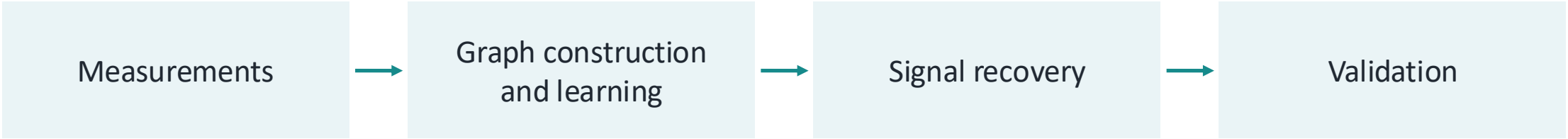
Lecture 20 Graph learning and signal recovery

Learning relationships from measurements

Recovering values with those relationships



One framework for geometry, smoothness, and reliable estimation

Lecture structure and recording schedule

Section 1

1 Measurements and geometric graphs

Profiles, scaling, neighbors, and kernel weights

Section 2

2 Learning Laplacian weights

Energy, graph scale, degree regularization, and a worked solution

Section 3

3 Recovering graph signals

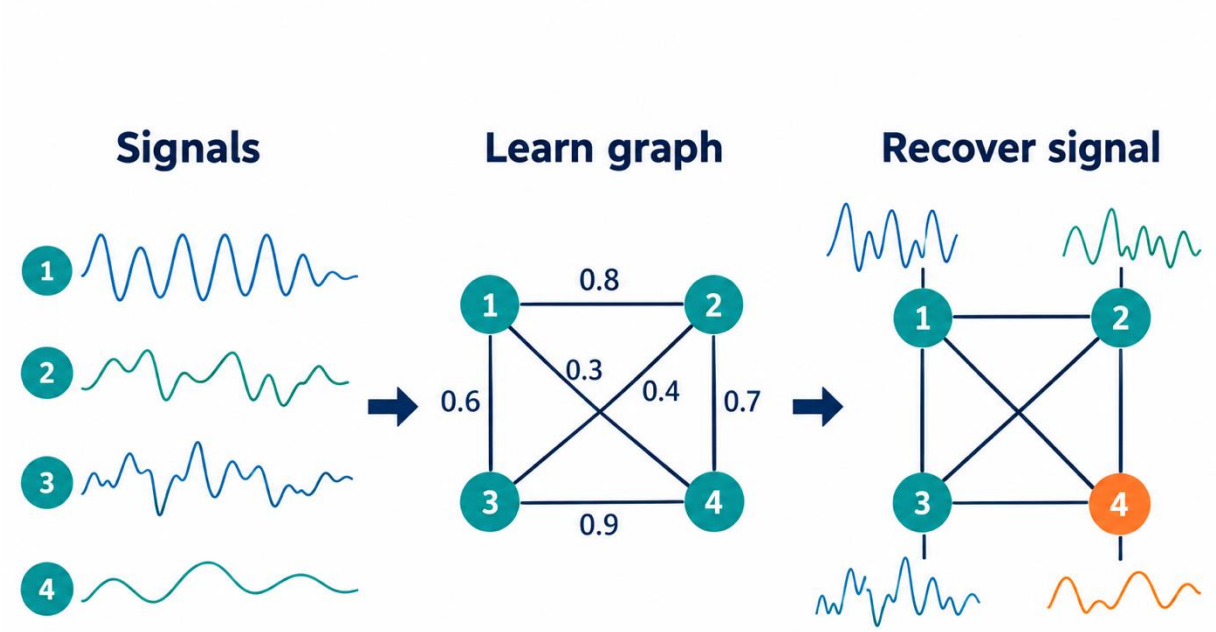
Quadratic estimation, harmonic extension, and label propagation

Section 4

4 Robustness and joint estimation

Robust penalties, recent results, and current research directions

Construction, learning, and recovery



Learning the graph

Known measurements help us estimate useful connections.

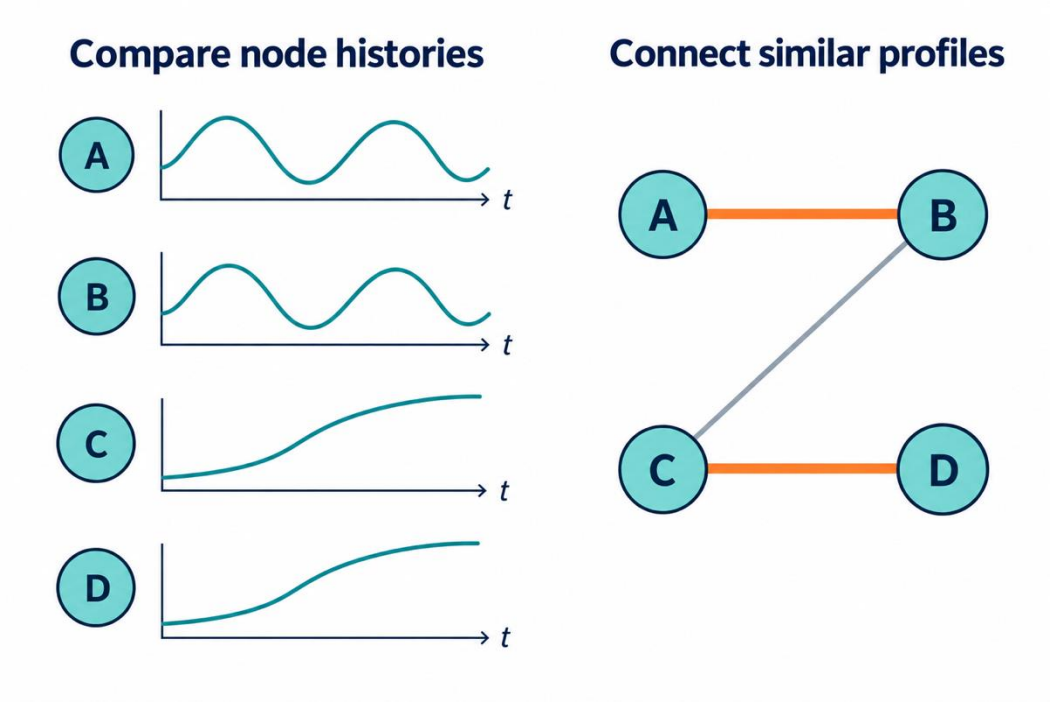
Recovering a signal

The connections then help estimate missing node values.

Always identify which quantity is known and which is unknown

Rows identify nodes, columns identify measurements

$$X \in \mathbb{R}^{n \times M}, \qquad X_{i:} = r_i^T$$



Read a column

One signal across all n nodes, such as temperatures at one time.

Read a row

The M measurements for one node. Compare these profiles to choose edges.

The same matrix shape can also store document embeddings or numeric attributes

A four-node running example

Node	Measurement 1	Measurement 2
A	0	0
B	0	1
C	3	3
D	3	4

Before calculating

A and B look similar.
C and D look similar.

Graph-construction choices

The neighbor rule and weights determine connections within and between groups.

These four profiles will make distances and learned weights concrete

Feature scaling defines graph similarity

$$d_{ij}^2 = \sum_{\ell} \frac{(r_{i\ell} - r_{j\ell})^2}{s_\ell^2}$$

$s_\ell > 0$ Larger scale → less influence on distance

Comparable units

Prevent numerical units alone from dominating distance.

Meaningful importance

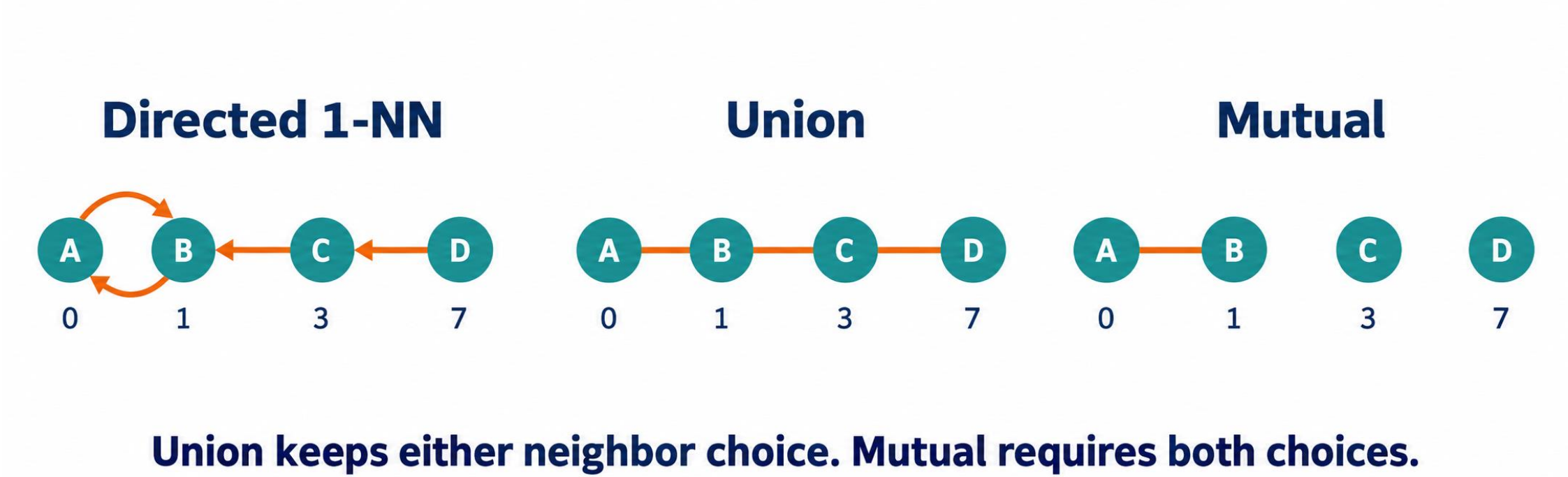
Choose scales that reflect which differences matter.

Consistent use

Use the same transform for all nodes.
Fit scales on training data.

Scaling changes distances, which determine neighbors and edge weights.

Directed, union, and mutual nearest neighbors



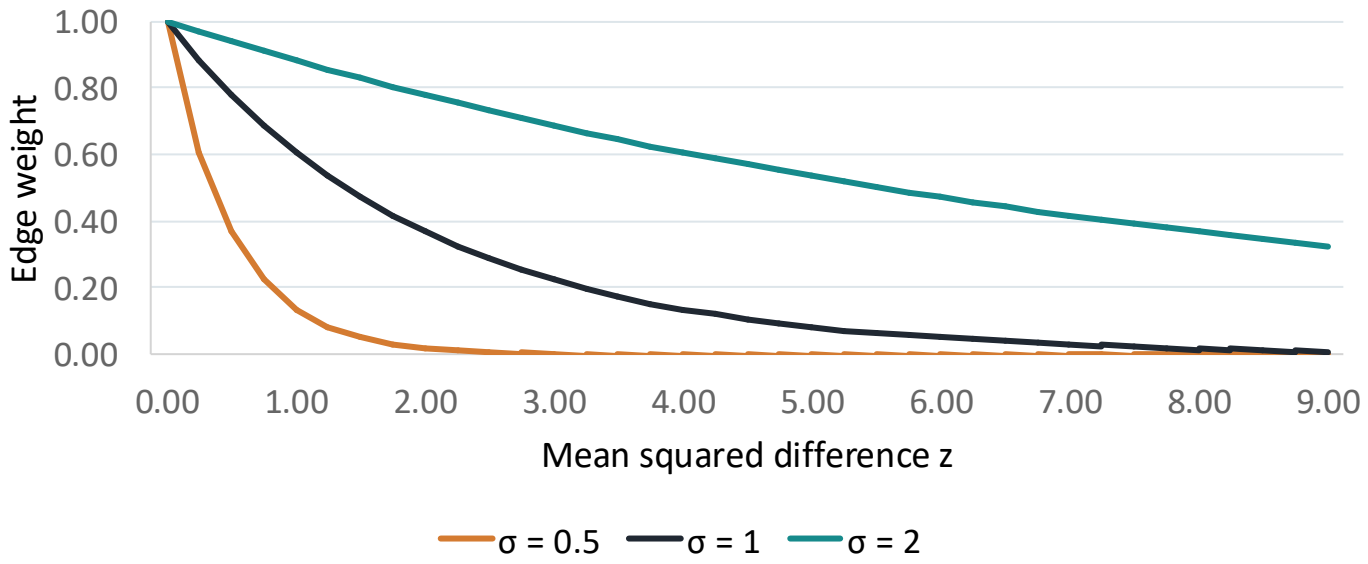
Use the printed coordinates to compute distance. Node spacing is schematic.

Union keeps one-sided choices. Mutual requires both nodes to choose each other.

Gaussian weights use a defined profile distance

$$w_{ij} = \exp\left(-\frac{z_{ij}}{2\sigma^2}\right)$$

$$z_{ij} = \frac{1}{M}\|r_i - r_j\|^2$$



Average squared profile difference

r_i: measurements at node i
M: number of measurements
Small z_{ij} means similar profiles.

σ controls the decay

Small σ: faster decay.
Large σ: slower decay.
Excluded pairs keep weight zero.

Gaussian construction sets weights from z_{ij}; graph learning will use z_{ij} as a cost.

Same profile distances, learned edge weights

Average squared profile differences become fixed costs for learning edge weights.

$$z_{ij} = \frac{1}{M}\sum_{m=1}^{M}(X_{im} - X_{jm})^2$$

$$\frac{1}{M}\operatorname{tr}(X^{\mathrm{T}}LX) = \sum_{i<j} w_{ij}z_{ij}$$

Given: observed profiles and distances

Learn: the graph Laplacian L

Compute z_{ij} from the M columns of X.
Keep these distances fixed.

Optimize the edge weights w_{ij}.
Use graph constraints and regularization.

Exact conversion: average over measurements, then group terms by edge.

The combinatorial graph Laplacian

$$L = D - W, \qquad D = \mathrm{diag}(W1)$$

W describes connections

Symmetric, nonnegative weights.
The diagonal is zero.
D stores the sum of weights incident to each node.

L measures disagreement

Diagonal entries are degrees.
Off-diagonal entries are negative edge weights.
Each row sums to zero.

A constant signal has no edge differences, so L sends it to zero

The energy becomes a cost on every edge

$$x^T L x = \sum_{i<j} w_{ij}\left(x_i - x_j\right)^2$$

One signal

Add the weighted squared difference across each edge once.

$$\frac{1}{M}\operatorname{tr}(X^{\mathrm{T}} L X) = \sum_{i<j} w_{ij} z_{ij}$$

Many signals

Average over columns. z_{ij} is the mean squared difference between node profiles.

Why this helps learning

The observed z values become costs. Large weights are cheaper between nodes with similar histories.

The $i < j$ sum counts unordered edges, so no factor of one half is needed

Why smoothness alone learns an empty graph

Every edge cost is nonnegative. Reducing any weight can only reduce the smoothness cost.

$$w = 0 \quad \Rightarrow \quad L = 0 \quad \Rightarrow \quad \operatorname{tr}(X^{\mathrm{T}} L X) = 0$$

The trivial solution

With no edges, there is no disagreement to penalize. Every signal has zero energy.

The missing requirement

Fix graph scale or penalize vanishing degrees. Otherwise the objective rewards shrinking the graph away.

A low training energy does not by itself show that the graph is useful

Classical graph learning: regularized smoothness

W is the symmetric weight matrix; w stores each undirected candidate edge once.

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} \\ w_{12} & 0 & w_{23} \\ w_{13} & w_{23} & 0 \end{bmatrix} \quad \leftrightarrow \quad w = (w_{12}, w_{13}, w_{23})^T$$

$$\min_{w \geq 0} z^T w - \alpha \sum_i \log d_i + \frac{\beta}{2} \|w\|^2$$

Profile agreement

$z^{\mathrm{T}}w$: strong edges cost less when profiles are similar.

Positive degrees

$-\alpha \Sigma log d_{\mathrm{i}}$ prevents zero degrees. d_{i} is the sum of incident weights.

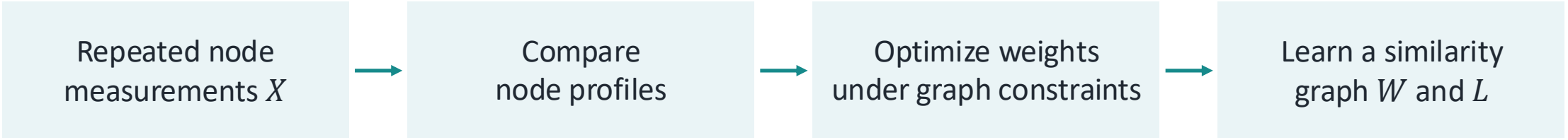
Controlled magnitude

$\beta \|w\|^2/2$ limits large edge weights.

$\alpha, \beta > 0$. Positive degrees prevent isolated nodes; multiple components may remain.

Kalofolias. How to Learn a Graph from Smooth Signals. AISTATS, 2016.

When to learn a graph from signals



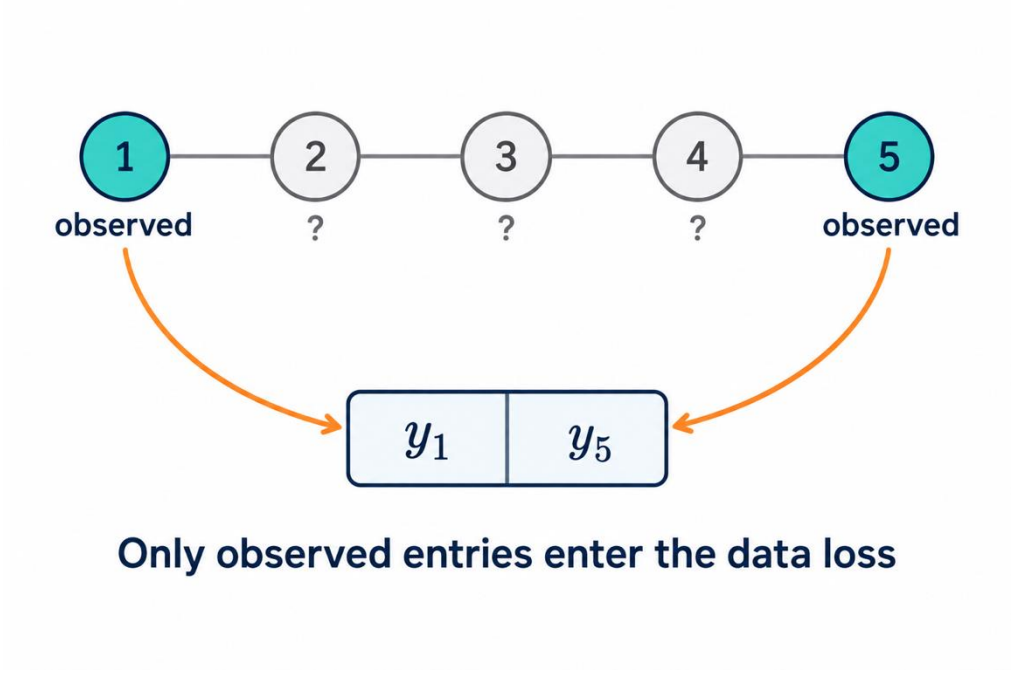
Use when	Relationships are unknown, and repeated signals are plausibly smooth on one graph.
Typical task	Learn connections among sensors from their historical measurement profiles.
Interpretation	Edges represent modeled similarity. Assess them through held-out recovery or graph evidence.

A transparent optimization baseline and a building block for joint graph–signal estimation.

Kalofolias. How to Learn a Graph from Smooth Signals. AISTATS, 2016.

Only observed nodes enter the data loss

$$y = Px + \eta, \qquad P \in \mathbb{R}^{m \times n}, \qquad y \in \mathbb{R}^{m}$$



Unknown and observed

x has n node values.
P selects the m measured entries.
y stores those observations.

Noise and graph

η is measurement noise.
The fixed Laplacian L supplies information about neighboring values.

P selects observed nodes; m denotes the number of signal observations.

Shuman et al. (2013)

Classical recovery: Laplacian-regularized least squares

Known graph L + noisy or partial observations y → estimate the full signal x.

$$\hat{x} = \operatorname{argmin}_x \frac{1}{2}\|Px - y\|^2 + \frac{\mu}{2} x^T L x$$

Data fit

$Px - y$ compares predictions with the values observed.

Graph energy

$x^{\mathrm{T}} L x$ penalizes differences across strongly weighted edges.

When it fits

Continuous signals with smooth graph variation. Squared loss suits approximately Gaussian noise.

Use for denoising or filling missing readings on a meaningful similarity graph.

Solve the same estimator through its normal equations

The linear system computes the preceding objective's minimizer; it is not a separate model.

$$P^T(P\hat{x} - y) + \mu L\hat{x} = 0$$

Differentiate

The first term pulls toward measurements.
The second pulls toward neighboring values.

$$(P^T P + \mu L)\hat{x} = P^T y$$

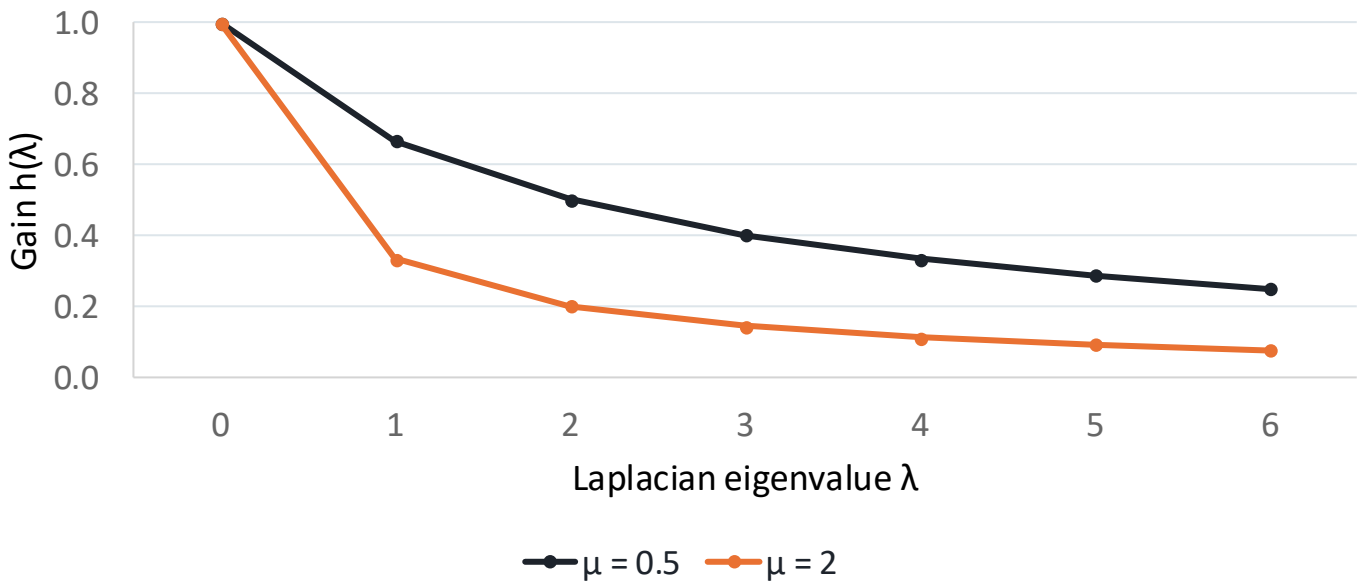
Practical solution

Solve the sparse system. μ balances observation fit and graph smoothness.

With $\mu > 0$ and node sampling, each connected component needs an observation for uniqueness.

Full observations give a low-pass graph filter

$$\hat{x} = (I + \mu L)^{-1} y, \qquad h(\lambda) = \frac{1}{1 + \mu\lambda}$$



Read the curves

High-frequency components have larger λ and are attenuated more. At $\lambda = 0$, the gain is one.

Assumption

Here $P = I$. Partial observations generally do not give a scalar spectral filter of L.

Increasing μ strengthens smoothing while preserving the constant component

Shuman et al. (2013)

Harmonic extension fixes reliable observations

$$\operatorname{argmin}_{x:\, x_O = y_O} \frac{1}{2} x^{\mathrm{T}} L x$$

$$L_{UU} x_U = -L_{UO} y_O$$

$$x_i = \frac{\sum_j w_{ij} x_j}{d_i}$$

Fix the boundary

O contains observed nodes. Their values stay exactly equal to y.

Solve only the unknowns

U contains unobserved nodes. The graph connects these unknowns to the fixed boundary.

Local meaning

Each unobserved value equals the weighted average of its neighbors.

Each connected component needs an observed boundary node

Choosing a recovery method from observation quality

Noisy observations

Laplacian-regularized least squares
Balances measurement fit with graph smoothness.

Useful for denoising or recovering a smooth signal from noisy, partial readings.

Trusted observations

Harmonic extension
Fixes observed values and minimizes the energy of the missing values.

Useful for filling gaps when known values are reliable and graph neighbors have compatible values.

Connection to semi-supervised learning: propagate class scores under a homophily assumption, then apply a class decision rule.

Zhu, Ghahramani & Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. ICML, 2003.

Robust recovery begins with the source of uncertainty

Unreliable measurements

Use a robust data loss, such as Huber, when some observations contain large errors.

Genuine signal boundaries

Use graph total variation when clean values form regions separated by sharp changes.

Uncertain graph connections

Review or learn the graph; use a graph-robust model when edge uncertainty can be characterized.

Choose the data term for measurement errors and the graph term for signal structure.

Graph total variation for signals with sharp boundaries

$$TV(x) = \sum_{i<j} w_{ij} |x_i - x_j|$$

$$\min_x \frac{1}{2} \|Px - y\|^2 + \lambda\, TV(x)$$

What changes

Absolute differences penalize a large edge jump linearly rather than quadratically.

When it helps

Use for approximately piecewise-constant signals whose regions align with the graph.

Choosing the strength

Stronger regularization fuses more values. Validate how well meaningful boundaries are retained.

TV models signal boundaries; a robust data loss can additionally handle outliers.

Wang et al. Trend Filtering on Graphs. Journal of Machine Learning Research, 2016.

Joint estimation when both graph and signal are uncertain

$$\min_{L\in\mathcal{L},Z} \frac{1}{2}\|P_\Omega(Z-Y)\|_F^2 + \alpha \mathrm{tr}(Z^{\mathrm{T}}LZ) + R(L)$$



Problem

Noisy or incomplete signals make graph construction uncertain. Estimate clean signals Z and a constrained graph L together.

When to use it

Useful with repeated observations and a plausible shared graph. Alternate signal and graph updates.

Graph constraints and held-out validation help control feedback between the two estimates.

Dong et al. Learning Laplacian Matrix in Smooth Graph Signal Representations. IEEE Transactions on Signal Processing, 2016.

One layer: update the signal, then the graph

Current estimates Z^{k} and L^{k} → Update signal using current graph → Update graph using new signal → Next estimates Z^{k+1} and L^{k+1}

Signal

$$Z^{k+1} = Z^{k} - \eta_k \left[(Z^{k} - Y) + \mu_k L^{k} Z^{k} \right]$$

Graph

$$L^{k+1} = \Pi_{\mathcal{L}} \left[L^{k} - \tau_k \left(\tfrac{1}{2}\mu_k Z^{k+1} (Z^{k+1})^{T} + \beta_k L^{k} \right) \right]$$

η, τ: update step sizes μ: signal smoothness β: graph penalty
Projection $\Pi\mathcal{L}$ keeps the graph valid and its scale fixed.

Repeat K layers. Parameters stay fixed during the pass; signal and graph estimates change.

Monga, Li & Eldar. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. IEEE Signal Processing Magazine, 2021.

Kojima et al. Algorithm Unrolling-Based Denoising of Multimodal Graph Signals. IEEE TSIPN 12, 544–555, 2026.

Training learns parameters; inference uses them

Noisy training input → Run K layers and final prediction → Compare with clean target → Backpropagate Update parameters

$$\mathcal{L}_{train}(\Theta) = (1/N) \sum_s \|\hat{X}_\Theta(Y_s) - X_s\|_F^2$$

Θ = all learned step sizes and regularization weights.
Training repeats: forward recovery → error → parameter update.

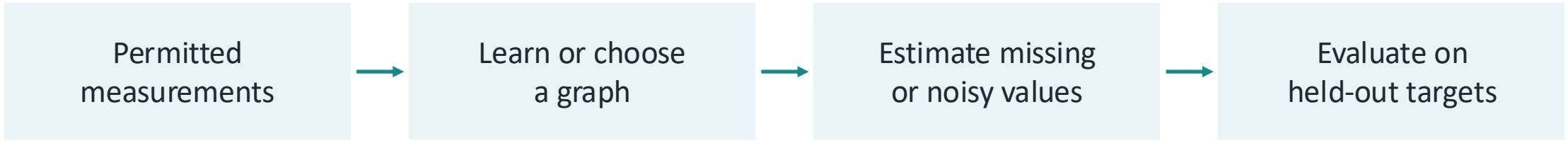
After training: keep Θ fixed

New noisy input → same signal and graph updates → recovered signal and graph.

Monga, Li & Eldar. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. IEEE Signal Processing Magazine, 2021.

Kojima et al. Algorithm Unrolling-Based Denoising of Multimodal Graph Signals. IEEE TSIPN 12, 544–555, 2026.

A complete learning-and-recovery workflow



Graph learning: fixed signals, unknown relationships

Signal recovery: fixed relationships, unknown values

Joint estimation: update both under explicit constraints

References for learning and recovery

Shuman et al. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. IEEE Signal Processing Magazine, 2013.

Dong et al. Learning Laplacian Matrix in Smooth Graph Signal Representations. IEEE Transactions on Signal Processing, 2016.

Kalofolias. How to Learn a Graph from Smooth Signals. AISTATS, 2016.

Zhu, Ghahramani & Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. ICML, 2003.

Huber. Robust Estimation of a Location Parameter. Annals of Mathematical Statistics, 1964.

Wang et al. Trend Filtering on Graphs. Journal of Machine Learning Research, 2016.

Tsukiashi & Ono. Robustifying Graph Laplacian Regularization Against Edge Weight Uncertainties: An Infimal Convolution Approach. ICASSP, 2026.

Shi & Mishne. Graph Laplacian Learning with Exponential Family Noise. IEEE TSIPN 11, 641–654, 2025.

Kojima et al. Algorithm Unrolling-Based Denoising of Multimodal Graph Signals. IEEE TSIPN 12, 544–555, 2026.

Yokota & Tanaka. Multivariate Signal Restoration via Multifold Graph Learning. arXiv:2609.36854, Sep 2026.

Kojima et al. Unrolled Time-Varying Graph Signal Restoration under Spatiotemporal Smoothness Priors. arXiv:2610.11359, Oct 2026.