GRAPH-BASED MACHINE LEARNING

Adrien Weihs

Inverse Problems and Artificial Intelligence in Medicine, Bath UK, July 2025

OUTLINE

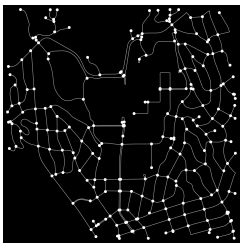
GRAPHS AS DATA STRUCTURES FOR LEARNING

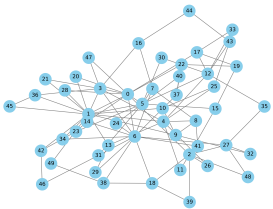
DEFINITION OF GRAPHS

Definition of graphs

A graph $G = (V, E)$ consists of a set of **vertices** V and a set of **edges** E, where each edge $e \in E$ is a pair of vertices, i.e., $e = \{u, v\} \subseteq V$

- **Real-world examples:**
  - Social network: users (nodes), friendships (edges)
  - Geographic road network: intersections (nodes), roads (edges)





GRAPHS ENCODE PAIRWISE RELATIONSHIPS

Why graphs?

Graphs encode (pairwise) relationships between vertices

- Relationships between vertices v_i and v_j can be of various nature:
  - **directional**, i.e. is the relationship between v_i and v_j the same as the relationship between v_j and v_i?

    $\Rightarrow$ if yes, **undirected graphs**; if no, **directed graphs**
  - **weighted**, i.e. are all relationships worth the same?

    $\Rightarrow$ if no, **weighted graphs**

Beyond graphs

Beyond graphs: Other related structures include multigraphs, hypergraphs, and simplicial complexes

GRAPHS PROVIDE STRUCTURE IN NON-EUCLIDEAN SETTINGS

- Examples of data
  - that is embedded in metric space: vectors in $\mathbb{R}^d$, images, etc.
  - that is not embedded in metric space: users in social network, bag of words etc.

Graph allow analysis in non-geometrical settings

Even when not embedded in an obvious metric space, graphs equip data with structure that can be leveraged for various analyses

- By equipping the data with a graph structure, we can perform machine learning tasks (e.g. clustering, learning) on the graph directly

$\Rightarrow$ we **lift the data into graph space**

$\Rightarrow$ we need to define notions such as **distance** or **continuity** on graphs to do analysis

$\Rightarrow$ **Caution**: the graph needs to be carefully constructed/chosen to reflect meaningful information

GRAPH CONSTRUCTION I

- Some datasets naturally have a graph structure (e.g. social networks, road networks, molecules)

- **Question**: For other datasets, how do we define the graph?

Weighted graphs

A weighted graph G is a tuple $G = (V, W)$ where V is a set of n vertices and $W = \{w_{ij}\}_{i,j=1}^{n}$ is a matrix. The entry w_{ij} is the weight of the edge between vertices v_i and v_j

- For datasets in metric spaces, w_{ij} is often a function of $\|v_i - v_j\|$, i.e. $w_{ij} = \eta(\|v_i - v_j\|)$:
  - ε-graphs: $w_{ij} = \mathbb{1}_{\{\|v_i - v_j\| \leq \varepsilon\}} \Rightarrow$ **vertices only interact within a ε-ball**
  - k-nn graphs: $w_{ij} = \eta(\|v_i - v_j\|)$ if v_i is among the k-nearest neighbors of v_j or v_j is among the k-nearest neighbors of $v_i \Rightarrow$ **vertices only interact with their k-nearest neighbors**

Graph design considerations

Choice of norm $\|\cdot\|$ and function η is crucial in graph construction and application dependent

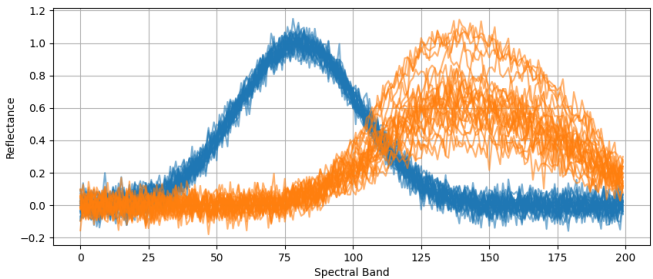
GRAPH CONSTRUCTION: EXAMPLE OF HYPERSPECTRAL DATA I

- A **hyperspectral image** [1] HSI $\in \mathbb{R}^{H \times W \times d}$ is a 3D data cube capturing detailed spectral information at each pixel
- $H \times W$ are the spatial dimensions (image height and width) and d is the number of spectral bands (typically 100–300)
- each pixel is a high-dimensional vector $x_i \in \mathbb{R}^d$, called a **spectral signature**
- Principle of hyperspectral imaging:
  - Each material interacts with light differently across wavelengths, producing a characteristic reflectance pattern/spectral signature
  - Spectral shape encodes chemical or physical properties of materials.

    $\Rightarrow$ **Distinct materials yield distinct spectral signatures** and hyperspectral imaging can help to distinguish between materials with similar color but distinct spectral signatures — which RGB imaging cannot (applications in remote sensing, medical imaging etc.)

GRAPH CONSTRUCTION: EXAMPLE OF HYPERSPECTRAL DATA II

- Magnitude of spectral signature can vary due to lighting, shadows or sensor variability



- **Goal:** Identify similar spectral signatures (i.e., same materials) despite variability in lighting or sensor exposure.

GRAPH CONSTRUCTION: EXAMPLE OF HYPERSPECTRAL DATA III

- We define the **cosine distance**: $\text{cosine\_dist}(x_i, x_j) = 1 - \frac{\langle x_i, x_j \rangle}{\|x_i\| \, \|x_j\|}$

  $\Rightarrow$ **Illumination Invariance** and **Shape Sensitivity**: Cosine distance ignores vector magnitude and captures angular similarity between spectral signatures.

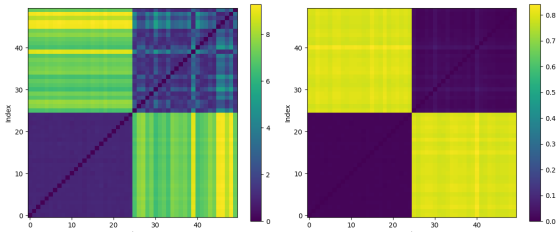


Figure: Pairwise distance of spectral signatures. Left: Euclidean. Right: Cosine.

Similarity graph

Similarity between spectral signatures is better encoded in graph space if we use the cosine distance compared to the Euclidean one

GRAPH CONSTRUCTION II

- For datasets that are not in a metric space, one needs to resort to other constructions
- **Example**: Bag of words
  - Each vertex is a collection of words
  - Connect vertices v_i and v_j if they have a word in common

MACHINE LEARNING ON GRAPHS: OVERVIEW OF TASKS I

- **Node-level tasks**
  - **Node classification:** predict a label for each node (e.g., user type in a social network)
  - **Node clustering:** group nodes into unsupervised communities based on structure or features
  - **Node regression:** predict continuous attributes for nodes (e.g., temperature at sensor nodes)
- **Edge-level tasks**
  - **Link prediction:** predict whether an edge should exist between two nodes (e.g., friend recommendations)
  - **Edge classification:** assign a label or type to each edge (e.g., type of relationship or interaction)

MACHINE LEARNING ON GRAPHS: OVERVIEW OF TASKS II

- **Graph-level tasks**
  - **Graph classification:** assign a label to the entire graph (e.g., molecule is toxic or not)
  - **Graph regression:** predict a continuous value for a graph (e.g., solubility of a molecule)

Focus of this course

Non-deep methods for node clustering and node classification

- Lots of deep-learning methods to solve all tasks efficiently [47]
- Non-deep methods are computationally tractable, interpretable, and often allow rigorous theoretical analysis (e.g., spectral clustering, Laplace learning).

THE GRAPH HOMOPHILY ASSUMPTION

- We focus on **node classification** and **node clustering** on **similarity graphs**, where:
  - Nodes represent data points (e.g. pixels, users)
  - Edges represent pairwise similarity between nodes
- This setting assumes that the graph is **homophilous** [29]:
  - **Homophily:** adjacent nodes tend to share the same label or belong to the same cluster
  - This underlies many graph learning methods (e.g. spectral clustering, Laplace learning, GCNs)
- In **heterophilious** graphs (where connected nodes are often dissimilar), these methods may fail or require adaptation [21]

GRAPH HOMOPHILY AND HETEROPHILY

Figure: Examples of graph. Left: Homophily. Right: Heterophily

- Homophilous graphs: graphs on hyperspectral data, citation networks
- Heterophilious graphs: fraudster graphs

TAKEAWAY: GRAPHS AS DATA STRUCTURES FOR LEARNING

- Graphs enable learning in both **geometric** and **non-geometric** settings
- Graphs encode prior knowledge (e.g., similarity, structure) in data
- We will explore how to construct and learn from graphs without relying on deep neural networks
- We focus on node-level tasks

GRAPH LEARNING

TYPES OF LEARNING PARADIGMS

- Machine learning tasks differ based on the availability of labels
- **Key distinction:** how much supervision (i.e., labeled data) is available

Paradigm	Input Data
Unsupervised	Only inputs $x_1, \ldots, x_n$
Semi-supervised	Few labels (x_i, y_i), mostly x_i
Supervised	Labeled pairs (x_i, y_i)

UNSUPERVISED LEARNING AND CLUSTERING

CLUSTERING IN UNSUPERVISED LEARNING

- In **unsupervised learning**, we are given only data points:

$$\{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^d$$

  with **no access to labels** y_i.

- The goal of **clustering** is to group the points into K clusters:

$$\text{Find } y_1, y_2, \ldots, y_n \in \{1, 2, \ldots, K\}$$

  such that:

  - Similar points x_i, x_j are assigned the same label $y_i = y_j$,
  - Dissimilar points are assigned different labels $y_i \neq y_j$

CLUSTERING ON SIMILARITY GRAPHS

- When data is represented as a **similarity graph**, i.e.

$$G = (V, W), \quad w_{ij} \text{ encodes similarity between } x_i \text{ and } x_j$$

  clustering becomes a **graph partitioning** problem.

Key assumptions

- We assume that our graph is undirected, i.e. $w_{ij} = w_{ji}$, and $w_{ij} \geq 0$
- A reasonable model should lead to large w_{ij} when v_i and v_j are close

- Desired partitioning:
  - **Low edge weights** between different groups (dissimilar points)
  - **High edge weights** within the same group (similar points)

GRAPH CUTS: MINCUT I

- For $A, B \subseteq V$, we define

$$W(A,B) := \sum_{i \in A, j \in B} w_{ij}, \quad |A| = \{\text{number of vertices in A}\} \quad \text{and} \quad \overline{A} = \text{ complement of A in } V$$

$\Rightarrow W(A,B)$ sums the weights of all the edges connecting A and B

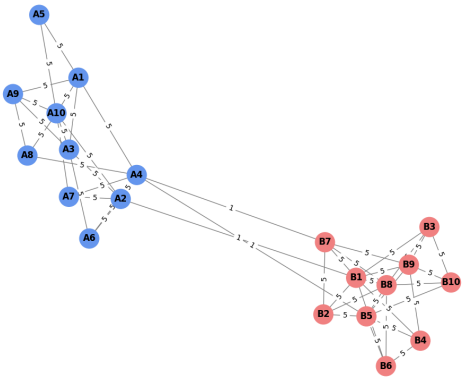
Mincut [42]

For a given number of K subsets, find a partition $A_1, \ldots, A_K$ of V that minimizes

$$\text{cut}(A_1, \ldots, A_K) := \frac{1}{2} \sum_{k=1}^{K} W(A_k, \overline{A_k})$$

$\Rightarrow$ Minimize sum of edge weights going from a cluster A_k to its complement

GRAPH CUTS: MINCUT II



- For $K = 2$ and a partition A_1, A_2 of V, we have $A_2 = \overline{A_1}$

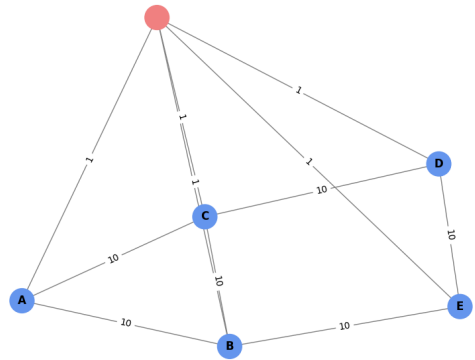
  $\Rightarrow \mathrm{cut}(A_1, A_2) = W(A, \overline{A})$

  $\Rightarrow$ minimizing $\mathrm{cut}(A_1, A_2)$ is equivalent to having low edge weights between two clusters A_1 and $\overline{A_1}$ and high edge weights within the same group

GRAPH CUTS: MINCUT III

- In practice, mincut often leads to **unbalanced partitions**, e.g. one node separated from the rest

  $\Rightarrow$ **Solution**: Add set-size constraint into the minimization problem



RATIOCUT

RatioCut [38]

For a given number of K subsets, find a partition $A_1, \ldots, A_K$ of V that minimizes

$$\mathrm{RatioCut}(A_1, \ldots, A_K) := \frac{1}{2} \sum_{k=1}^{K} \frac{W(A_k, \overline{A_k})}{|A_k|}$$

- Dividing by the size of the sets $|A_k|$ favors the selection of **balanced** clusters measured by the number of vertices (other variants to balance by the sum of edge weights in each cluster Ncut)

  $\Rightarrow$ The problem is NP-hard to solve

- **Question**: is there an efficient way to compute this partition?

GRAPH LAPLACIAN

Definition of graph Laplacian [34]

Given a weighted undirected graph $G = (V, W)$ with $|V| = n$, we define the degree matrix $D = \mathrm{diag}(d_1, \ldots, d_n)$, with $d_i = \sum_{j=1}^{n} w_{ij}$. The (unnormalized) graph Laplacian is defined as $L := D - W$ and satisfies:

$$\text{for all } u \in \mathbb{R}^n,\ u^T L u = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2, \text{ i.e. } L \text{ is positive semi-definite}$$

Proof.

$$u^\top L u = u^\top D u - u^\top W u = \sum_{i=1}^{n} d_i u_i^2 - \sum_{i,j=1}^{n} u_i u_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i u_i^2 - 2 \sum_{i,j=1}^{n} u_i u_j w_{ij} + \sum_{j=1}^{n} d_j u_j^2 \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2.$$

□

RELAXATION OF RATIOCUT I

- For ease of exposition, consider $K = 2$
- Consider a partition $V = A \cup \overline{A}$ and define $u \in \mathbb{R}^n$ ($f \in \mathbb{R}^n \cong \{f(v_i) \mid f : V \mapsto \mathbb{R}\}_{i=1}^n$):

$$u_i = \begin{cases} \sqrt{\frac{|\bar{A}|}{|A|}} & \text{if } i \in A \\ -\sqrt{\frac{|A|}{|\bar{A}|}} & \text{if } i \in \bar{A} \end{cases} \quad \Rightarrow \quad (u_i - u_j)^2 = \begin{cases} 0 & \text{if } i, j \in A \\ \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 & \text{if } i \in A, j \in \overline{A} \end{cases}$$

- Then,

$$\begin{aligned} u^T L u &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2 \\ &= \frac{1}{2} \sum_{i \in A, j \in \overline{A}} w_{ij} \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 + \frac{1}{2} \sum_{i \in \overline{A}, j \in A} w_{ij} \left(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \\ &= \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \sum_{i \in A, j \in \overline{A}} w_{ij} = \left(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\right)^2 \operatorname{cut}(A, \overline{A}) \end{aligned}$$

RELAXATION OF RATIOCUT II

- We continue:

$$\begin{aligned} u^T L u &= \mathrm{cut}(A, \bar{A}) \left(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2 \right) \\ &= \mathrm{cut}(A, \bar{A}) \left(\frac{|\bar{A}| + |A|}{|A|} + \frac{|A| + |\bar{A}|}{|\bar{A}|} \right) \\ &= n \cdot \mathrm{cut}(A, \bar{A}) \left(\frac{1}{|A|} + \frac{1}{|\bar{A}|} \right) \\ &= n \cdot \mathrm{RatioCut}(A, \bar{A}) \end{aligned}$$

- We also have

$$\mathbb{1}^T u = \sum_{i=1}^{n} u_i = \sum_{i \in A} \sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i \in \bar{A}} \sqrt{\frac{|A|}{|\bar{A}|}} = |A| \sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}| \sqrt{\frac{|A|}{|\bar{A}|}} = 0$$

and

$$\|u\|^2 = \sum_{i=1}^{n} u_i^2 = |A| \frac{|\bar{A}|}{|A|} + |\bar{A}| \frac{|A|}{|\bar{A}|} = |\bar{A}| + |A| = n$$

RELAXATION OF RATIOCUT III

Equivalence of RatioCut

Minimizing $\mathrm{RatioCut}(A, \overline{A})$ is equivalent the finding

$$\min_{A \subset V} u^T L u \text{ subject to } \mathbb{1}^T u = 0,\ u \text{ defined with respect to } A,\ \|u\| = \sqrt{n}$$

- This is a hard problem, since u can only take two values $\Rightarrow$ relax further

Relaxation of RatioCut [45]

We want to solve

$$\min_{u \in \mathbb{R}^n} u^T L u \text{ subject to } \mathbb{1}^T u = 0,\ \|u\| = \sqrt{n} \tag{1}$$

- From $u^T L u = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2$, we see that $\mathbb{1}$ is an eigenvector of L

  $\Rightarrow$ (1) is minimized by **the second eigenvector of L**, ψ_2, called the Fiedler vector

RELAXATION OF RATIOCUT IV

- $[\psi_2]_i$ should be the class assignment of vertex v_i but $[\psi_2]_i \in \mathbb{R}$
  $\Rightarrow$ we cluster $\{[\psi_2]_i\}_{i=1}^n$ into two classes using **k-means clustering** and use these classes as final partition for our graph

k-means Clustering

Input: Data points $x_1, \ldots, x_n \in \mathbb{R}^d$, number of clusters K
Output: Clusters $C_1, \ldots, C_K$
Initialize cluster centers $\mu_1, \ldots, \mu_K$ (e.g., randomly);
repeat
  Assign each point to its closest center: $c_j = \arg\min_i \|x_j - \mu_i\|^2$ for $j = 1, \ldots, n$;
  Update each center: $\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$ for $i = 1, \ldots, K$;
until *convergence*;
return $C_1, \ldots, C_K$

k-MEANS CLUSTERING

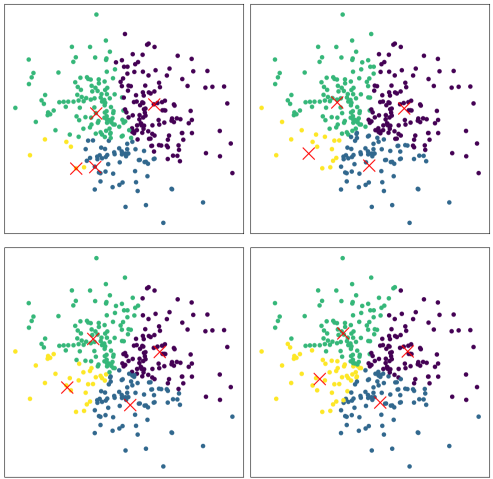


Figure: Four steps of the k-means algorithm with $K = 4$

SPECTRAL CLUSTERING I

- Similar argument works for K clusters which leads to the following algorithm

Unnormalized Spectral Clustering [35]

Input: Weight matrix W, number of clusters K
Output: Clusters $A_1, \ldots, A_K$
Compute the unnormalized Laplacian $L = D - W$;
Compute the first K eigenvectors $u_1, \ldots, u_K$ of L;
Form matrix $U \in \mathbb{R}^{n \times K}$ with columns $u_1, \ldots, u_K$ and let y_i be the i-th row of U;
Cluster the points $y_1, \ldots, y_n$ in $\mathbb{R}^K$ using k-means into $C_1, \ldots, C_K$;
return $A_i = \{j \mid y_j \in C_i\}$ *for* $i = 1, \ldots, K$

SPECTRAL CLUSTERING II

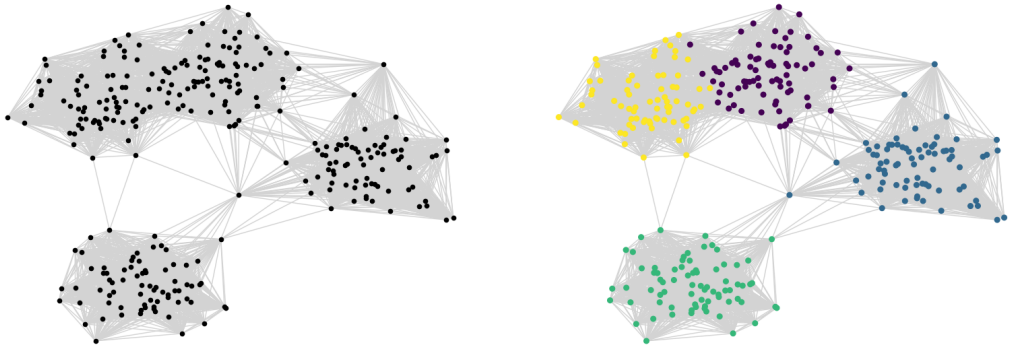


Figure: Spectral clustering with $K = 4$

CONNECTED COMPONENTS AND SPECTRAL PROPERTIES

Number of connected components

Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.

SPECTRALNET: GENERALIZING SPECTRAL CLUSTERING I

- **Limitation of Spectral Clustering**
  - Embedding is computed from full similarity graph
  - Cannot generalize to new data
- **SpectralNet** [37]:
  - Learns a neural network $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ to approximate the spectral embedding
  - Trains using pairwise similarity loss and orthogonality constraints on output

    $\Rightarrow$ The output vectors $y_i \in \mathbb{R}^k$ **approximate the rows of the eigenvectors matrices**

Advantages

- Enables out-of-sample prediction
- Scales to large datasets

SPECTRALNET: GENERALIZING SPECTRAL CLUSTERING II

- **Learn Embedding**
  - Train neural network f_θ to embed data points: $x_i \mapsto y_i = f_\theta(x_i)$
- **Clustering via k-means**
  - Apply k-means to $\{y_i\}_{i=1}^n$ to assign cluster labels

Generalization

- New data point x_{new} is mapped to $f_\theta(x_{\text{new}})$
- Assign cluster using trained k-means model, i.e. x_{new} gets the label of the nearest center to $f_\theta(x_{\text{new}})$

REMARKS ON SPECTRAL CLUSTERING RELAXATION

- **No optimality guarantee**: the solution of relaxed spectral clustering is **not guaranteed** to be close to the original combinatorial partitioning problem
- **Relaxation is not unique**: for instance, one can derive a completely different relaxation via semidefinite programming
- **Popularity**: spectral relaxation yields a **standard linear algebra problem** (eigenvector computation) that is easy to solve in practice

GRAPH TOTAL VARIATION AND CUT FUNCTIONAL

- We define the **graph total variation (TV)** [19] as:

$$\mathrm{TV}(u) = \frac{1}{2} \sum_{i,j} w_{ij} |u_i - u_j|$$

- For $u = \mathbb{1}_A$, we recover the **cut functional**:

$$\mathrm{TV}(\mathbb{1}_A) = \mathrm{cut}(A, \bar{A}) = \sum_{i \in A, j \in \bar{A}} w_{ij}$$

- **Relaxation:** we let $u_i \in \mathbb{R}^n \Rightarrow \mathrm{argmin}_{u \in \mathbb{R}^n} \mathrm{TV}(u) =$ any constant and problem is trivial

  $\Rightarrow$ we need to add regularization term

GINZBURG–LANDAU FUNCTIONAL ON GRAPHS

- We want to minimize the **Ginzburg–Landau functional** [4]:

$$\sum_{i,j} w_{ij}|u_i - u_j| + \sum_i P(u_i)$$

  where $P : \mathbb{R} \mapsto [0, \infty)$ is a double-well potential, i.e. $P(\pm 1) = 0$ (e.g. $\frac{1}{4}(u^2 - 1)^2$)

- **Interpretation:**
  - First term promotes **smoothness of u on the graph**
  - Second term encourages $u_i \in \{-1, +1\}$, **approximating binary segmentation**

COMPARISON OF RELAXATIONS

Aspect	Spectral Methods	Variational Methods
Objective	Relaxed quadratic form	Regularized total variation
Optimization	Solves eigenvalue problem (efficient)	Iterative optimization (e.g. MBO)
Faithfulness to Cut	Poor in some cases	More faithful to original cut

Table: Comparison between spectral and variational methods for graph clustering

TAKEAWAY: CLUSTERING

- **Goal**: Partition unlabeled data into K meaningful clusters by exploiting pairwise similarities
- Graph Cut Approaches:
  - **MinCut**: minimizes edges between clusters — may lead to unbalanced partitions
  - **RatioCut**: balances cluster size — NP-hard
- **Spectral Relaxation**:
  - Solve eigenvalue problem $\min u^\top L u$ under constraints
  - Use second eigenvector and k-means for clustering
- **Variational Methods**:
  - Use graph total variation + double-well potential (Ginzburg–Landau)

SEMI-SUPERVISED LEARNING

SEMI-SUPERVISED LEARNING PRINCIPLE

- **Aim:** Learn a prediction function using a small set of labeled data and a large set of unlabeled data
  - Labeled data: $\{(x_i, y_i)\}_{i=1}^{N}$, where y_i is known
  - Unlabeled data: $\{x_j\}_{j=N+1}^{n}$, where y_j is unknown

Key Idea

Leverage the structure of the unlabeled data to improve learning

$\Rightarrow$ create **relationships** between labelled and unlabelled samples

SUPERVISED AND SEMI-SUPERVISED LEARNING

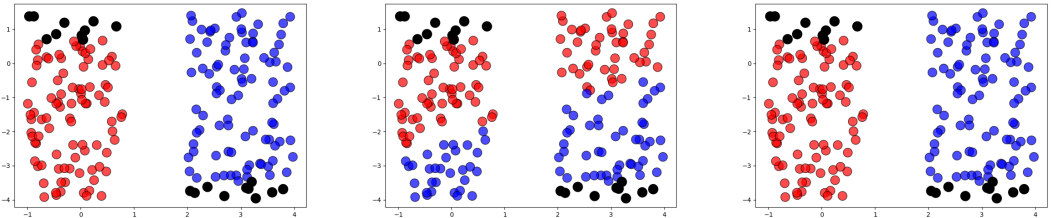


Figure: Left: Ground truth. Middle: Supervised. Right: Semi-supervised. Labelled points used for learning are marked in black.

GRAPH DIRICHLET ENERGY I

- We structure our points $x_1, \dots, x_n \overset{\text{iid}}{\sim} \mu \in \mathcal{P}(\Omega)$ in a similarity graph $G = (V, W)$
- **Recall**: homophily assumption, i.e. labels that are close on the graph should have similar labels
- We relate all the points in the graph by

Graph Dirichlet energy

For a vector $u \in \mathbb{R}^n$, the graph Dirichlet energy is:

$$\mathcal{E}(u) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2 = u^T L u$$

where $L = D - W$ is the graph Laplacian.

- **Observation**: $\mathcal{E}$ does not involve any labels

GRAPH DIRICHLET ENERGY II

- Graph Dirichlet energy $\frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2 = u^T L u$:
  - **Measures the smoothness** of u over the graph
  - Small energy leads similar values of u on neighboring nodes

$\Rightarrow$ if $u \in \{0,1\}^n$ is the label assignment, we want to minimize $\mathcal{E}$ over $u \in \{0,1\}^n$

- We relax to $u \in \mathbb{R}^n$ and will then threshold

$$y_i = \begin{cases} 0 & \text{if } u_i < 0.5 \\ 1 & \text{else} \end{cases}$$

- This is similar to the spectral clustering relaxation for unsupervised learning

LAPLACE LEARNING

Laplace learning [50]

Laplace learning aims to find:

$$u = \underset{u \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i,j=1}^{n} w_{ij}(u_i - u_j)^2 \quad \text{such that } u_i = y_i \text{ for } 1 \leq i \leq N$$

- Laplace learning is a very popular technique and works well in many applications
- Laplace learning can also be extended to the multi-class setting
- **Question**: How do we compute the solution?

DERIVATION OF THE CLOSED-FORM SOLUTION I

- Let $\mathcal{L} = \{1, \dots, N\}$ and $\mathcal{U} = \{N+1, \dots, n\}$ be the sets of labelled and unlabelled indices
- We write

$$u = \begin{bmatrix} u_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} L_{\mathcal{L}\mathcal{L}} & L_{\mathcal{L}\mathcal{U}} \\ L_{\mathcal{U}\mathcal{L}} & L_{\mathcal{U}\mathcal{U}} \end{bmatrix}$$

- For Laplace learning, we minimize $\mathcal{E}(u)$ such that $u_{\mathcal{L}} = y$
  - We have

$$\begin{aligned} \mathcal{E}(u) &= \begin{bmatrix} u_{\mathcal{L}}^\top & u_{\mathcal{U}}^\top \end{bmatrix} \begin{bmatrix} L_{\mathcal{L}\mathcal{L}} & L_{\mathcal{L}\mathcal{U}} \\ L_{\mathcal{U}\mathcal{L}} & L_{\mathcal{U}\mathcal{U}} \end{bmatrix} \begin{bmatrix} u_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix} \\ &= u_{\mathcal{L}}^\top L_{\mathcal{L}\mathcal{L}} u_{\mathcal{L}} + 2 u_{\mathcal{L}}^\top L_{\mathcal{L}\mathcal{U}} u_{\mathcal{U}} + u_{\mathcal{U}}^\top L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}} \end{aligned}$$

  - Since $u_{\mathcal{L}} = y$, **the first term is constant** and we only minimize over $u_{\mathcal{U}}$

DERIVATION OF THE CLOSED-FORM SOLUTION II

- We proceed as follows:
  - we compute the gradient of $2u_{\mathcal{L}}^\top L_{\mathcal{L}\mathcal{U}} u_{\mathcal{U}} + u_{\mathcal{U}}^\top L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}}$ with respect to $u_{\mathcal{U}}$, which yields

$$2L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}} + 2L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}}$$

  - Setting the gradient to zero, we obtain:

Solution to Laplace learning

The solution to Laplace learning is given by $u = [y, u_{\mathcal{U}}]$ where

$$u_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{U}}^{-1} L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}}$$

- In practice, for large datasets, one solves $L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}}$ through iterative methods

DISCRETE LAPLACE EQUATION

- We have

$$\begin{bmatrix} L_{\mathcal{L}\mathcal{L}} & L_{\mathcal{L}\mathcal{U}} \\ L_{\mathcal{U}\mathcal{L}} & L_{\mathcal{U}\mathcal{U}} \end{bmatrix} \begin{bmatrix} u_{\mathcal{L}} \\ u_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} L_{\mathcal{L}\mathcal{L}} u_{\mathcal{L}} + L_{\mathcal{L}\mathcal{U}} u_{\mathcal{U}} \\ L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}} + L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}} \end{bmatrix}$$

- Since $u_{\mathcal{U}} = -L_{\mathcal{U}\mathcal{U}}^{-1} L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}}$, we have $L_{\mathcal{U}\mathcal{L}} u_{\mathcal{L}} + L_{\mathcal{U}\mathcal{U}} u_{\mathcal{U}} = 0$ or, re-written differently:

Discrete Laplace equation

The solution to Laplace learning can be characterized equivalently by

$$[Lu]_{\mathcal{U}} = 0 \quad \text{and} \quad u_{\mathcal{L}} = y$$

CONTINUUM HARMONIC FUNCTIONS

- Let $u : \Omega \to \mathbb{R}$ solve the Dirichlet problem:

$$\Delta u = 0 \quad \text{in } \Omega, \qquad u = g \quad \text{on } \partial\Omega$$

- **Mean-Value Property:** For any point $x \in \Omega$ and small ball $B(x, r) \subset \Omega$, the solution satisfies

$$u(x) = \frac{1}{|\partial B(x, r)|} \int_{\partial B(x,r)} u(y)\, dS(y)$$

  $\Rightarrow$ the value at x is the average of u over a small neighborhood

- **Maximum Principle:** The solution attains its maximum and minimum on the boundary:

$$\max_{x\in\Omega} u(x) = \max_{x\in\partial\Omega} u(x), \quad \min_{x\in\Omega} u(x) = \min_{x\in\partial\Omega} u(x)$$

- These properties characterize **harmonic functions** and imply uniqueness of the solution

DISCRETE MEAN-VALUE PROPERTY

- Assume $[Lu]_{\mathcal{U}} = 0$
  - Then, for $i \in \mathcal{U}$

$$[Lu]_i = \sum_{j=1}^{n} w_{ij}(u_i - u_j) = 0$$

  - Rearranging, we obtain:

Discrete maximum principle

If u is the solution to Laplace learning, it satisfies the discrete mean-value property:

$$u_i = \frac{1}{d_i} \sum_{j=1}^{n} w_{ij} u_j$$

$\Rightarrow$ **"The label at point x_i is the weighted average of the labels of its neighbors"** (modulo thresholding)

DISCRETE MAXIMUM PRINCIPLE

- One can also show

Discrete maximum principle

Assume G is connected, i.e. there exists a path between every pair of vertices x_i. Assume that $u \in \mathbb{R}^n$ satisfies $[Lu]_{\mathcal{U}} \leq 0$. Then,

$$\max_{1 \leq i \leq n} [u]_i = \max_{i \in \mathcal{L}} [u]_i$$

Proof sketch. Analogous to the continuum proof, i.e. use the mean value property to show that if the maximum is attained in $\mathcal{U}$, then u must be constant. □

$\Rightarrow$ As a corollary, we obtain that the solutions to the discrete Laplace equation, i.e. **solutions to Laplace learning, are unique**

CONVERGENCE OF LAPLACIANS

- We have shown that solution to Laplace learning is a **discrete harmonic function**

  $\Rightarrow$ What is the link between L and Δ?

- Several ways to characterize/clarify the link between L and Δ: the most intuitive is through **pointwise consistency** as follows

Consistency of Laplacians [10]

Assume that $w_{ij} = \frac{1}{\varepsilon_n}\eta(\|x_i - x_j\|/\varepsilon_n)$. Then, for $u : \Omega \mapsto \mathbb{R}$ smooth enough and high probability, we have

$$\max_{1\leq i\leq n} |[L_n u|_{V_n}](x_i) - \Delta u(x_i)| \leq C\|u\|_{\mathrm{C}^3(\Omega)}\varepsilon_n$$

- **Conclusion**: the Laplacian matrix L is a discrete approximation of the Laplace operator Δ

SPECTRAL FORMULATION OF GRAPH DIRICHLET ENERGY

- L is a real symmetric positive semi-definite matrix $\Rightarrow$ it has an **orthonormal eigenbasis**:

$$L\psi_k = \lambda_k \psi_k, \quad 0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$$

- Every $u \in \mathbb{R}^n$ can be written as $u = \sum_{k=1}^n (u^T \psi_k)\psi_k =: \sum_{k=1}^n u_k \psi_k$

$$\Rightarrow \mathcal{E}(u) = \left(\sum_{k=1}^n u_k \psi_k^T\right)\left(\sum_{k=1}^n u_k \lambda_k \psi_k\right) = \sum_{k=1}^n \lambda_k u_k^2$$

SPECTRAL FORMULATION OF LAPLACE LEARNING

Spectral formulation of Laplace learning

Laplace learning can also be formulated as finding

$$u = \underset{u \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{k=1}^{n} \lambda_k u_k^2 \quad \text{such that } u_{\mathcal{L}} = y$$

- We know that $\mathcal{E}$ measures the smoothness on the graph, i.e. the lower $\mathcal{E}(v)$ is, the smoother v is on the graph
  - $\mathcal{E}(\psi_k) = \psi_k^T L \psi_k = \lambda_k \|\psi_k\|_2 = \lambda_k$

    $\Rightarrow$ **ψ_k is decreasingly smooth on the graph as k increases**
- Minimizing $\sum_{k=1}^{n} \lambda_k u_k^2$ will lead to large/small u_k when λ_k is small/large

  $\Rightarrow$ **Laplace learning is a low-pass filter** (similarly to what can be done using Fourier analysis)

EIGENVECTOR SMOOTHNESS

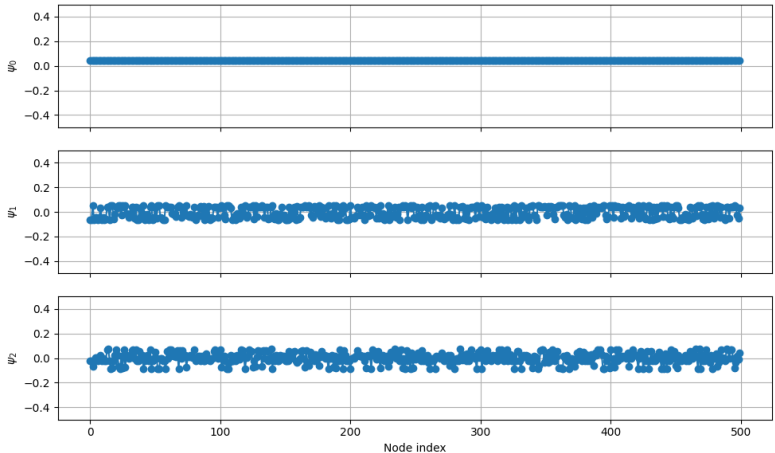
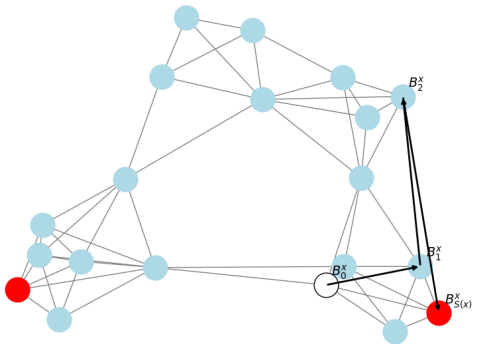


Figure: First three eigenvectors of graph Laplacian of a random geometric graph

RANDOM WALKS ON GRAPHS



- Let $G_n = (\Omega_n, W)$ be a graph with edge weights $W = (w_{ij})$.
- Let B_t^x be a random walk on Ω_n, starting from $B_0^x = x \in \Omega_n$, with transitions given by:

$$\mathbb{P}(B_{t+1}^x = x_j \mid B_t^x = x_i) = \frac{w_{ij}}{d_i}$$

$\Rightarrow$ the random walk can only move between connected nodes

- Define the **hitting time of the labelled set** (a stopping time)

$$S(x) = \min\{t \in \mathbb{N} \mid B_t^x \in \{x_i\}_{i \in \mathcal{L}}\}$$

RANDOM WALK FORMULATION OF LAPLACE LEARNING

Random walk formulation of Laplace learning

Assume the graph G_n is connected. Define

$$u(x) = \mathbb{E}\left[\sum_{j\in\mathcal{L}} y_j \cdot \mathbf{1}_{\{B^x_{S(x)}=x_j\}}\right] = \sum_{j\in\mathcal{L}} y_j \mathbb{P}(\{B^x_{S(x)} = x_j\}),$$

i.e. the expected label at hitting time when starting at node x. Then, u is the solution to Laplace learning.

Proof sketch. By definition of the hitting time, $u_{\mathcal{L}} = y$. On the unlabelled set $\mathcal{U}$, we show that u satisfies the discrete mean-value property, which implies that $[Lu]_{\mathcal{U}} = 0$. We conclude that u solves the discrete Laplace equation. □

LAPLACE LEARNING IS ILL-POSED

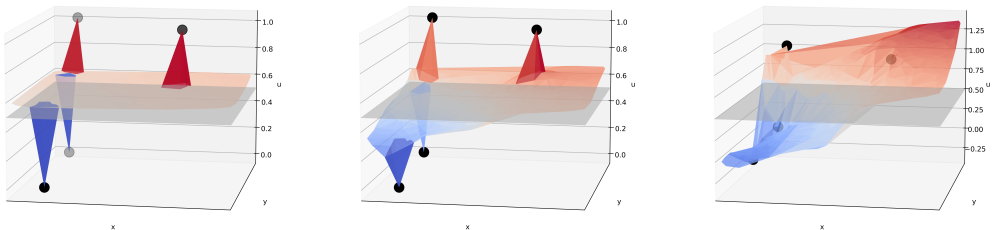


Figure: Different labelling functions. The label assignement threshold is in grey. Left: Laplace learning with radius 1. Middle: Laplace learning with radius 2. Right: Desired labelling function

TOWARDS WELL-POSED EXTENSIONS

- When labeled points are **sparse**, i.e. $|\mathcal{L}| \ll n$, the solution to Laplace learning can become **degenerate**
  - **Sharp spikes at labeled points**
  - **Flat or uninformative behavior elsewhere**
- We seek extensions to Laplace learning which:
  - **lead to better accuracy in SSL**
  - **provably remove degeneracies**
  - **are efficient to compute**

EXPLAINING THE DEGENERACIES

- There are two main ways to explain the ill-posedness of Laplace learning when $|\mathcal{L}| \ll n$:
  - **discrete arguments**
  - **large data limit arguments**
- Both lead to separate extensions which we will further explore below

STATIONARY DISTRIBUTION OF RANDOM WALKS

Stationary distribution of a random walk

The stationary distribution of a random walk on a graph, is the distribution $\pi \in \mathbb{R}^n$ such that $\pi = P\pi$, where P is the transition matrix of the walk. On our graphs, $[\pi]_i = \frac{d_i}{\sum_j d_j}$.

Proof. In order to describe the random walk, one usually proceeds as follows:

- we prescribe the initial probability $p(0)$ of a walker at every node, i.e. $p(0) \in \mathbb{R}^n$ such that $[p(0)]_i \geq 0$ and $\sum_{i=1}^n [p(0)]_i = 1$
- **Note**: if we want the walker to start at node x_i, we set $[p(0)]_j = \delta_{ij}$
- The probability of the walker at time 1 $p(1)$ is given by $p(1) = WD^{-1}p(0) =: Pp(0)$: indeed,

$$[p(1)]_i = \sum_{j=1}^{n} \frac{w_{ij}}{d_j} [p(0)]_j \Rightarrow \text{we recover the random walk defined previously}$$

MIXING TIME OF A RANDOM WALK

Proof continued.

- Iterating the above argument, we obtain that $p(t) = P^t p(0)$
- It is straight-forward to check that $\pi = P\pi$

□

Mixing time of random walks

Let P be the transition matrix of a random walk on a finite graph with stationary distribution π. Then the **mixing time** with respect to a norm $\|\cdot\|$ is defined by:

$$t_{\text{mix}}^{\|\cdot\|}(\varepsilon) := \min\left\{ t \in \mathbb{N} \;\middle|\; \sup_{p(0)} \|P^t p(0) - \pi\| \leq \varepsilon \right\},$$

where $p(0)$ is any initial probability distribution over the nodes.

LAPLACE RANDOM WALK LEADS TO CONSTANT SOLUTIONS

- We assume a nice setting, e.g. the **mixing times are finite**

- The random walk interpretation of Laplace learning explains the ill-posedness at the discrete level: **if $|\mathcal{L}| \ll n$, then the probability of hitting a label is low**

  $\Rightarrow$ for x unlabelled, the hitting time of the labelled set $S(x)$ may be **greater than "the" mixing time** of the random walk

  $\Rightarrow$ the distribution of a walker starting at x unlabelled for $t \gg 1$ is π

  $\Rightarrow$ we have

$$u(x_i) = \sum_{j \in \mathcal{L}} y_j \mathbb{P}(\{B^x_{S(x)} = x_j\}) \approx \frac{\sum_{j \in \mathcal{L}} y_j d_j}{\sum_{r=1}^{n} d_r} =: c$$

  $\Rightarrow$ **on the unlabelled set, we will have a constant and spikes on the labelled set**

POISSON RANDOM WALK I

- Poisson learning suggests to assign labels based on a different random walk

- **Idea**:
  - we start random walks from every labelled nodes $\{x_j\}_{j\in\mathcal{L}}$
  - for every unlabelled node x_i and number of steps t, we count how many times $B_t^{x_j} = x_i$
  - the label assigned to x_i **after T steps** is the average of the labels $\{y_j\}_{j\in\mathcal{L}}$ weighted by the count (previous point) at each time step

    $$\Rightarrow u_T(x_i) = \mathbb{E}\left[\sum_{t=0}^{T}\sum_{j\in\mathcal{L}} y_j \mathbb{1}_{\{B_t^{x_j}=x_i\}}\right] = \sum_{t=0}^{T}\sum_{j\in\mathcal{L}} y_j \mathbb{P}(\{B_t^{x_j}=x_i\})$$

- **Question**: the Laplace learning random walk became ill-posed **at very low label rates** when the number of steps $t \gg 1$, what about the Poisson learning random walk?

POISSON RANDOM WALK II

- When $t \gg 1$, the random walk will be distributed similarly to the stationary distribution π

$$\Rightarrow \ \mathbb{P}(\{B_t^{x_j} = x_i\}) \approx \frac{d_j}{\sum_{r=1}^{n} d_r}$$

- This implies that the label assigned to x_i after t steps is approximately

$$\sum_{j \in \mathcal{L}} y_j \mathbb{P}(\{B_t^{x_j} = x_i\}) \approx \frac{d_i \sum_{j \in \mathcal{L}} y_j}{\sum_{r=1}^{n} d_r} =$$

$\Rightarrow$ our random walk idea only works for small t

$\Rightarrow$ we need to **correct the label assignment when t is large**

POISSON LEARNING

Poisson learning [8]

Consider

$$u_T(x_i) = \mathbb{E}\left[\sum_{t=0}^{T} \frac{1}{d_i} \sum_{j \in \mathcal{L}} \left(y_j - \frac{1}{N} \sum_{r \in \mathcal{L}} y_r \right) \mathbb{1}_{\{B_t^{x_j} = x_i\}}\right].$$

We assign labels $u(x_i) := \lim_{T \to \infty} u_T(x_i)$

- This corrects for large step time behavior (let $\bar{y} = \frac{1}{N} \sum_{r \in \mathcal{L}} y_r$):

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{d_i} \sum_{j \in \mathcal{L}} (y_j - \bar{y}) \mathbb{1}_{\{B_t^{x_j} = x_i\}}\right] &= \frac{1}{d_i} \sum_{j \in \mathcal{L}} y_j \mathbb{P}(\{B_t^{x_j} = x_i\}) - \frac{\bar{y}}{d_i} \sum_{j \in \mathcal{L}} \mathbb{P}(\{B_t^{x_j} = x_i\}) \\
&\approx \frac{1}{d_i} \frac{d_i \sum_{j \in \mathcal{L}} y_j}{\sum_{r=1}^{n} d_r} - \frac{\bar{y}}{d_i} \frac{N d_i}{\sum_{r=1}^{n} d_r} \\
&= \frac{1}{\sum_{r=1}^{n} d_r} (N\bar{y} - N\bar{y}) = 0
\end{aligned}$$

POISSON EQUATION

- The above label assignment u is hard to compute with the random walk definition
- Similarly to Laplace learning, we can show the following

Poisson equation

The function u defined through the random walk formulation is the solution of

$$Lu(x_i) = \sum_{j \in \mathcal{L}} (y_i - \overline{y})\delta_{ij} \text{ for } 1 \leq i \leq n \text{ and such that } \sum_{i=1}^{n} d_i u(x_i) = 0$$

- **Question**: does Poisson learning work at very low label rates?

EMPIRICAL PERFORMANCE OF POISSON LEARNING

Table: MNIST: Average accuracy scores over 100 trials with standard deviation in brackets.

# LABELS PER CLASS	**1**	**2**	**3**	**4**	**5**
LAPLACE/LP [50]	16.1 (6.2)	28.2 (10.3)	42.0 (12.4)	57.8 (12.3)	69.5 (12.2)
NEAREST NEIGHBOR	55.8 (5.1)	65.0 (3.2)	68.9 (3.2)	72.1 (2.8)	74.1 (2.4)
RANDOM WALK [49]	66.4 (5.3)	76.2 (3.3)	80.0 (2.7)	82.8 (2.3)	84.5 (2.0)
MBO [17]	19.4 (6.2)	29.3 (6.9)	40.2 (7.4)	50.7 (6.0)	59.2 (6.0)
VOLUMEMBO [25]	89.9 (7.3)	95.6 (1.9)	96.2 (1.2)	96.6 (0.6)	96.7 (0.6)
WNLL [39]	55.8 (15.2)	82.8 (7.6)	90.5 (3.3)	93.6 (1.5)	94.6 (1.1)
CENTERED KERNEL [30]	19.1 (1.9)	24.2 (2.3)	28.8 (3.4)	32.6 (4.1)	35.6 (4.6)
SPARSE LP [27]	14.0 (5.5)	14.0 (4.0)	14.5 (4.0)	18.0 (5.9)	16.2 (4.2)
P-LAPLACE [16]	72.3 (9.1)	86.5 (3.9)	89.7 (1.6)	90.3 (1.6)	91.9 (1.0)
Poisson	90.2 (4.0)	93.6 (1.6)	94.5 (1.1)	94.9 (0.8)	95.3 (0.7)
PoissonMBO	**96.5 (2.6)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**	**97.2 (0.1)**

ASYMPTOTIC ARGUMENT FOR DEGENERACIES

- Degeneracies occur at very low label rates, i.e. when $|\mathcal{L}| \ll n$

Key idea

We let $n \to \infty$ to see if we can deduce properties of the situation $|\mathcal{L}| \ll n$ in the finite setting

- **Recall**: n is the dataset size, so we assume that we get increasingly more data samples but only **a fixed amount are labeled**

SCALING LAWS IN MACHINE LEARNING

- In modern machine learning, all quantities are large:
  - Model size (e.g. in deep learning: width, depth, parameter count)
  - Dataset size
  - Training time and compute budget
- Scaling laws aim to understand the behavior of algorithms as these quantities **tend to infinity** [22, 23, 44, 11, 40]
- This approach has deep roots:
  - In numerical analysis: mesh refinement, convergence analysis, etc.
  - In statistics: asymptotic consistency, minimax theory, etc.

THEORETICAL VALUE OF SCALING LIMITS

- In terms of modeling, large-scale behavior approximates the infinite limit:

  “Everything is large” $\approx$ “Everything is infinite”

- **Qualitative** or **quantitative** discrete-to-continuum convergence results can make this precise

- Studying **continuum** limits can simplify theoretical analysis

SCALING LAWS INFORM ALGORITHM DESIGN

- Scaling laws yield a **theory-guided approach** to model and algorithm design:
  - How wide, how deep, how much data?
  - Which hyperparameters matter most?
- Enables more **efficient model selection**:
  - Reduce reliance on grid search or brute-force tuning
  - Predict diminishing returns from increasing parameters

$\Rightarrow$ Scaling laws help build models that are **predictable, robust, and cost-effective** at scale

SCALING WEIGHT MODEL

- As $n \to \infty$, we get increasingly more samples in the graph

  $\Rightarrow$ we need a systematic way to define the graph, i.e **the graph weights**

Scaling weight model

Let $\varepsilon_n \to 0$ be a positive sequence and $\eta : [0, \infty) \mapsto [0, \infty)$ be non-increasing. We define

$$w_{\varepsilon_n, ij} = \frac{1}{\varepsilon_n^d} \eta \left(\frac{\|x_i - x_j\|}{\varepsilon_n} \right)$$

- We let $\varepsilon_n \to 0$ (other models are possible):
  - **analytically nice**: discrete difference become derivatives
  - **computationally nice**: we control the number of neighbors/density of W

INFORMAL CONTINUUM LIMIT OF GRAPH DIRICHLET ENERGY

- Consider the simplified case where $\eta(x) = \mathbb{1}_{\{[0,1]\}}(x)$, samples come from the uniform distribution on Ω and let $u : \Omega \mapsto \mathbb{R}$ be smooth with right boundary conditions

$$\frac{1}{n^2\varepsilon_n^2} u^T L u = \frac{1}{n^2\varepsilon_n^2} \sum_{i,j=1}^{n} \frac{1}{\varepsilon_n^d} \mathbb{1}_{\{|x_i - x_j| \le \varepsilon_n\}} \left(u(x_i) - u(x_j)\right)^2$$

$$\xrightarrow{n\to\infty} \frac{1}{\varepsilon_n^{2+d}} \int_\Omega \int_\Omega \mathbb{1}_{\{|y-x|\le\varepsilon_n\}} (u(y) - u(x))^2 \, \mathrm{d}y \mathrm{d}x$$

$$= \frac{1}{\varepsilon_n^2} \int_\Omega \int_{\{|z|\le 1\}} (u(x + \varepsilon_n z) - u(x))^2 \, \mathrm{d}z \mathrm{d}x$$

$$= \int_\Omega |\nabla u(x)|^2 \int_{\{|z|\le 1\}} z^2 \, \mathrm{d}z \mathrm{d}x + o(\varepsilon_n)$$

$$\xrightarrow{\varepsilon_n \to 0} C \int_\Omega |\nabla u(x)|^2 \, \mathrm{d}x = C \langle u, \Delta u \rangle_{\mathrm{L}^2(\Omega)} \quad \text{where } \Delta \text{ is Laplace operator}$$

CONTINUUM LAPLACE LEARNING

- The analogue continuum problem to Laplace learning is to find

$$u \in \operatorname*{argmin}_{v:\Omega \mapsto \mathbb{R}} \int_{\Omega} |\nabla v(x)|^2 \, \mathrm{d}\mu(x) \quad \text{such that } v(x_i) = y_i \text{ for } 1 \leq i \leq N$$

Sobolev Spaces

Let $\Omega \subset \mathbb{R}^d$ be an open set. For $k \in \mathbb{N}_0$ and $1 \leq p \leq \infty$, the Sobolev space $\mathrm{W}^{k,p}(\Omega)$ is defined as

$$\mathrm{W}^{k,p}(\Omega) := \{u \in \mathrm{L}^p(\Omega) \mid D^\alpha u \in \mathrm{L}^p(\Omega) \text{ for all multi-indices } \alpha \text{ with } |\alpha| \leq k\},$$

where $D^\alpha u$ denotes the weak derivative of u of order α.

- Every function u for which $\int_\Omega |\nabla u(x)|^2 \, \mathrm{d}x < \infty$ is in $\mathrm{W}^{1,2}(\Omega)$

SOBOLEV EMBEDDINGS

- We know that the solution u to the continuum function Laplace learning problem is in $\mathrm{W}^{1,2}$
- **Note**: u also satisfies pointwise constraints $u(x_i) = y_i$ for $1 \leq i \leq N$

$\Rightarrow$ u must be at least continuous

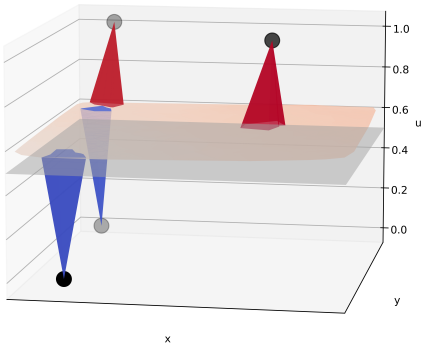
Sobolev embeddings

If $k > \frac{d}{p}$, then

$$W^{k,p}(\Omega) \hookrightarrow C^{0,\alpha}(\overline{\Omega}), \quad \text{for some } \alpha > 0.$$

$\Rightarrow$ Functions in $\mathrm{W}^{1,2}$ are only continuous if $1 > \frac{d}{2} \Leftrightarrow d < 2$

LAPLACE LEARNING IS FUNDAMENTALLY ILL-POSED



- In most applications, $d \gg 1$

  $\Rightarrow$ minimizers of $\int_\Omega |\nabla v(x)|^2 \, \mathrm{d}\mu(x)$ will not be continuous

  $\Rightarrow$ solutions of continuum learning will only be minimizers of $\int_\Omega |\nabla v(x)|^2 \, \mathrm{d}\mu(x)$, i.e. **constants**

p-LAPLACE LEARNING

- **Question**: can we design a discrete graph-learning algorithm which has a continuum limit allowing for better **well-posedness assumptions** than Laplace learning?

p-Laplace learning [13]

p-Laplace learning aims to find:

$$u = \operatorname*{argmin}_{v \in \mathbb{R}^n} \frac{1}{n^2 \varepsilon_n^p} \sum_{i,j=1}^{n} w_{\varepsilon_n, ij} |u_i - u_j|^p \quad \text{such that } u_i = y_i \text{ for } 1 \leq i \leq N$$

- We can show that the continuum p-Laplacian learning problem is to find

$$u \in \operatorname*{argmin}_{v : \mapsto \mathbb{R}} \int_{\Omega} |\nabla v(x)|^p \, \mathrm{d}\mu(x) \quad \text{such that } v(x_i) = y_i \text{ for } 1 \leq i \leq N$$

$\Rightarrow u \in \mathrm{W}^{1,p}$ so as long as $p > d$, u should be continous

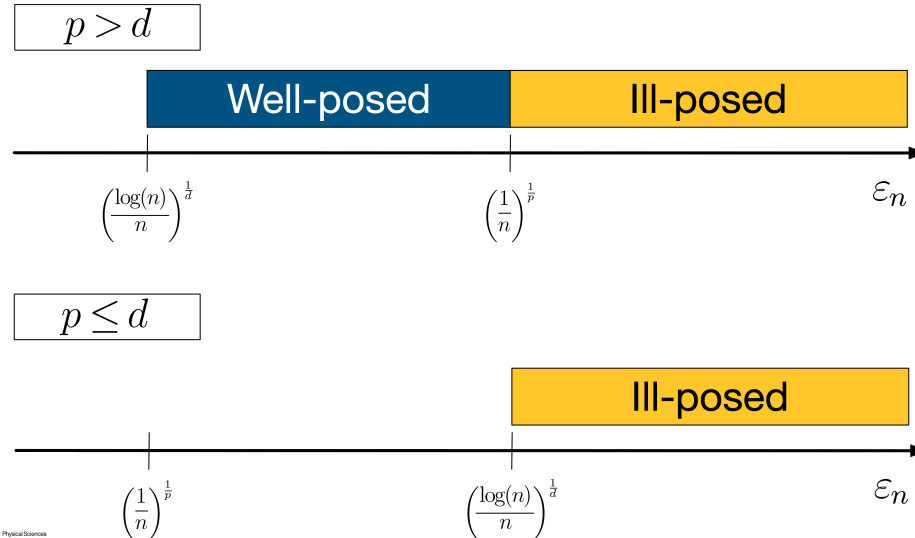
WELL/ILL-POSEDNESS CHARACTERIZATION I

- **Sobolev embedding intuition** is useful but not sufficient to understand when p-Laplace learning is well/ill-posed in the continuum

- True characterization depends on scaling of $\varepsilon_n \to 0$ [40], i.e. **graph construction**

  $\Rightarrow$ discrete-to-continuum analysis helps us to choose the right hyperparameter ε_n to ensure correct behaviour of learning problem

WELL/ILL-POSEDNESS CHARACTERIZATION II



WELL/ILL-POSEDNESS CHARACTERIZATION III

- Geometric interpretation of bounds:
  - $\varepsilon_n \gg \left(\frac{\log(n)}{n}\right)^{1/d} \Rightarrow$ graph should be **at least connected**

    $\Rightarrow$ we want to capture the geometry of the samples in a meaningful manner
  - $\varepsilon_n \ll \left(\frac{1}{n}\right)^{1/p} \Rightarrow$ graph should not be **too connected**
- For $p \neq 2$, solutions to p-Laplace learning are not exact anymore
- **Question**: can we have another higher-regularization method who leads to exact solutions?

FRACTIONAL LAPLACE LEARNING

Fractional Laplace learning [46]

For $s > 0$, fractional Laplace learning aims to find

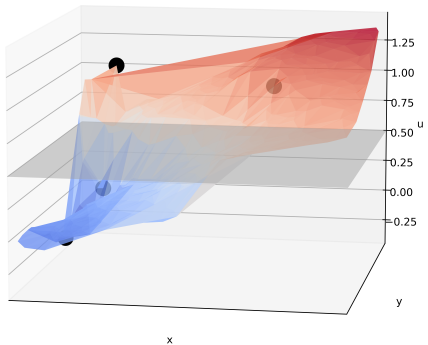
$$u = \underset{u \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{k=1}^{n} \lambda_k^s u_k^2 \quad \text{such that } u_{\mathcal{L}} = y$$

- Fractional Laplace learning has a quadratic energy: $u^T L^s u$

  $\Rightarrow$ efficient numerical solvers

WELL/ILL-POSEDNESS CHARACTERIZATION IV

- We can show that solution to the continuum fractional Laplace problem are in $\mathrm{W}^{s,2}$
- **Similar characterization** to p-Laplace learning can be proven for well/ill-posedness of fractional Laplace learning



SPECTRAL KERNEL DESIGN

- We can generalize fractional Laplace learning to

Spectral kernel design [41]

The spectral kernel design method aims to find

$$u = \operatorname*{argmin}_{u \in \mathbb{R}^n} \sum_{k=1}^{n} r(\lambda_k) u_k^2 \quad \text{such that } u_{\mathcal{L}} = y$$

for some function r

- Examples:
  - $r(x) = (1+x)^s$: regularized fractional Laplace learning
  - $r(x) = e^{\sigma x}$: diffusion process
  - $r(x) = P(x)$ where $P(x)$ is a polynomial: this is the filter also used in the original GCN paper
  - PageRank, Graph wavelets, etc.

GRAPH REGULARITY AND OUT-OF-SAMPLE EXTENSIONS

- **Idea**: we want a smooth labelling function and impose **smoothness on graph space**
- **Drawback**: our labelling function is only defined on the graph and we cannot do **out-of-sample** evaluations
- **Question**: can we define a function that is smooth on the whole of Ω but also consider the graph geometry?

  $\Rightarrow$ Manifold regularization

SUPERVISED LEARNING

- Linear regression aims to find:

$$w^* = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{N} (w^\top x_i) - y_i)^2$$

- In order to **avoid overfitting**, it is also common to consider the ridge regression problem:

$$w^* = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{N} (w^\top x_i) - y_i)^2 + \lambda \|w\|_2^2$$

- **Note**: This constrains our prediction f to be of the form $f(x) = w^T x$, i.e. a linear function

  $\Rightarrow$ for a better fit, we want to consider a more general class of functions $\mathcal{H}$:

$$f^\star = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2$$

- **Question**: How do we pick $\mathcal{H}$ so that the problem remains tractable?

KERNEL RIDGE REGRESSION

- Kernel ridge regression provides a structured way to choose the class of functions $\mathcal{H}$
- In particular, $\mathcal{H}$ will be parametrized by a **kernel function**
- Even when $\mathcal{H}$ is infinite-dimensional, we can transform the problem into a finite-dimensional one by the **representer theorem**

REPRODUCING KERNEL HILBERT SPACES I

RKHS

Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric, positive semi-definite function, i.e. such that for each n, $\{x_i\}_{i=1}^n \subseteq \mathcal{X}$ and $\{\alpha_i\}_{i=1}^n \subseteq \mathbb{R}$ we have $\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j) \geq 0$. The RKHS $\mathcal{H}_K$ associated with K is defined as the unique Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ such that:

1. For every $x \in \mathcal{X}$, the function $K(x, \cdot) \in \mathcal{H}_K$
2. For every $f \in \mathcal{H}_K$ and every $x \in \mathcal{X}$, the *reproducing property* holds: $f(x) = \langle f, K(x, \cdot) \rangle_{\mathcal{H}_K}$
3. The space $\mathcal{H}_K$ is the closure of the set of all finite linear combinations of kernel sections:

$$\left\{ f = \sum_{i=1}^{n} \alpha_i K(x_i, \cdot) \,\middle|\, n \in \mathbb{N},\ \alpha_i \in \mathbb{R},\ x_i \in \mathcal{X} \right\},$$

with inner product defined by:
$\left\langle \sum_{i=1}^n \alpha_i K(x_i, \cdot), \sum_{j=1}^m \beta_j K(x_j, \cdot) \right\rangle = \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j K(x_i, x_j)$

REPRODUCING KERNEL HILBERT SPACES II

- For $f \in \mathcal{H}$, we have

$$|f(x) - f(y)| = |\langle f, K(x,\cdot) - K(y,\cdot)\rangle_{\mathcal{H}_K}| \leq \|f\|_{\mathcal{H}_K} \|K(x,\cdot) - K(y,\cdot)\|_{\mathcal{H}_K}$$

  $\Rightarrow$ The smaller $\|f\|_{\mathcal{H}_K}$, the smoother f

- **Question**: what is $\|f\|_{\mathcal{H}_K}$?

- Every symmetric, positive semi-definite function K is the kernel of a RKHS, so the norm will depend on K

  $\Rightarrow$ we give one example such that $\|f\|_{\mathcal{H}_K}$ is the Sobolev norm

KERNEL THROUGH DIFFERENTIAL OPERATORS

Green function

Let D be a differential operator. Consider the differential equation

$$Dg = f,$$

if there exists a function $k(x, y)$ such that $g(x) = \int k(x, y)f(y)\,dy$, then k is called the Green function of D

- The following theorem links the Green function of an operator to a RKHS.

RKHS derived from a differential operator [31]

Let D be a differential operator defined on some class of functions $\mathcal{H}$ such that, endowed with the inner product

$$\langle f, g\rangle_{\mathcal{H}} = \langle Df, Dg\rangle_{L^2},$$

it is a Hilbert space. Then, $\mathcal{H}$ is the RKHS associated with the kernel K which is the Green function of D.

FOURIER DEFINITION OF SOBOLEV SPACES

- **Recall**:
  - $f \in \mathrm{W}^{s,2} \Leftrightarrow \sum_{|\alpha| \leq m} \|D^\alpha f\|_{L^2}^2 < \infty$
  - Fourier transform of a weak derivative is:
    $$\widehat{D^\alpha f}(\xi) = (2\pi i \xi)^\alpha \hat{f}(\xi).$$
  - Plancherel Theorem: $\|f\|_{\mathrm{L}^2} = \|\hat{f}\|_{\mathrm{L}^2}$

- Sobolev norm via Plancherel:
  $$\sum_{|\alpha| \leq m} \|D^\alpha f\|_{L^2}^2 = \sum_{|\alpha| \leq m} \int |\xi^\alpha|^2 |\hat{f}(\xi)|^2 d\xi \sim \int (1 + \|\xi\|^2)^m |\hat{f}(\xi)|^2 d\xi.$$

Fourier definition of $\mathrm{W}^{s,2}$

For $s \in \mathbb{R}$, we have $\mathrm{W}^{s,2}(\mathbb{R}^d) = \left\{ f \in L^2 \,\middle|\, \int_{\mathbb{R}^d} (1 + \|\xi\|^2)^s |\hat{f}(\xi)|^2 d\xi < \infty \right\}$

SOBOLEV SPACES VIA FRACTIONAL OPERATORS

- $f \in \mathcal{S}(\mathbb{R}^d)$, the Fourier transform of the Laplacian is:

$$\widehat{-\Delta f}(\xi) = \|\xi\|^2 \hat{f}(\xi)$$

$\Rightarrow$ We define the Fourier transform of the fractional operators $(I - \Delta)^{s/2}$ as

$$\widehat{(I - \Delta)^{s/2} f}(\xi) = (1 + \|\xi\|^2)^{s/2} \hat{f}(\xi)$$

Differential operator definition of Sobolev spaces

We define the Sobolev norm via

$$\|f\|^2_{\mathrm{W}^{s,2}(\mathbb{R}^d)} := \|(I - \Delta)^{s/2} f\|^2_{L^2} = \int_{\mathbb{R}^d} (1 + \|\xi\|^2)^s |\hat{f}(\xi)|^2 \, d\xi$$

and have the inner product

$$\langle f, g \rangle_{\mathrm{W}^{s,2}} = \langle (I - \Delta)^{s/2} f, (I - \Delta)^{s/2} g \rangle_{\mathrm{L}^2}$$

$\mathrm{W}^{s,2}$ IS A RKHS

- We know that $\mathrm{W}^{s,2}$ is a Hilbert space
- Equivalent RKHS definition

Evaluation operator

$\mathcal{H}$ Hilbert space is RKHS if $E_x : \mathcal{H} \mapsto \mathbb{R}$, $E_x(f) = f(x)$ is a bounded functional

$\Rightarrow$ we know by Sobolev embeddings, that the evaluation operator bounded on $\mathrm{W}^{s,2}$ if $s > \frac{d}{2}$

- By above theorem

$\mathrm{W}^{s,2}$ is a RKHS [14]

For $s > \frac{d}{2}$, $\mathrm{W}^{s,2}$ is a RHKS with kernel K being the Green function of $(I - \Delta)^s$ and $\|f\|_{\mathcal{H}_K}$ is equivalent to Sobolev norm

MATÉRN KERNELS

Matérn kernels

The Matérn kernel is a family of positive-definite kernels parameterized by a smoothness parameter $\nu > 0$:

$$K_\nu(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell} \right)^\nu K_\nu^* \left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell} \right),$$

where K_ν^* is the modified Bessel function of the second kind, and $\ell > 0$ is the length scale

- K_ν is the **Green's function of the fractional operator** $(I - \Delta)^{\nu + \frac{d}{2}}$ on $\mathbb{R}^d$

$\Rightarrow$ Picking the kernel $K_{s-d/2}$ and having $\|f\|_{\mathcal{H}_{K_{s-d/2}}} < \infty$ is equivalent functions being in $\mathrm{W}^{s,2}$

SUPERVISED LEARNING IN RKHS

Kernel Ridge Regression

Given labeled data $\{(x_i, y_i)\}_{i=1}^N \subseteq \Omega$, and a reproducing kernel K, we aim to find

$$f^\star = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}_K}^2$$

- **Limitation:** This formulation does not exploit the geometry of the data distribution, i.e. we just try to find a good fit (the second term avoids overfitting)
- **Note**: in contrast to graph learning, f^* is defined on the whole of Ω
- **Goal**: Incorporate geometric prior into learning, i.e. go from supervised to semi-supervised learning

THE MANIFOLD HYPOTHESIS

Manifold regularization [3]

We add a second regularization term penalizing f based on the geometry of the data distribution:

$$f^\star = \underset{f \in \mathcal{H}_k}{\operatorname{argmin}} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda_{\text{ext}} \|f\|^2_{\mathcal{H}_k} + \lambda_{\text{int}} \|f\|^2_I$$

- Very common assumption in machine learning is

Manifold hypothesis

Real-world data lies near a low-dimensional manifold $\mathcal{M} \subset \mathbb{R}^d$

- **Examples**: images of the same object under different poses, handwritten digits
  $\Rightarrow$ we want f to be smooth on $\mathcal{M}$: $\|f\|^2_I = \|\nabla_{\mathcal{M}} f\|^2_I$

DISCRETIZING THE INTRINSIC NORM

- Often, we don't know what $\mathcal{M}$ is but we have a lot of unlabelled samples

  $\Rightarrow$ Construct the graph and penalize smoothness through the graph dirichlet energy

- Manifold regularization becomes:

$$f^\star = \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda_{\text{ext}} \|f\|^2_{\mathcal{H}_k} + \lambda_{\text{int}} f^\top L f.$$

- **Conclusion**:
  - $f*$ is regularized on the whole ambient space Ω and guaranteed to be in $\mathcal{H}_K \Rightarrow$ we can do **out-of-sample evaluations**
  - f takes into account the **geometry of the data** through the regularization on the graph

NUMERICAL CONSIDERATIONS I

- The manifold regularization is a minimization problem on the possibly infinite-dimensional Hilbert space $\mathcal{H}_K$
- **Question**: how do we compute f^*?
- Similarly to RKHS ridge regression theory, we have

Representer theorem

f^* can be written as $f^*(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$

NUMERICAL CONSIDERATIONS II

- Let $k = \{K(x_i, x_j)\}_{i,j=1}^{n}$ and $k_{\mathcal{L}} = \{K(x_i, x_j)\}_{i,j=1}^{i=N,j=n}$
- By the definition of the RKHS inner product, we have:

$$\|f\|_{\mathcal{H}_k}^2 = \langle \sum_{i=1}^{n} \alpha_i K(x_i, x), \sum_{i=1}^{n} \alpha_i K(x_i, x) \rangle = \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = \alpha^T k \alpha$$

- We have $(f(x_1), \ldots, f(x_n)) = k\alpha$ so

$$f^\top L f = \alpha^T k L k \alpha$$

- We have $\sum_{i=1}^{N} (f(x_i) - y_i)^2 = \|k_{\mathcal{L}}\alpha - y\|_2^2$

NUMERICAL CONSIDERATIONS III

Finite-dimensional manifold regularization

Manifold regularization can be written as the following finite-dimensional optimization problem

$$\alpha^\star = \underset{\alpha \in \mathbb{R}^{N+|\mathcal{U}|}}{\operatorname{argmin}} \; \|k_{\mathcal{L}}\alpha - y\|^2 + \lambda_{\text{ext}}\alpha^\top k\alpha + \lambda_{\text{int}}\alpha^\top kLk\alpha$$

SCALING THE NUMBER OF LABELS IN SSL

- Until now, we have always considered that $|\mathcal{L}|$ is fixed as $n \to \infty$
- Let β denote the labelling rate, i.e. the ratio $\frac{|\mathcal{L}|}{n}$
- We consider the case when $\beta_n \to 0$, i.e. the situation when obtaining new labelled data is much more difficult than obtaining new unlabelled data
- **Question**: what is the smallest necessary number of labels to ensure well-posedness?

Scaling of β_n [10]

It can be shown that

- if $\beta_n \ll \varepsilon_n^2$, then Laplace learning is ill-posed
- if $\beta_n \gg \varepsilon_n^2$, then Laplace learning is well-posed and we get quantitative rates of convergence between discrete and continuum solution

TAKEAWAY: SEMI-SUPERVISED LEARNING

- **Goal**: Find a labelling on the whole graph given only a subset of label
- **Laplace learning**:
  - minimizes smoothness on the graph
  - can be shown to be degenerate in very low labelling rates
- **Extensions**:
  - **Poisson learning** motivated by random walk formulation
  - **Higher order methods** motivated by scaling laws
- **Out-of-sample evaluations**:
  - **Manifold regularization** and RKHS

SUPERVISED LEARNING

SUPERVISED LEARNING

- Suppose that we are given noisy labels $y_i = g(x_i) + \xi_i$ where $\xi_i \in \mathbb{R}$ are independent and identically distributed sub-Gaussian centered noise
- We consider minimizers $u_{n,\tau}^{(\mathbf{y}_n)}$ of

$$\frac{1}{n}\sum_{i=1}^{n} |v_n(x_i) - y_i|^2 + \tau v_n^T L^s v_n$$

- The function $u_{n,\tau}^{(\mathbf{y}_n)}$ is the best approximation of g on the graph which is regularized using the fractional Laplace energy

RATES OF CONVERGENCE

- One can obtain **rates of convergence** between $u_{n,\tau}^{(\mathbf{y}_n)}$ and g [18]

$\Rightarrow$ This is a quantitative result while the previous continuum limit results are only asymptotic/qualitative

- **Question**: why use graphs in supervised learning?

GRAPHS ARE CONVENIENT NUMERICAL SCHEMES ON COMPLICATED DOMAINS

Nonlocal characterization of $\mathrm{W}^{1,p}$ [6]

For $f \in \mathrm{L}^p$ and a family of mollifiers $(\rho_\varepsilon)_\varepsilon$,

$$\lim_{\varepsilon \to 0} \int_\Omega \int_\Omega \frac{|f(x) - f(y)|^p}{|x-y|^p} \rho_\varepsilon(|x-y|)\, dy\, dx = C \|f\|^p_{\mathrm{W}^{1,p}}$$

- By discretizing the above formula, we recover p-Laplace learning

  $\Rightarrow$ graphs can be used to approximate Sobolev semi-norms/regularizing terms used in inverse problems

- **Advantage**: compared to other numerical schemes, graphs are mesh-free and only rely on the possibility of sampling points

  $\Rightarrow$ graphs can be used to solve inverse problems on complicated manifolds [24]

BAYESIAN FORMULATIONS AND ACTIVE LEARNING

FROM VARIATIONAL TO BAYESIAN LEARNING

- For now, we have considered the variational problem

$$\arg\min_{v} \; J(v) + \Psi(v, y)$$

  where

  - $J(v)$: regularizer (e.g., smoothness through graph-Dirichlet energy)
  - $\Psi(v, y)$: data-fidelity term (e.g. L^2-loss in supervised learning)

- Solving such a problem yields a single estimator v^*, but offers no confidence in this prediction

- Bayesian methods replace point estimates with **distributions over functions**

- This allows us to quantify **uncertainty**, not just prediction

BAYESIAN PRINCIPLES

- **Incorporate prior knowledge**: Learning problems come with known structure (e.g., smoothness because of homophily) and we define a prior $\mu_0(v)$ encoding such assumptions

  $\Rightarrow$ we want to bias the solution toward plausible functions

- **Data refines belief**: The likelihood $\mu_1(y \mid v)$ measures how well a candidate function v explains the observed data

- **Posterior belief**: The posterior $\mu_2(v \mid y)$ is our updated belief about v after seeing data; it quantifies how likely each candidate function v is, given y

- **Question:** Priors and likelihoods are often easy to model—how does one compute the posterior?

BAYES THEOREM

Bayes Theorem

Given a prior distribution $\mu_0(v)$ over functions v, and a likelihood $\mu_1(y \mid v)$ modeling the probability of observing labels y given v, Bayes' theorem gives the posterior:

$$\mu_2(v \mid y) = \frac{1}{Z}\,\mu_1(y \mid v)\,\mu_0(v)$$

where $Z = \int \mu_1(y \mid v)\mu_0(v)\,dv$ is the normalization constant making it a probability distribution

- The posterior is a probability distribution over functions — that is, over their function values:
  - Each draw from the posterior corresponds to an entire function v
  - For every point x in the domain, we can compute the mean and variance of $v(x)$

    $\Rightarrow$ Instead of a **single estimate** $v^*(x)$, we obtain a **distribution** over possible values of $v(x)$

- **Question**: What is the most likely function v?

MAXIMUM A POSTERIORI ESTIMATOR

MAP estimator

The Maximum A Posteriori (MAP) estimator is the most likely function v under the posterior distribution $\mu_2(v \mid y)$:

$$\mathrm{MAP} = \operatorname*{argmax}_v \mu_2(v \mid y)$$

- Bayesian prior, likelihood and posterior in practice
  - Prior: $\mu_0(v) \propto e^{-J(v)}$
  - Likelihood: $\mu_1(y|v) \propto e^{-\Psi(v,y)}$
  - Posterior: $\mu_2(v|y) \propto e^{-J(v)-\Psi(v,y)}$

$\Rightarrow$ We have

$$\mathrm{MAP} = \operatorname*{argmax}_v e^{-J(v)-\Psi(v,y)} = \operatorname*{argmax}_v -J(v) - \Psi(v,y) = \operatorname*{argmin}_v J(v) + \Psi(v,y)$$

SAMPLING FROM GAUSSIAN POSTERIORS

- In order to leverage the Bayesian formulation, we should be able to **sample the posterior efficiently**

- Common methods: Markov Chain Monte Carlo, Langevin dynamics (used in diffusion models), variational inference etc.

- Example: Preconditioned Crank–Nicolson MCMC [5]
  - Robust to small noise and large datasets
  - Only requires:
    - Prior samples: if prior is Gaussian, this can efficiently be obtained through Karhunen–Loève expansion
    - Evaluation of $\Psi(v, y)$

BAYESIAN FORMULATION ON GRAPHS

Graph Priors

On graphs, we often consider the quadratic form

$$J(v) = \langle v, (\lambda \,\mathrm{Id} + L_n)^s v \rangle_n$$

which leads to a Gaussian prior $\mu_0(v) = \mathcal{N}(0, (\lambda \,\mathrm{Id} + L_n)^{-s})$

- Consider likelihood with hard constraints $\Psi(y, v) = \begin{cases} 0 & \text{if } v_{\mathcal{L}} = y \\ +\infty & \text{else} \end{cases}$

$\Rightarrow$ Posterior is proportional to

$$\text{Gaussian on } v \cdot e^{-\Psi(y,v)} = \text{Gaussian on } v \text{ with constraint } \{v_{\mathcal{L}} = y\}$$

CONDITIONAL DISTRIBUTION OF GAUSSIAN

Conditional Distribution of a Multivariate Normal

Let

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right)$$

Then the conditional distribution of Y given $X = x$ is:

$$Y \mid X = x \sim \mathcal{N}\left(\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(x - \mu_X),\ \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\right)$$

$\Rightarrow$ If we have $v = [v_{\mathcal{L}}, v_{\mathcal{U}}]^T \sim \mathcal{N}(0, Q^{-1})$ (here Q is $(\lambda \operatorname{Id} + L)$ or $(\lambda \operatorname{Id} + L)^s$) we can show

$$v_{\mathcal{U}} \mid v_{\mathcal{L}} = y \sim \mathcal{N}(-Q_{\mathcal{U}\mathcal{U}}^{-1} Q_{\mathcal{U}\mathcal{L}} \cdot y, Q_{\mathcal{U}\mathcal{U}}^{-1})$$

$\Rightarrow$ **Posterior mean is the solution to Laplace learning**

UNCERTAINTY IN LABELLING

- We use the **posterior variance** to compute the uncertainty of the labelling function [5]

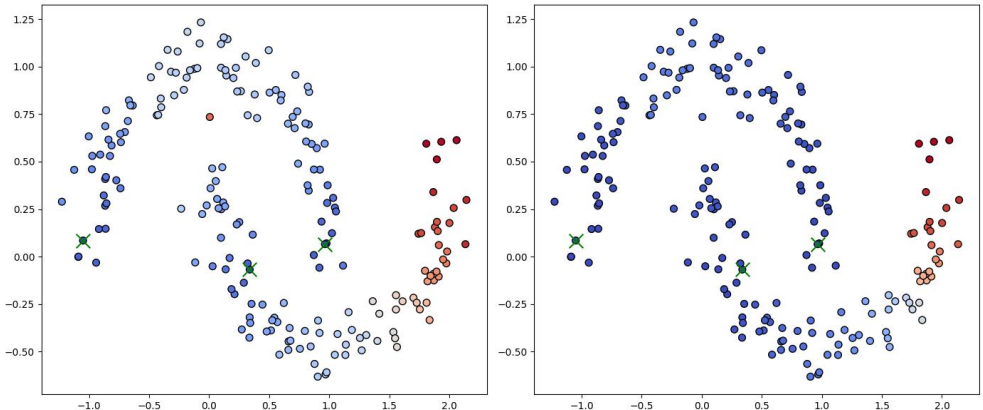


Figure: Left: Posterior variance using $(\lambda \mathrm{Id} + L)$. Right: Posterior variance using $(\lambda \mathrm{Id} + L)^2$

ACTIVE LEARNING

WHAT IS ACTIVE LEARNING?

- **Active learning** is a framework for efficient learning when labeled data is scarce or expensive (e.g. medical diagnosis)
- Instead of passively using a random training set, the algorithm **actively selects the most informative data points** to label
- **Goal**: achieve high accuracy with as few labeled examples as possible

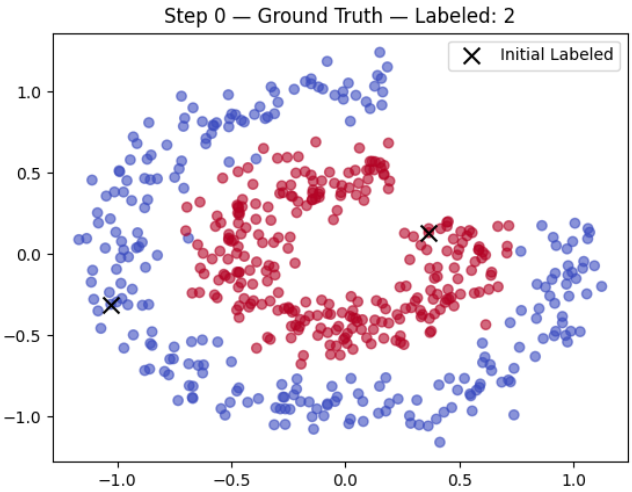
COMPONENTS OF ACTIVE LEARNING

- **Learner:** model trained on the current labeled set $\mathcal{L}$
- **Oracle:** source of ground truth labels (e.g. human annotator, simulation)
- **Acquisition Function:** chooses which unlabeled point to label next, based on some notion of utility

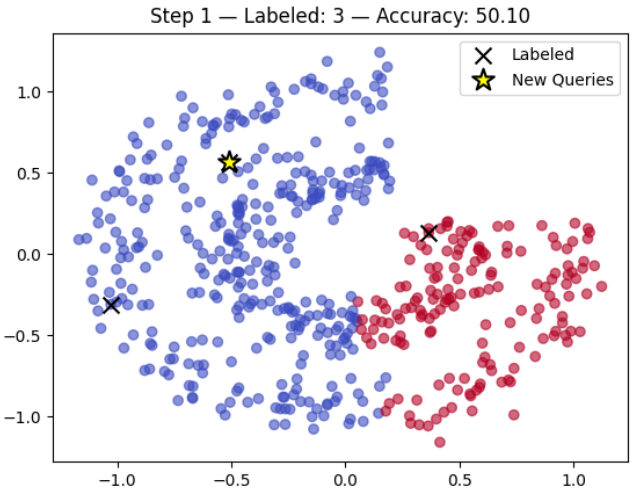
THE ACTIVE LEARNING LOOP

1. Start with a small labeled set $\mathcal{L} \subset \mathcal{X} \times \mathcal{Y}$, and unlabeled pool $\mathcal{U} \subset \mathcal{X}$
2. Train the model on $\mathcal{L}$
3. Use acquisition function to pick point $x^* \in \mathcal{U}$
4. Query oracle for y^*, add (x^*, y^*) to $\mathcal{L}$
5. Repeat until labeling budget is exhausted

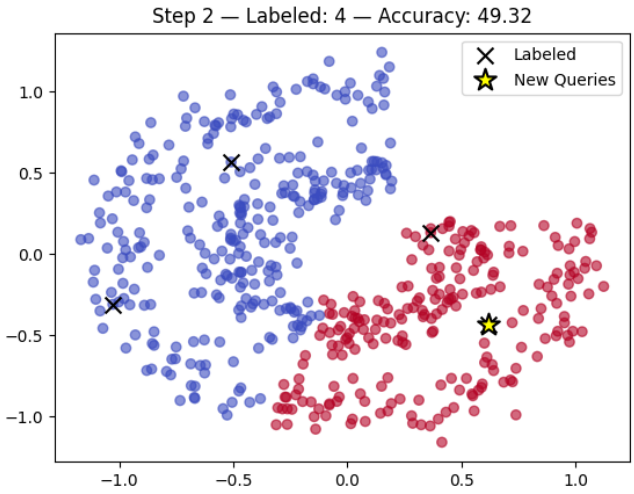
ACTIVE LEARNING EXAMPLE



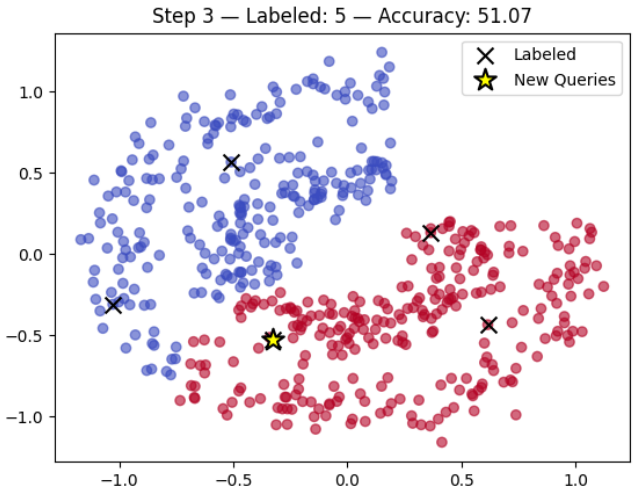
ACTIVE LEARNING EXAMPLE



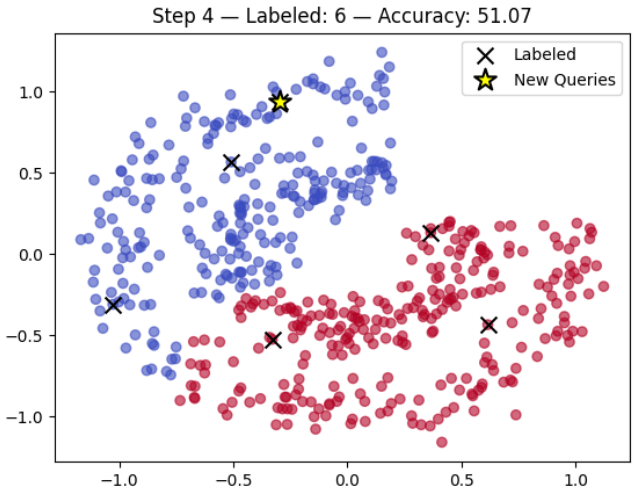
ACTIVE LEARNING EXAMPLE



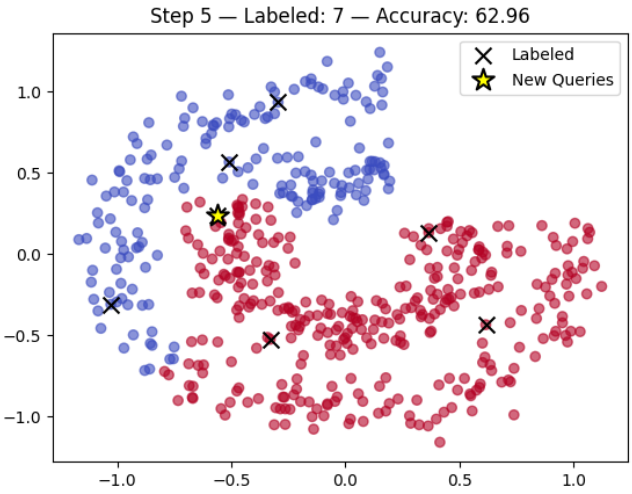
ACTIVE LEARNING EXAMPLE



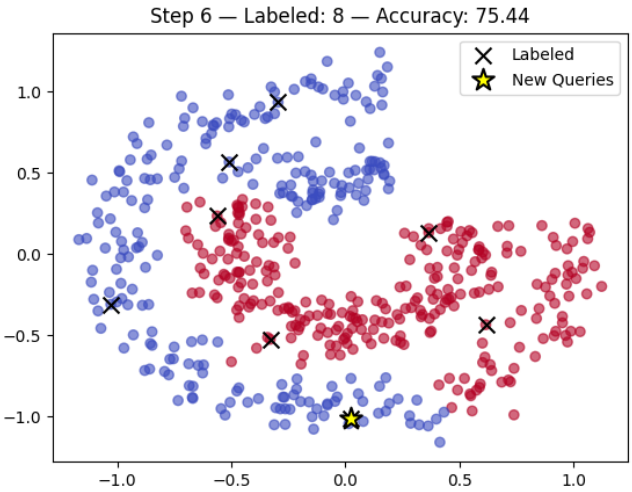
ACTIVE LEARNING EXAMPLE



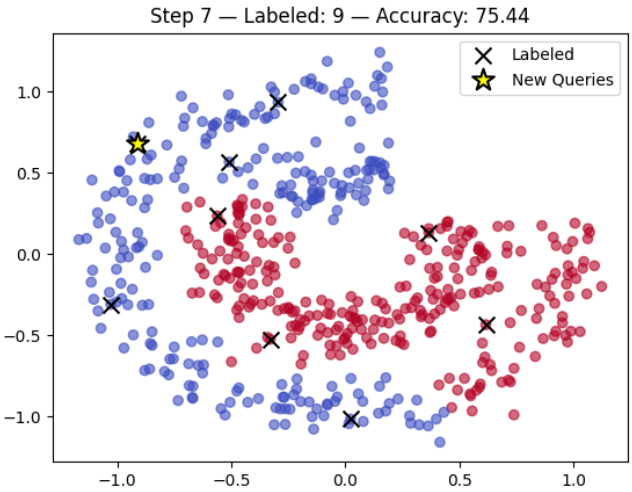
ACTIVE LEARNING EXAMPLE



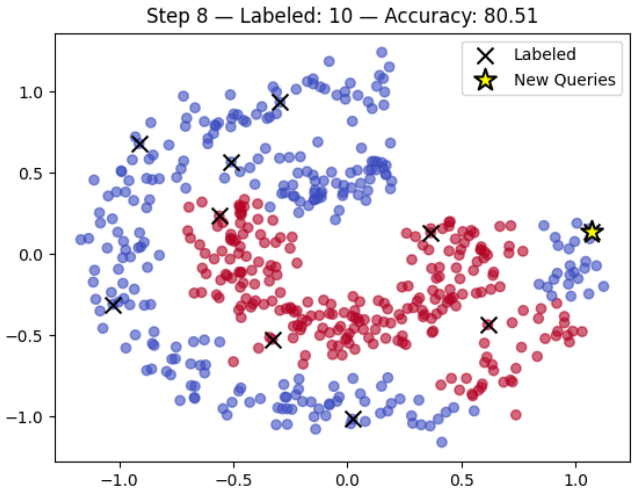
ACTIVE LEARNING EXAMPLE



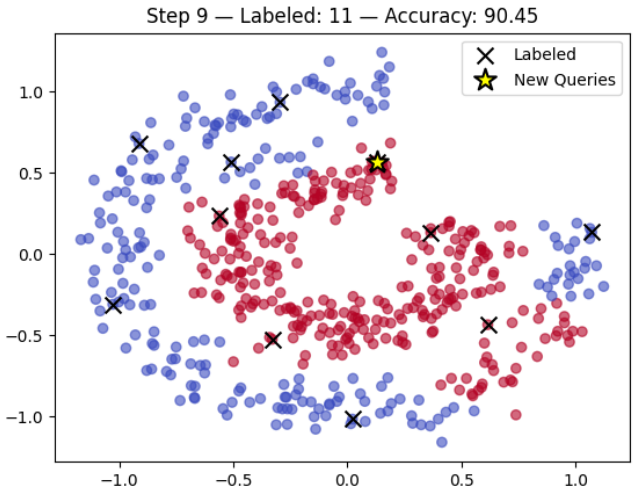
ACTIVE LEARNING EXAMPLE



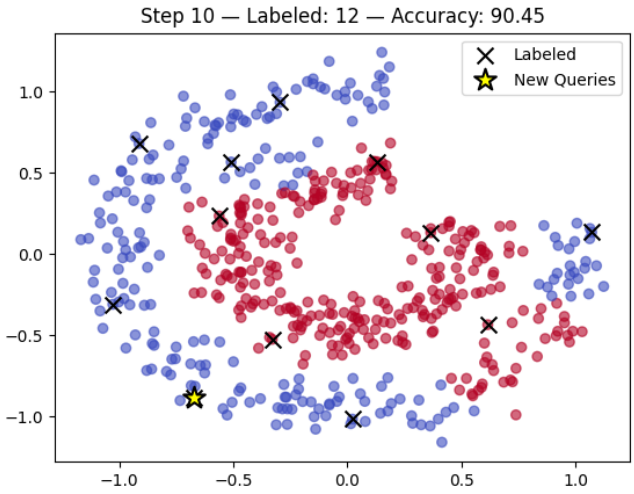
ACTIVE LEARNING EXAMPLE



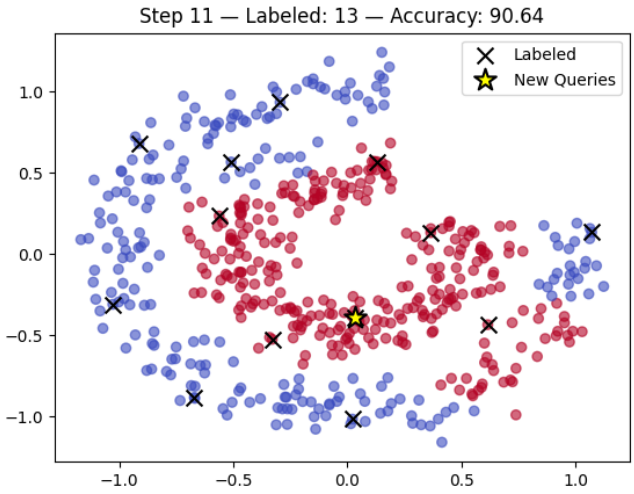
ACTIVE LEARNING EXAMPLE



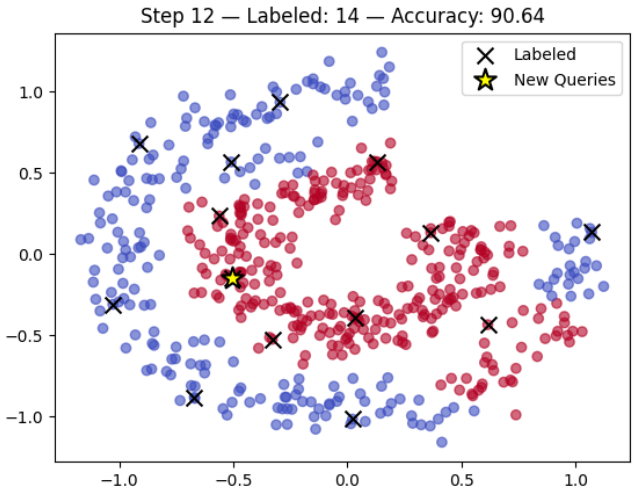
ACTIVE LEARNING EXAMPLE



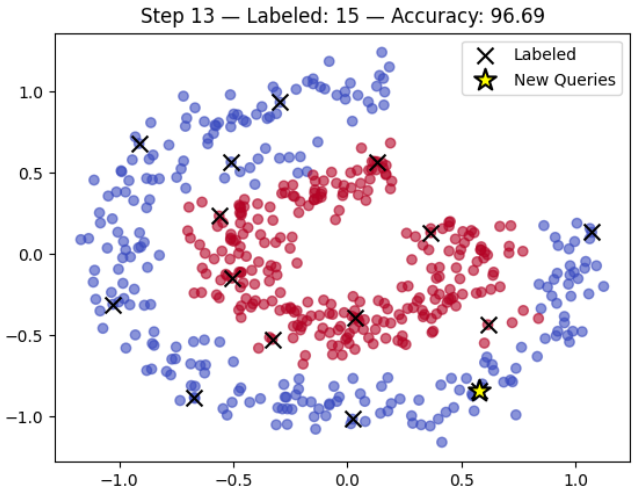
ACTIVE LEARNING EXAMPLE



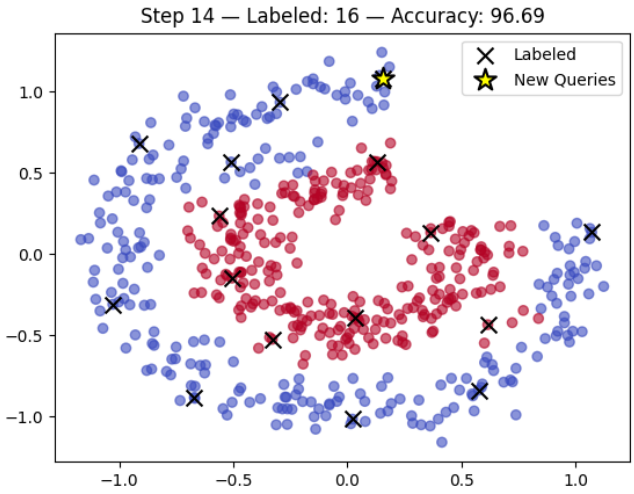
ACTIVE LEARNING EXAMPLE



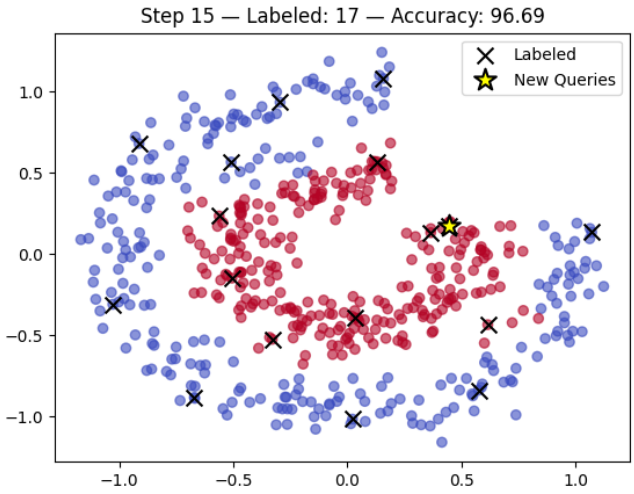
ACTIVE LEARNING EXAMPLE



ACTIVE LEARNING EXAMPLE



ACTIVE LEARNING EXAMPLE



ACQUISITION STRATEGIES

Trade-off

Balance exploration (getting representative labels of the whole dataset) and exploitation (getting labels that are useful in maximizing the performance)

- Most acquisition functions are based on the Bayesian formulation of graph learning:
  - **Uncertainty Sampling:** query where the model is least confident [36]
  - **Expected Model Change:** pick point that would cause the largest update to the model (**Bayesian based**) [33]
  - **Variance Reduction:** select point that reduces posterior uncertainty (**Bayesian based**)
- Some methods also try to explicitly balance exploration and exploitation by adding terms in the model/acquisition function [32]

V-OPTIMALITY CRITERION

- Given a Gaussian posterior $v_{\mathcal{U}} \mid v_{\mathcal{L}} = y \sim \mathcal{N}(-Q_{\mathcal{U}\mathcal{U}}^{-1} Q_{\mathcal{U}\mathcal{L}} \cdot y, Q_{\mathcal{U}\mathcal{U}}^{-1})$, uncertainty at each node is captured by the diagonal entries $Q_{\mathcal{U}\mathcal{U}}^{-1}$.

V-Opt [26]

The V-opt criterion chooses new query point x_h with

$$h = \operatorname*{argmin}_{k \in \mathcal{U}} \operatorname{Tr}((Q_{\mathcal{U}\setminus\{k\}\mathcal{U}\setminus\{k\}})^{-1})$$

- This selects query points that **minimize the posterior variance** over all unlabeled nodes
- **Question**: how does one compute $\operatorname{Tr}((Q_{\mathcal{U}\setminus\{k\}\mathcal{U}\setminus\{k\}})^{-1})$ efficiently?

EFFICIENT V-OPT COMPUTATION VIA LOW-RANK UPDATE I

- It can be shown that

Inverse of matrix with one row/column removed

$$(Q_{V\setminus\{k\}V\setminus\{k\}})^{-1} = (Q_{VV}^{-1})_{-k,-k} - \frac{(Q_{VV}^{-1})_{-k,k}\,(Q_{VV}^{-1})_{k,-k}}{(Q_{VV}^{-1})_{kk}}$$

- Let $C^V = Q_V^{-1} \in \mathbb{R}^{n\times n}$ be the posterior covariance matrix over all nodes V and set

$$C^{V\setminus\{k\}} = C^V - \frac{C^V_{:k} C^V_{k:}}{C^V_{kk}}$$

- $C^V_{:k}$ denotes the k-th column of C^V, and $C^V_{k:}$ its k-th row

EFFICIENT V-OPT COMPUTATION VIA LOW-RANK UPDATE II

- We are interested in the restriction to unlabeled nodes, i.e., the minor excluding row and column k

Restriction to unlabeled set (excluding node k)

$$(C^{V\setminus\{k\}})_{-k,-k} = (C^V)_{-k,-k} - \frac{C^V_{-k,k}\, C^V_{k,-k}}{C^V_{kk}} = (Q_{V\setminus\{k\}V\setminus\{k\}})^{-1}$$

$\Rightarrow$ we used a rank-one update of the covariance matrix to avoid recomputing full inverse

EFFICIENT V-OPT COMPUTATION VIA LOW-RANK UPDATE III

- We have

$$\begin{aligned}
\mathrm{Tr}(C^{V\setminus\{k\}}) &= \underbrace{\mathrm{Tr}((C^{V\setminus\{k\}})_{-k,-k})}_{\text{Correct objective to minimize}} + \mathrm{Tr}((C^{V\setminus\{k\}})_{k,k}) \\
&= \mathrm{Tr}((C^{V\setminus\{k\}})_{-k,-k}) + \mathrm{Tr}\left((C^V)_{k,k} - \frac{(C^V_{:k}C^V_{k:})_{k,k}}{C^V_{kk}}\right) \\
&= \mathrm{Tr}((C^{V\setminus\{k\}})_{-k,-k}) + \mathrm{Tr}\left((C^V)_{k,k} - \frac{(C^V_{kk})^2}{C^V_{kk}}\right) \\
&= \mathrm{Tr}((C^{V\setminus\{k\}})_{-k,-k})
\end{aligned}$$

- **Question**: why not just compute $\mathrm{Tr}((C^{V\setminus\{k\}})_{-k,-k})$ directly?

EFFICIENT V-OPT COMPUTATION VIA LOW-RANK UPDATE IV

- The reason is

$$\begin{aligned}\operatorname{Tr}((C^{V\setminus\{k\}})_{-k,-k}) &= \operatorname{Tr}(C^{V\setminus\{k\}}) \\ &= \operatorname{Tr}(C^V) - \operatorname{Tr}\left(\frac{C^V_{:k}C^V_{k:}}{C^V_{kk}}\right) \\ &= \underbrace{\operatorname{Tr}(C^V)}_{\text{independent of } k} - \frac{\|C^V_{:k}\|_2^2}{C^V_{kk}}\end{aligned}$$

- This allows us to show that

Simplification of V-Opt

$$h = \operatorname*{argmin}_{k\in\mathcal{U}} \operatorname{Tr}((Q_{\mathcal{U}\setminus\{k\}\mathcal{U}\setminus\{k\}})^{-1}) = \operatorname*{argmax}_{k\in\mathcal{U}} \frac{\|C^V_{:k}\|_2^2}{C^V_{kk}}$$

$\Rightarrow$ efficient active learning iterations since C^V is only computed once

TAKEAWAY: BAYESIAN FORMULATION AND ACTIVE LEARNING

- **Goal**: Formulate graph learning problem in Bayesian framework
- **Graph Bayesian setting**:
  - Prior: given by the (fractional) Laplace learning energy
  - Can be used to quantify uncertainty in predictions
- **Active learning**:
  - Select most useful point to label
  - Based on Bayesian formulation
  - Requires efficient matrix computations

GRAPH TOPOLOGY

STRUCTURE OF VARIATIONAL OBJECTIVES ON GRAPHS

- Variational methods on graphs often separate:
  - **Where nodes interact**: encoded by the graph structure, such as edge weights w_{ij}
  - **How nodes interact**: encoded by the interaction mechanism, such as differences in function values across edges:

    $$\sum_{i,j} w_{ij}\, \Phi(f_i - f_j)$$

    where Φ could be $|f_i - f_j|^2$, $|f_i - f_j|^p$, etc.
- All the extensions of Laplace learning that we considered until now modify the interaction part
- **Question**: what can be said about the graph structure part?

MODIFYING THE GRAPH STRUCTURE

Modifying the graph structure

Changes in graph structure can lead to significant improvements in learning behavior and model expressiveness

- **Mitigating spiking in Laplace learning**:
  - Reweighted Laplacians provably remove the degeneracies of Laplace learning [9]
- **Hypergraph Learning**:
  - Graphs only model pairwise relations — this can be limiting in datasets with natural higher-order structure (e.g., co-authorships, group interactions).
  - Hypergraphs generalize graphs by allowing edges (hyperedges) to connect any number of nodes.

HYPERGRAPH LEARNING

HYPERGRAPH SETTING

- A hypergraph G is defined as $G = (V, E)$ where V is a set of vertices and E a family of subsets e – known as **hyperedges** – of V with $|e| \geq 2$

- **Intuition**: since $|e| \geq 2$, we capture higher order relationships between samples, e.g. similarity of researchers based on paper authorship

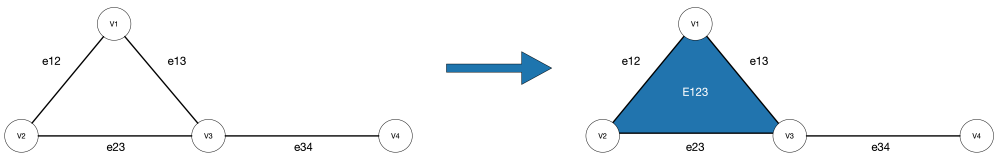


Figure: From graphs to hypergraphs

HYPERGRAPH DECOMPOSITION

- Let $E^{(k)} = \{\{x_i, x_j\} \,|\, \text{there exists } e \in E \text{ with } |e| = k+1 \text{ and } \{x_i, x_j\} \in e\}$

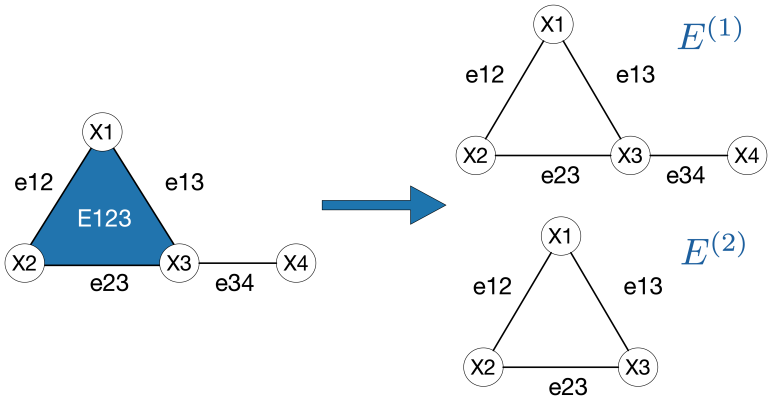


Figure: Skeleton graphs $G = (\Omega, E^{(k)})$

HYPERGRAPH LEARNING

- The following method generalizes Laplace learning to hypergraphs

Hypergraph learning [48]

Hypergraph learning aims to find

$$u = \operatorname*{argmin}_{v:V\to\mathbb{R}} \sum_{e\in E} \sum_{\{x_i,x_j\}\subseteq e} \frac{w_0(e,x_i,x_j)}{|e|}(v(x_i)-v(x_j))^2 \text{ such that } v(x_i)=\ell_i \text{ for } i\leq N$$

where w_0 is the hyperedge weight function

- **Key observation**: for each hyperedge e, we penalize the smoothness of v between each pair or vertices $\{x_i, x_j\} \subseteq e$

$\Rightarrow$ Basically Laplace learning on each skeleton graph

HIGHER ORDER HYPERGRAPH LEARNING

- The next method does fractional Laplace learning on hypergraphs

Higher order hypergraph learning

Higher order hypergraph learning aims to find

$$u = \operatorname*{argmin}_{v:V\mapsto\mathbb{R}} \sum_{k=1}^{q} \lambda_k v^T (L_n^{(k)})^k v = v^T \left[\sum_{k=1}^{q} \lambda_k (L_n^{(k)})^k \right] v =: v^T \mathcal{L}_n^{(q)} v$$

where $L_n^{(k)}$ is the Laplacian of the skeleton graphs $G_n^{(k)} = (V, E_n^{(k)})$ and $q = \max_{e\in E} |e| - 1$

$\Rightarrow$ Laplace learning on E_1, fractional Laplace learning with $s = 2$ on E_2, etc.

VARIATIONAL CONSISTENCY

- Most algorithms look for the solutions of **variational problems** on the (hyper)graph with n vertices i.e. labelling functions u_n are minimizers of some functional/energy $\mathcal{E}_n$

- **Intuition**: in the large data limit, i.e. as $n \to \infty$, the discrete samples $\{x_i\}_{i=1}^n \subseteq \Omega$ "converge" to the continuum set Ω

$\Rightarrow$ It is therefore reasonable to assume that a limiting energy/learning problem $\mathcal{E}_\infty$ is defined for functions $v : \Omega \to \mathbb{R}$

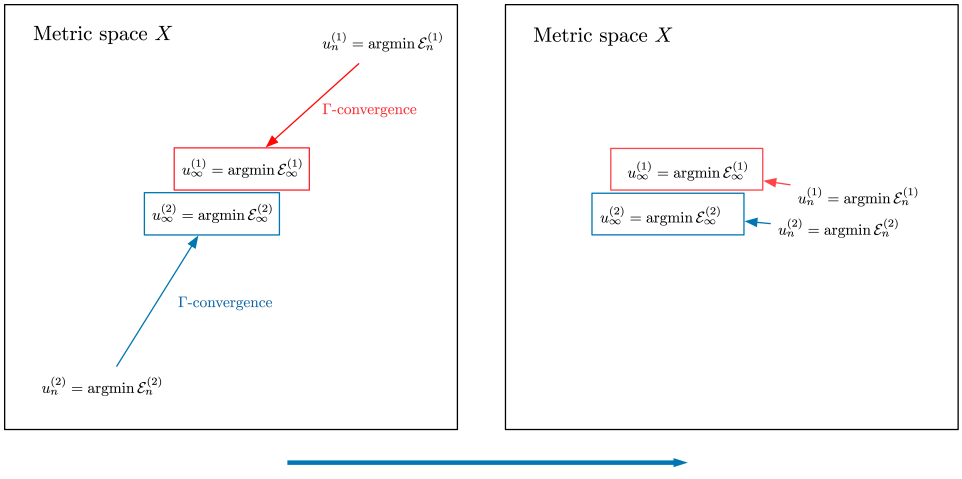
- We want to establish relationships between $\mathcal{E}_n$, $\operatorname{argmin} \mathcal{E}_n$ and $\mathcal{E}_\infty$, $\operatorname{argmin} \mathcal{E}_\infty$

- For **Laplace learning**:
  - $\mathcal{E}_n =$ graph Dirichlet energy with constraints
  - $\mathcal{E}_\infty = \mathrm{W}^{1,2}$-semi-norm with constraints

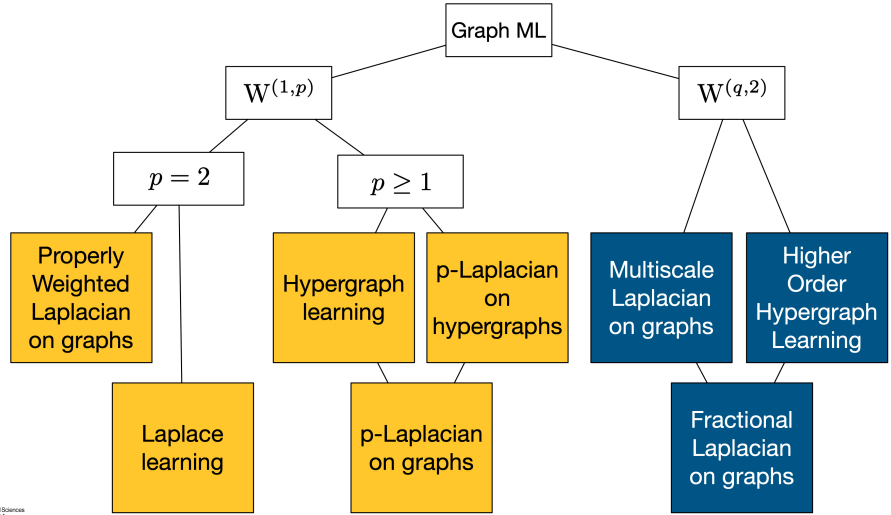
CONVERGENCE OF u_n THROUGH Γ-CONVERGENCE

- **Convergence of minimizers of functionals** is established through Γ-convergence [7]
- Γ-convergence is a property of a sequence of functionals: $\mathcal{E}_n$ Γ-converges to $\mathcal{E}_\infty$
- **Fundamental property of Γ-convergence**: "compactness of $u_n = \operatorname{argmin} \mathcal{E}_n$ in X" + "Γ-convergence of functionals $\mathcal{E}_n$ to $\mathcal{E}_\infty$" = "convergence in X of u_n to $u_\infty = \operatorname{argmin} \mathcal{E}_\infty$"
- Metric space X is TL^p-space [19] in which we can define a distance (similar to the Wasserstein distance) allowing us to **compare discrete and continuum functions**

VARIATIONAL CONSISTENCY TO CLUSTER ALGORITHMS



CLASSIFICATION OF VARIOUS (HYPER)GRAPH ALGORITHMS



HOHL IS LAPLACE LEARNING ON A SPECIAL GRAPH

- **Recall**: HOHL energy is $\langle v, \sum_{k=1}^{q} \lambda_k (L_n^{(k)})^k v \rangle_n = \langle v, \mathcal{L}_n^{(q)} v \rangle_n$
- There exists a graph whose Laplacian is equal to $\mathcal{L}_n^{(q)}$

  $\Rightarrow$ **HOHL is Laplace learning on this graph and a quadratic form**

- **Consequence**: HOHL can serve as a drop-in replacement for Laplace learning in existing pipelines (e.g. SSL, Active learning etc.)

APPLICATION EXAMPLE: ACTIVE LEARNING

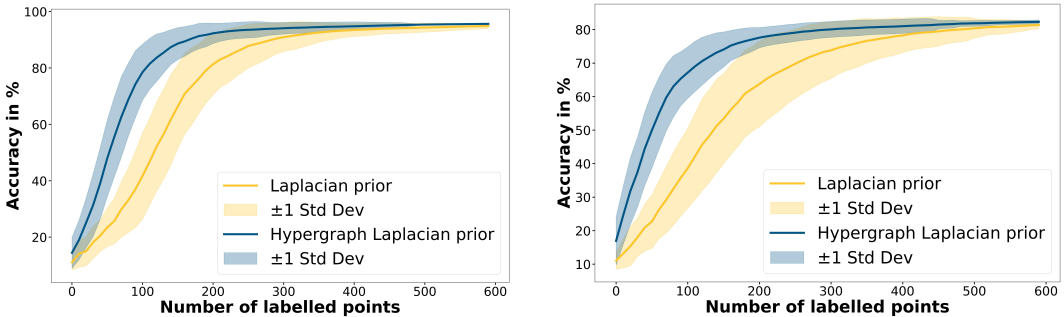


Figure: Accuracy in active learning using Laplacian and HOHL priors. Left: MNIST dataset. Right: fashionMNIST dataset.

TAKEAWAY: HYPERGRAPH LEARNING

- **Goal**: Extend graph learning to more complicated structures
- **Hypergraph learning**:
  - Can be analyzed in the same manner as graph learning
  - Can sometime lead to better results than graph learning because it better captures underlying geometry of the data

OVER-SMOOTHING AND OVER-SQUASHING

FEATURES AND GRAPH STRUCTURE

- In many semi-supervised learning problems on graphs, we are given:
  - A **graph structure** $G = (V, E)$, encoding relationships between nodes
  - A **feature matrix** $X \in \mathbb{R}^{n \times d}$, with X_i describing attributes of node v_i
- Citation Network:
  - Nodes = papers, Edges = citations
  - Features = bag-of-words representation of paper content
- Social Networks:
  - Nodes = users, Edges = friendships or follows
  - Features = user attributes (e.g., age, location, interests)

GRAPH NEURAL NETWORKS FOR SEMI-SUPERVISED LEARNING

- In most of the examples we saw previously, the features were the vertices directly
- Graph neural networks are able to deal with the most general situation where the graph and features differ

GNN-based SSL Approach

Train a GNN $f_\theta : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times C}$ to output soft labels:

$$\hat{Y} = f_\theta(X, W) \quad \text{with } \hat{Y}_i \in \left\{ \mathbf{p} \in \mathbb{R}^C \;\middle|\; \sum_{i=1}^{C} p_i = 1,\ p_i \geq 0 \text{ for all } i \right\}$$

where θ are learnable weights and C is the number of classes

- Weights θ are obtained through optimizing cross-entropy loss on the labeled nodes:

$$\mathcal{L}_{\text{sup}}(\theta) = \sum_{i\in\mathcal{L}} \text{CE}(\hat{Y}_i, y_i)$$

STRUCTURE OF A GRAPH NEURAL NETWORK

- A Graph Neural Network is a deep architecture
- Each layer updates node representations using both their own features and those of their neighbors
- **Layer structure**:

$$H^{(k+1)} = \sigma\left(\underbrace{\tilde{W}H^{(k)}A^{(k)}}_{\text{message passing + linear map}}\right)$$

  - $H^{(0)} = X$: initial features
  - $A^{(k)}$: learnable weight matrix at layer k
  - σ: non-linear activation function (e.g. ReLU)
  - $\tilde{W}$: $F(W)$ for W the weight matrix
- The output $H^{(K)} \in \mathbb{R}^{n \times C}$ gives predictions for each node after K layers.

MESSAGE PASSING AS LAPLACIAN SMOOTHING

- Message passing in GNNs corresponds to applying a step of a **random walk** on the graph to the **features**:

$$\widetilde{W} = WD^{-1} = P$$

- **Recall**: P is the **Laplace random walk transition matrix**

$\Rightarrow$ understanding the properties of the random walk can help us understand the behavior of GNNs

OVER-SMOOTHING

- The Laplace random walk converges to constants for large time steps
- The same behavior can be therefore observed when the number of layers goes to infinity

  $\Rightarrow$ The same ideas that lead to the Poisson random walk can be used to **correct the degeneracies in GNNs**

- This leads to an **improved GNN architecture** [43]

Over-smoothing

The phenomenon where the graph features converge to a constant is called over-smoothing

OVER-SMOOTHING DEPENDS ON THE GRAPH STRUCTURE

- Over-smoothing depends on the speed of convergence to the stationary distribution of the Laplace random walk

Relationship between convergence speed and eigenvalues [12]

For a graph, any initial distribution f and $t \in \mathbb{N}$, we have:

$$\|P^t f - \pi\| \leq e^{-t\lambda_2} \frac{\max_i \sqrt{d_i}}{\min_i \sqrt{d_i}}$$

where $\tilde{\lambda}_2$ is the second eigenvalue of the normalized Laplacian $D^{-1/2}LD^{-1/2}$

$\Rightarrow$ Stacking layers leads to exponential convergence to the stationary distribution

$\Rightarrow$ **Over-smoothing is directly linked to the graph structure**

- **However**: depth should lead to better learning
- **Question**: how can we control over-smoothing from a graph-structural perspective?

CONTROLLING OVER-SMOOTHING

Relationship between eigenvalues and graph sparsity [12]

We have

$$\tilde{\lambda}_2 \leq 1 - 2\frac{\sqrt{(\max_i d_i) - 1}}{\max_i d_i}\left(1 - \frac{2}{\varnothing}\right) + \frac{2}{\varnothing}$$

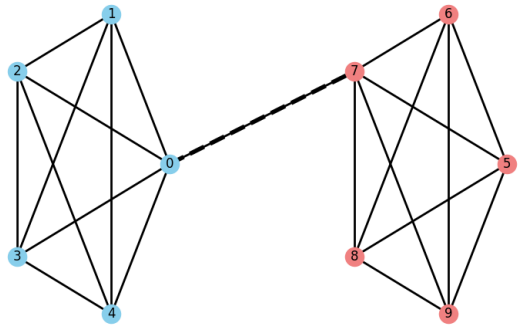
where $\varnothing$ denotes the longest shortest path between any two vertices

$\Rightarrow$ We can reduce λ_2, i.e. oversmoothing, by reducing $(\max_i d_i)$, i.e. having **a sparser graph**

- **Question**: given a graph, which edges should I remove?

BOTTLENECKS

- If we remove the wrong edges, we might end up with **bottlenecks**



BOTTLENECKS LIMIT INFORMATION FLOW

Key Insight

Let $G = (V, E)$ be a graph with two dense regions V_1 and V_2, connected by a single edge (i, j), where $i \in V_1$, $j \in V_2$. In a random walk with transition matrix $P = WD^{-1}$, the crossing probability is small:

$$P_{ij} = \frac{w_{ij}}{d_i} \ll 1$$

$\Rightarrow$ In graphs with two well-connected clusters joined by a sparse **bottleneck**, random walks tend to stay trapped within clusters

$\Rightarrow$ **Information propagation across clusters is hindered** (message passing, random walk etc.)

$\Rightarrow$ Diffusion-based methods (e.g. Laplace learning, GNNs) may fail to generalize across clusters

OVER-SQUASHING

Over-squashing

The inability of distant nodes to influence each other is called over-squashing

- The presence of bottlenecks/over-squashing can also theoretically studied through the Cheeger constant

Cheeger constant

The Cheeger constant is:

$$h(G) = \min_{\substack{S \subset V \\ 0 < |S| \leq \frac{n}{2}}} \frac{W(S, \overline{S})}{\mathrm{vol}(S)} \quad \text{with} \quad \mathrm{vol}(S) := \sum_{i \in S} d_i$$

- RatioCut normalizes using node counts while Cheeger normalizes using degree volume

CHEEGER INEQUALITY

- $h(G)$ quantifies the **difficulty of separating** the graph into two large pieces

  $\Rightarrow$ Small $h(G)$ implies the presence of a **narrow bottleneck**

Cheeger Inequality

We have

$$2h(G) \geq \tilde{\lambda}_2 \geq \frac{h(G)^2}{2}$$

$\Rightarrow$ in order to have less "bottleneckless" we need to have a large $h(G)$ which implies a large λ_2

OVER-SMOOTHING AND OVER-SQUASHING TRADE-OFF

- We can summarize the previous results as follows

Over-smoothing and over-squashing trade-off [20]

- Removing edges reduces over-smoothing by reducing $\tilde{\lambda}_2$ but increases over-squashing by reducing $h(G)$
- Adding edges reduces over-squashing by increasing $h(G)$ but increases over-smoothing by increasing $\tilde{\lambda}_2$

- **Question**: how do I select the right edges to add/remove?

  $\Rightarrow$ Graph-rewiring in graph neural networks [2]

SPECTRAL APPROACH

- λ_2 is the central quantity to control for over-smoothing and over-squashing
- By using matrix perturbation theory, we can for example show the following

First-order approximation of change in second eigenvalue [28]

We have

$$\xi_2(M + \delta M) \approx \xi_2(M) + \Xi_2(M)^T(\delta M)\Xi_2(M)$$

where ξ_2 and Ξ_2 are the second eigenvalue and eigenfunction functions

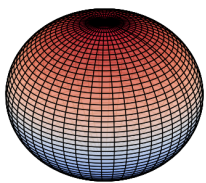
$\Rightarrow$ we can select edges based on **maximizing/minimizing** $\xi_2(M + \delta M)$

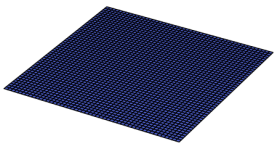
CURVATURE

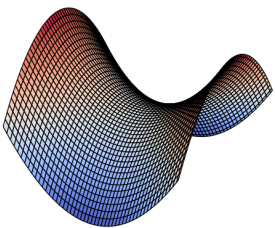
Curvature intuition

If geodesics starting at nearby points with same velocity

- remain parallel = space has zero curvature
- converge = space is positively curved
- diverge = space is negatively curved







GRAPH CURVATURE

- The same concept of curvature can be transposed to graphs
  - Edge (v_i, v_j) has **positive curvature** if
    - Edge connects nodes with many shared neighbors
    - Local structure resembles triangle
    - **Intuition**: Walkers starting at nodes v_i and v_j have a high probability of converging to same vertex
  - Edge (v_i, v_j) has **negative curvature** if
    - Edge bridges sparsely connected regions
    - Few (or no) common neighbors between endpoints
    - **Intuition**: Walkers starting at nodes v_i and v_j have a low probability of converging to same vertex

GRAPH CURVATURE APPROACH

- True definition of graph curvature depends on optimal transport
- Many relaxations exist to allow for efficient computation
- We can [15]
  - add edges around edges with low curvature to remove **bottlenecks** and decrease **over-squashing**
  - remove edges around edges with high curvature to decrease **over-smoothing**

TRANSFERRING GNN IDEAS TO GRAPH LEARNING

- We have seen that graph learning ideas can help in GNNs

Open research question

Can some structural graph topological approaches help inspire new methods in graph learning?

TAKEAWAY: OVER-SMOOTHING AND OVER-SQUASHING

- **Goal**: Understand how the graph structure influences learning in GNNs
- **Over-smoothing**:
  - all features converge to some constant
  - happens when convergence of random walk to stationary distribution is very fast
- **Over-squashing**:
  - information is blocked in the graph
  - happens when bottlenecks are present
- **Trade-off** can be resolved using:
  - spectral methods
  - curvature methods

UCLA College | Physical Sciences Mathematics

Adrien Weihs
Hedrick Assistant Adjunct Professor
weihs@math.ucla.edu

References

References I

[1] J. M. Amigo, H. Babamoradi, and S. Elcoroaristizabal. Hyperspectral image analysis. a tutorial. *Analytica Chimica Acta*, 2015.

[2] A. Arnaiz-Rodríguez and A. Velingker. Graph learning: Principles, challenges, and open directions. Tutorial at the 41st International Conference on Machine Learning (ICML), July 2024. Hall A8, Vienna, Austria.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006.

[4] A. L. Bertozzi and A. Flenner. Diffuse interface models on graphs for classification of high dimensional data. *SIAM Review*, 58(2):293–328, 2016.

[5] A. L. Bertozzi, X. Luo, A. M. Stuart, and K. C. Zygalakis. Uncertainty quantification in graph-based classification of high dimensional data. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):568–595, 2018.

[6] J. Bourgain, H. Brezis, and P. Mironescu. Another look at Sobolev spaces. In *Optimal Control and Partial Differential Equations*, pages 439–455, 2001.

[7] A. Braides. Γ-*convergence for Beginners*. Oxford University Press, 2002.

References II

[8] J. Calder, B. Cook, M. Thorpe, and D. Slepčev. Poisson learning: graph based semi-supervised learning at very low label rates. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[9] J. Calder and D. Slepčev. Properly-weighted graph laplacian for semi-supervised learning. *Applied Mathematics & Optimization*, 82(3):1111–1159, 2020.

[10] J. Calder, D. Slepčev, and M. Thorpe. Rates of convergence for Laplacian semi-supervised learning with low labeling rates. *to appear in Research in the Mathematical Science, preprint arXiv:2006.02765*, 2020.

[11] L. Chizat and F. R. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, pages 3040–3050, 2018.

[12] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, Providence, RI, 1997.

[13] A. El Alaoui, X. Cheng, A. Ramdas, M. J. Wainwright, and M. I. Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. In *Proceedings of the Conference on Learning Theory*, pages 879–906, 2016.

References III

[14] G. E. Fasshauer and Q. Ye. Reproducing kernels of generalized sobolev spaces via a green function approach with distributional operators. *Numerische Mathematik*, 119(3):585–611, 2011.

[15] L. Fesser and M. Weber. Mitigating over-smoothing and over-squashing using augmentations of forman-ricci curvature. In S. Villar and B. Chamberlain, editors, *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 19:1–19:28. PMLR, 27–30 Nov 2024.

[16] M. Flores, J. Calder, and G. Lerman. Analysis and algorithms for ℓ_p-based semi-supervised learning on graphs. *Applied and Computational Harmonic Analysis*, 60:77–122, 2022.

[17] C. García-Cardona, E. Merkurjev, A. L. Bertozzi, A. Flenner, and A. G. Percus. Multiclass data segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014.

[18] N. García Trillos, R. Murray, and M. Thorpe. Rates of convergence for regression with the graph poly-laplacian, 2022.

[19] N. García Trillos and D. Slepčev. Continuum limit of total variation on point clouds. *Archive for Rational Mechanics and Analysis*, 220(1):193–241, 2016.

References IV

[20] J. H. Giraldo, K. Skianis, T. Bouwmans, and F. D. Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 566–576, New York, NY, USA, 2023. Association for Computing Machinery.

[21] C. Gong, Y. Cheng, J. Yu, C. Xu, C. Shan, S. Luo, and X. Li. A survey on learning from graphs with heterophily: Recent advances and future directions, 2024.

[22] B. Hanin. Random neural networks in the infinite width limit as gaussian processes, 2021.

[23] B. Hanin and A. Zlokapa. Bayesian interpolation with deep linear networks, 2023.

[24] J. Harlim, D. Sanz-Alonso, and R. Yang. Kernel methods for bayesian elliptic inverse problems on manifolds. *SIAM/ASA Journal on Uncertainty Quantification*, 8(4):1414–1445, 2020.

[25] M. Jacobs, E. Merkurjev, and S. Esedoglu. Auction dynamics: A volume-constrained MBO scheme. *Journal of Computational Physics*, 354:288–310, 2018.

[26] M. Ji and J. Han. A variance minimization criterion for active learning on graphs. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 556–564, Mar. 2012.

References V

[27] A. Jung, A. O. Hero III, A. Mara, and S. Jahromi. Semi-supervised learning via sparse label propagation. *arXiv preprint arXiv:1612.01414*, 2016.

[28] K. Karhadkar, P. K. Banerjee, and G. Montufar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] Y. Ma, X. Liu, N. Shah, and J. Tang. Is homophily a necessity for graph neural networks?, 2023.

[30] X. Mai. A random matrix analysis and improvement of semi-supervised learning for large dimensional data. *Journal of Machine Learning Research*, 19(79):1–27, 2018.

[31] J. Mairal, M. Arbel, A. Rudi, and J.-P. Vert. Machine learning with kernel methods — lecture slides. Online course materials, MVA/MASH, ENS Paris-Saclay & University of Paris Dauphine, PSL, 2022. Accessed 2025-06-19.

[32] K. Miller and J. Calder. Poisson reweighted laplacian uncertainty sampling for graph-based active learning. *SIAM Journal on Mathematics of Data Science*, 5(4):1160–1190, 2023.

[33] K. S. Miller and A. L. Bertozzi. Model change active learning in graph-based semi-supervised learning. *Communications on Applied Mathematics and Computation*, 6(2):1270–1298, 2024.

References VI

[34] B. Mohar. Some applications of laplace eigenvalues of graphs. In L. W. Beineke and R. J. Wilson, editors, *Graph Symmetry: Algebraic Methods and Applications*, volume 497 of *Mathematics and Its Applications*, pages 225–275. Springer, Dordrecht, 1997.

[35] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[36] B. Settles. *Active Learning*, volume 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, Palo Alto, CA, USA, 2012.

[37] U. Shaham, K. Stanton, H. Li, R. Basri, B. Nadler, and Y. Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations*, 2018.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[39] Z. Shi, S. Osher, and W. Zhu. Weighted nonlocal laplacian on interpolation from sparse data. *Journal of Scientific Computing*, 73(2):1164–1177, 2017.

[40] D. Slepčev and M. Thorpe. Analysis of p-Laplacian regularization in semisupervised learning. *SIAM Journal on Mathematical Analysis*, 51(3):2085–2120, 2019.

References VII

[41] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In B. Schölkopf and M. K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 144–158, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[42] M. Stoer and F. Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, July 1997.

[43] M. Thorpe, T. M. Nguyen, H. Xia, T. Strohmer, A. Bertozzi, S. Osher, and B. Wang. GRAND++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2022.

[44] M. Thorpe and Y. van Gennip. Deep limits of residual neural networks. *Research in the Mathematical Sciences*, 10(1):6, 2022.

[45] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[46] A. Weihs and M. Thorpe. Consistency of fractional graph-laplacian regularization in semisupervised learning with finite labels. *SIAM Journal on Mathematical Analysis*, 56(4):4253–4295, 2024.

[47] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph Learning: A Survey . *IEEE Transactions on Artificial Intelligence*, 2(02):109–127, Apr. 2021.

References VIII

[48] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[49] D. Zhou and B. Schölkopf. Learning from labeled and unlabeled data using random walks. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *Pattern Recognition*, pages 237–244, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[50] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, 2003.