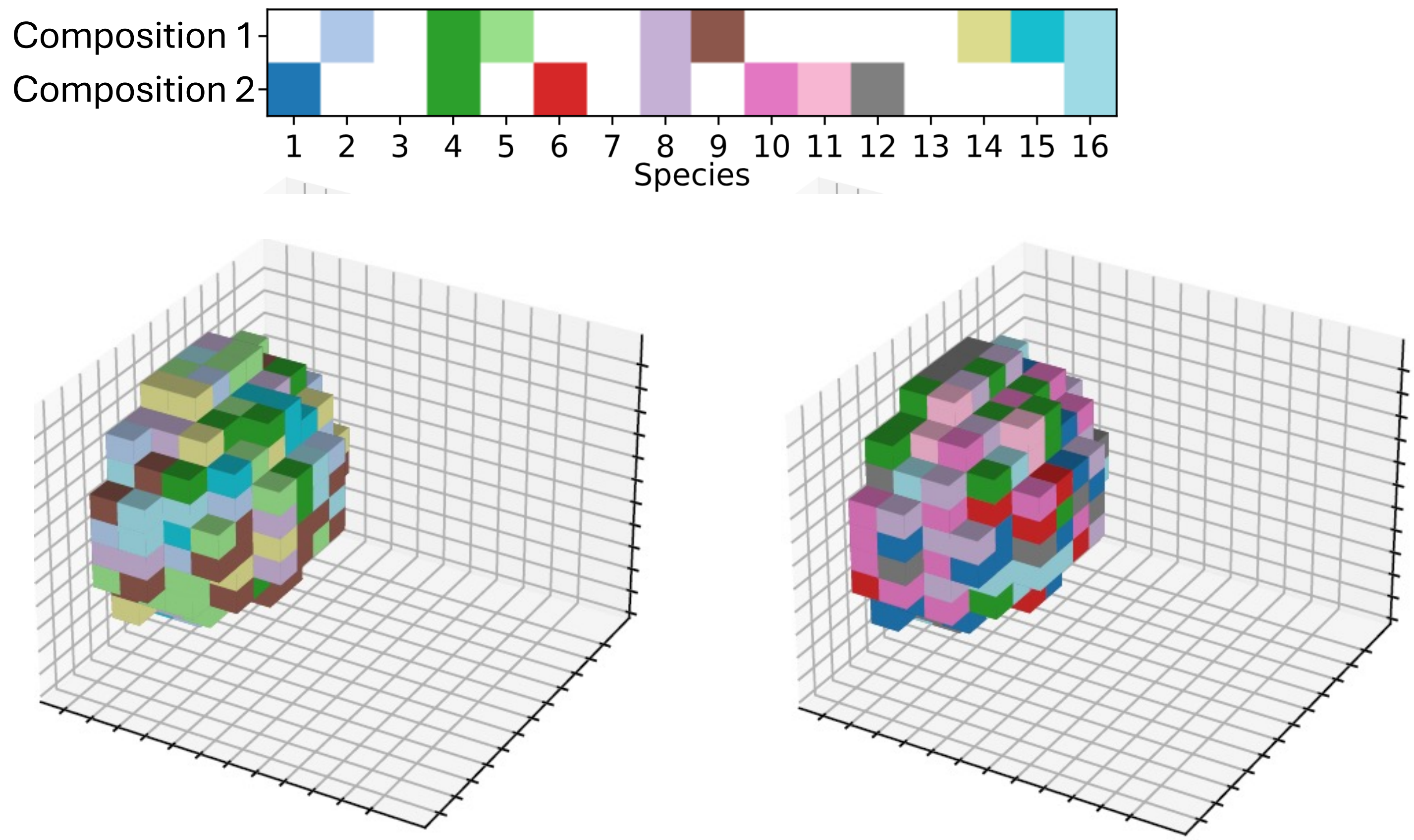


## Nonorthogonal condensates

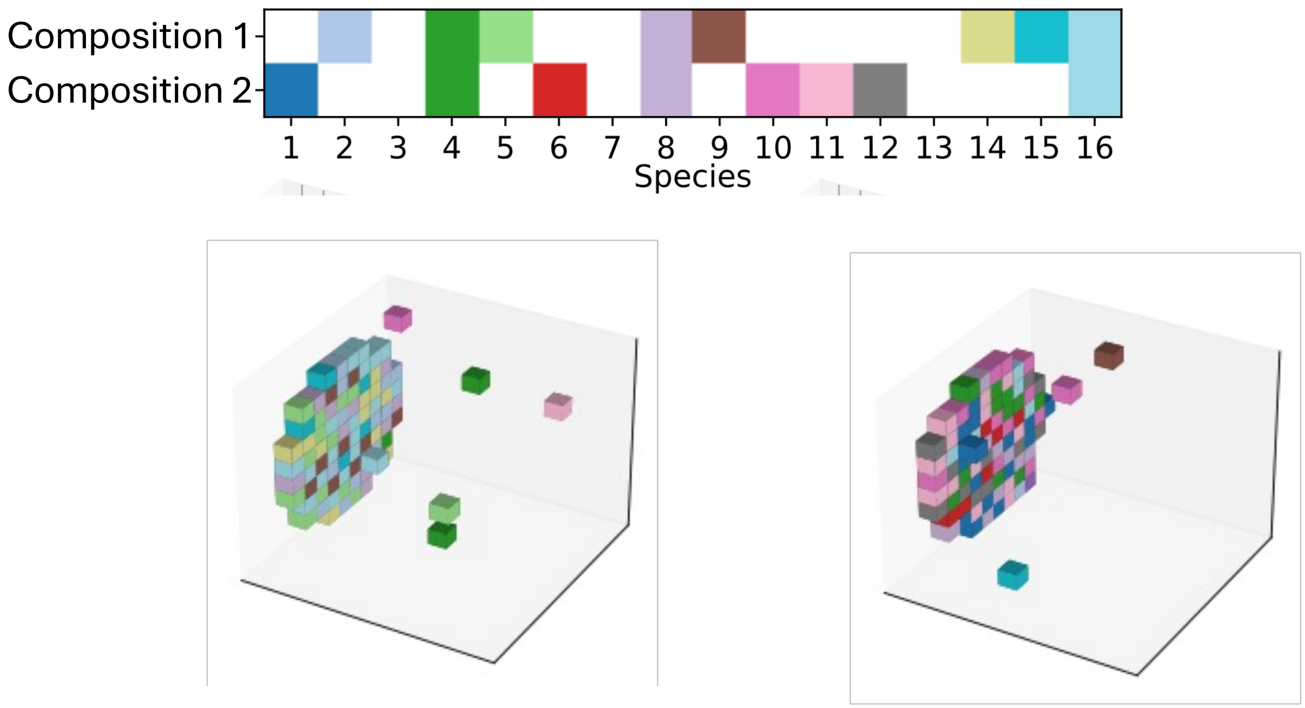


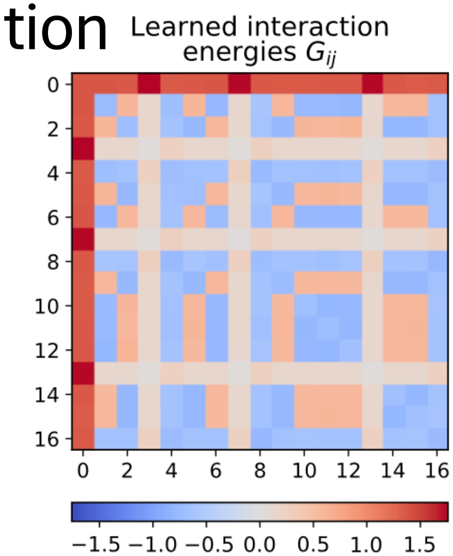
Study in bulk case: Texeira et al., 2023

# Nonorthogonal condensates share some species



# Condensate composition matches surface composition

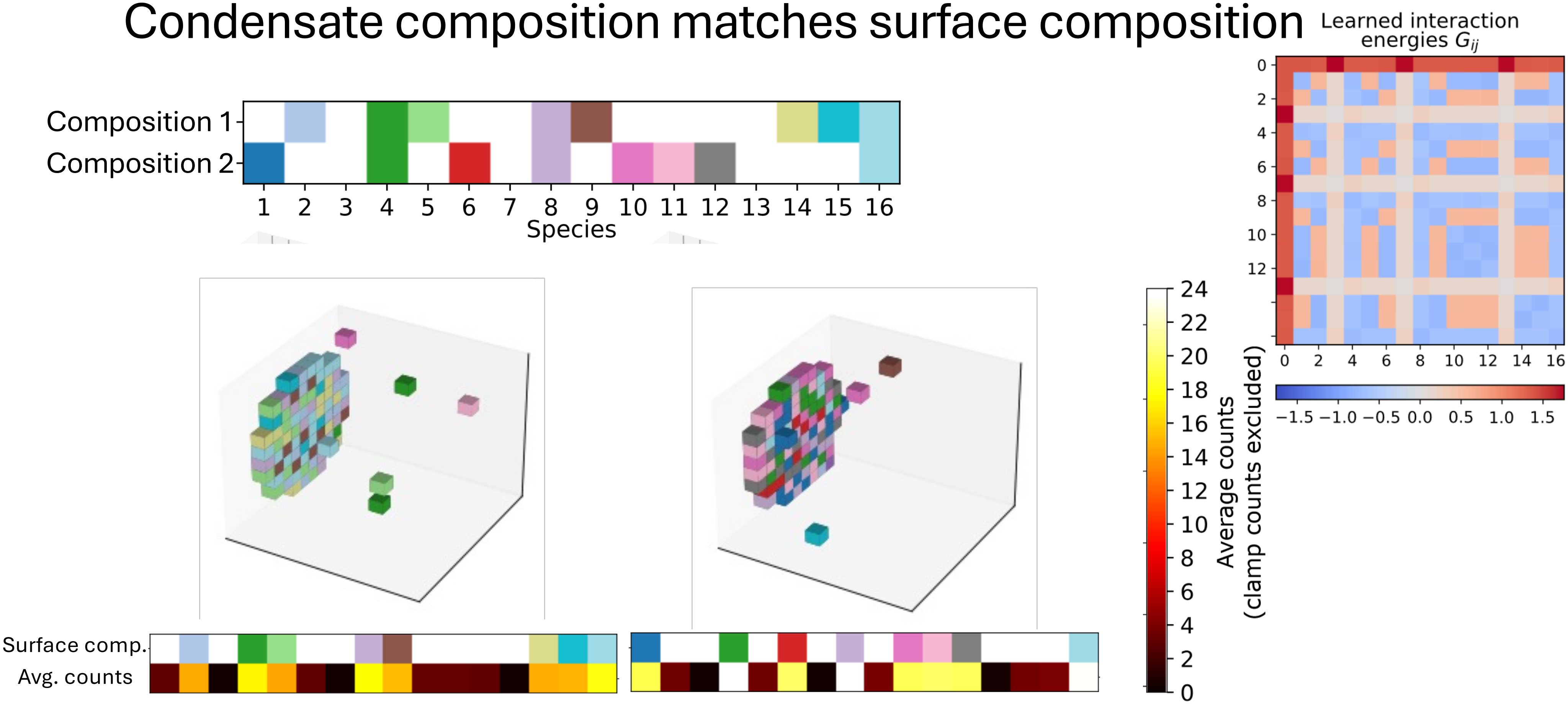


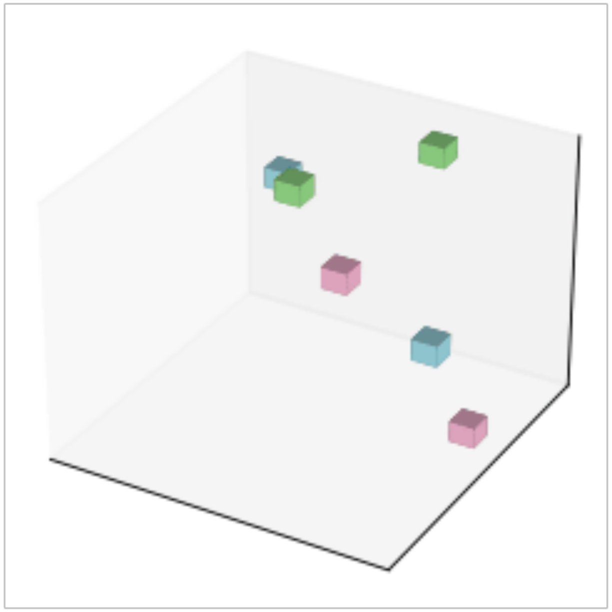

Clamping the surface: the environment provides energy

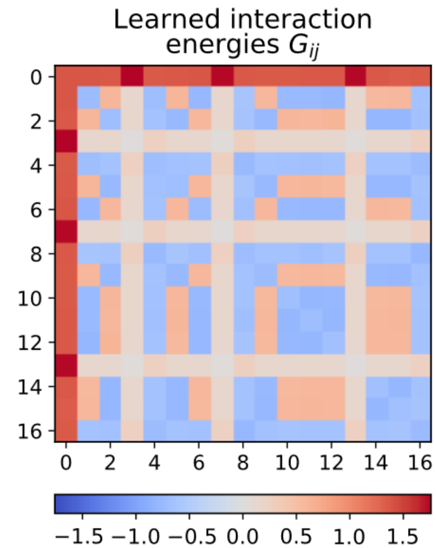
Settling back to equilibrium: the “cell” does inference without using energy

# Condensate composition matches surface composition

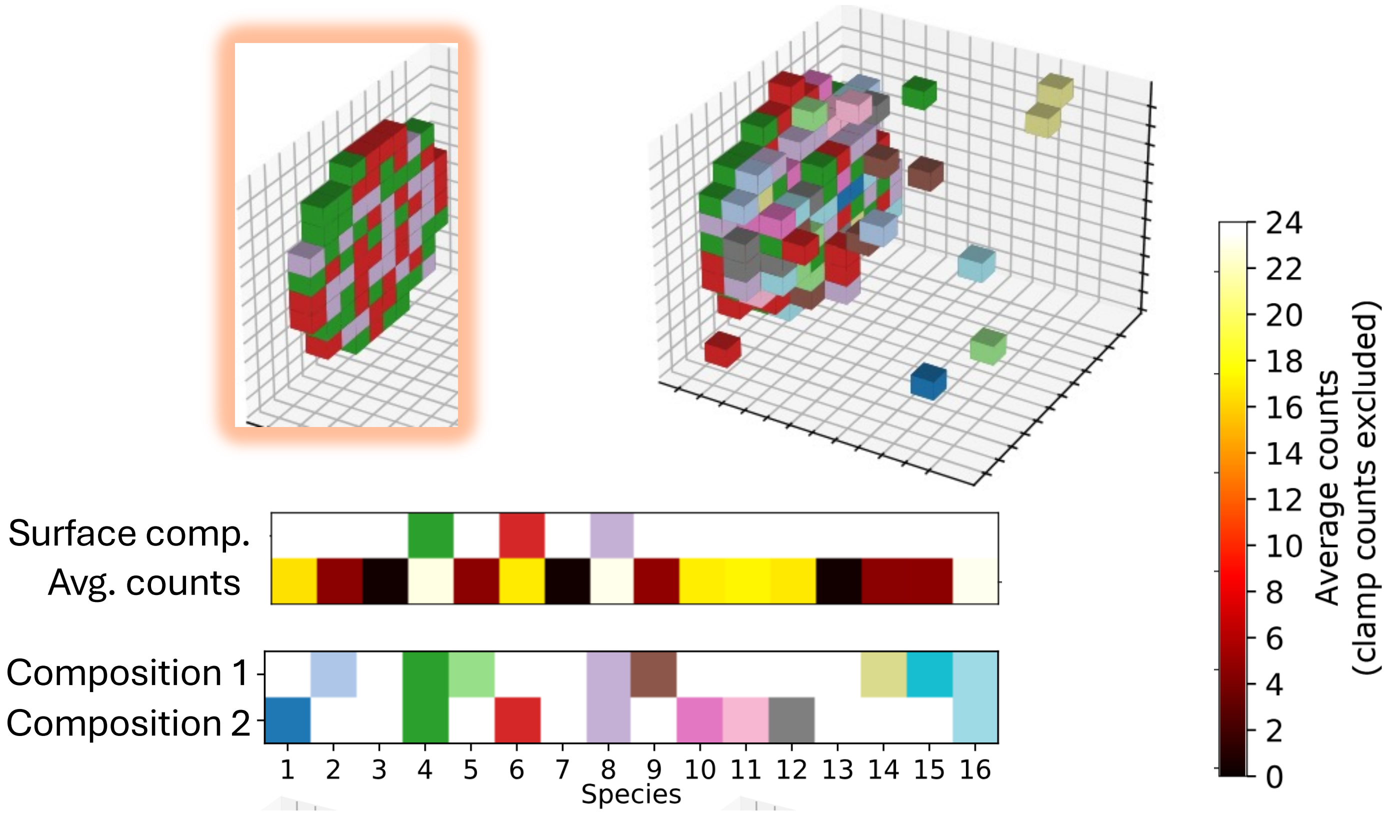


# No surface clamp results in no condensation

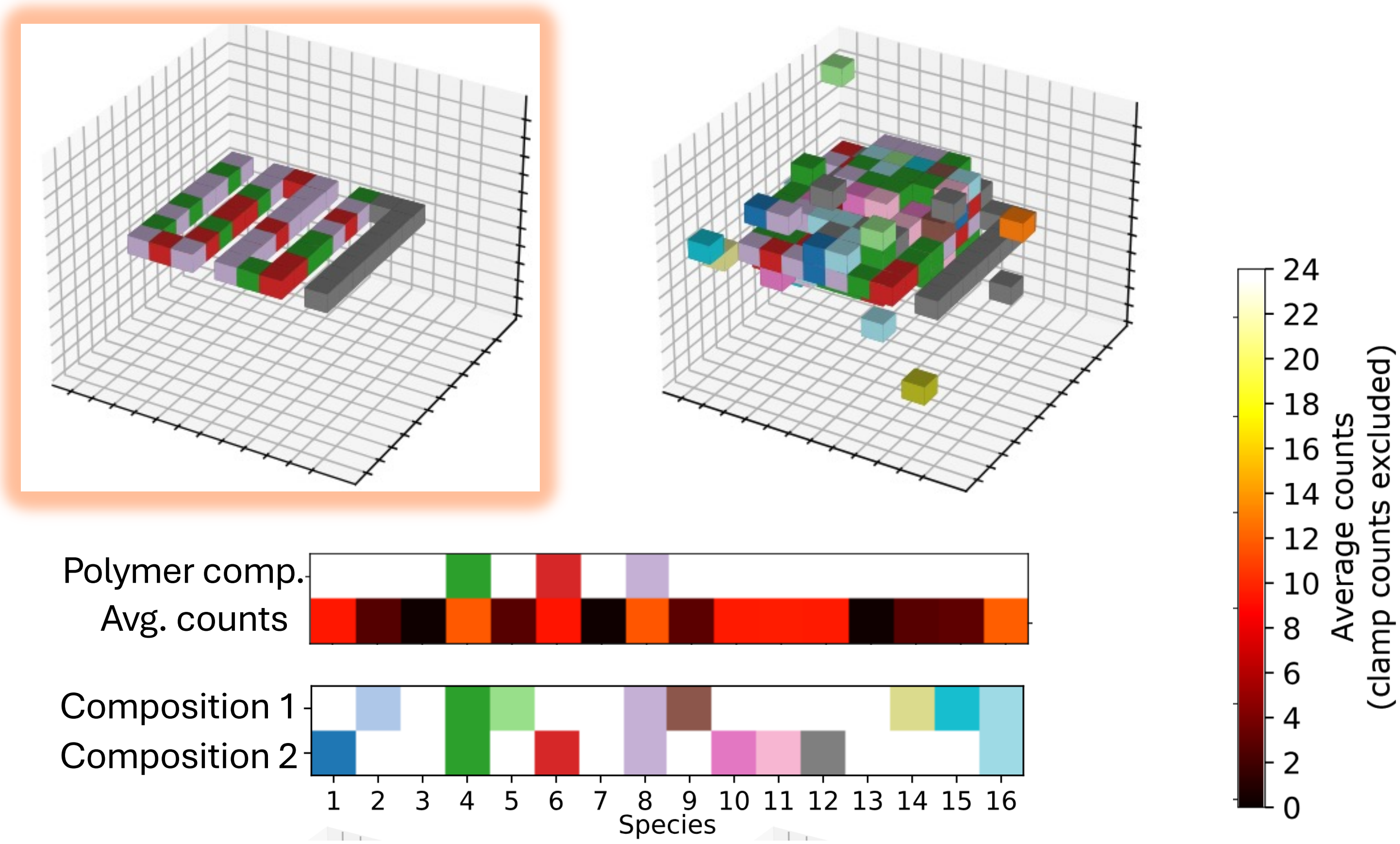




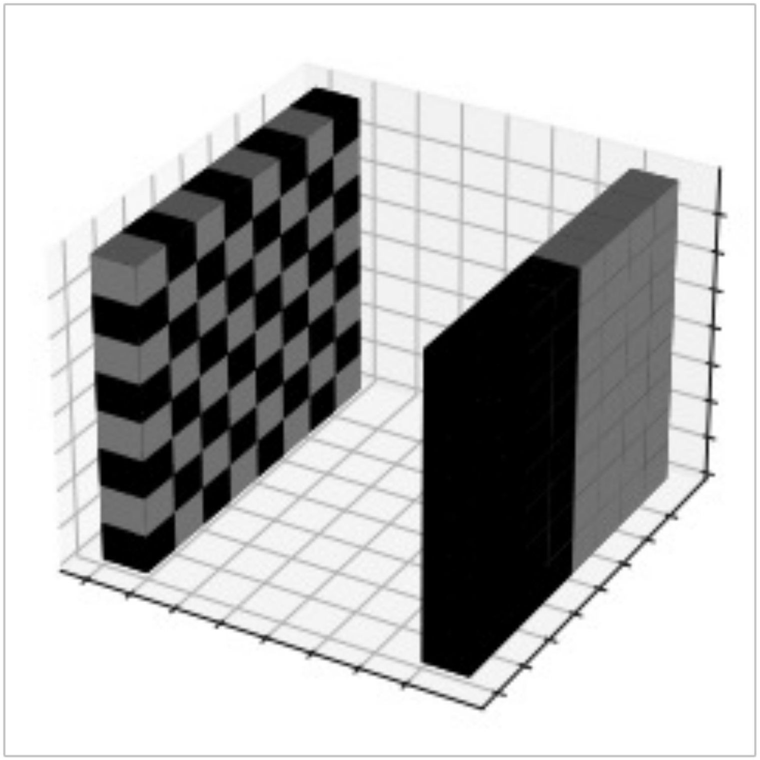
# A surface of partial composition recalls full composition



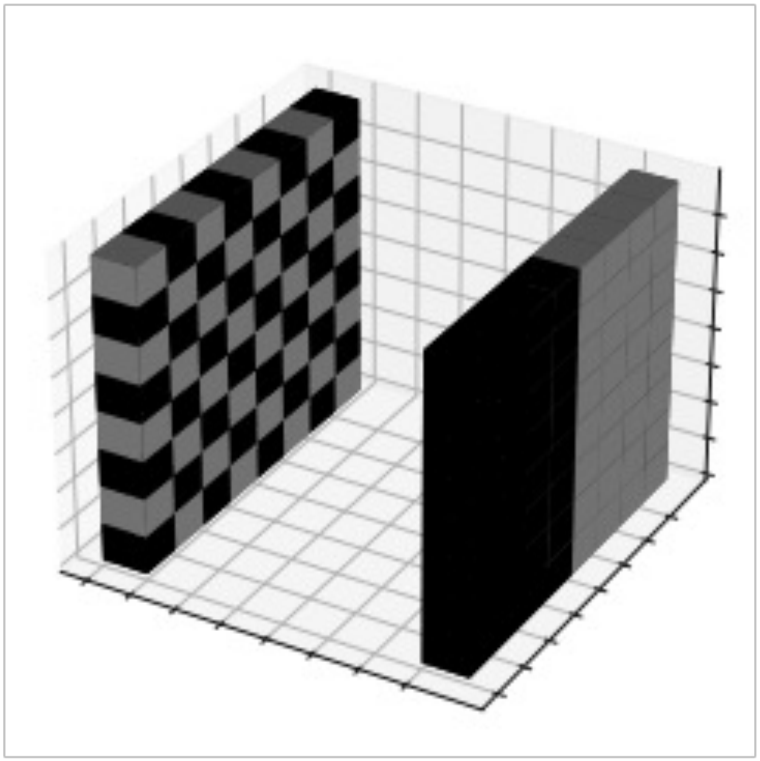
# A polymer of partial composition recalls full composition



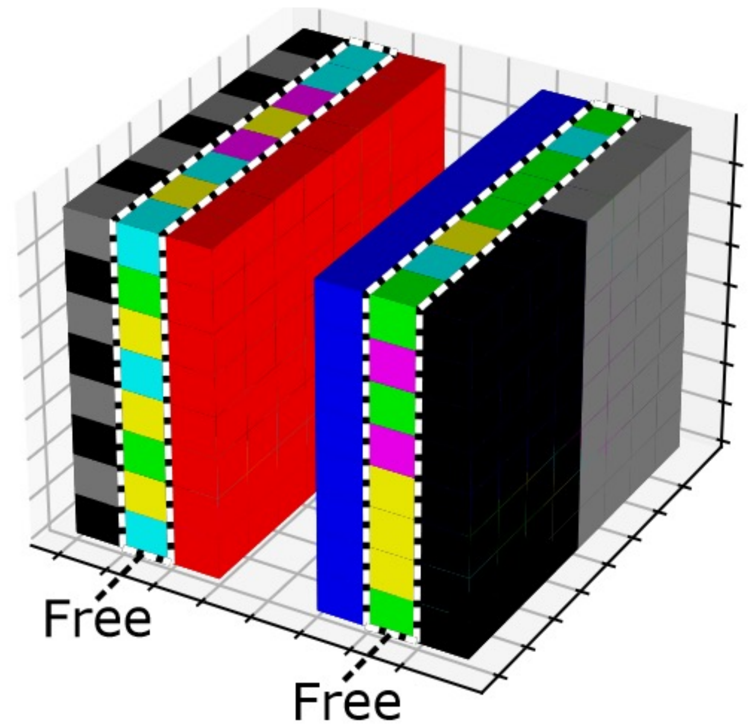
# Task 3: Recognizing surface *arrangement* instead of composition



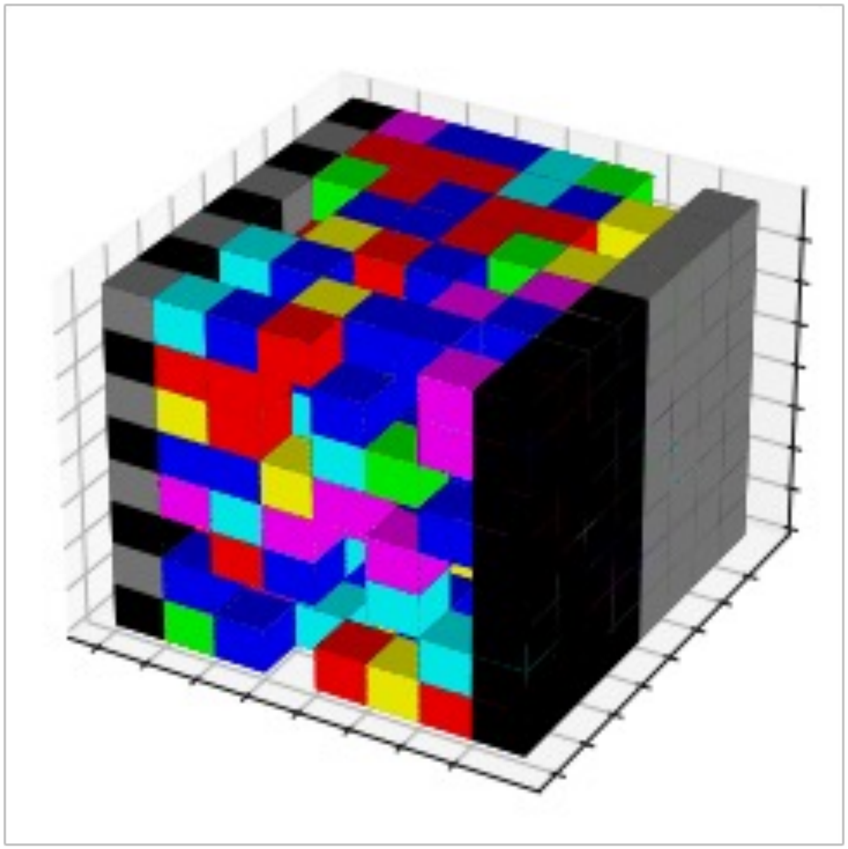
# Task 3: Recognizing surface *arrangement* instead of composition

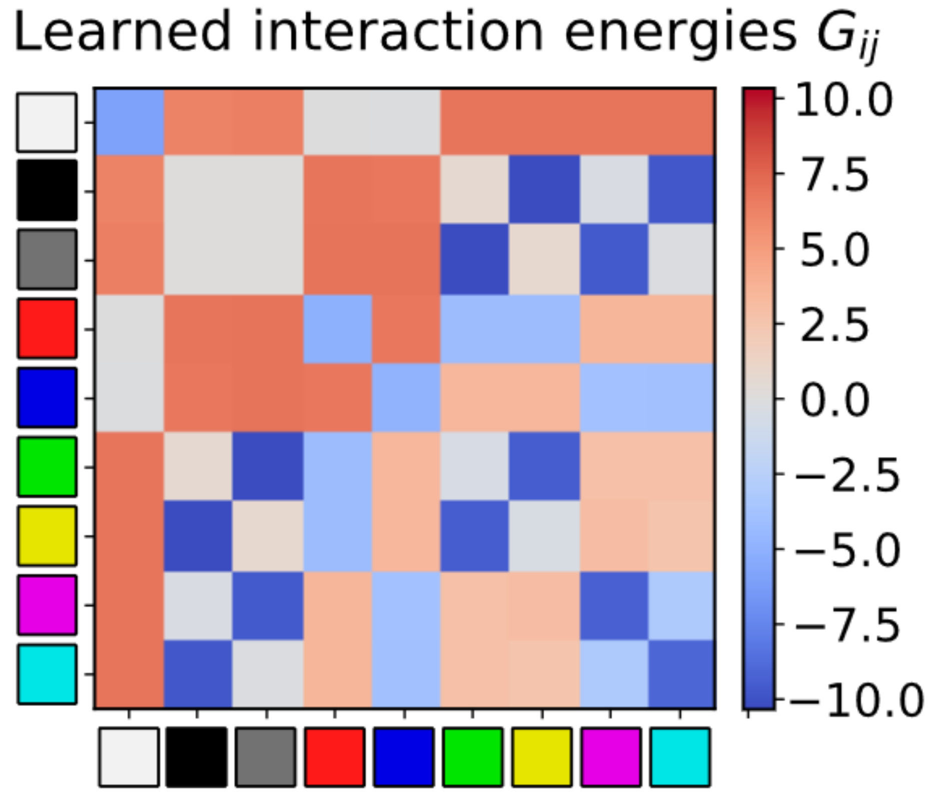


Wake phase of training

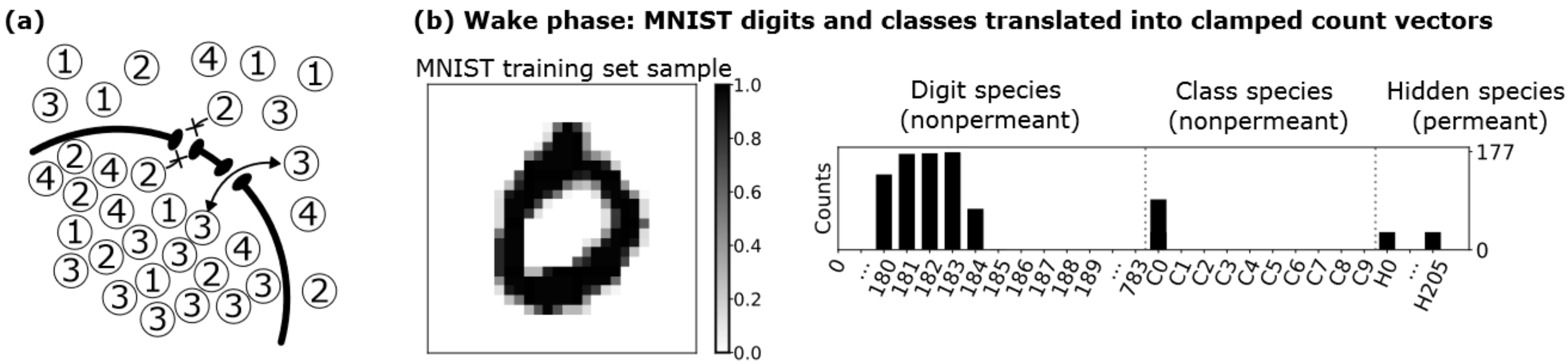


# Molecules recognize surface arrangement after training

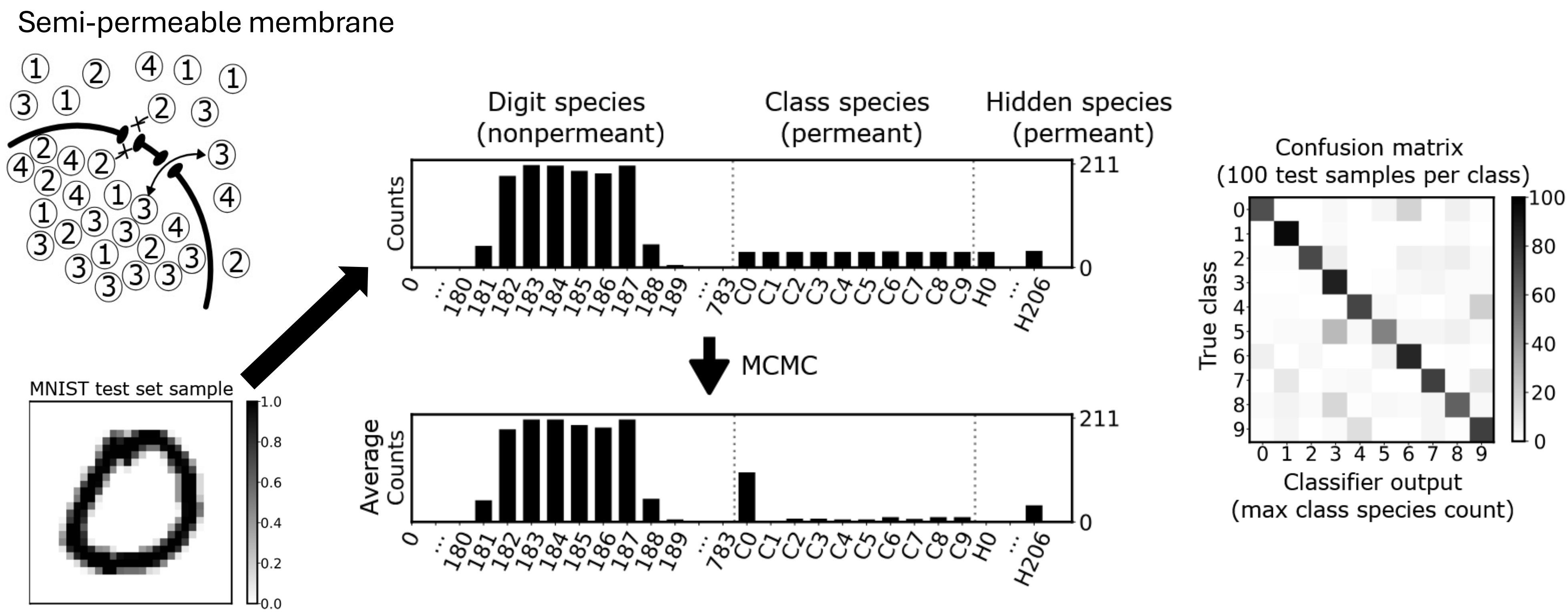




# Task 4: Learning distributions over macroscopic observable: molecular counts

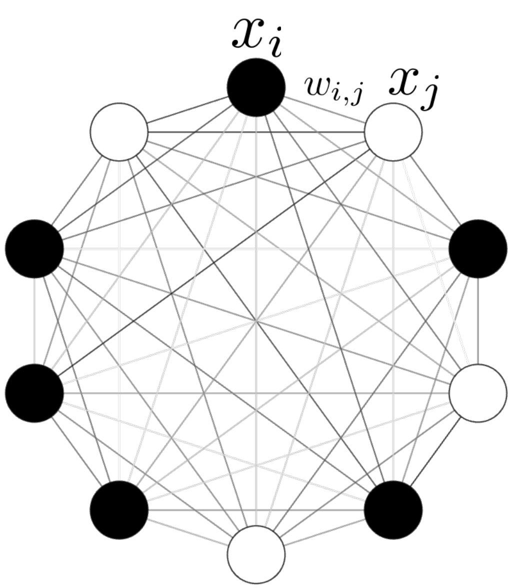


# Semipermeable membranes which recognize handwritten digits



# Biomolecular systems are much like neural networks

(1) Their parameters encode probability distributions
(2) They do inference: compute conditional distributions
(3) Their parameters can be learned
(4) Learned parameters generalize beyond their training set




# The Boltzmann liquid model generalizes

Macroscopic observables for learning and clamping can be *any function* mapping microstates (configurations) to a value

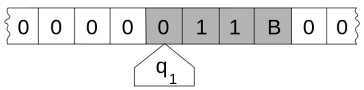
Model supports polymers: can study polymer folding (proteins, genome) and its relation to condensate formation

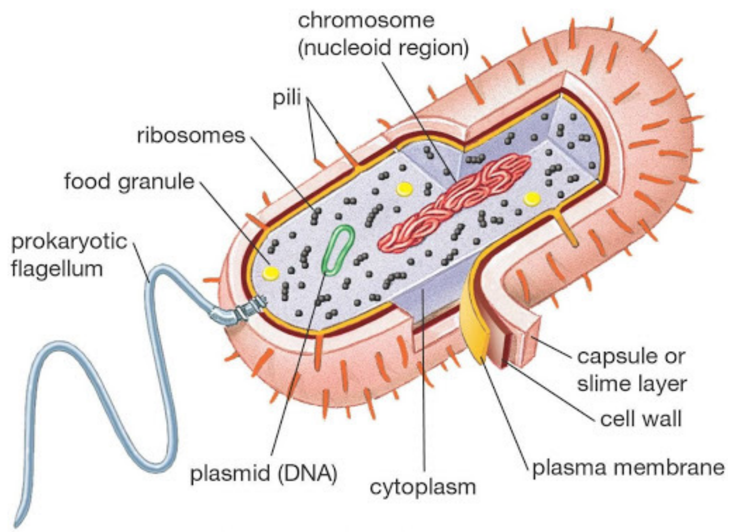
Supports reactions, simulates dilute systems and self-assembly accurately (w.r.t. equilibrium)

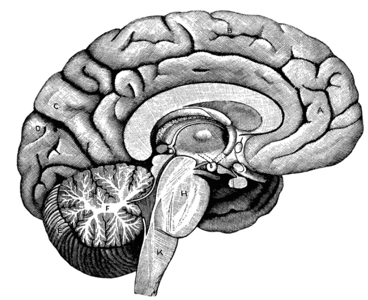
Anisotropic molecules

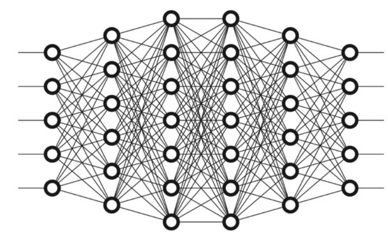
Lattice not important: math is done on general graph

# Everything is code, but what kind of code is it?





What are “natural” models for biomolecular algorithms?

How do we look for them and where do we see them?

Thank you for listening!