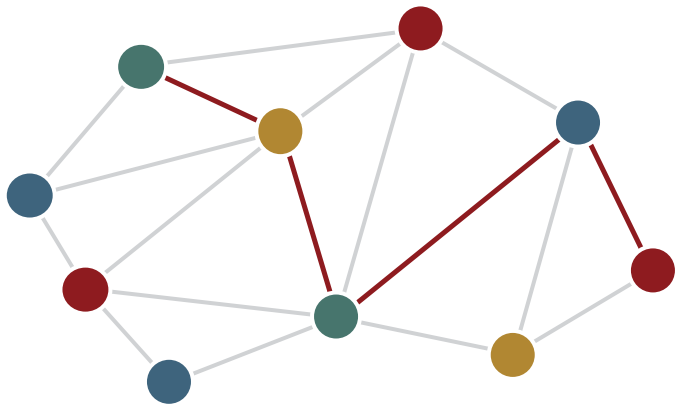
# Computing on sparse graphs and Contextual SBM



September 17

Zhiyang Wang · Electrical & Systems Engineering · WashU

Class 8 · when we can recover from weak community pattern

Sign up for the presentation.
Choose a date that we have classes.
Except for Oct 27, 29, Nov 3,5.
Coordinate for more presentations per class if needed.

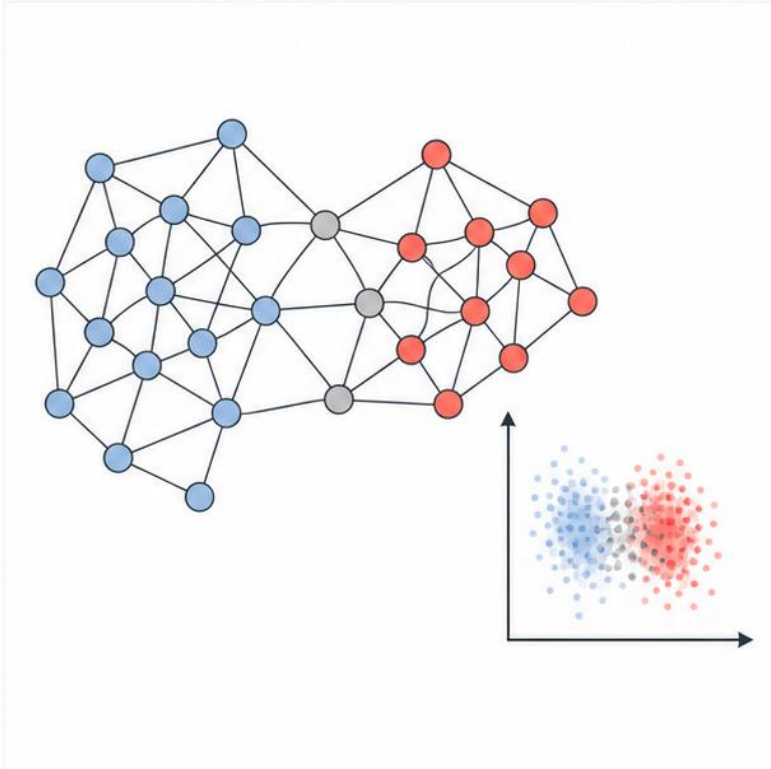
# Computing on sparse graphs and Contextual SBM

01 Separate signal from random edges

02 Add node features when topology is weak

03 Learn the filter with a few labels

04 Exploit sparse matrix structure



**A model can be statistically identifiable yet difficult for a chosen algorithm to recover.**

# Three questions guide us

**01** **WHEN IS RECOVERY POSSIBLE?**

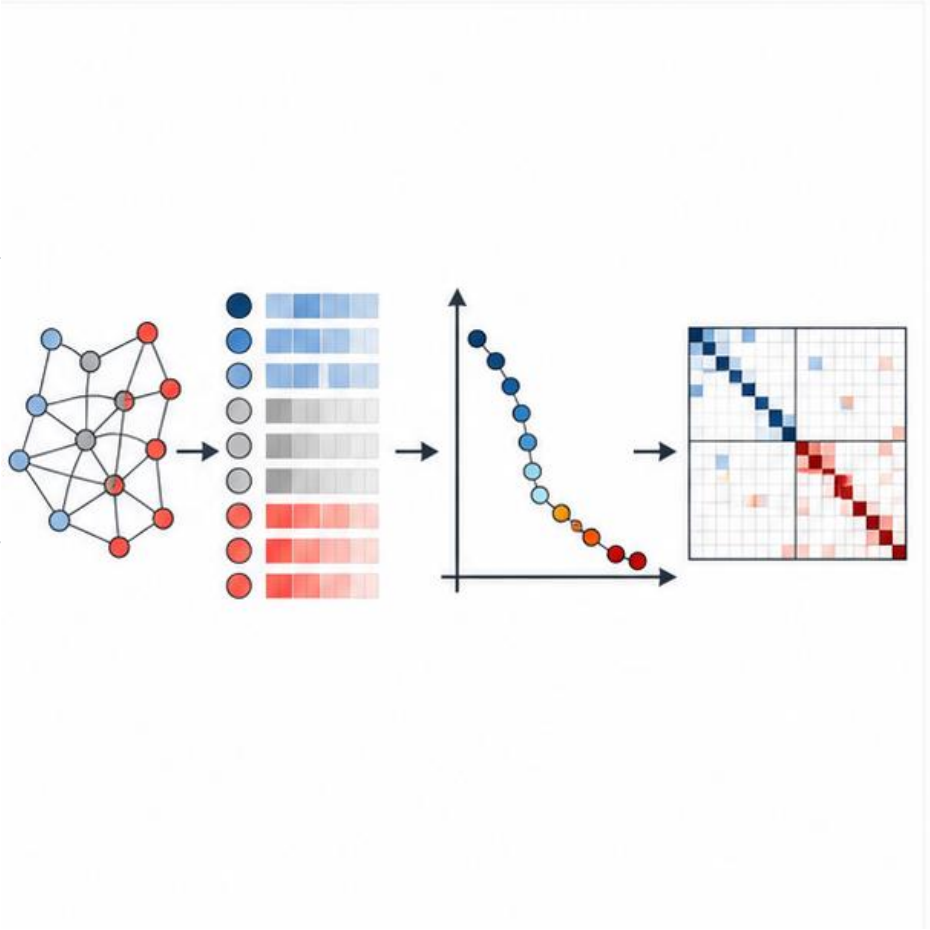
Derive the sparse-SBM threshold separating detectable and impossible regimes.

**02** **HOW DO FEATURES HELP?**

Use a contextual SBM and feature-aware spectral embedding to combine evidence.

**03** **HOW DO WE COMPUTE IT?**

Express graph learning as repeated sparse matrix products in COO or CSR form.



# The balanced SBM sets the baseline

**EDGE PROBABILITIES**

$$y_i \in \{-1, +1\}, \qquad p = \frac{a}{n}, \qquad q = \frac{b}{n}$$

Nodes in the same community connect with probability $p$; different communities connect with $q$.

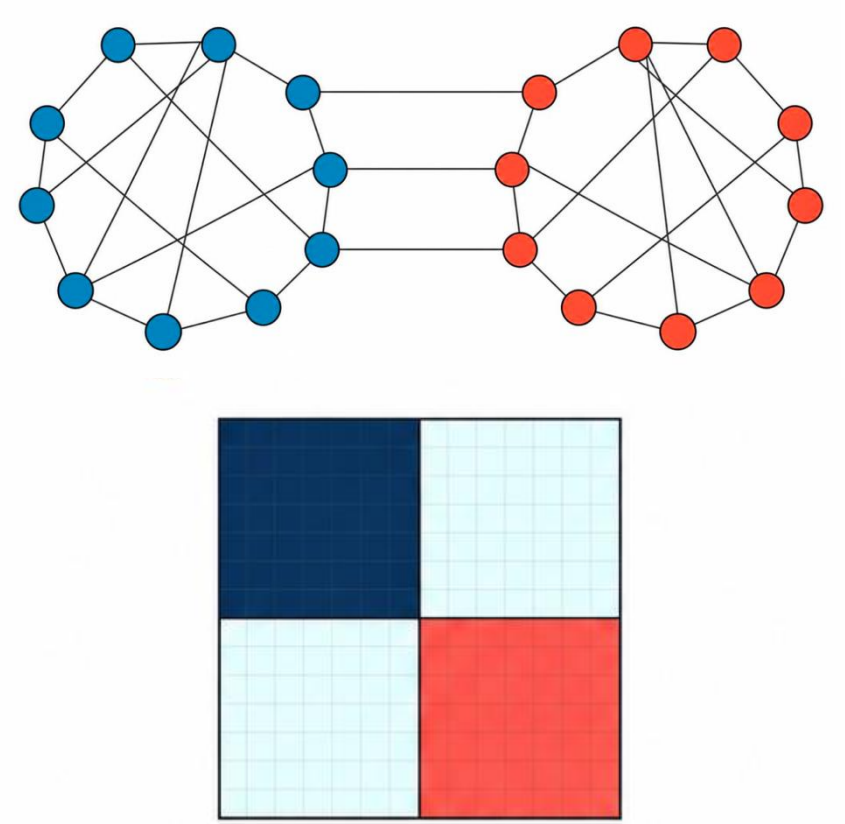
*a/2 ≈ a node's expected neighbors inside its own community,*
*b/2 ≈ a node's expected neighbors across communities.*

*Communities are assortative when a > b.*

**EXPECTED DEGREE**

$$\bar{d} \approx \frac{a+b}{2}$$

**TASK** *Input: one sparse adjacency $\boldsymbol{A}$.*
*Output: one estimated label $\hat{y}_i$ for every node.*



**Sparse scaling keeps local evidence finite even as the graph grows.**

# Recovery has three distinct goals

**1 · DETECTION**

*Decide whether communities exist.*

*Distinguish the SBM from an Erdős–Rényi graph with the same average degree.*

**2 · WEAK RECOVERY**

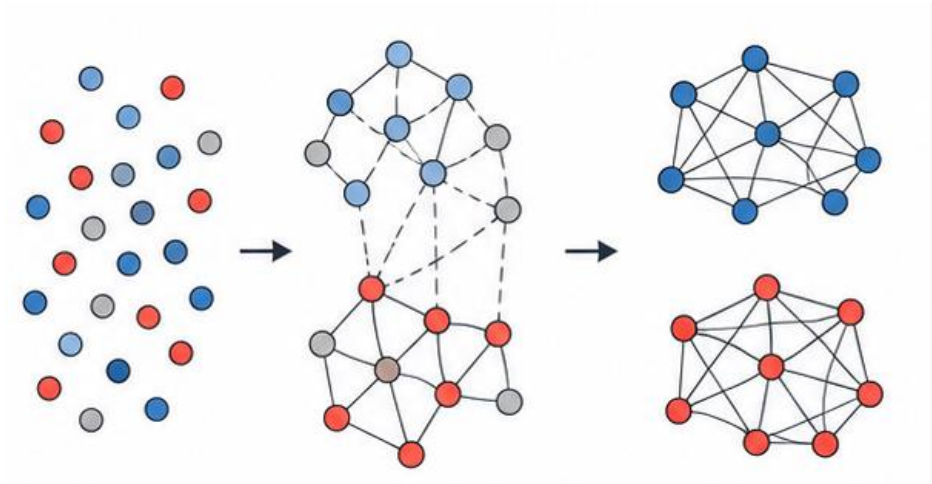
*Label the nodes strictly better than random guessing.*

*Any positive correlation with the true labels $y$ is enough.*

**3 · EXACT RECOVERY**

*Every node's label is correct, with high probability.*

*The strongest goal — needs average degree growing like $\log n$.*



**Below the detection threshold, even weak recovery is information-theoretically impossible.**

# Signal-to-noise predicts recoverability

**COMMUNITY SIGNAL**

Difference in expected connectivity: $(p-q)^2$

*The useful signal: the systematic gap between within- and across-community connection rates.*

**EDGE NOISE**

Random degree fluctuations scale with $2(p+q)$

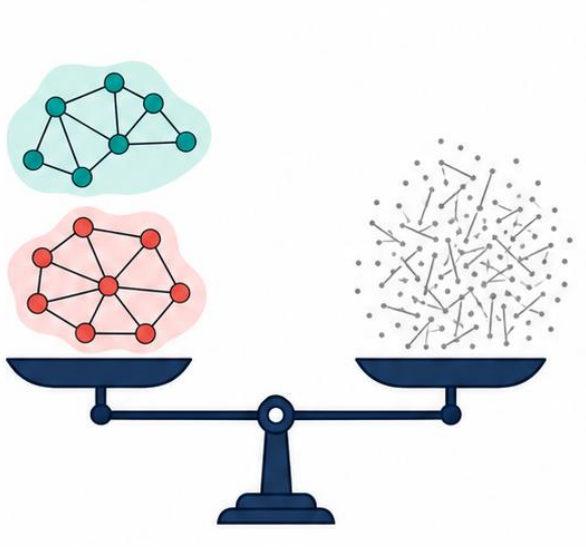
*The noise: sparse edges are random coin flips, so degrees fluctuate with this variance.*

**THE DETECTION TEST**

$$\textbf{SNR} = \frac{(p-q)^2}{2(p+q)} > \frac{1}{n} \iff (a-b)^2 > 2(a+b)$$

Above the threshold → communities are detectable and weak recovery is possible.

Below it → indistinguishable from Erdős–Rényi; no algorithm beats chance.

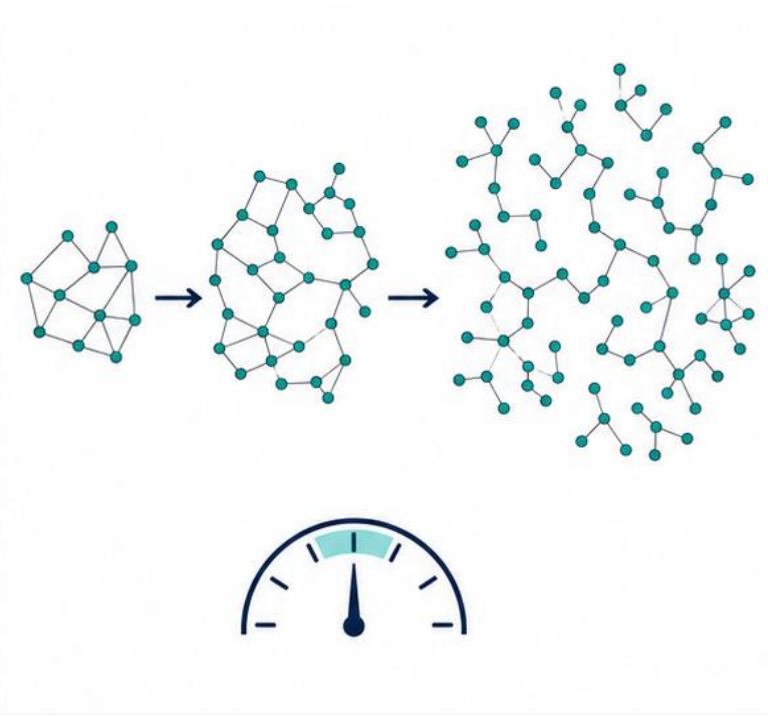


**This is the Kesten–Stigum threshold: recovery is possible only when signal beats noise.**

# Sparse scaling exposes the threshold

Substitute $p = a/n$ and $q = b/n$ into the signal-to-noise ratio.

$$n \cdot SNR = \frac{(a-b)^2}{2(a+b)}$$



**How to read the formula**

- $a/2$ = average edges to your own community
- $b/2$ = average edges to the other community
- Node degrees stay constant as $n$ grows (sparse regime)

Because $n$ cancels out, detectability depends only on $a$ and $b$ — not on the size of the network.

**Weak recovery is impossible when $(a-b)^2 < 2(a+b)$ — the Kesten–Stigum threshold.**

# Algorithms can fail above the information limit

**INFORMATION LIMIT**

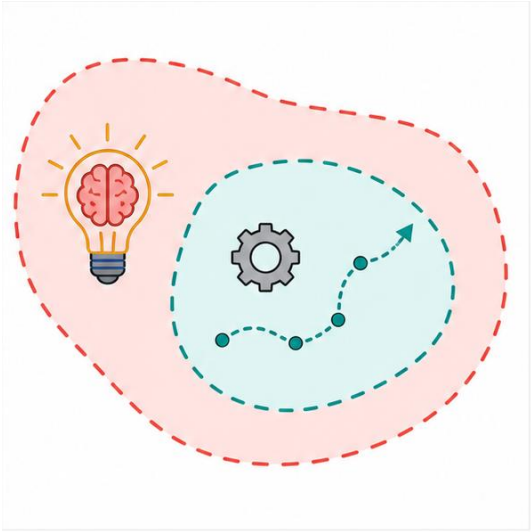
*Below it, no estimator works.*

*Above it, some estimator exists.*

**LIMIT OF CHOSEN METHODS**

*A chosen spectral method may require more separation than the model does.*

**Outer region:** the graph carries enough information — recovery is possible in principle.



**Inner region:** what efficient algorithms (e.g., spectral methods) actually recover.

*The ring in between — recovery is possible in principle, but not efficiently.*

**Algorithm failure above the threshold does not imply the graph contains no information.**

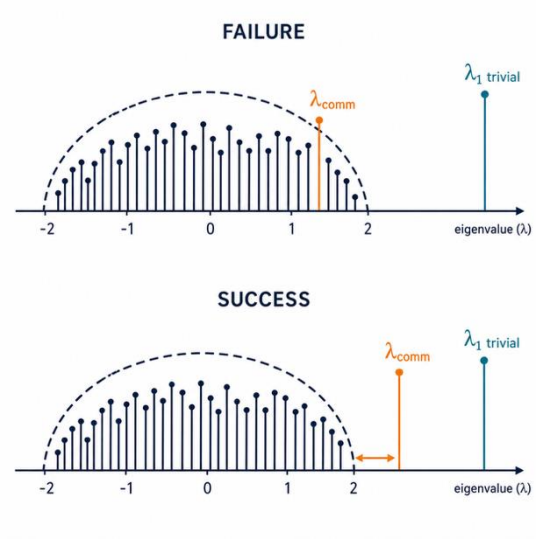
# Detection depends on a nontrivial spectral outlier

**TRIVIAL LEADING MODE**

*The largest eigenvalue $\lambda_1$ follows average degree and the near-constant vector — it does not reveal the partition.*

**INFORMATIVE COMMUNITY MODE**

*Detection succeeds when $\lambda_2$ leaves the noise bulk; its eigenvector can then correlate with the labels.*




**Failure (top): $\lambda_2$ remains inside the bulk. The separated rightmost spike is $\lambda_1$, but that mode is trivial.**

**Success (bottom): $\lambda_2$ separates from the bulk while $\lambda_1$ remains farther right as the trivial leading mode.**

*$u_1$ is approximately the normalized all-ones vector. A useful community eigenvector must correlate with $y$, the ±1 label vector.*

**Ignore $\lambda_1$. Recover communities only when $\lambda_2$ separates from the bulk.**

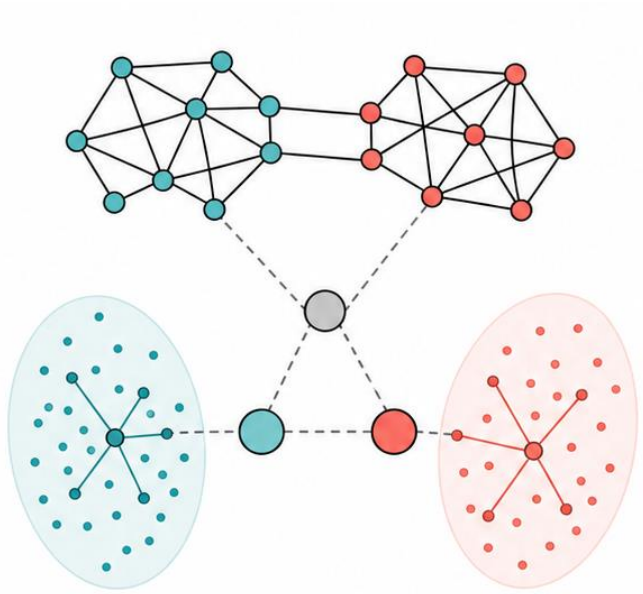
# Contextual SBM adds node features

The same latent label $y_i$ controls both the graph neighborhood and the feature distribution.

$$\mathbf{A} \sim SBM(y; a, b), \qquad \mathbf{X} \mid y \sim \text{Gaussian mixture}$$

**GRAPH EVIDENCE — A**

*Edges form as in the plain SBM*



**FEATURE EVIDENCE — X**

*Each node also draws a feature vector from a Gaussian cloud whose center depends on its community.*

The gray node: its edges are ambiguous, but its features still pick a side.

**Topology and attributes are two noisy measurements of one hidden partition.**

# Features form a noisy Gaussian mixture

Let $u$ be an unknown feature direction shared by all nodes; $z_i$ is isotropic noise in $\mathbb{R}^d$.

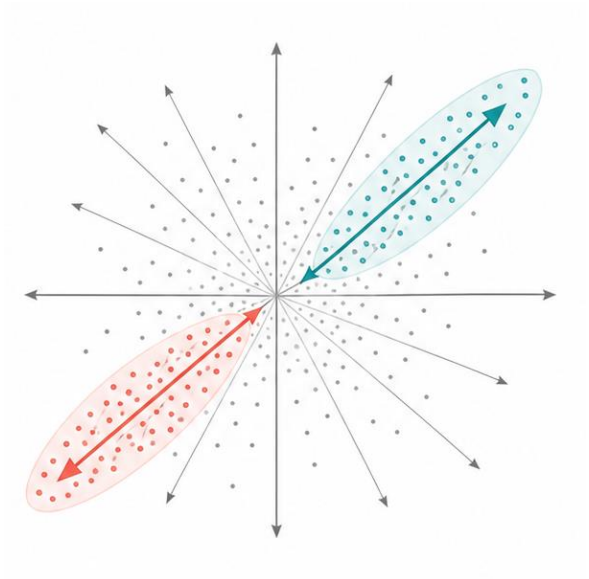
$$x_i = \sqrt{\frac{\mu}{n}} y_i\, u + \frac{z_i}{\sqrt{d}}, \quad u \sim \mathcal{N}(0, I_d/d), .\ z_i \sim \mathcal{N}(0, I_d)$$

**In the formula:**

*$y_i = \pm 1$ picks the community — it flips the mean to $+u$ or $-u$.*

*$\mu$ sets how far apart the two class means sit.*

*The $\sqrt{\mu/n}$ scaling makes features deliberately weak as $n$ grows — the analogue of $p = a/n$ for edges.*



**In the picture:**

*the teal and red clouds are the two communities, centered at $+u$ and $-u$ along the one informative direction.*

*The gray arrows are pure-noise directions: the noise budget is split across all $d$ of them, so more dimensions mean thinner noise along $u$.*

$$\mathbb{E}[x_i \mid y_i, u] = \sqrt{\frac{\mu}{n}}\, y_i u, \qquad \mathrm{Cov}(x_i \mid y_i, u) = \frac{1}{d} I_d$$

Larger $\mu$ separates the class means; larger $d$ adds coordinates but raises the dimension-to-sample ratio $d/n$.

# Graph and feature evidence add

**GRAPH CONTRIBUTION**

$$SNR_A = \frac{(a-b)^2}{2(a+b)}$$

Topology separates connection profiles.

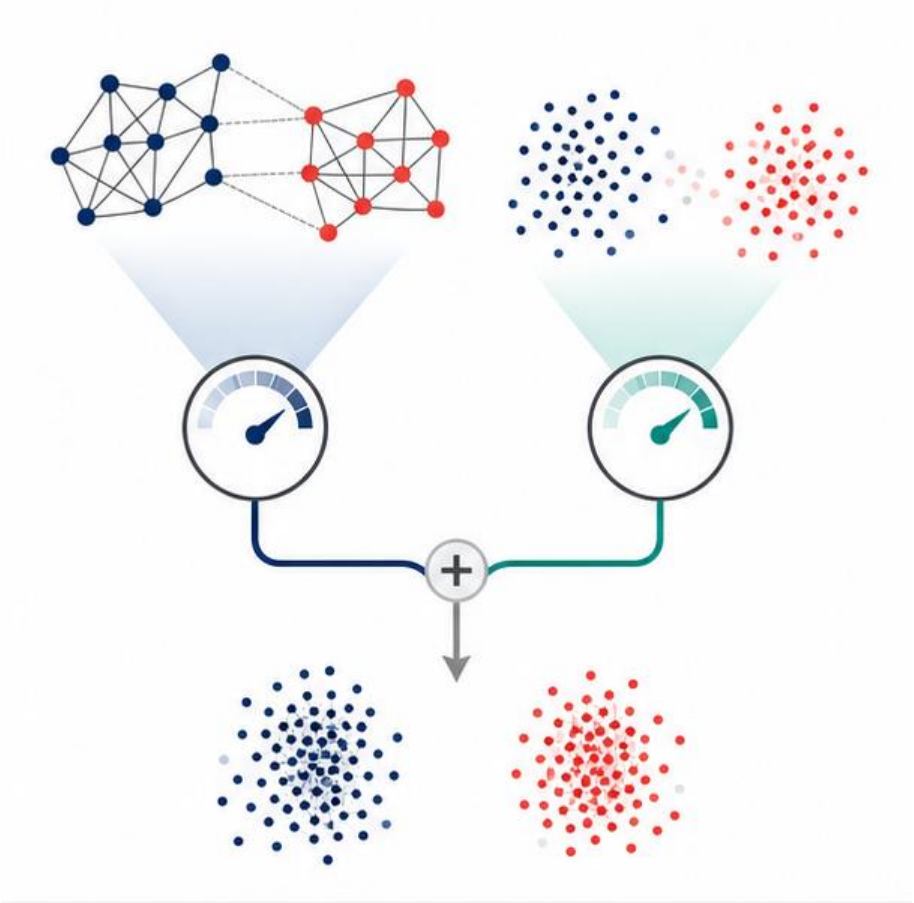
**FEATURE CONTRIBUTION**

$$SNR_X = \frac{\mu^2 d}{n}$$

Attributes separate class means.

**COMBINED CRITERION**

Recovery is possible when $SNR_A + SNR_X > 1$.



**Weak sources can become detectable together because their normalized evidence adds.**

# Four regimes clarify which evidence works

**ONE SOURCE STRONG**

*Topology alone: graph SNR>1.*

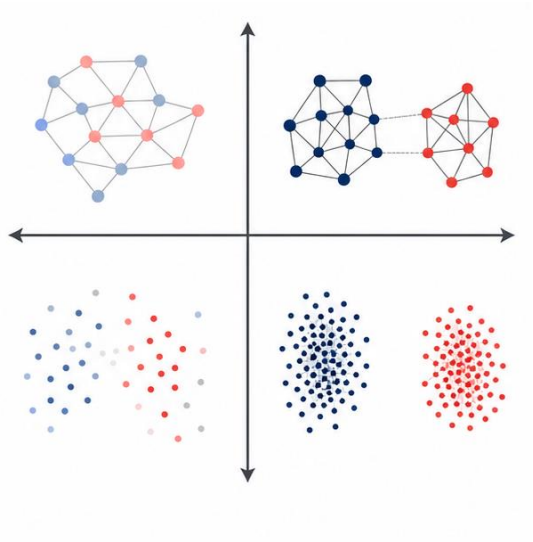
*Features alone: feature SNR>1.*

**COMPLEMENTARY OR IMPOSSIBLE**

*Neither alone, but the sum >1: fuse them.*

*If the sum is ≤1: no recovery.*

**Graph SNR > 1**
— topology alone recovers



**Both > 1**
— either source works

**Both < 1**
— fuse if the sum > 1;
if the sum ≤ 1, no method works

**Feature SNR > 1**
— features alone recover

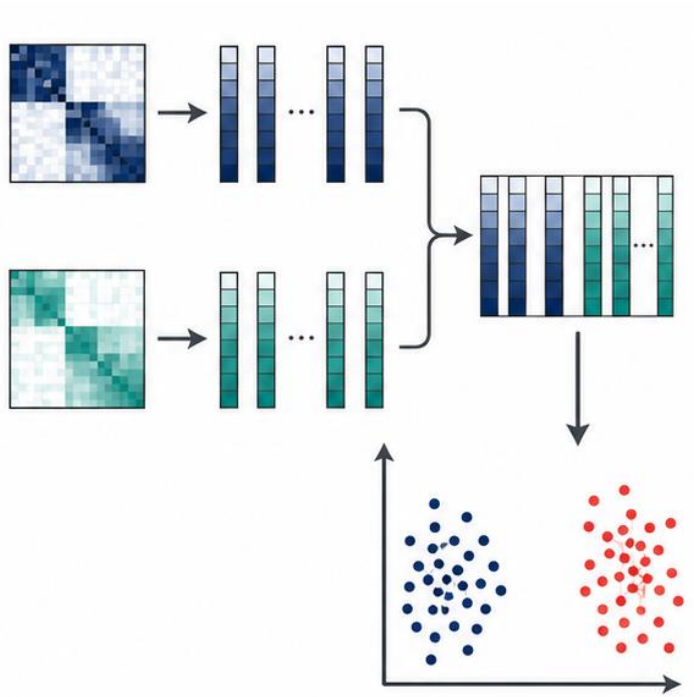
**Model selection should reflect which evidence stream carries the signal.**

# Feature-aware embeddings join two spectra

**Diagonalize the graph and feature Gram matrix**

$$\mathbf{V} = \left[\mathbf{V}_{graph}(\mathbf{A}) \mid \mathbf{V}_{feat}(\mathbf{X}\mathbf{X}^{\top})\right] \in \mathbb{R}^{n\times(C+k)}$$

- Graph eigenvectors encode connection profiles.
- Feature eigenvectors encode shared attribute directions.
- **Choose $C$ graph modes and $k$ feature modes.**
- **Concatenate columns before learning.**



**The representation preserves evidence that either operator could miss alone.**

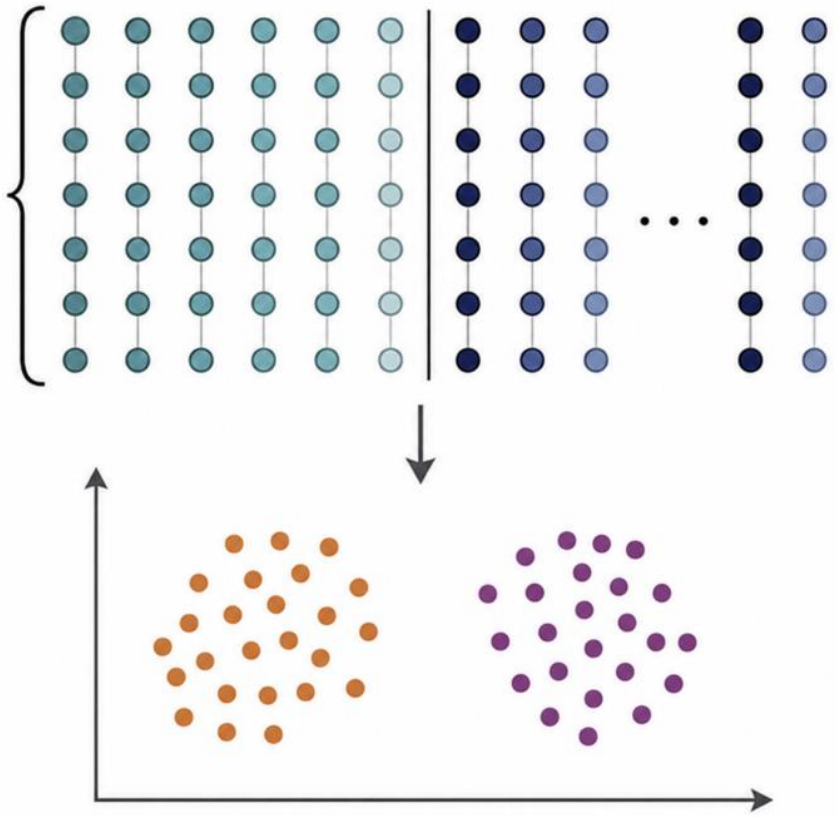
# Concatenation creates node embeddings

**COLUMN VIEW**

$C$ graph modes plus $k$ feature modes form $V$.

**ROW VIEW**

Row $v_i \in \mathbb{R}^{c+k}$ is the joint embedding of node $i$.



**Nodes are close when both topology and attributes support similar latent labels.**

# A linear head learns community names

JOINT EMBEDDING

$$\mathbf{V} = \left[\mathbf{V}_{graph}(\mathbf{A}) \mid \mathbf{V}_{feat}(\mathbf{X}\mathbf{X}^{\top})\right]$$

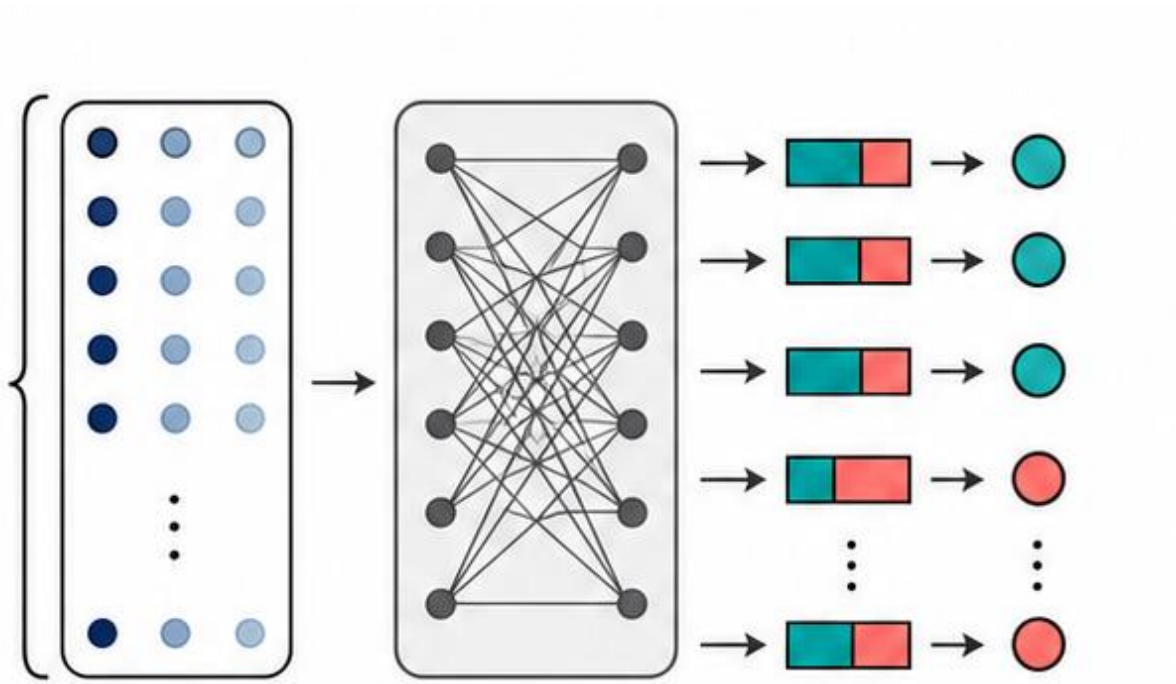
remains fixed during training.

TRAINABLE MAP

$$\mathbf{Z} = softmax(\mathbf{V}\mathbf{W})$$

$W \in \mathbb{R}^{(C+k)\times m}$ maps rows to logits.

*m = number of communities (m = 2 here)*



**Supervision selects a useful combination of graph and feature coordinates.**

# Ten labels classify every node

**01** **INPUT**

*A sparse graph $\boldsymbol{A}$, features $\boldsymbol{X}$, and 10 labeled nodes $J$.*

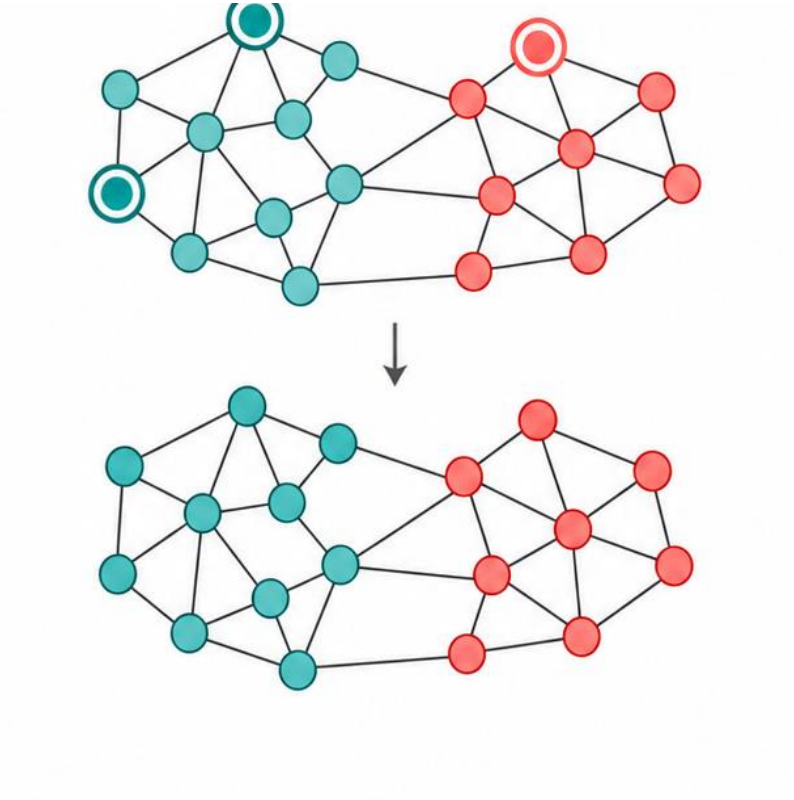
**02** **GRAPH MODEL**

*A contextual SBM or attributed network with two latent groups.*

**03** **OUTPUT + LOSS**

*Class probabilities for every node; cross-entropy only on $J$.*

$$L = -\sum_{i \in J} log Z_{i,y_i}$$



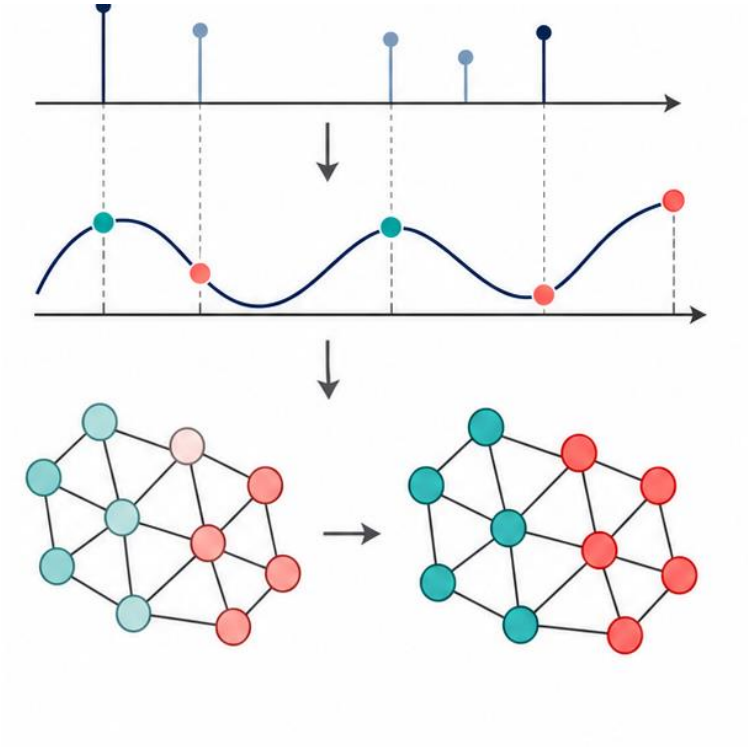
# Polynomial filters can express the labels

If the graph shift has distinct eigenvalues and every spectral component $\hat{x}_i$ is nonzero, a finite polynomial can interpolate any desired response.

$$\mathbf{y} = H(\mathbf{S})\mathbf{x},$$
$$H(\mathbf{S}) = h_0\mathbf{I} + h_1\mathbf{S} + \cdots + h_{K-1}\mathbf{S}^{K-1}$$

$$\hat{y}_i = h(\lambda_i)\hat{x}_i, \qquad h(\lambda) = \sum_{k=0}^{K-1} h_k\,\lambda^k.$$

$$h(\lambda_i) = \frac{\hat{y}_i}{\hat{x}_i}$$



*Each power of S = one round of neighbor averaging — exactly what a GNN layer computes.*

**A graph convolution can approximate the community assignment without explicitly selecting one eigenvector.**

# A linear GNN learns the filter coefficients

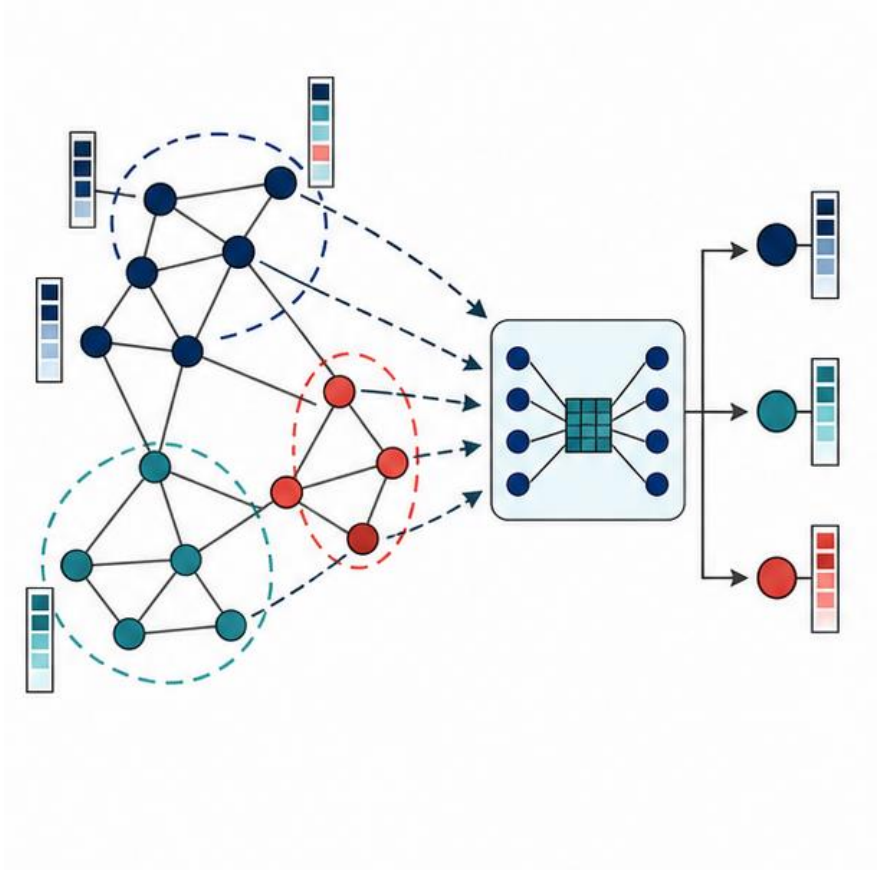
**FILTER VIEW**

$$H(\mathbf{S})\mathbf{x} = h_0\mathbf{x} + h_1\mathbf{S}\mathbf{x} + \cdots + h_{K-1}\mathbf{S}^{K-1}\mathbf{x}$$

*learns a spectral response.*

**MESSAGE-PASSING VIEW**

*Each $\boldsymbol{S}$ multiplication exchanges information across one hop.*



**More filter taps enlarge the neighborhood and the spectral degrees of freedom.**

# Cross-entropy trains on labeled nodes

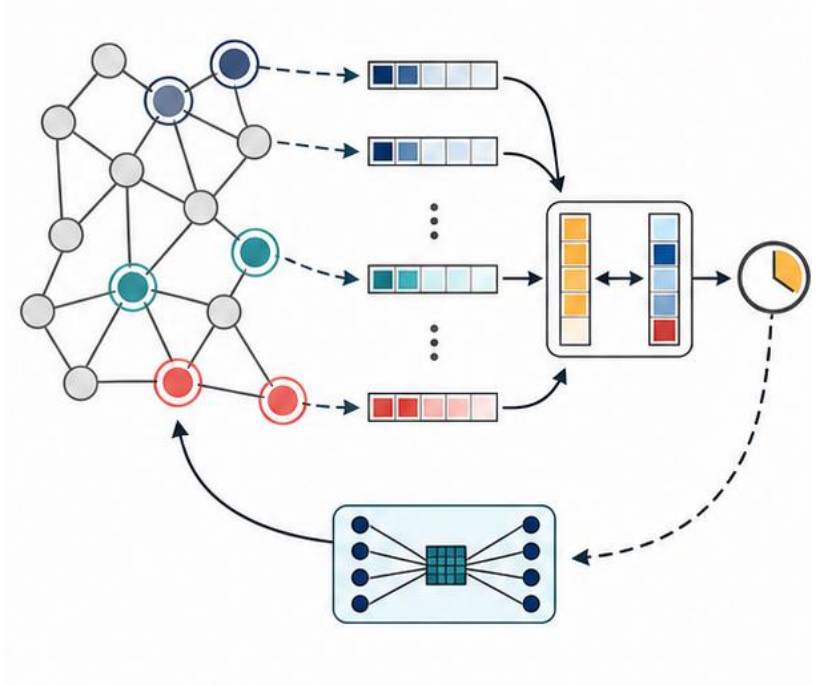
**PREDICTIONS**

$Z = \Phi(x; S, H)$ *provides one probability vector per node.*

**SUPERVISED OBJECTIVE**

$$L(\theta) = -\sum_{i \in J} CE(y_i, Z_i) = \sum_{i \in J} \log(Z_i[y_i])$$

*Cross-Entropy loss. Unlabeled nodes still affect messages.*



**WHY THIS LOSS?**

- Maximum likelihood — it makes the observed labels most probable.
- $-log$ punishes confident mistakes hard.
- Convex for the linear head — a single global optimum.
- Gradient = prediction − truth: simple and stable.

**One shared graph model transfers supervision through structure and features.**

# Every GNN layer is a sparse matrix product

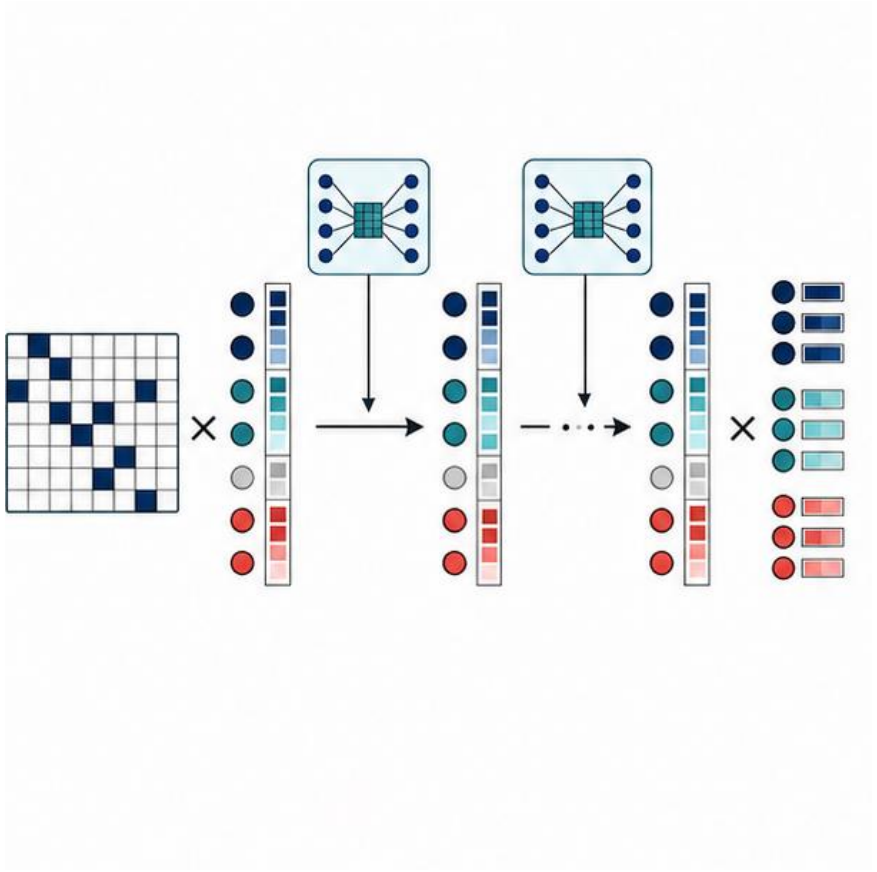
**PROPAGATION**

$\boldsymbol{Y} = \boldsymbol{SX}$ *aggregates neighboring features.*

$\boldsymbol{S}$ *has the graph sparsity pattern.*

**FEATURE MIXING**

$\boldsymbol{Z} = \sigma(\boldsymbol{YW})$ *mixes channels after aggregation.*



**Efficient message passing depends on multiplying only the nonzero entries of S.**

# Sparse storage cuts the per-layer cost

**DENSE MATRIX**

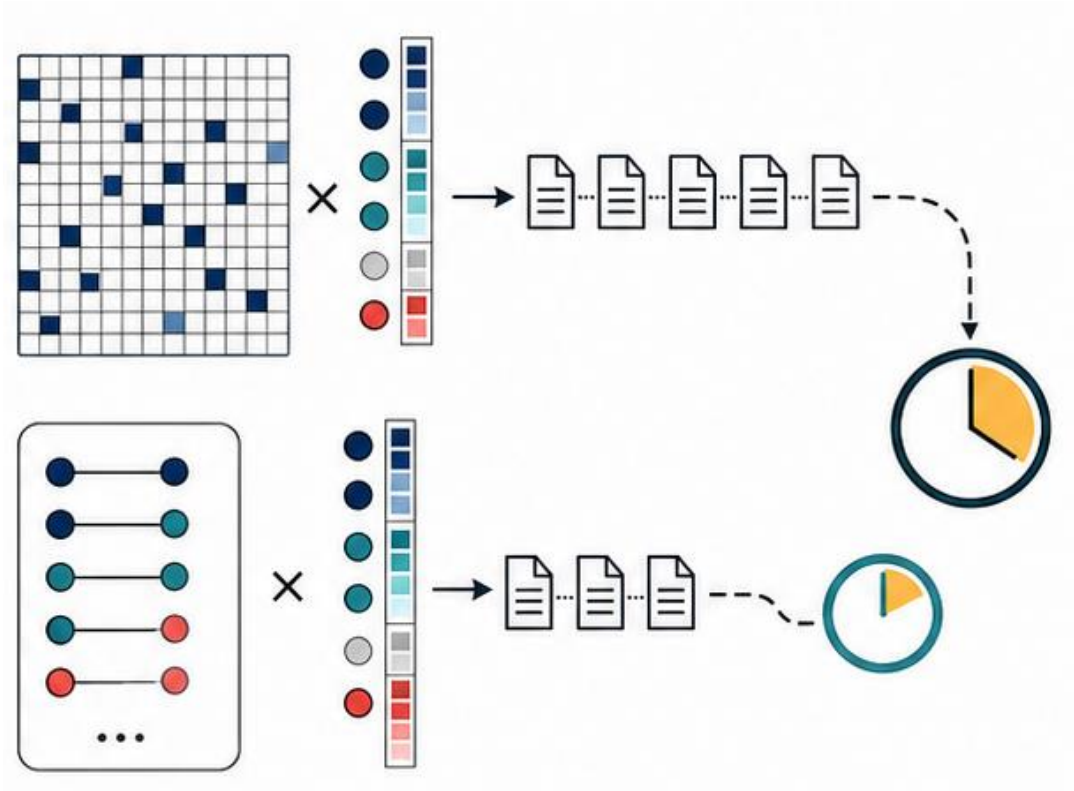
*Store $n^2$ entries.*

*Compute matrix shift $\boldsymbol{SX}$ in $O(n^2F)$.*

**SPARSE GRAPH**

*Store $|\mathcal{E}|$ non-zero entries.*

*Compute matrix shift $\boldsymbol{SX}$ in $O(|\mathcal{E}| \cdot F)$.*



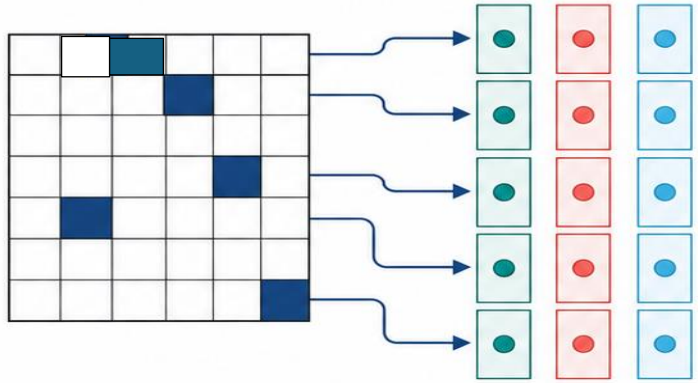
**For bounded-degree graphs, sparse propagation scales approximately linearly with $n$.**

# COO stores one Coordinate per nonzero

**THREE ARRAYS**

*row[k], col[k], value[k]*

*for* $k = 0, \ldots, |\mathcal{E}| - 1.$



**EXAMPLE**

Graph: 0 — 1 — 2

```
S = 0 1 0
    1 0 2
    0 2 0
```

```
(row, col, val):
(0, 1, 1)
(1, 0, 1)
(1, 2, 2)
(2, 1, 2)
```

*4 triples — each edge appears twice.*

**ENGINEERING TRADEOFF**

*Easy to construct and concatenate.*

*Duplicate coordinates may need coalescing.*

**WHAT IS COO — AND WHY USE IT?**

- "Coordinate" format: one triple (row, col, value) per nonzero entry.
- For S this is literally the edge list — two endpoints plus a weight.
- SH becomes one pass over the list — only real edges do any work.

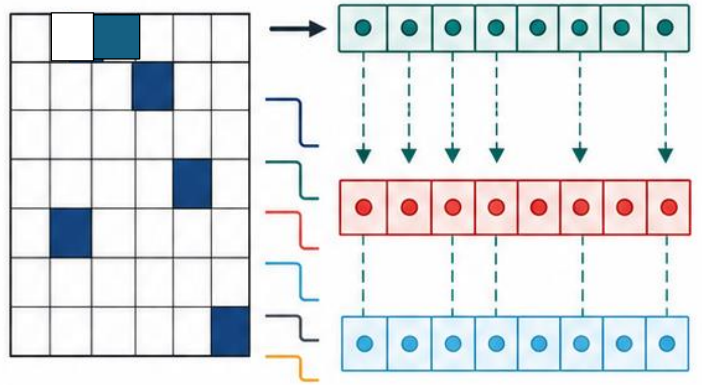
**COO is an edge list with an optional value attached to every edge.**

# CSR compresses row boundaries

Column indices and values are stored consecutively by row.

$crow[i+1] - crow[i] = |\mathcal{E}|\ in\ row\ i$

- crow has length $n+1$; col and value each have length $|\mathcal{E}|$. Row $i$ occupies positions crow$[i]$ through crow$[i+1]-1$.



**EXAMPLE**

Same graph 0 — 1 — 2, four nonzeros:

```
crow = [0, 1, 3, 4]
col  = [1, 0, 2, 1]
val  = [1, 1, 2, 2]
```

*Row 1 sits at positions crow[1] … crow[2]−1 = 1 … 2.*

```
S = 0 1 0
    1 0 2
    0 2 0
```

CSR supports efficient row traversal without storing a row index per edge.

# CSR computes one row locally

**01** **LOCATE**

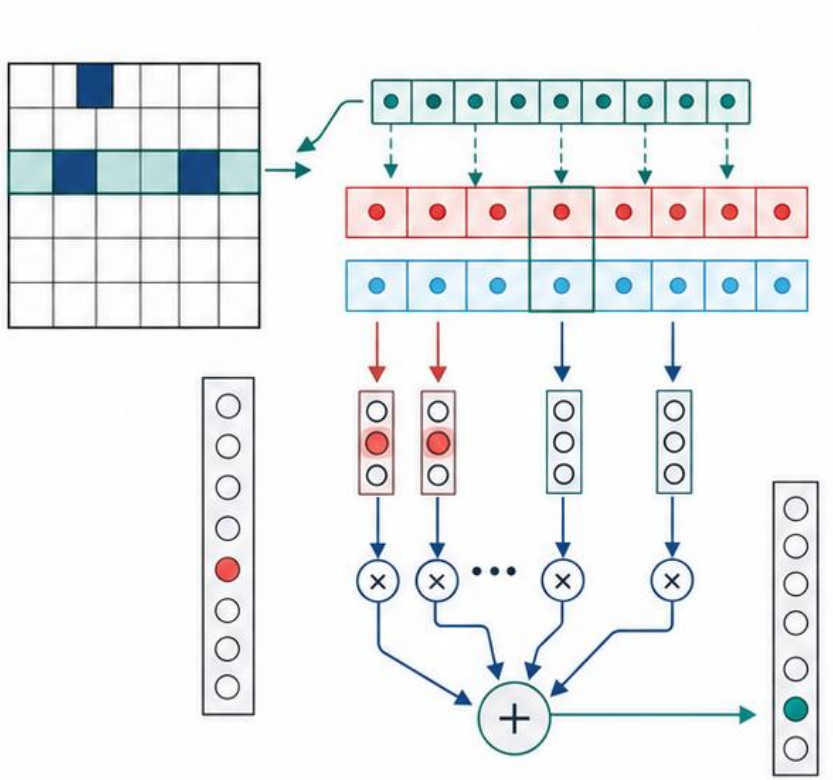
Read start=crow[$i$] and stop=crow[$i+1$].

**02** **GATHER**

For $k \in [\text{start}, \text{stop})$, in that segment, fetch $x[col[k]]$.

**03** **ACCUMULATE**

Set $y[i] = \Sigma\, value[k] \cdot x[col[k]]$.



**EXAMPLE**

Same graph 0 — 1 — 2, four nonzeros:

```
crow = [0, 1, 3, 4]
col  = [1, 0, 2, 1]
val  = [1, 1, 2, 2]
```

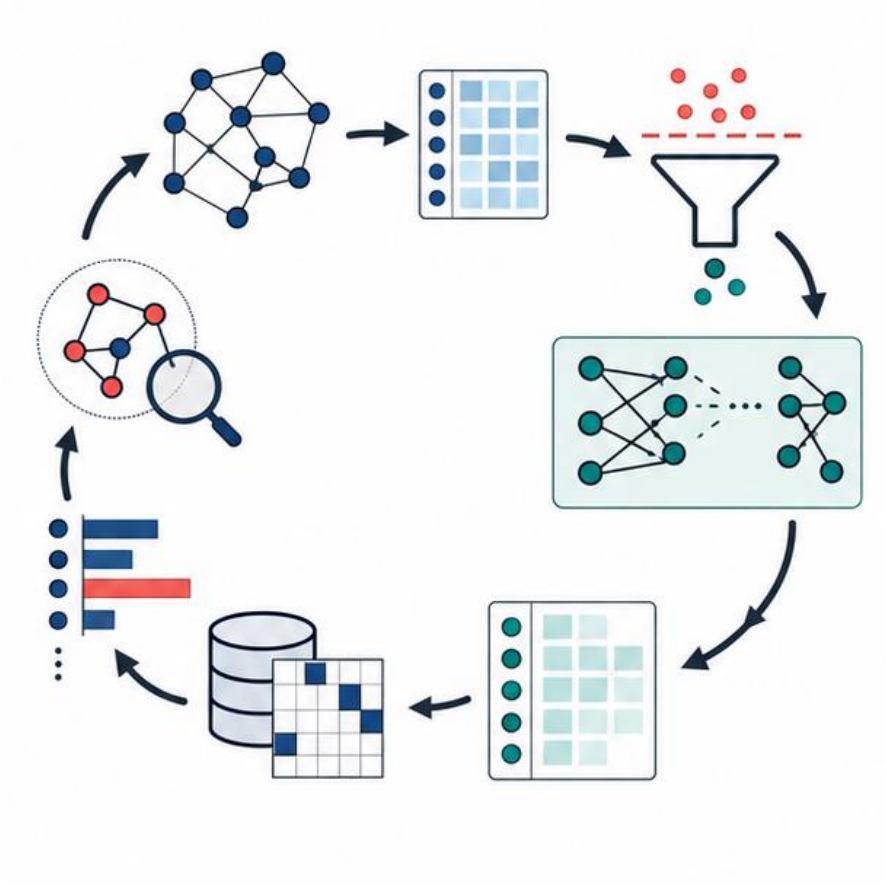
```
S = 0 1 0
    1 0 2
    0 2 0
```

$x = (10,20,30)^{\mathsf{T}}$,

*Row 1 sits at positions crow[1] … crow[2]−1 = 1 … 2.*

TAKEAWAYS

# Combine signal, learning, and sparse computation.



01 Thresholds separate possible from impossible

02 Features add evidence to weak topology

03 Graph filters turn embeddings into predictors

04 COO and CSR make propagation scalable

**A successful graph-learning system needs enough signal, an operator that exposes it, and an implementation that visits only the edges.**