

Graph Neural Networks (GNNs)

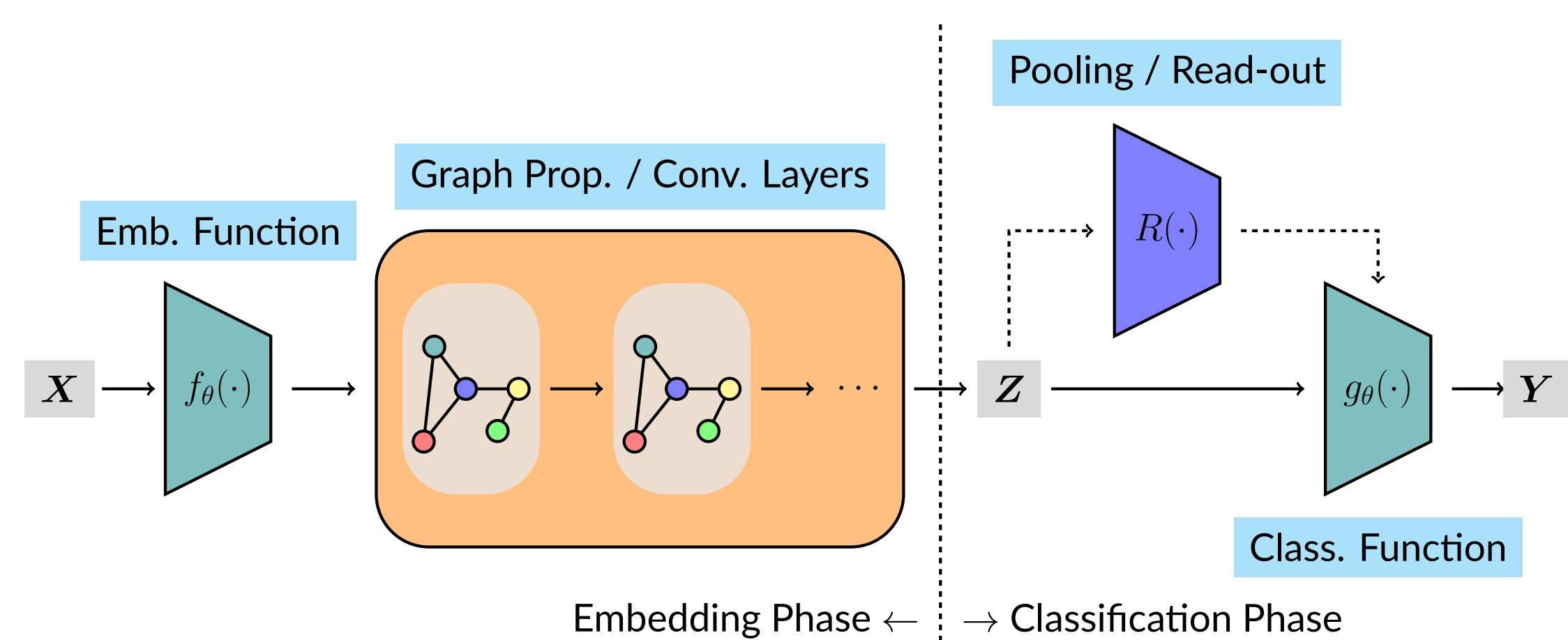


Figure 1. A typical structure of GNNs.

- Embedding function $f(\mathbf{X})$: map features $\mathbf{X} \in \mathbb{R}^{n \times F}$ to initial embeddings $\mathbf{Z}^{(0)} \in \mathbb{R}^{n \times d}$;
- Graph propagation / convolution $\text{GP}(\mathbf{Z}^{(k)}; \mathbf{A})$: propagate embedding K times;
- Read-out / pooling $\text{R}(\mathbf{Z}^{(K)}; \mathbf{A})$ (optional): pooling for graph-level tasks;
- Classification function $g(\cdot)$: final classification to generate predictions \mathbf{Y} .

Expressive Power & Universality of GNNs

- Spectral GNNs: designing **universal filters** (e.g., ChebNet, GPRGNN, BernNet, ...).
- Spatial GNNs: designing GNNs bounded by k -WL tests (e.g., GIN, ...).

There are works exploring relations between GNNs and geometric objects (e.g., curvature, cellular sheaves) and physical concepts (e.g., oscillators).

Q: Can we define universality of spatial GNNs from a geometric perspective?

Preliminaries

Definition (Equivalent). For a given graph $G = (V, E)$, two node embedding matrices $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)} \in \mathbb{R}^{n \times d}$ are equivalent if for all $(i, j) \in E$, $\|\mathbf{Z}_i^{(1)} - \mathbf{Z}_j^{(1)}\|_2 = \|\mathbf{Z}_i^{(2)} - \mathbf{Z}_j^{(2)}\|_2$ holds.

Definition (Congruent). For a given graph $G = (V, E)$, two node embedding matrices $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)} \in \mathbb{R}^{n \times d}$ are congruent if for all $i, j \in V$, $\|\mathbf{Z}_i^{(1)} - \mathbf{Z}_j^{(1)}\|_2 = \|\mathbf{Z}_i^{(2)} - \mathbf{Z}_j^{(2)}\|_2$ holds.

Definition (Globally Rigid). For a given graph $G = (V, E)$, an embedding matrix \mathbf{Z} is globally rigid if all its equivalent embedding matrices \mathbf{Z}' are also congruent to \mathbf{Z} .

Definition (Rigid). For a given graph $G = (V, E)$, an embedding matrix \mathbf{Z} is rigid if all equivalent embeddings that can be obtained by **continuous** motion from \mathbf{Z} are congruent to \mathbf{Z} .

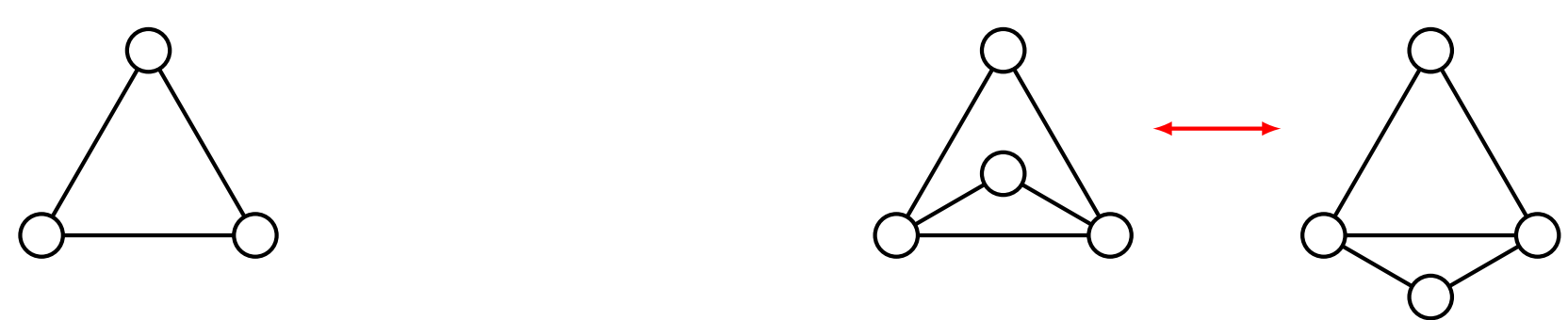
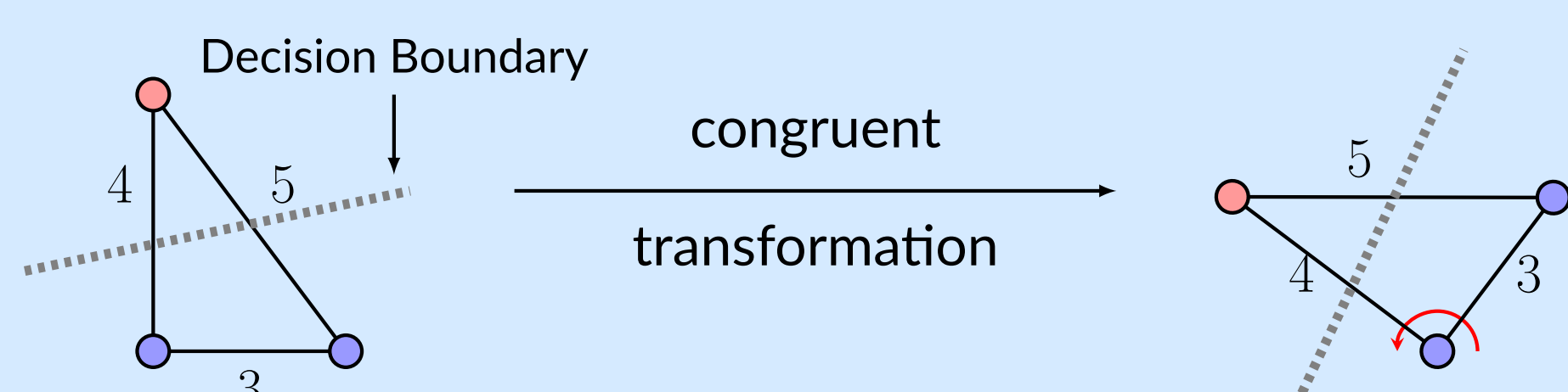


Figure 2. Left: A globally rigid graph in \mathbb{R}^2 ; Right: A rigid but not globally rigid graph in \mathbb{R}^2 , since it has an equivalent but not congruent embedding.

Definition (Metric Matrix). The metric matrix of an embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$ is defined as $\mathbf{M}_{\mathbf{Z}} = (\|\mathbf{Z}_i - \mathbf{Z}_j\|_2)_{ij}$. We also define the mapping from an embedding matrix \mathbf{Z} to its metric matrix $\mathbf{M}_{\mathbf{Z}}$ as $\mathbf{M}_{\mathbf{Z}} = \mathbf{M}(\mathbf{Z})$.

Defining Spatial-Universality

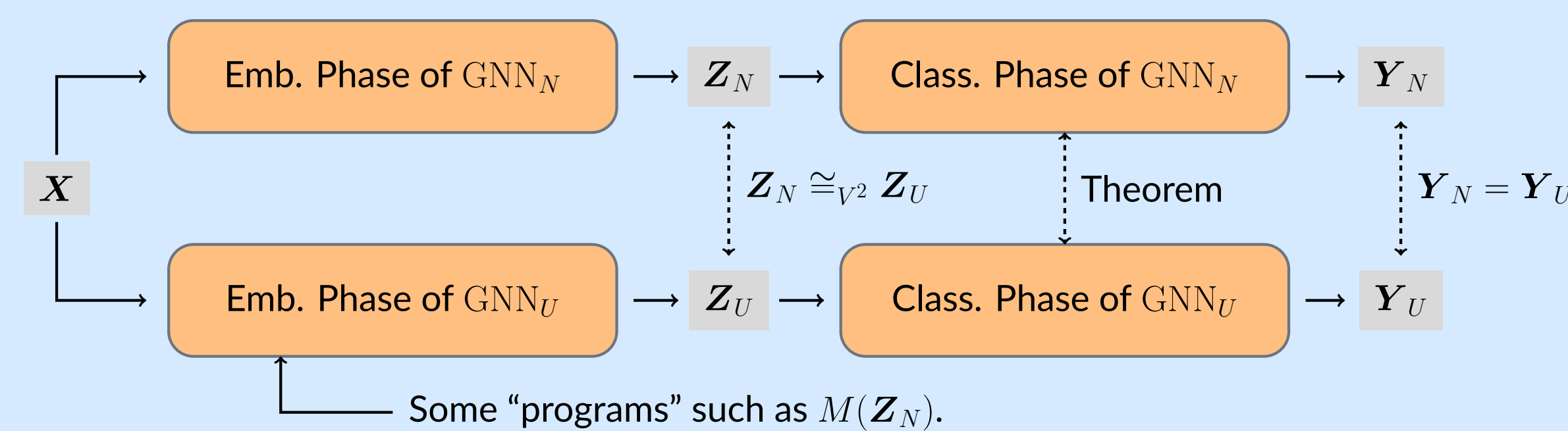
Observation:



Defining Spatial-Universality (Cont'd)

Theorem (MLPs Are Congruent-Insensitive). Given two congruent embedding matrices \mathbf{Z}_1 and \mathbf{Z}_2 , for any MLP_M (with biases), there always exists another MLP_N (also with biases) such that $\text{MLP}_M(\mathbf{Z}_1) = \text{MLP}_N(\mathbf{Z}_2)$.

Idea:



Spatial-Universal: It can arrange nodes with a given metric matrix!

This idea is closely related to the **Distance Geometry Problem (DGP)**.

Distance Geometry Problem (DGP)

Given a positive integer d , a graph $G = (V, E)$, and a symmetric non-negative matrix \mathbf{M} , decide whether there exists an embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$, such that

$$\forall (i, j) \in E, \|\mathbf{Z}_i - \mathbf{Z}_j\| = M_{ij}.$$

Optimization Objective

About the Optimization Objective

- **Full** metric matrix: $O(n^2) \Rightarrow$ **partial** metric matrix on edges: $O(m)$.
- For **globally rigid** graphs, partial metric matrix is enough to determine the “shape” of the embedding, this modification does not weaken the expressive power.
- However, solving the DGP is **NP-Hard**. We can not directly arrange the nodes, thus we introduce an **error-tolerant objective**:

$$E_p(\mathbf{Z}; \mathbf{M}, E) = \frac{1}{2} \|\mathbf{A} \odot (\mathbf{M}(\mathbf{Z}) - \mathbf{M})\|_F^2 = \sum_{(i,j) \in E} \frac{1}{2} (\|\mathbf{Z}_i - \mathbf{Z}_j\|_2 - M_{ij})^2.$$

- This function is closely related to the raw **Stress function** σ_r in the **Multidimensional Scaling (MDS)** problem and the potential energy of **spring networks**.

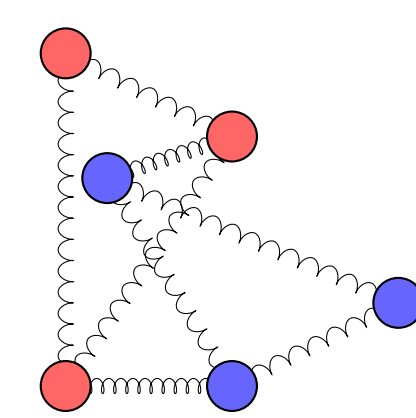


Figure 3. A spring network.

- To align with other representative GNNs, we modify E_p and add a regularization term to get the final objective:

$$\begin{aligned} \mathcal{L}(\mathbf{Z}; \mathbf{Z}^{(0)}, \mathbf{M}, E) &= (1 - \alpha) \tilde{E}_p(\mathbf{Z}; \mathbf{M}, E) + \alpha \|\mathbf{Z} - \mathbf{Z}^{(0)}\|_F^2 \\ &= (1 - \alpha) E_p(\mathbf{D}^{1/2} \mathbf{Z}; \mathbf{M}, E) + \alpha \|\mathbf{Z} - \mathbf{Z}^{(0)}\|_F^2. \end{aligned}$$

About the Metric Matrix

- For scenarios with prior knowledge about distances between nodes (e.g., molecular conformation generation, or graph drawing), directly use them; for other scenarios, learn a metric matrix.
- Our idea is to increase the distances between dissimilar nodes and reduce the distances between similar nodes:
 1. Introduce edge attention $\alpha_{ij} \in [-1, 1]$, when $\alpha_{ij} \rightarrow 1 \Leftrightarrow i, j$ tend to belong to the same class, and when $\alpha_{ij} \rightarrow -1 \Leftrightarrow i, j$ tend to belong to different classes;
 2. Map the initial embedding matrix $\mathbf{Z}^{(0)}$ (defined later) to a hidden matrix \mathbf{H} ;
 3. Use attention mechanisms, such as $\alpha_{ij} = \tanh(\mathbf{a}^\top [\mathbf{H}_i^\top; \|\mathbf{H}_j^\top\|])$ or $\alpha_{ij} = \tanh(\mathbf{H}_i \cdot \mathbf{W} \mathbf{H}_j^\top)$ to learn the edge attention;
 4. Then we can set $M_{ij} = \frac{1 - \alpha_{ij}}{1 + \alpha_{ij} + \varepsilon} \|\mathbf{Z}_i^{(0)} - \mathbf{Z}_j^{(0)}\|$, where ε is a small positive number.

Framework

The Embedding Function

- A linear layer $f(\mathbf{X}) = \mathbf{X} \mathbf{W} + \mathbf{1} \mathbf{b}^\top$ in linear GNNs, or
- A two-layer MLP $f(\mathbf{X}) = \sigma(\sigma(\mathbf{X} \mathbf{W}_1 + \mathbf{1} \mathbf{b}_1^\top) \mathbf{W}_2 + \mathbf{1} \mathbf{b}_2^\top)$ in spectral GNNs.

Propagation

- Our goal is to design a propagation method that minimizes the objective.
- It's non-convex. Following related works, we employ the **stationary point iteration** method.
- By computing the gradient, setting it to zero, rearranging the terms, rewriting it as an iteration form, substituting $1 - \alpha$ with β to allow more flexibility, it leads to:

$$\mathbf{Z}^{(k+1)} = (1 - \alpha) \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \mathbf{Z}^{(k)} + \beta \mathbf{D}^{-1/2} \mathbf{L}_H \mathbf{D}^{-1/2} \mathbf{Z}^{(k)} + \alpha \mathbf{Z}^{(0)},$$

where $\mathbf{H} = \mathbf{A} \odot \mathbf{M} \odot \mathbf{M}(\mathbf{D}^{-1/2} \mathbf{Z})^{\odot -1}$, and $\mathbf{L}_H = \text{diag}(\mathbf{H} \mathbf{1}) - \mathbf{H}$.

- And we have the message-passing form:

$$\mathbf{Z}_i^{(k+1)} = (1 - \alpha) \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{Z}_j^{(k)}}{\sqrt{d_i d_j}} + \beta \sum_{j \in \mathcal{N}(i)} \frac{M_{ij} (\mathbf{Z}_i^{(k)} - \mathbf{Z}_j^{(k)})}{\sqrt{d_i d_j} \|\mathbf{Z}_i^{(k)} / \sqrt{d_i} - \mathbf{Z}_j^{(k)} / \sqrt{d_j}\|_2} + \alpha \mathbf{Z}_i^{(0)}.$$

Optional Linear and Non-linear Transformations

We may incorporate linear and non-linear transformations after each propagation step. In our experiments without pre-designed metric matrices, such as node classification, we utilize three designs from the GCNII model: a linear transformation, the identity mapping, and a non-linear transformation (ReLU).

The Classification Function

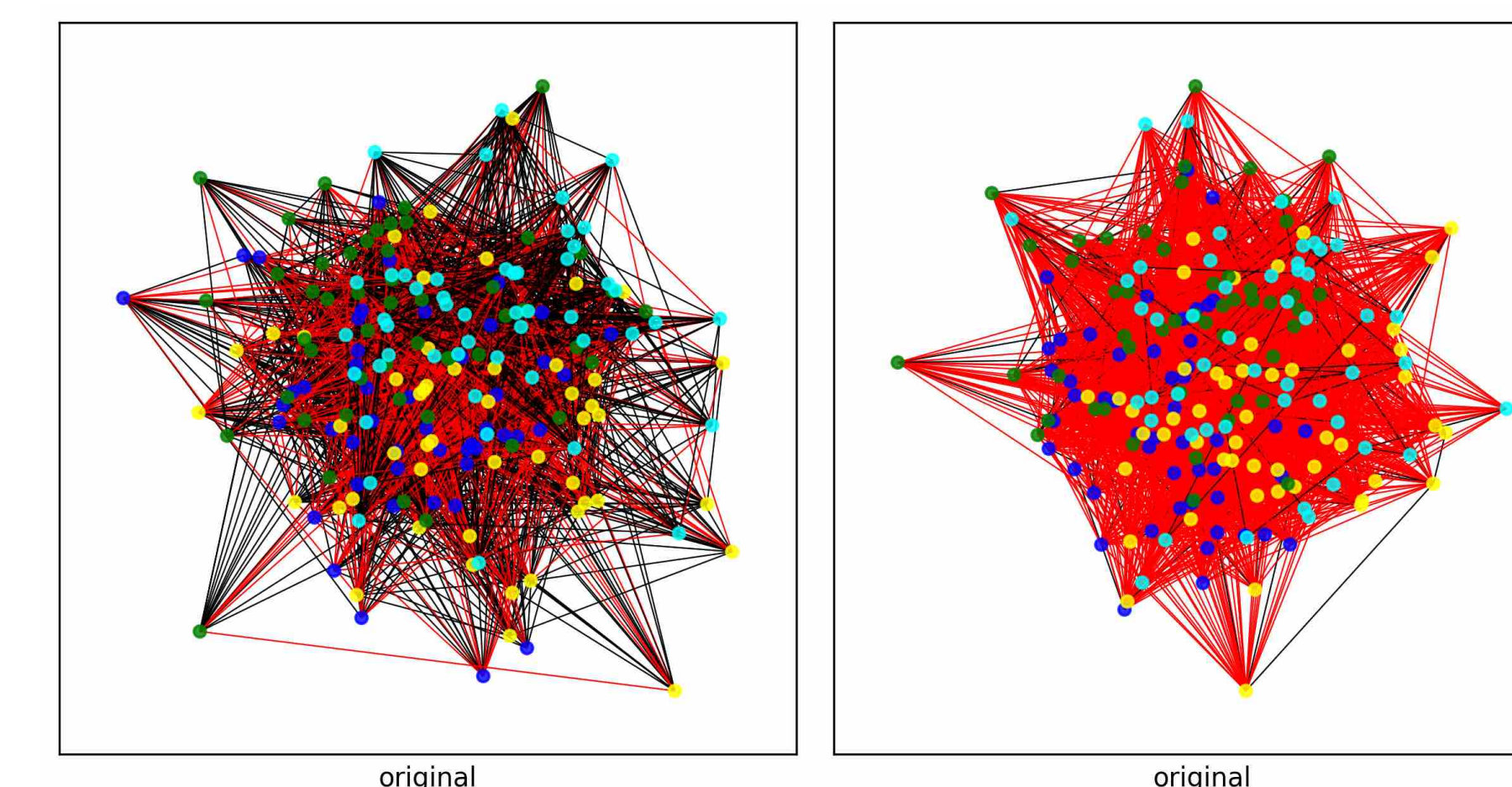
We use a linear layer $g(\mathbf{Z}^{(K)}) = \mathbf{Z}^{(K)} \mathbf{W} + \mathbf{1} \mathbf{b}^\top$ as the final classification function.

Experiments

We have done the “Arranging Nodes with Given Metric Matrices” experiments on synthetic graphs, supervised node classification and graph regression experiments on real-world graphs.

Arranging Nodes with Given Metric Matrices

- We generate two **Stochastic Block Model (SBM)** graphs, one homophilic and one heterophilic, consisting of four blocks with 50 nodes in each block.
- The node features are sampled from two 2-dimensional Gaussian distributions.



- If i and j are in the same class, we set $M_{ij} = 0$; otherwise, we set $M_{ij} = 5$.
- We pass the node features through 8 MGNN layers, with $\alpha = 0.05$ and $\beta = 0.5$.
- For visualization results, please refer to the smaller posters below or our paper.

Supervised Node Classification and Graph Regression

- Our MGNN model performs well, and the results are promising.
- For experiment details and results, please refer to the smaller posters below or our paper.