Graph Representation Learning

Gonzalo Mateos

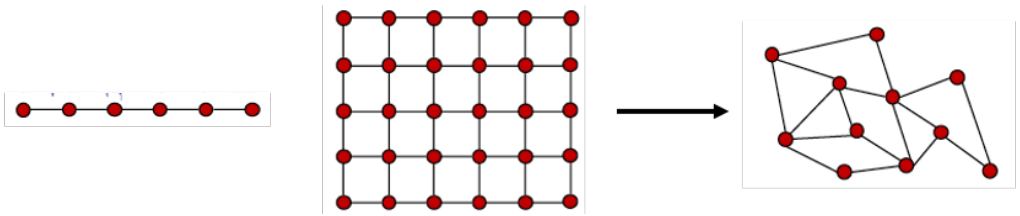
Dept. of ECE and Goergen Institute for Data Science
University of Rochester
gmateosb@ece.rochester.edu
http://www.ece.rochester.edu/~gmateosb/

Acknowledgment: Y. Li, S. S. Saboksayr and M. Wasserman

Facultad de Ingeniería, UdelaR
Montevideo, Uruguay
February 11, 2021

Machine learning on graphs: Motivation

- Successful models for representation learning of structured data
  - Sequences (e.g., text, videos) via recurrent neural networks (RNNs)
  - Image classification via convolutional neural networks (CNNs)



- Data not always regular $\Rightarrow$ Complex relational structures
  - Graphs with social networks, computational chemistry, biology, ...
- Challenge: apply models designed for regular data to graphs
  - Graph structures can be arbitrary and vary across scenarios
  - Convolutions do not generalize to irregular graph domains

M. Bronstein et al, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, 2017

What is this lecture about?

Graph representation learning (GRL)

- Learn low-dimensional vectors (embeddings) for graph data
- Learning types:
  - Supervised: learn representations for node or graph classification
  - Unsupervised: learn representations that preserve graph structure
- Underlying graph domain:
  - Transductive: fixed graph structure (e.g., a large social network)
  - Inductive: input graphs can vary (e.g., multiple molecules)
- Information from graph nodes:
  - Featureless: no additional information (i.e., graph signals)
  - With features: nodes possess usable attributes

Roadmap

The network embedding problem

A taxonomy of graph embedding models

Unsupervised graph embedding

Applications

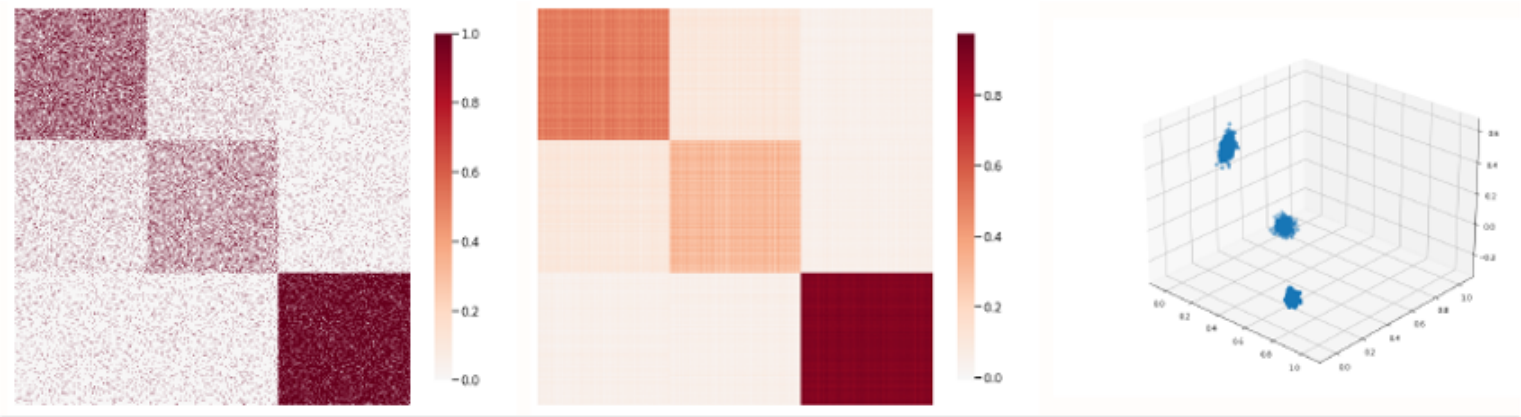
Network embedding

- Learn a mapping from a discrete graph to a continuous domain
- Given $G(\mathcal{V}, \mathcal{E})$ with weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{N_v \times N_v}$
- **Goal**: learn (low) d-dimensional vector representation $\{\mathbf{z}_i\}_{i \in \mathcal{V}}$
  $\Rightarrow$ Criterion is to preserve local and global graph properties
- Output is node embedding matrix $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_v}]^\top \in \mathbb{R}^{N_v \times d}$
  $\Rightarrow$ Pick $d \ll N_v$ for scalability
  $\Rightarrow$ Effectively a dimensionality reduction technique
- Extensions to embed the whole graph via $\mathbf{z} \in \mathbb{R}^d$ possible

Adjacency spectral embedding

- Ex: SBM with $N_v = 1500$, $Q = 3$ and mixing parameters

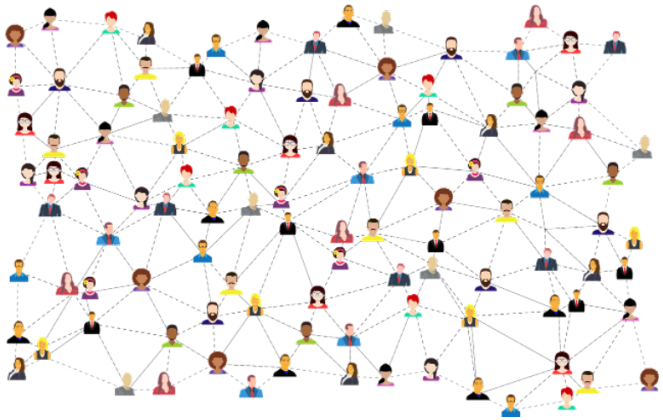
$$\boldsymbol{\alpha} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \quad \boldsymbol{\Pi} = \begin{bmatrix} 0.5 & 0.1 & 0.05 \\ 0.1 & 0.3 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$



- Sample adjacency (left), $\mathbf{Z}\mathbf{Z}^\top$ (center), rows of $\mathbf{Z}$ (right)
- Use embeddings to bring to bear geometric methods of analysis

Role of graph signals

- Graph signals (a.k.a. node attributes or features) $\mathbf{X} \in \mathbb{R}^{N_v \times F}$



  - Ex: Age, gender in social network, fMRI signals, product ratings

- Embeddings capture structural and semantic graph information

$$\{\mathbf{W}, \mathbf{X}\} \mapsto \mathbf{Z}$$

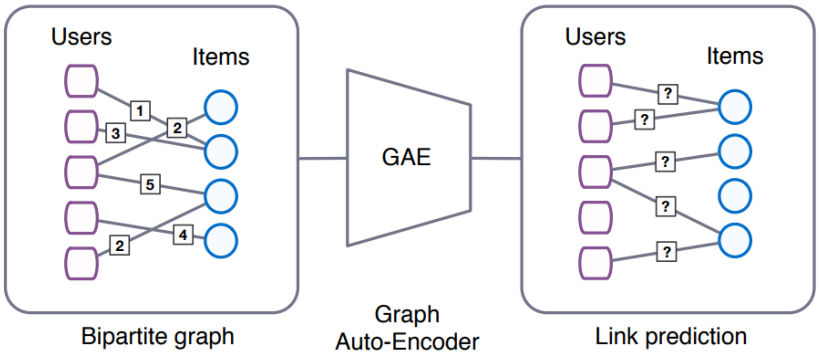
- Absent $\mathbf{X}$, the embedding $\{\mathbf{W}\} \mapsto \mathbf{Z}$ is termed featureless

  $\Rightarrow$ Mapping only preserves structural information

Transductive and inductive embeddings

Transductive network embedding

- Embed nodes within a fixed (often large) graph
  - Ex: Friend or product recommendation via link prediction
  - Ex: Node classification in semi-supervised learning



- Given new nodes, need to update and re-train the model

Inductive network embedding

- Learn mapping to representations that generalize to unseen graphs
  - Ex: Embed brain graphs for subject classification
  - Ex: Embed dynamic graphs for temporal clustering
- Signals **X** typically needed for inductive embedding

Unsupervised and supervised embeddings

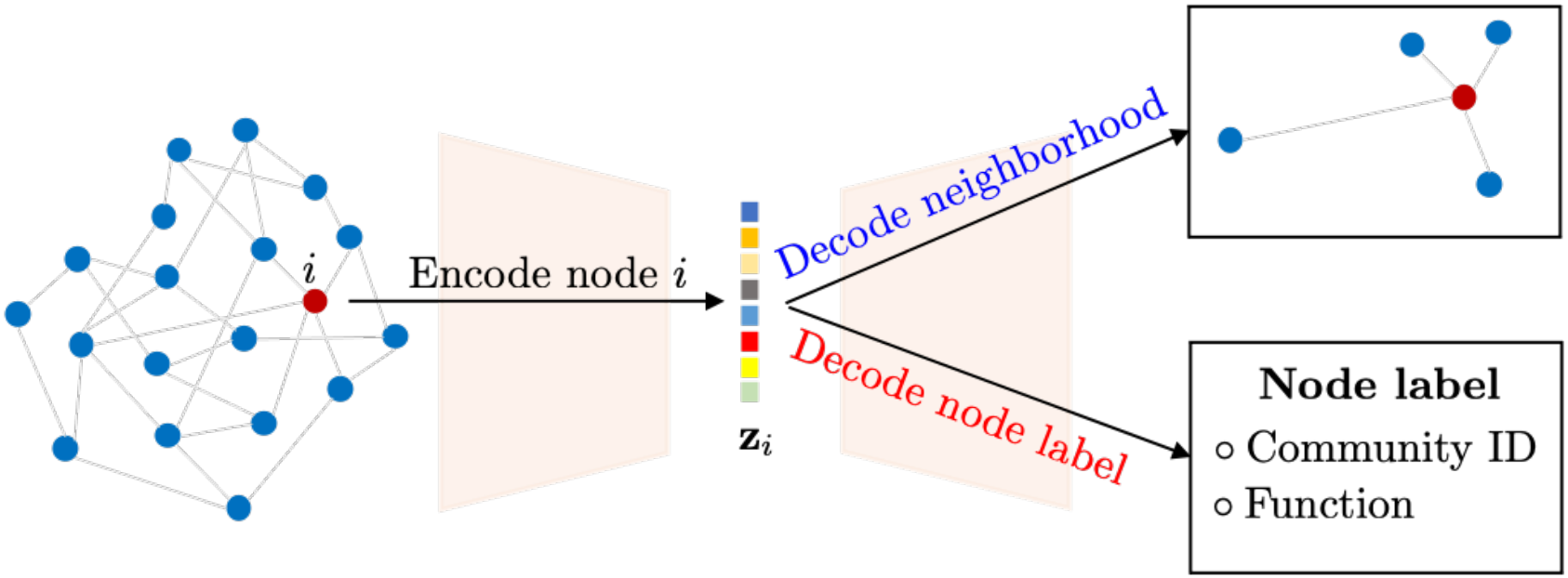
Unsupervised network embedding

- Only graph topology $\mathbf{W}$ is given
  - Preserve graph structures by optimizing a reconstruction loss
  - Decode embedding $\mathbf{Z}$ to approximate $\mathbf{W}$ well
- Ex: compression, visualization, clustering, link prediction

Supervised network embedding

- In addition to $\mathbf{W}$ (and $\mathbf{X}$), node or graph labels $\mathbf{y}^S$ available
  - Optimize embeddings for downstream tasks
  - Combine reconstruction and task-specific loss functions
- Ex: node classification, graph classification

An encoder-decoder perspective



W. L. Hamilton et al, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, 2018

Roadmap

The network embedding problem

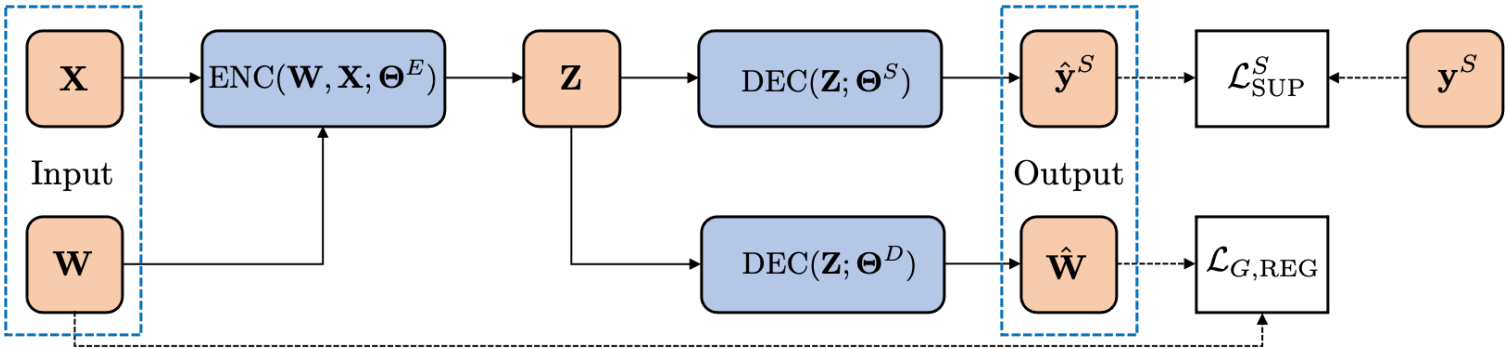
A taxonomy of graph embedding models

Unsupervised graph embedding

Applications

Encompassing graph embedding model

- Graph Encoder Decoder Model (GraphEDM)
  - $\Rightarrow$ Unifying framework to review and compare GRL methods
  - $\Rightarrow$ Open-source library with methods and applications



I. Chami et al, "Machine learning on graphs: A model and comprehensive taxonomy," *arXiv:2005.03675 [cs.LG]*, 2020

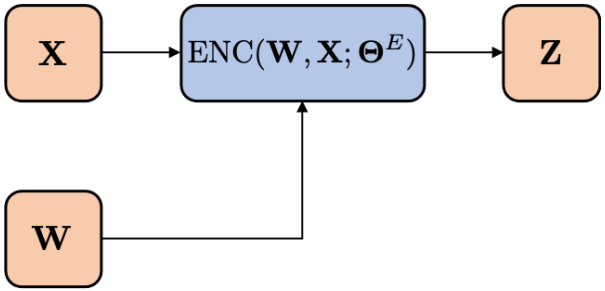
- **Q:** What are the framework's constituent components?

https://github.com/google/gcnn-survey-paper

Input

- Undirected graph $G(\mathcal{V}, \mathcal{E})$, with $|\mathcal{V}| = N_v$ and $|\mathcal{E}| = N_e$
  $\Rightarrow$ Weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{N_v \times N_v}$
- Optional graph signals (node features) $\mathbf{X} \in \mathbb{R}^{N_v \times F}$
- For (semi)-supervised learning tasks, also need target labels of:
  - Nodes (N), for node classification and clustering
  - Edges (E), for relationship classification or link prediction
  - Graphs (G), for graph clustering and classification
- Supervision signal (labels) denoted as $\mathbf{y}^S$, where $S \in \{N, E, G\}$

Encoder



- Graph encoder network

$$\text{ENC}_{\boldsymbol{\Theta}^E} : \mathbb{R}^{N_v \times N_v} \times \mathbb{R}^{N_v \times F} \mapsto \mathbb{R}^{N_v \times d}$$

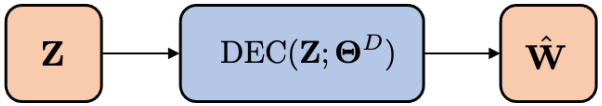
  $\Rightarrow$ Learnable parameters $\boldsymbol{\Theta}^E$

- Combines graph structure with graph signals to produce an embedding

$$\mathbf{Z} = \text{ENC}(\mathbf{W}, \mathbf{X}; \boldsymbol{\Theta}^E)$$

  $\Rightarrow$ Captures different graph properties based on type of supervision

Decoder



- Graph decoder network

$$\mathrm{DEC}_{\boldsymbol{\Theta}^D} : \mathbb{R}^{N_v \times d} \mapsto \mathbb{R}^{N_v \times N_v}$$

  $\Rightarrow$ Learnable parameters $\boldsymbol{\Theta}^D$

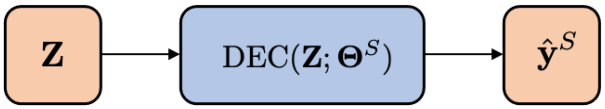
- Uses $\mathbf{Z}$ to produce (dis)similarity scores $\hat{W}_{ij}$ for all $\{i,j\} \in \mathcal{V}^{(2)}$

$$\hat{\mathbf{W}} = \mathrm{DEC}(\mathbf{Z};\boldsymbol{\Theta}^D)$$

  $\Rightarrow$ Unsupervised graph reconstruction

  $\Rightarrow$ Approximate $\mathbf{W}$ or general (dis)similarity matrix $s(\mathbf{W})$

Decoder



- Classification network

$$\mathrm{DEC}_{\boldsymbol{\Theta}^S} : \mathbb{R}^{N_v \times d} \mapsto \mathbb{R}^{N_v \times |\mathcal{Y}|}$$

$\Rightarrow$ Learnable parameters $\boldsymbol{\Theta}^S$, label space $\mathcal{Y}$

- Uses $\mathbf{Z}$ to produce node-wise distributions over labels

$$\hat{\mathbf{y}}^S = \mathrm{DEC}(\mathbf{Z};\boldsymbol{\Theta}^S)$$

$\Rightarrow$ (Semi)-supervised learning for node/graph classification

Output

- Reconstructed graph similarity matrix $\hat{\mathbf{W}} \in \mathbb{R}^{N_v \times N_v}$
  $\Rightarrow$ Used to train unsupervised embedding algorithms
- For (semi)-supervised learning tasks, outputs are predicted labels $\hat{\mathbf{y}}^S$
  - The label output space varies depending on the type of supervision
- Node-level: $\hat{\mathbf{y}}^N \in \mathcal{Y}^{N_v}$ or $\hat{\mathbf{Y}}^N \in [0,1]^{N_v \times |\mathcal{Y}|}$
  $\Rightarrow$ When $|\mathcal{Y}| = d$, can use softmax activation on $\mathbf{Z}$'s rows
- Edge-level: $\hat{\mathbf{Y}}^E \in \mathcal{Y}^{N_v \times N_v}$, where typically $\mathcal{Y} = \{0,1\}^{\#\text{relation types}}$
  $\Rightarrow$ When $\#\text{relation types} = 1$ (i.e., link prediction), output $\hat{\mathbf{W}}$
- Graph-level: $\hat{y}^G \in \mathcal{Y}$
  $\Rightarrow$ Using $\mathbf{W}$, convert $\mathbf{Z}$ to $\hat{y}^G$ via graph pooling

Loss functions

- Supervised loss

  $\Rightarrow \mathcal{L}_{\text{SUP}}^{S}$ compares predicted labels $\hat{\mathbf{y}}^S$ to ground truth $\mathbf{y}^S$

  Ex: semi-supervised node classification ($S = N$, $\mathcal{V} = \mathcal{V}_{obs} \cup \mathcal{V}_{miss}$)

$$\mathcal{L}_{\text{SUP}}^{N}(\mathbf{y}^N, \hat{\mathbf{y}}^N; \mathbf{\Theta}) = \sum_{i \in \mathcal{V}_{obs}} \ell(y_i^N, \hat{y}_i^N; \mathbf{\Theta})$$

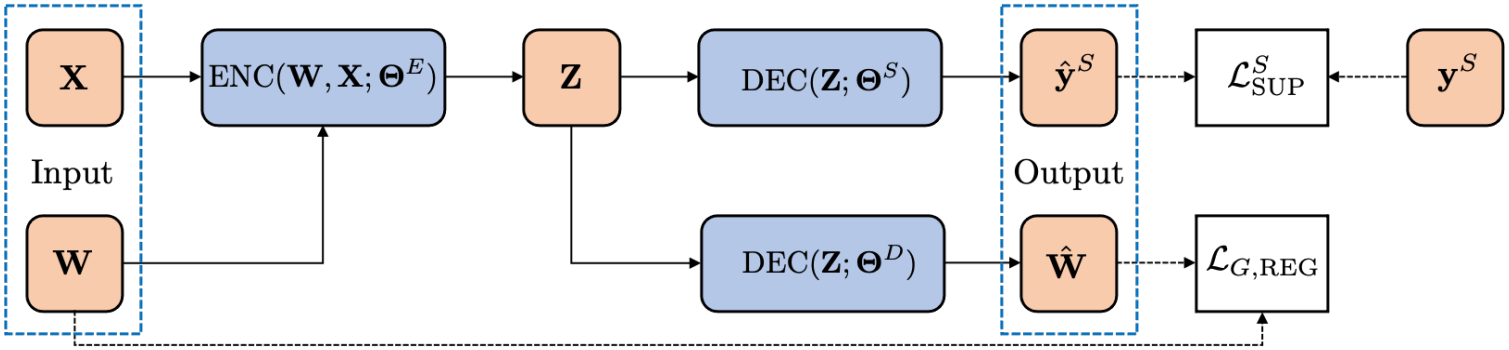
- Graph regularization loss

  $\Rightarrow \mathcal{L}_{G,\text{REG}}$ compares $\hat{\mathbf{W}}$ with target (dis)similarity matrix $s(\mathbf{W})$

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) = d_1(s(\mathbf{W}), \hat{\mathbf{W}})$$

  - $d_1(\cdot, \cdot)$: distance or dissimilarity function
  - Leverage G via $s(\mathbf{W})$ to regularize model parameters $\mathbf{\Theta}$

Objective function



- Weight regularization loss

  $\Rightarrow$ $\mathcal{L}_{\mathrm{REG}}$ regularizes trainable parameters $\mathbf{\Theta}$ to reduce overfitting

$$\mathcal{L}_{\mathrm{REG}}(\mathbf{\Theta}) = \sum_{\boldsymbol{\theta} \in \mathbf{\Theta}} \|\boldsymbol{\theta}\|_2^2$$

- Overall GraphEDM objective function

$$\mathcal{L}(\mathbf{\Theta}) = \alpha \mathcal{L}^S_{\mathrm{SUP}}(\mathbf{y}^S, \hat{\mathbf{y}}^S; \mathbf{\Theta}) + \beta \mathcal{L}_{G,\mathrm{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) + \mathcal{L}_{\mathrm{REG}}(\mathbf{\Theta})$$

  $\Rightarrow$ Train in a supervised ($\alpha \neq 0$) or unsupervised ($\alpha = 0$) fashion

- **Q**: End-to-end supervised learning or two-step learning?

Graph embedding taxonomy

- Categorize GRL methods based on encoder and loss function used
- Shallow embedding methods $\mathbf{Z} = \text{ENC}(\mathbf{\Theta}^E) = \mathbf{\Theta}^E$
  - A simple embedding lookup
- Graph auto-encoding methods $\mathbf{Z} = \text{ENC}(\mathbf{W}; \mathbf{\Theta}^E)$
  - Transductive like shallow embeddings, no $\mathbf{X}$ so works for fixed G
- Graph regularization methods $\mathbf{Z} = \text{ENC}(\mathbf{X}; \mathbf{\Theta}^E)$
  - Leverage $\mathbf{W}$ via $\mathcal{L}_{G,\text{REG}}$ to regularize node embeddings
- Neighborhood aggregation methods $\mathbf{Z} = \text{ENC}(\mathbf{W}, \mathbf{X}; \mathbf{\Theta}^E)$
  - Use $\mathbf{W}$ to propagate information among nodes and learn $\mathbf{Z}$

Roadmap

The network embedding problem

A taxonomy of graph embedding models

Unsupervised graph embedding

Applications

Unsupervised graph embedding

- **Goal:** Learn node embeddings that preserve graph structure
- Optimize to reconstruct some node (dis)similarity matrix $s(\mathbf{W})$

$$\mathcal{L}(\mathbf{\Theta}) = \alpha \mathcal{L}_{\text{SUP}}^{S}(\mathbf{y}^{S}, \hat{\mathbf{y}}^{S}; \mathbf{\Theta}) + \beta \mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) + \mathcal{L}_{\text{REG}}(\mathbf{\Theta}), \quad \alpha = 0$$

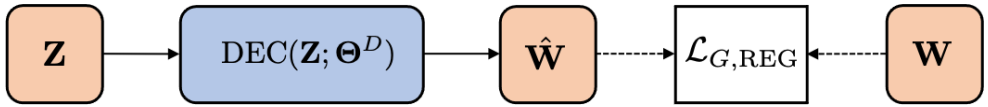
  - Decoder network outputs $\hat{\mathbf{W}}$, with $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j)$
  - Graph regularization loss $\mathcal{L}_{G,\text{REG}} = d_1(s(\mathbf{W}), \hat{\mathbf{W}})$
  - Optimize over training set $\{i, j\} \in \mathcal{V}_{obs}^{(2)}$, use SGD or spectral methods
- Target pairwise node similarity matrix $s(\mathbf{W})$ can take many forms

  Ex: Reconstruct first-order proximity via $[s(\mathbf{W})]_{ij} = W_{ij}$

  Ex: Higher-order proximity $[s(\mathbf{W})]_{ij} = |\mathcal{N}_i \cap \mathcal{N}_j|$, Jaccard, Adamic-Adar

  Ex: Prob. $[s(\mathbf{W})]_{ij} = \mathrm{P}\left(v_j \,\middle|\, v_i\right)$ that $i, j \in \mathcal{V}$ co-occur on random walks

Shallow embeddings



- Shallow embedding methods

$$\mathbf{Z} = \mathrm{ENC}(\mathbf{\Theta}^E) = \mathbf{\Theta}^E \in \mathbb{R}^{N_v \times d}$$

  $\Rightarrow$ A simple embedding lookup, optimize $\mathbf{Z}$ directly

- Two classess based on the type of decoder $\hat{\mathbf{W}} = \mathrm{DEC}(\mathbf{Z}; \Theta^D)$
  - Distance-based methods: $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j)$
  - Outer product-based methods: $\hat{\mathbf{W}} = \mathbf{Z}\mathbf{Z}^\top \Rightarrow \hat{W}_{ij} = \mathbf{z}_i^\top \mathbf{z}_j$

- Inspired by dimensionality reduction via low-rank matrix factorization

  $\Rightarrow$ More recent approaches rely on random walks (NLP analogies)

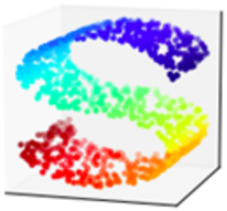
Distance-based methods

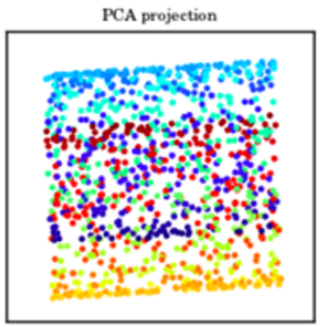
- **Idea:** embeddings preserve distances in G [encoded in $s(\mathbf{W})$]

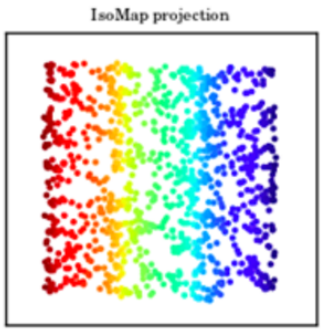
$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \boldsymbol{\Theta}) = d_1(s(\mathbf{W}), \hat{\mathbf{W}}) = \sum_{i,j \in \mathcal{V}_{obs}^{(2)}} ([\mathrm{s}(\mathbf{W})]_{ij} - \hat{W}_{ij})^2$$

$\Rightarrow$ Euclidean distance decoder: $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j) = \|\mathbf{z}_i - \mathbf{z}_j\|_2$

- Multi-dimensional scaling (MDS) preserves local connectivity
  - Set $[s(\mathbf{W})]_{ij} = 1$ (or W_{ij}) if $W_{ij} > 0$ and 0 otherwise
- IsoMAP preserves global geodesic distances in the manifold
  - Set e.g., $[s(\mathbf{W})]_{ij} = d_G(i, j)$, shortest-path distance between $i, j \in \mathcal{V}$







Laplacian eigenmaps

- Capture information in G via spectral properties of $\mathbf{L} = \mathbf{D} - \mathbf{W}$

  $\Rightarrow$ Locality-preserving nonlinear dimensionality reduction scheme

$$\min_{\mathbf{Z} \in \mathbb{R}^{N_v \times d}} \text{trace}(\mathbf{Z}^\top \mathbf{L} \mathbf{Z}), \quad \text{s. to } \mathbf{Z}^\top \mathbf{D} \mathbf{Z} = \mathbf{I}$$

- Equivalently written as a graph regularization term

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) = d_1(s(\mathbf{W}), \hat{\mathbf{W}}) = \sum_{i,j \in \mathcal{V}_{obs}^{(\mathbf{2})}} W_{ij} \hat{W}_{ij}^2$$

  $\Rightarrow$ Euclidean distance decoder: $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j) = \|\mathbf{z}_i - \mathbf{z}_j\|_2$

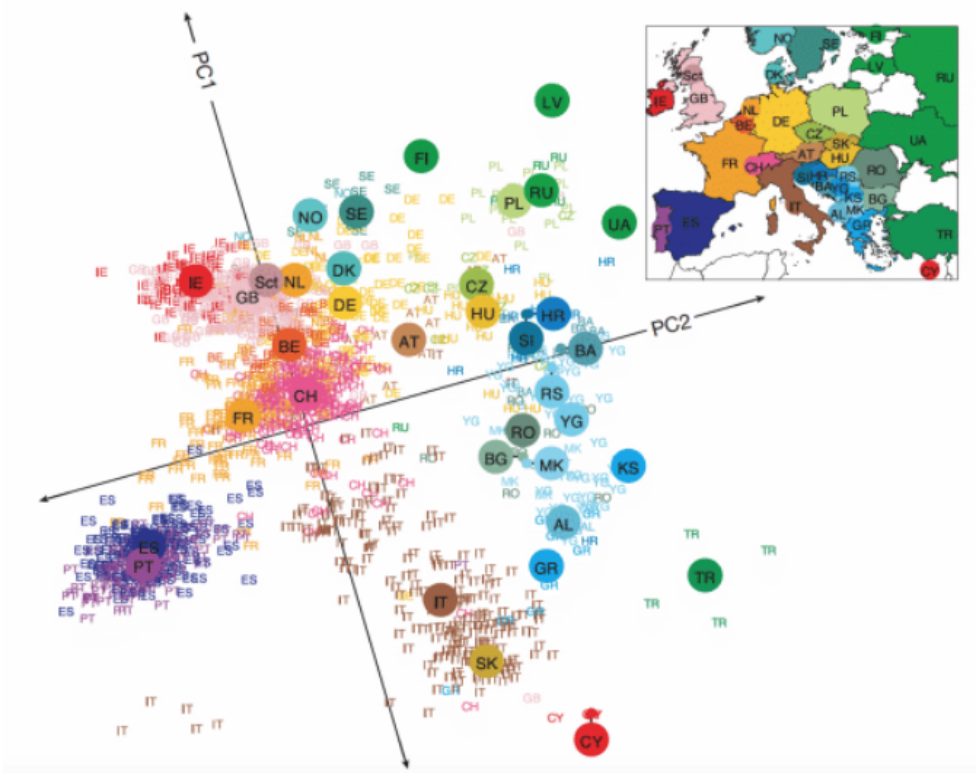
  $\Rightarrow$ Embeddings close in $\mathbb{R}^d$ if i, j well connected in G

M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, 2003.

Gene cartography

- Ex: Spectral embedding of 'gene similarity' matrix ($d = 2$)

  $\Rightarrow$ Consistent with origins of individuals in European map



J. Novembre, "Genes mirror geography within Europe," *Nature*, 2008

Non-Euclidean embedding spaces

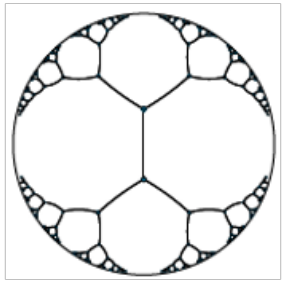
- **Idea:** embed graphs with hierarchical structure into hyperbolic space

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \boldsymbol{\Theta}) = d_1(s(\mathbf{W}), \hat{\mathbf{W}}) = -\sum_{i,j \in \mathcal{V}_{obs}^{(2)}} W_{ij} \log \frac{e^{-\hat{W}_{ij}}}{\sum_{k|\mathbf{W}_{ik}=0} e^{-\hat{W}_{ik}}}$$

$\Rightarrow$ Poincaré distance decoder:

$$\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j) = \text{arcosh}\left(1 + 2\frac{\|\mathbf{z}_i - \mathbf{z}_j\|_2^2}{(1 - \|\mathbf{z}_i\|_2^2)(1 - \|\mathbf{z}_i\|_2^2)}\right)$$

- Capture similiarty and hierarchy
- Use Riemannian optimization tools



M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," *NeurIPS,* 2017

Matrix factorization methods

- **Idea:** learn low-rank representation of similarity matrix $s(\mathbf{W})$

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) = d_1(s(\mathbf{W}), \hat{\mathbf{W}}) = \|s(\mathbf{W}) - \hat{\mathbf{W}}\|_F^2$$

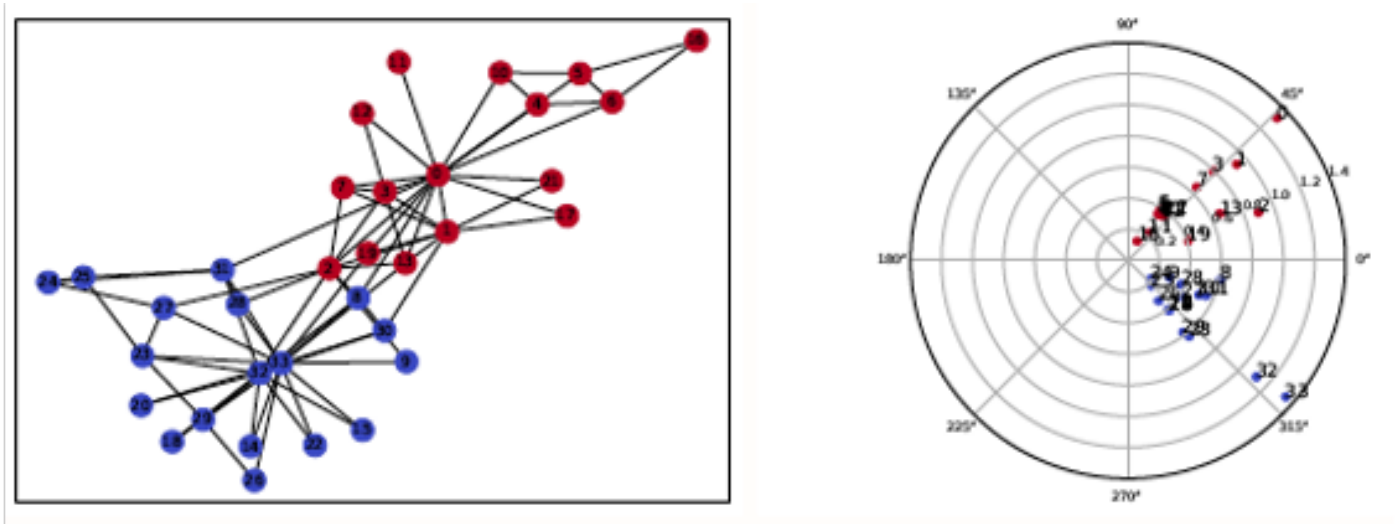
  $\Rightarrow$ Outer product decoder: $\hat{\mathbf{W}} = \text{DEC}(\mathbf{Z}; \mathbf{\Theta}^D) = \mathbf{Z}\mathbf{Z}^\top$

  $\Rightarrow$ Implies an inner-product approximation $[s(\mathbf{W})]_{ij} \approx \mathbf{z}_i^\top \mathbf{z}_j$

- Graph factorization (GF) preserves first-order similarity in G
  - Set $[s(\mathbf{W})]_{ij} = W_{ij}$ and evaluate $\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta})$ on $(i,j) \in \mathcal{E}$
- GraRep preserves higher-order similarity in G
  - Set e.g., $[s(\mathbf{W})]_{ij} = [\mathbf{W}^k]_{ij}$, $k \geq 2$, for length-k path counts
- HOPE preserves general similarity measures in (directed) G
  - Jaccard, Adamic-Adar and related neighborhood scores
- Closely related to adjacency spectral embedding (ASE) for RDPGs

Zachary's karate club

- Ex: Zachary's karate club graph with $N_v = 34$, $N_e = 78$ (left)



- ASE node embeddings (rows of $\mathbf{Z}$) for $d = 2$ (right)
  - Club's administrator ($i = 0$) and instructor ($j = 33$) are orthogonal
- Interpretability of embeddings a valuable asset for RDPGs

  $\Rightarrow$ Vector magnitudes indicate how well connected nodes are

  $\Rightarrow$ Vector angles indicate positions in latent space

From texts to graphs

- Permeate advances in language modeling and feature learning in NLP
  - Ex: Skip-gram neural network model for word2vec embeddings
  - From text corpora (word sequences) to graphs (node sequences)
- **Idea:** similar $\mathbf{z}_i$ to nodes that tend to co-occur in random walks over G
- View sentences in NLP as random walks over the vocabulary
  - Generate short random walks on G to sample node sequences
  - Learn node positional distributions just like words [Perozzi et al'14]
- Prob. $\mathrm{P}\left(j \,\middle|\, i\right)$ of visiting j in a length-T random walk from i

  $\Rightarrow$ Asymmetric similarity measure $[s(\mathbf{W})]_{ij}$ to decode from $\mathbf{Z}$

Random walk approaches

- Training pairs $\{i, j\} \in \mathcal{V}_{obs}^{(2)}$ sampled from short random walks
  - For each $i \in \mathcal{V}$, N pairs $\{i, j_1\}, \ldots, \{i, j_N\}$ sampled from $\mathrm{P}\left(j \,\middle|\, i\right)$
  - Length of each walk is $T \in \{2, \ldots, 10\}$
- Cross-entropy loss as graph regularization term

$$\mathcal{L}_{G,\mathrm{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \boldsymbol{\Theta}) = - \sum_{i,j \in \mathcal{V}_{obs}^{(2)}} \log \hat{W}_{ij}$$

$\Rightarrow$ Composition of softmax and outer product decoder

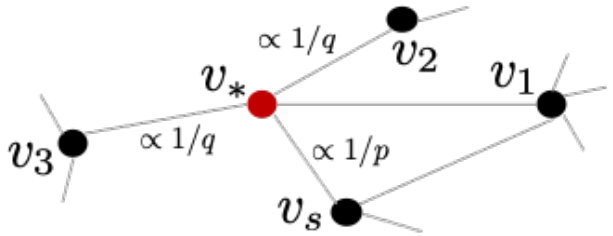
$$\hat{W}_{ij} = \frac{e^{\mathbf{z}_i^\top \mathbf{z}_j}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}}$$

$\Rightarrow$ Implies an approximation $\hat{W}_{ij} \approx [s(\mathbf{W})]_{ij} = \mathrm{P}\left(j \,\middle|\, i\right)$

- Evaluating the softmax denominator is challenging ($\mathcal{O}(N_v)$ complexity)

DeepWalk and node2vec

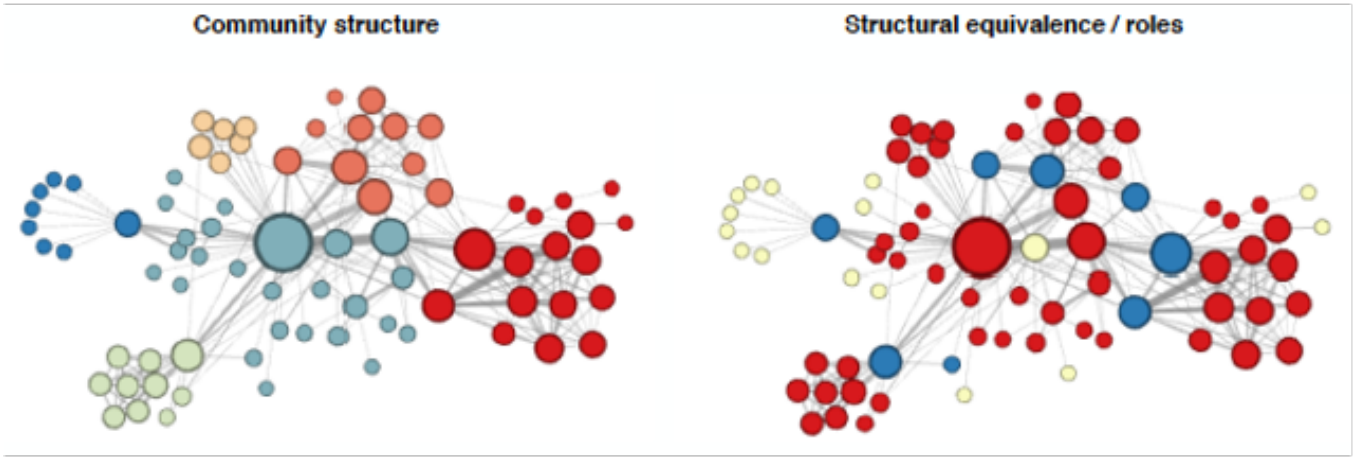
- DeepWalk samples unbiased random walks
  - Transition probability matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$
  - Hierchical softmax technique to form $\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}$ using binary trees
- Node2Vec offers a flexible definition of (biased) random walks
  - Smoothly interpolates between walks akin to BFS or DFS
  - Effective for capturing structural roles or community structures
  - Approximates $\sum_{k \in \mathcal{V}^*} e^{\mathbf{z}_i^\top \mathbf{z}_k}$ via samples $\mathcal{V}*$



- Hyperparameters p (return) and q (in-out). After $v_s \to v_*$
  - (i) Control probability of revisiting nodes ($v_* \to v_s$); or
  - (ii) Staying close to the preceding node ($v_* \to v_1$); or
  - (iii) Moving outward farther away ($v_* \to \{v_2, v_3\}$)

Biased random walks

- Ex: character interaction graph from the novel 'Les Miserables'



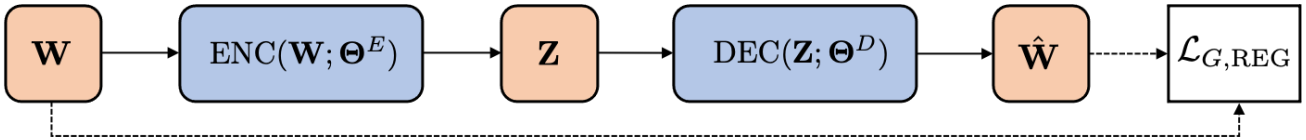
- node2vec interpolates between capturing global and local structure
  - Left coloring indicates membership to communities (global positions)
  - Right coloring indicates roles played within (local) neighborhoods

A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *KDD*, 2016

Limitations

- Shallow embeddings: encoder a simple embedding lookup
  - $\Rightarrow$ Directly optimizes a unique embedding $\mathbf{z}_i$ for each node $i \in \mathcal{V}$
- No parameters sharing between nodes in the encoder
  - Statistically inefficient, parameter sharing can act as a regularizer
  - Computationally inefficient, number of parameters is $\mathcal{O}(N_v)$
- Fails to leverage graph signals during encoding
  - Attributes highly informative w.r.t. the node's position and role in G
- Inherently transductive
  - Challenge for dynamic networks or large graphs not stored in memory
  - Does not generalize to other graphs beyond G (used for training)

Autoencoders



- Autoencoders

$$\mathbf{Z} = \text{ENC}(\mathbf{W};\mathbf{\Theta}^E)$$

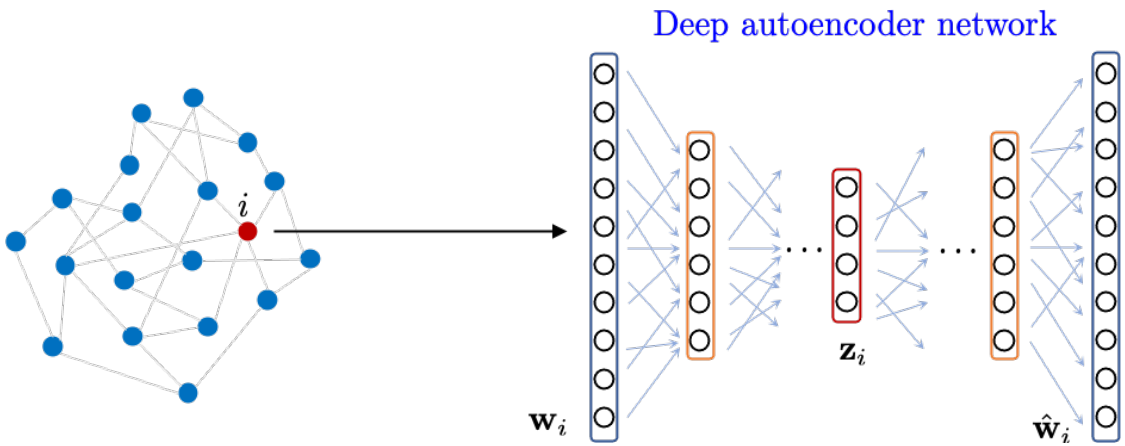
  $\Rightarrow$ Directly incorporate $\mathbf{W}$ into the encoder

- Use deep neural network encoder and decoder functions
  - $\Rightarrow$ Ability to model non-linearities
  - $\Rightarrow$ Leads to more complex representations
- Models trained by minimizing a reconstruction error objective

Neighborhood autoencoder methods

- Let $\mathbf{w}_i \in \mathbb{R}^{N_v}$ denote the i-th column of $\mathbf{W}$

  $\Rightarrow$ Captures neighborhood information of $i \in \mathcal{V}$



- Autoencoder objective: reconstruct $\mathbf{w}_i$ from learnt embedding $\mathbf{z}_i$

Structural deep network embedding

- Structural deep network embedding (SDNE) minimizes

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) = \sum_i \|\mathbf{w}_i - \hat{\mathbf{w}}_i\|_2^2 + \gamma \sum_{i,j} W_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2$$

  $\Rightarrow$ Incorporates the Laplacian eigenmaps objective

- Uses deep autoencoders per node (shared parameters)

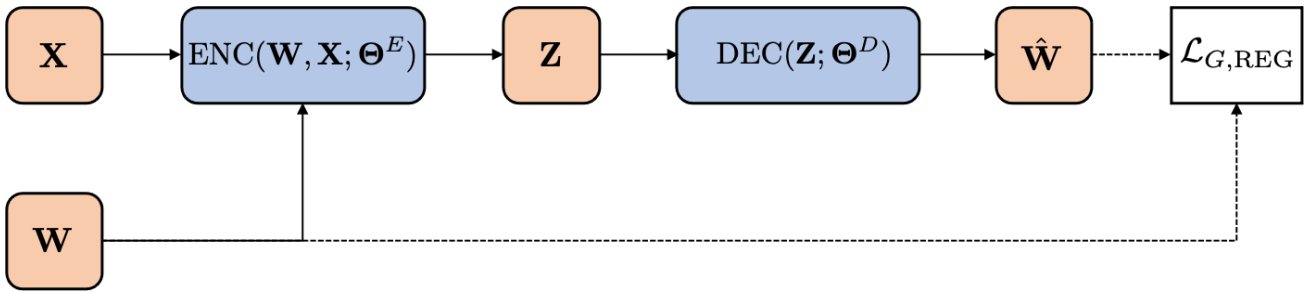
$$\mathbf{z}_i = \text{ENC}(\mathbf{w}_i; \mathbf{\Theta}^E), \quad \hat{\mathbf{w}}_i = \text{DEC}(\mathbf{z}_i; \mathbf{\Theta}^D)$$

  - Via $\mathbf{w}_i$, encoder regularized with G's topology

- Drawback: input dimension fixed to N_v, costly for large G

D. Wang et al, "Structural deep network embedding," *KDD*, 2016

Graph neural networks



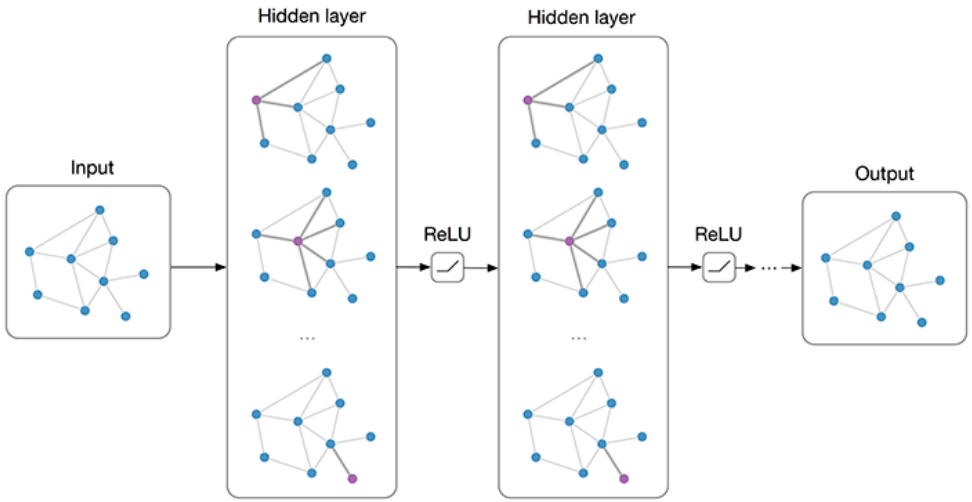
- Graph Neural Networks

$$\mathbf{Z} = \mathrm{ENC}(\mathbf{W}, \mathbf{X}; \mathbf{\Theta}^E)$$

$\Rightarrow$ Use graph signals $\mathbf{X}$ and topology $\mathbf{W}$ in encoder function

- Generate embedding $\mathbf{z}_i$ by aggregating signals within $\mathcal{N}_i$
  - Convolutional: local and distributed implementation
  - Efficiency: parameter dimensions independent of N_v
  - Regularization: effected via parameter sharing across nodes
  - Inductive: generate embeddings for nodes not seen in training

Convolutional graph autoencoders



- Graph autoencoders (GAE) use a GCN encoder to learn embeddings

$$\mathbf{Z} = \text{GCN}(\mathbf{W}, \mathbf{X}; \mathbf{\Theta}^E)$$

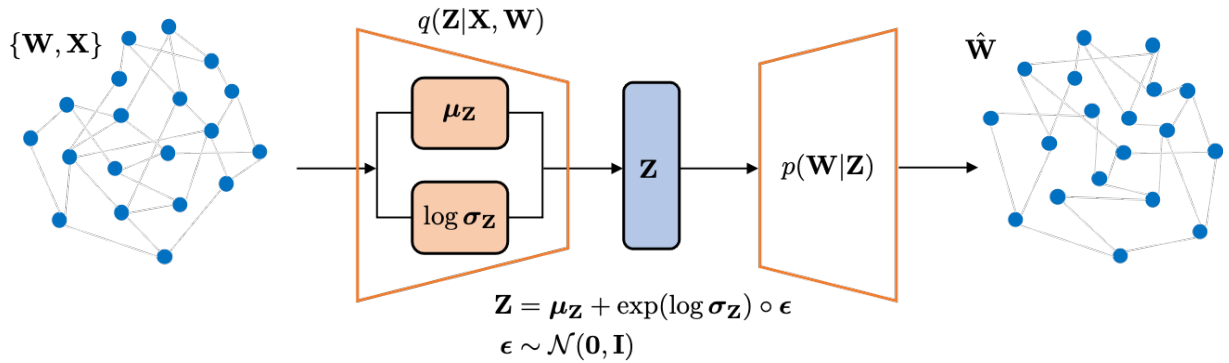
- Sigmoid cross entropy loss between $\mathbf{W}$ and decoder output $\hat{\mathbf{W}}$

$$\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta}) = -\sum_{i,j \in \mathcal{V}_{obs}^{(\mathbf{2})}} (1 - W_{ij}) \log(1 - \sigma(\hat{W}_{ij})) + W_{ij} \log \sigma(\hat{W}_{ij})$$

  $\Rightarrow$ Outer product decoder: $\hat{\mathbf{W}} = \text{DEC}(\mathbf{Z}; \mathbf{\Theta}^D) = \mathbf{Z}\mathbf{Z}^\top$

  $\Rightarrow$ Non-probabilistic model, suitable for unweighted graphs

Variational graph autoencoders



- **Goal:** train a probablistic decoder to generate realistic graphs

$$\hat{\mathbf{W}} \sim p(\mathbf{W} \mid \mathbf{Z})$$

given latent variables from a probabilistic encoder $\mathbf{Z} \sim q(\mathbf{Z} \mid \mathbf{X}, \mathbf{W})$

- Minimize reconstruction error given training graphs and signals
- Post training, drop the encoder and generate graphs $\hat{\mathbf{W}} \sim p(\mathbf{W} \mid \mathbf{Z})$
  - Given latent variables $\mathbf{Z} \sim p(\mathbf{Z})$ sampled from a prior distribution

T. N. Kipf and M. Welling, "Variational graph auto encoders," *arXiv:1611.07308 [stat.ML]*, 2016

Probabilistic encoder and decoder

- Encoder: simple inference model parameterized by GCNs

$$q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{W}) = \prod_{i \in \mathcal{V}} q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{W}), \text{ with } q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{W}) = \mathcal{N}(\mathbf{z}_i; \boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2))$$

  - Two separate GCNs to generate mean and variance parameters

$$\boldsymbol{\mu}_{\mathbf{Z}} = \text{GCN}_{\mu}(\mathbf{W}, \mathbf{X}), \quad \log \boldsymbol{\sigma}_{\mathbf{Z}} = \text{GCN}_{\sigma}(\mathbf{W}, \mathbf{X})$$

  - Given $\boldsymbol{\mu}_{\mathbf{Z}}$ and $\log \boldsymbol{\sigma}_{\mathbf{Z}}$, sample latent embeddings via

$$\mathbf{Z} = \boldsymbol{\mu}_{\mathbf{Z}} + \exp(\log \boldsymbol{\sigma}_{\mathbf{Z}}) \circ \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

- Decoder: generative model based on outer product decoder

$$p(\mathbf{W} \,|\, \mathbf{Z}) = \prod_{i,j \in \mathcal{V}^{(2)}} p(W_{ij} \,|\, \mathbf{z}_i, \mathbf{z}_j), \text{ with } p(W_{ij} = 1 \,|\, \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^{\top} \mathbf{z}_j)$$

  - $\sigma(\cdot)$ stands for the logistic sigmoid function

- Prior: latent node embeddings assumed $\mathbf{z}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$

Training the VGAE model

- Maximize the evidence likelihood lower bound (ELBO)

$$\mathcal{L}(\mathbf{\Theta}) = \sum_i \mathbb{E}_{q(\mathbf{Z}|\mathbf{X}_i,\mathbf{W}_i)} \left[\log p(\mathbf{W}_i \,|\, \mathbf{Z})\right] - \mathrm{KL}(q(\mathbf{Z} \,|\, \mathbf{X}_i, \mathbf{W}_i), p(\mathbf{Z}))$$

  - $\mathrm{KL}(\cdot,\cdot)$ stands for Kullback-Leibler divergence
  - Learnable parameters $\mathbf{\Theta}$ are the GCN filters $\mathbf{H}_k$
  - Requires a set of training graphs $\{\mathbf{W}_1, \mathbf{X}_1\}, \ldots, \{\mathbf{W}_P, \mathbf{X}_P\}$

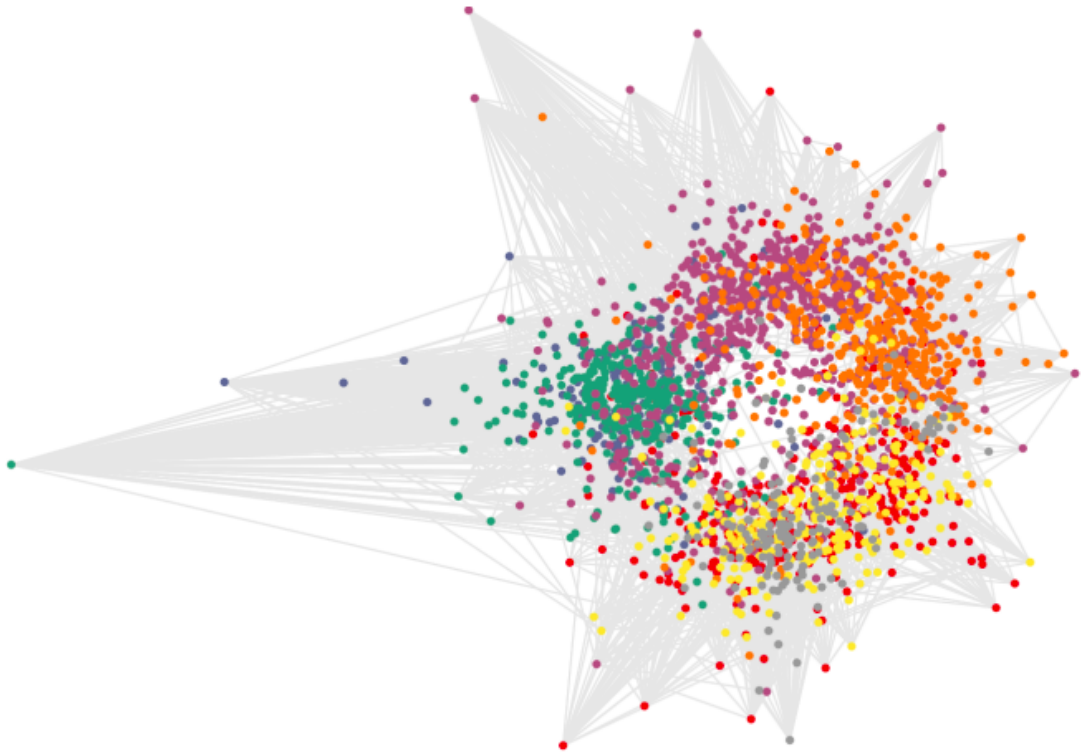
- Generate a distribution over $\mathbf{Z}$ to satisfy two (conflicting) goals
  (a) Sampled $\mathbf{Z}$ rich enough for the decoder to reconstruct $\mathbf{W}$
  (b) Distribution $q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{W})$ is as close as possible to the prior $p(\mathbf{Z})$

- Goal (b) is critical to generate new graphs after training

  $\Rightarrow$ Sample $\mathbf{Z} \sim p(\mathbf{Z}) \rightarrow$ Decode $\hat{\mathbf{W}} \sim p(\mathbf{W}_i \,|\, \mathbf{Z})$

VGAE in action

- Ex: Cora dataset with $N_v = 2708$ papers and $N_e = 5429$ citations
  - Graph signals: presence/absence of 1433 words from dictionary



- Learned latent space using the VGAE model for a link prediction task

  $\Rightarrow$ Colors indicate class labels (discipline, not used for training)

Roadmap

The network embedding problem

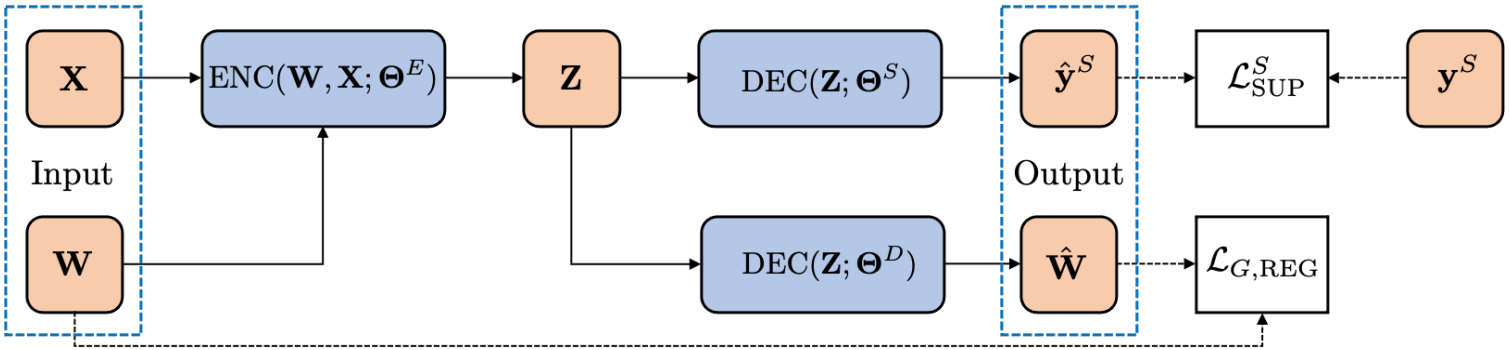
A taxonomy of graph embedding models

Unsupervised graph embedding

Applications

Applications

- GRL has been succesfully applied in a wide range of domains
  - Unsupervised learning to preserve graph structure
  - Supervised learning for prediction or classification



- Ex: brain network analysis for patient-control study
  - (Un)supervised GRL for graph reconstruction and classification
- Ex: social network analysis for temporal graph clustering
  - Unsupervised setting to learn graph-level representation

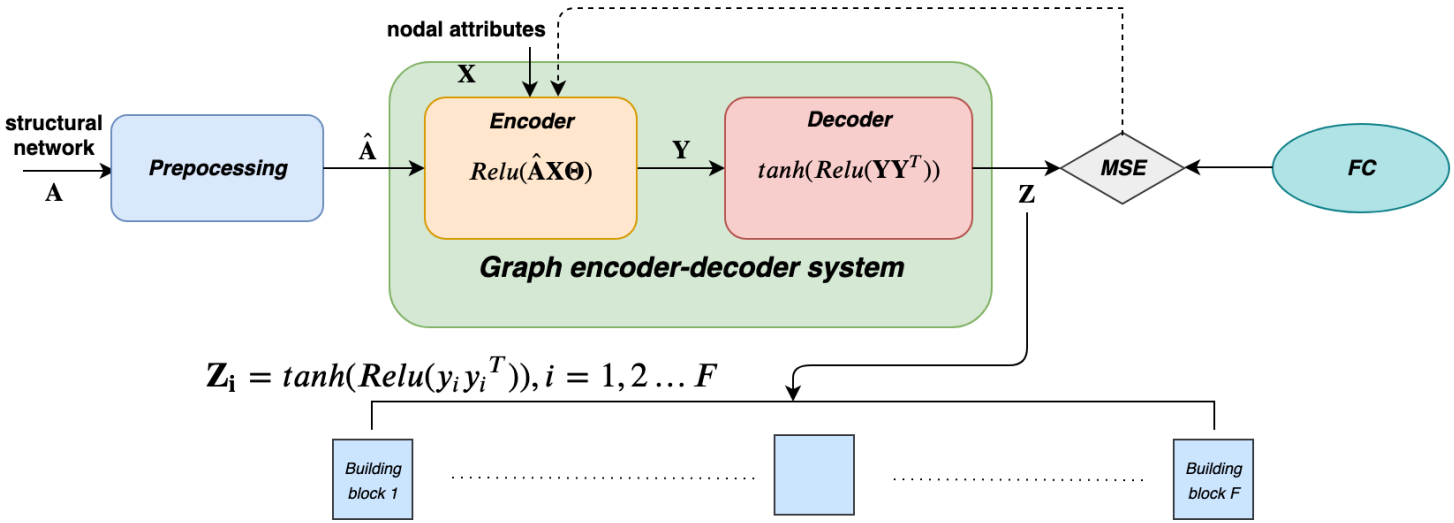
Networks of the brain

- Challenge: understand human brain function and structure
  - Neuroimaging advances $\Rightarrow$ Data increase in volume and complexity
  - Graph-centric analysis and methods of network science [Sporns'10]
- Brain networks can reflect two connectivity patterns
  - Structural connectivity (SC). How is the brain wired?
    $\Rightarrow$ Anatomical tracts connecting brain regions (DTI)
  - Functional connectivity (FC). How the brain functions?
    $\Rightarrow$ Correlation between neural signals in different regions (fMRI)
- Key problem: deciphering the relationship between SC and FC
  - Simulations of nonlinear cortical activity models [Honey et al'09]
  - Diffusion-based parametric inverse problem [Abdelnour et al'14]
  - Network deconvolution [Li-Mateos'19]
- **Goal:** pursue SC-to-FC mapping as a regression problem
  $\Rightarrow$ Reconstruct FC network from SC network

Problem statement

Study the generation of FC patterns from SC graphs

- **Goal**: learn the mapping from brain SC networks to FC networks
- Approach: reconstruct FC networks from the given SC networks
- Model: GCN-based encoder-decoder system



- Analysis: investigate latent variables within the system

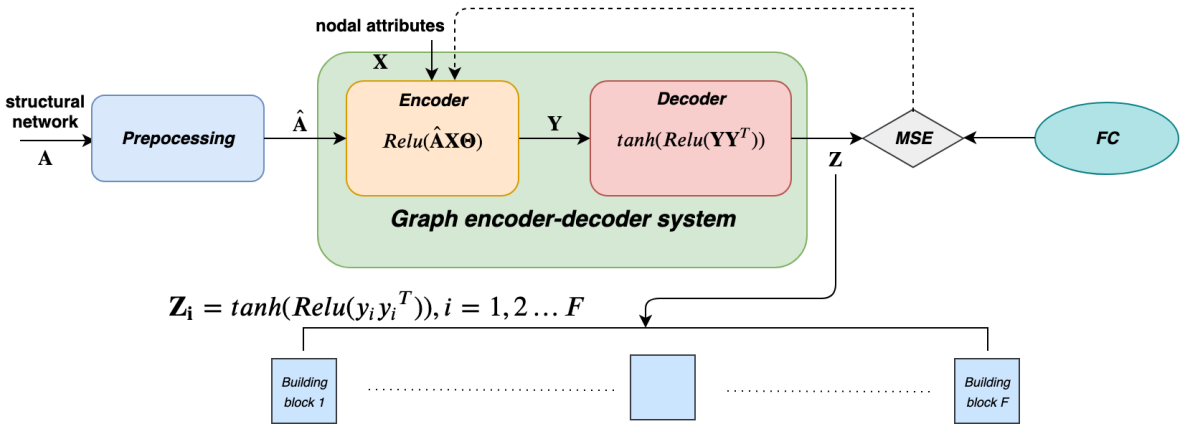
Model architecture: encoder

- Input SC network $\mathbf{A} \in \mathbb{R}^{N \times N}$, N regions from brain atlas
  - $\Rightarrow$ Edge weights represent SC between brain regions
  - $\Rightarrow$ Preprocessing: $\hat{\mathbf{A}} := \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{A}$
- Learn vertex representations (i.e., embeddings) that capture
  (i) Nodal attributes, e.g. intrinsic properties of brain regions
  (ii) Graph topology information, e.g. regional connection strengths
- A single-layer GCN used for encoder to learn node embeddings

$$\mathbf{Y} = \mathrm{ReLU}(\hat{\mathbf{A}}\mathbf{X}\boldsymbol{\Theta}) \in \mathbb{R}^{N \times F}$$

  - $\mathbf{X} \in \mathbb{R}^{N \times T}$: input signal matrix
  - $\boldsymbol{\Theta} \in \mathbb{R}^{T \times F}$: learnable GCN filter coefficients
  - $\mathrm{ReLU}(x) = \max(0, x)$ activation for training the network

Model architecture: decoder

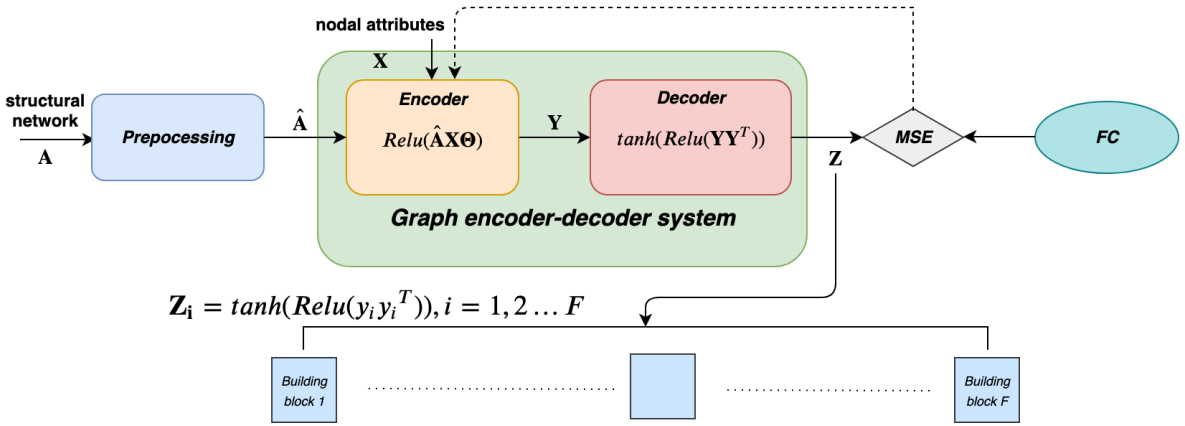


- Node embeddings $\mathbf{Y} \in \mathbb{R}^{N \times F}$ go through the outer-product decoder

$$\mathbf{Z} = \tanh(\mathrm{ReLU}(\mathbf{Y}\mathbf{Y}^T)) \in \mathbb{R}^{N \times N}$$

- Weights in empirical FC networks restricted to [0,1]
  - Ensure the output of the decoder in the same range
  - Choose tanh and ReLU over sigmoid
- Loss function: MSE between $\mathbf{Z}$ and empirical FC

Model architecture: latent variables



- YY^T: rank-F approximation of FC graph before activation
  - Extract and analyze each of the rank-1 components $y_i y_i^T$

$$Z_i = \tanh(\mathrm{ReLU}(y_i y_i^T)), \quad i = 1, \ldots, F$$

- $Z_i \Leftrightarrow$ outputs of individual filters in graph convolutional layer

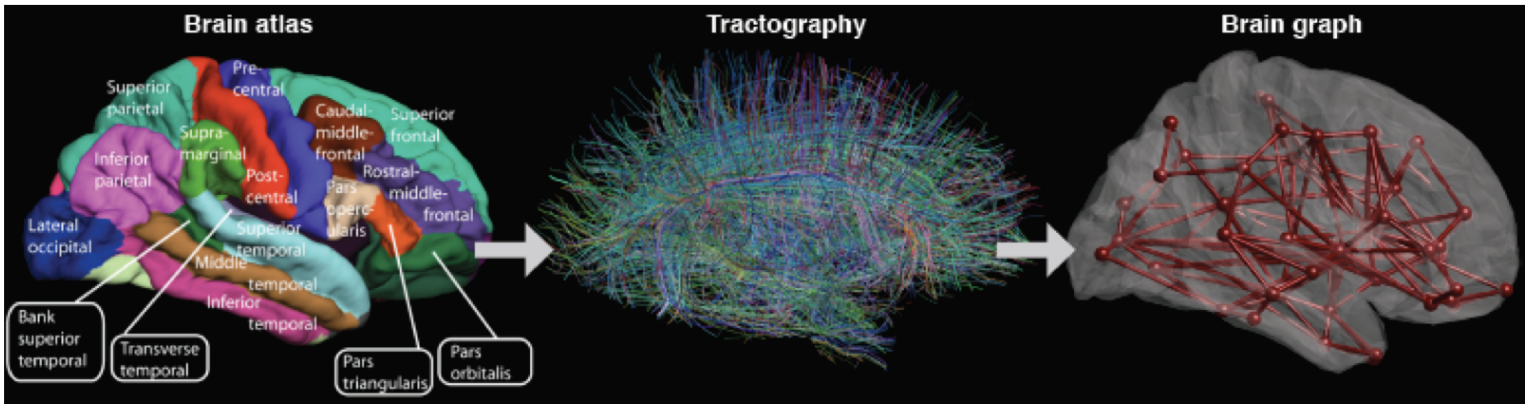
  $\Rightarrow$ View as building blocks of FC network

- Reveal details about generation of FC patterns from SC networks

Numerical tests: data

- $P = 1058$ healthy subjects from Human Connectome Project (HCP)
- Preprocessed SC network $\mathbf{A}$ from diffusion MRI

  $\Rightarrow$ Fiber counts between $N = 68$ cortical surface regions



- Preprocessed FC network from functional MRI

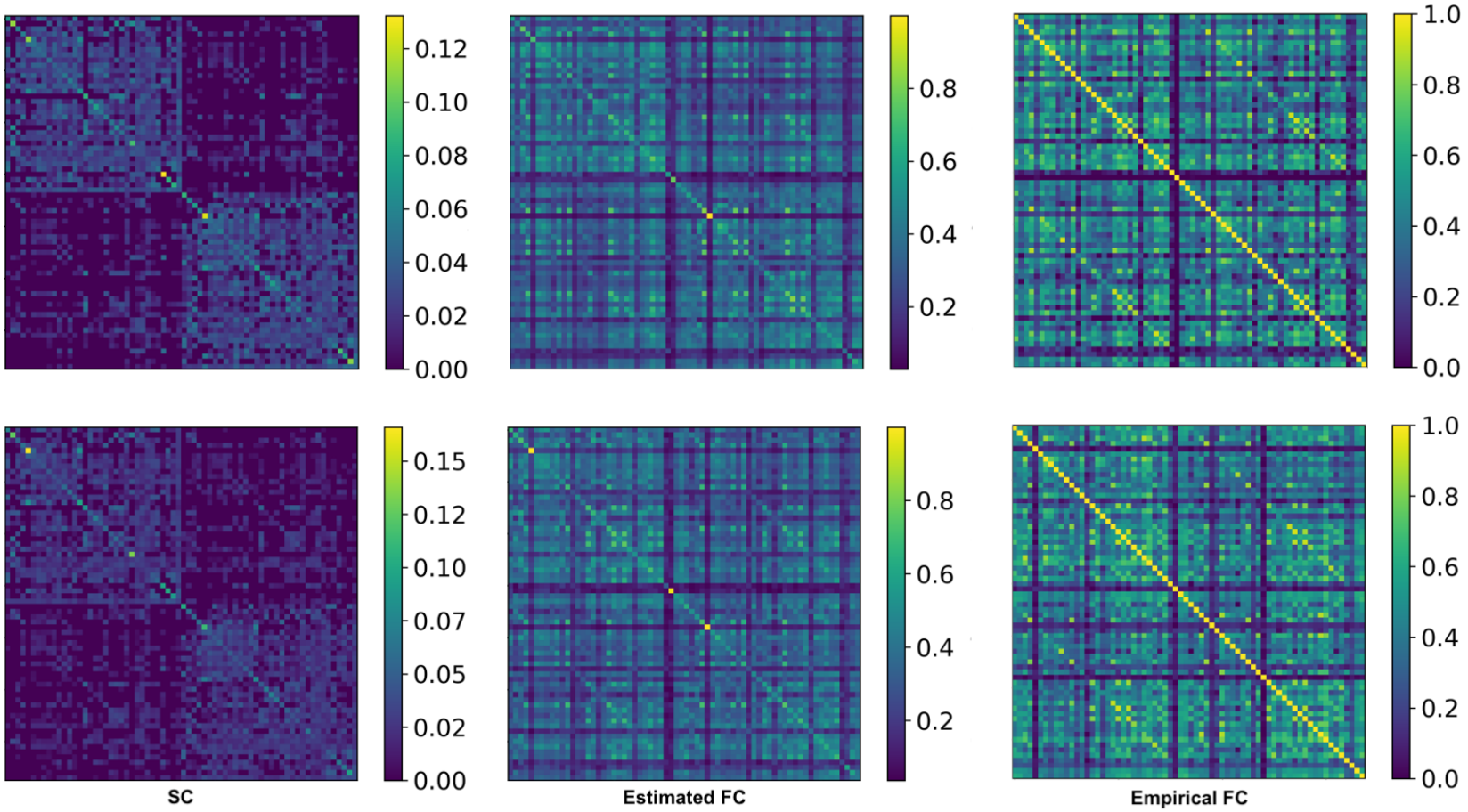
  $\Rightarrow$ Blood oxygen-level dependent (BOLD) signals

  $\Rightarrow$ Estimated FC $\Leftrightarrow$ Pearson correlation between BOLD signals

- One-hot encoding as the signal on each graph node $(\mathbf{X} = \mathbf{I}_N)$

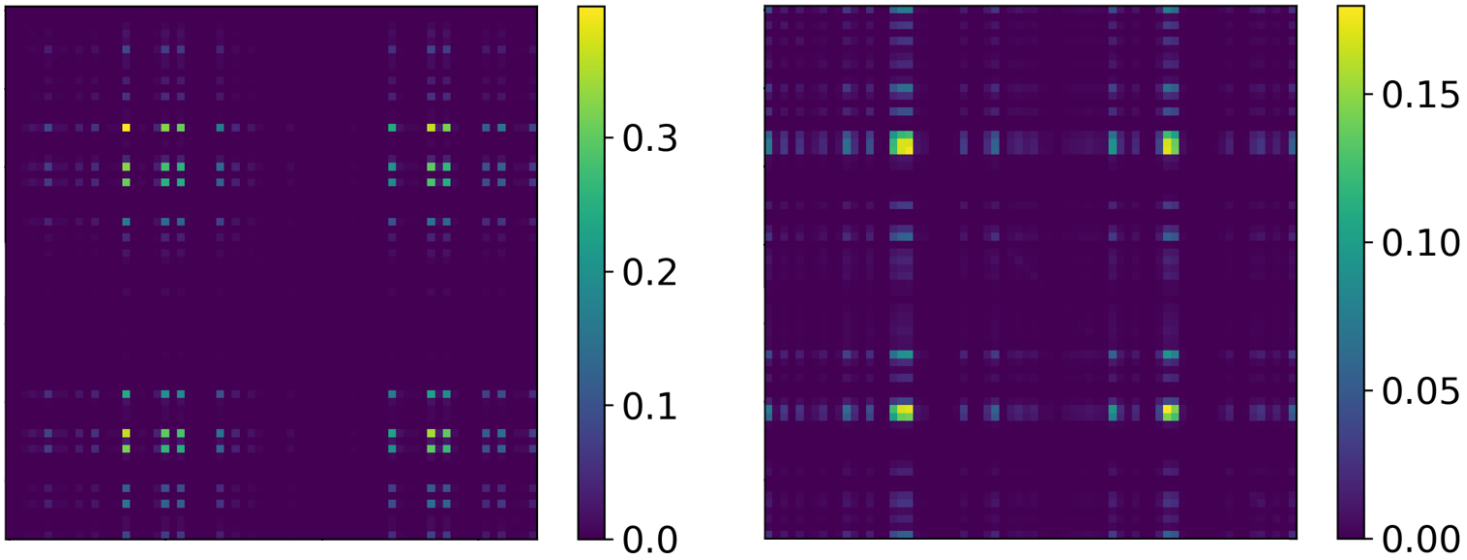
Numerical tests: FC reconstruction performance

- MSE between reconstructed and empirical FC networks
  - Average test reconstruction error = 0.0304 with std = 0.0011
  - Capture population patterns of SC-FC relationship



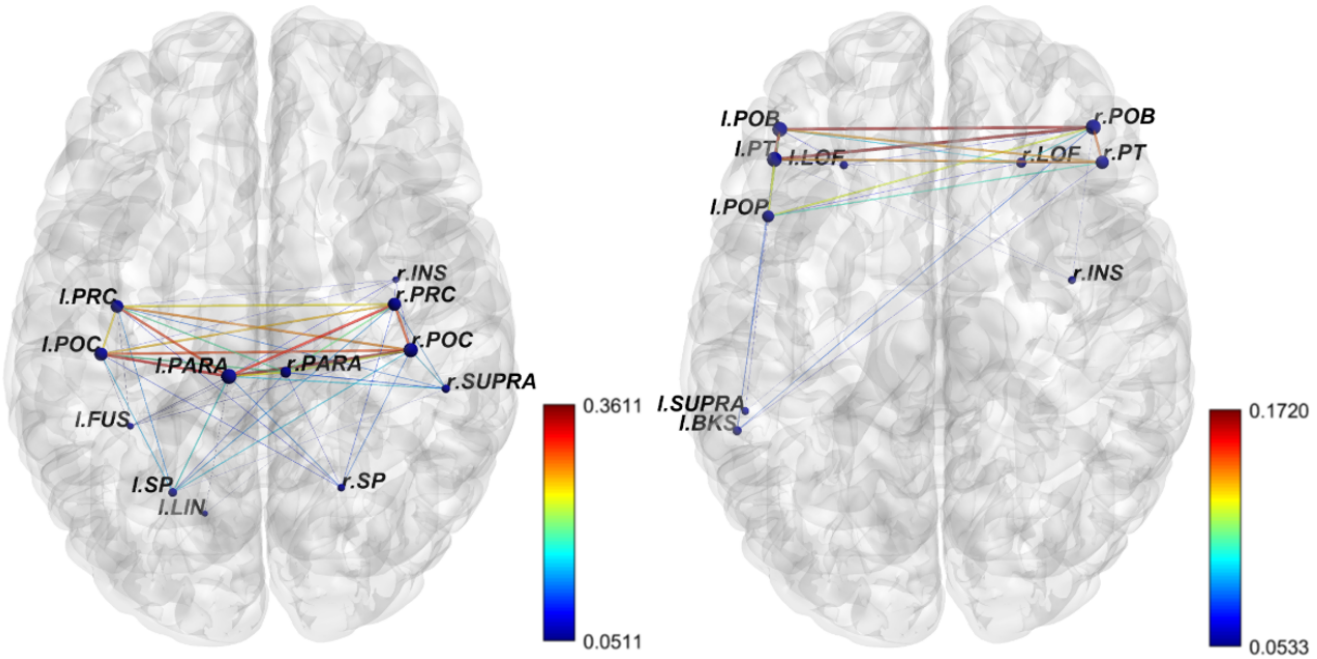
Numerical tests: component graphs

- Investigate the latent variables learnt during model training
  - Output of each graph filter in the graph convolution layer
  - Building blocks $\mathbf{Z}_i$ that generate reconstructed FC graph



- Subgraphs may reveal key insights about SC-to-FC mapping

Numerical tests: component graphs

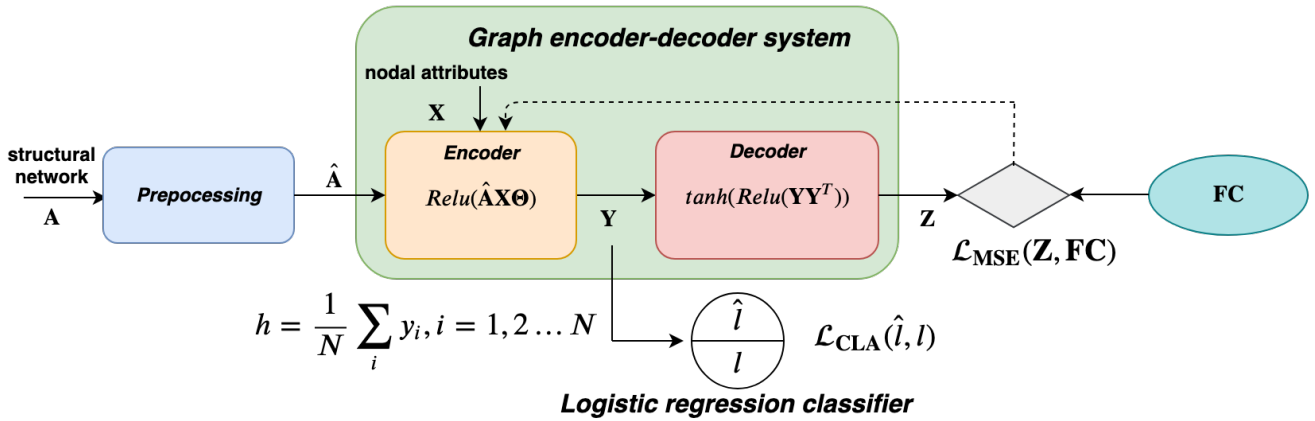


- Left: subnetwork of regions in frontal and parietal lobe
  - *Precentral (PRC), Paracentral (PARA)*, motor/sensory functions
  - *Postcentral (POC), Superior Parietal (SP)*, spatial/somatosensory
- Right: subnetwork of regions in Inferior Frontal Gyrus
  - *Parsopercularis (POP), Parsorbitalis (POB), Parstriangularis (PT)*
  - Critically involved in complex brain functions [Greenlee et al'07]

Supervised GRL for brain network classification

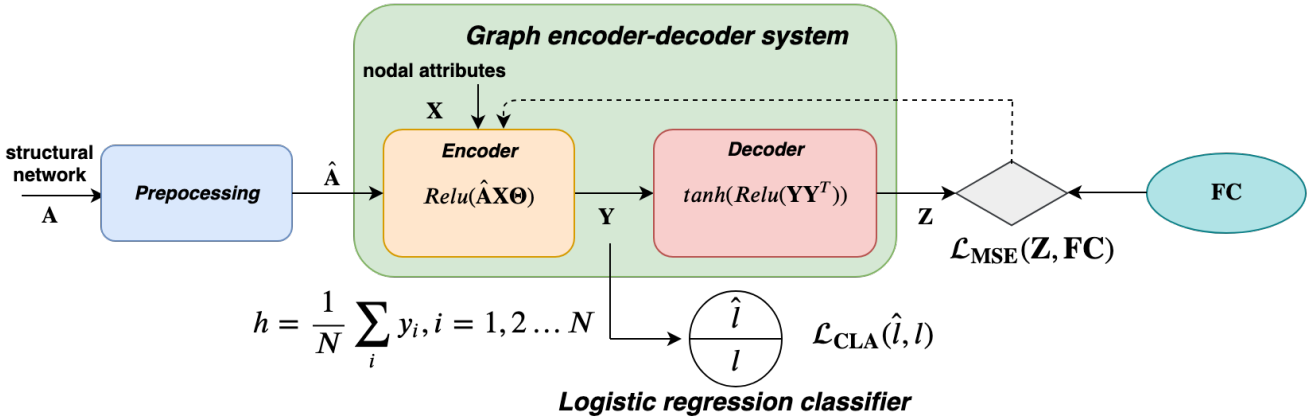
Model the relationship between brain structural and functional network

- **Goal**: summarize SC-FC relationship by simultaneously learning
  - Node embeddings to reconstruct FC from the given SC networks
  - Graph embeddings for graph classification
- Model: supervised graph encoder-decoder system



- Analysis: investigate group-wise difference within reconstructed FCs

Model architecture: classifier



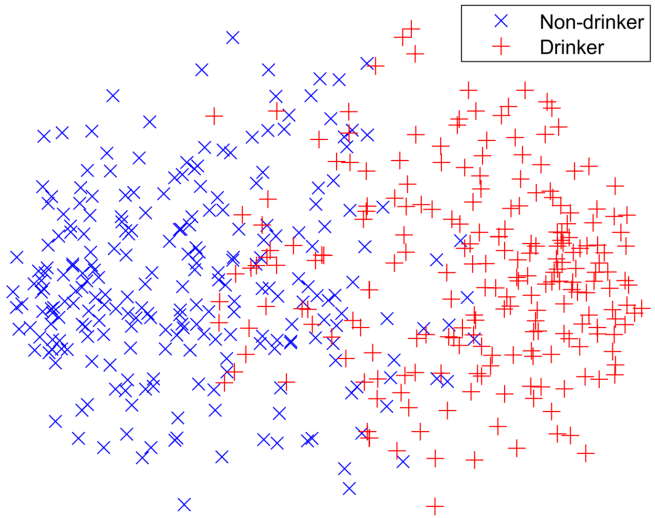
- Apply row-wise average-pooling on the encoder output **Y**

  $\Rightarrow$ Vector summarizing SC-FC relationship, i.e., graph embedding

- Construct logistic regression classifier to predict subject labels
  - Sigmoid cross-entropy loss between predicted and empirical labels
- Loss function: $\mathcal{L} = \mathcal{L}_{\mathbf{MSE}}(\mathbf{Z}, \mathbf{FC}) + \lambda \times \mathcal{L}_{\mathbf{CLA}}(\hat{l}, l)$

Numerical tests: results

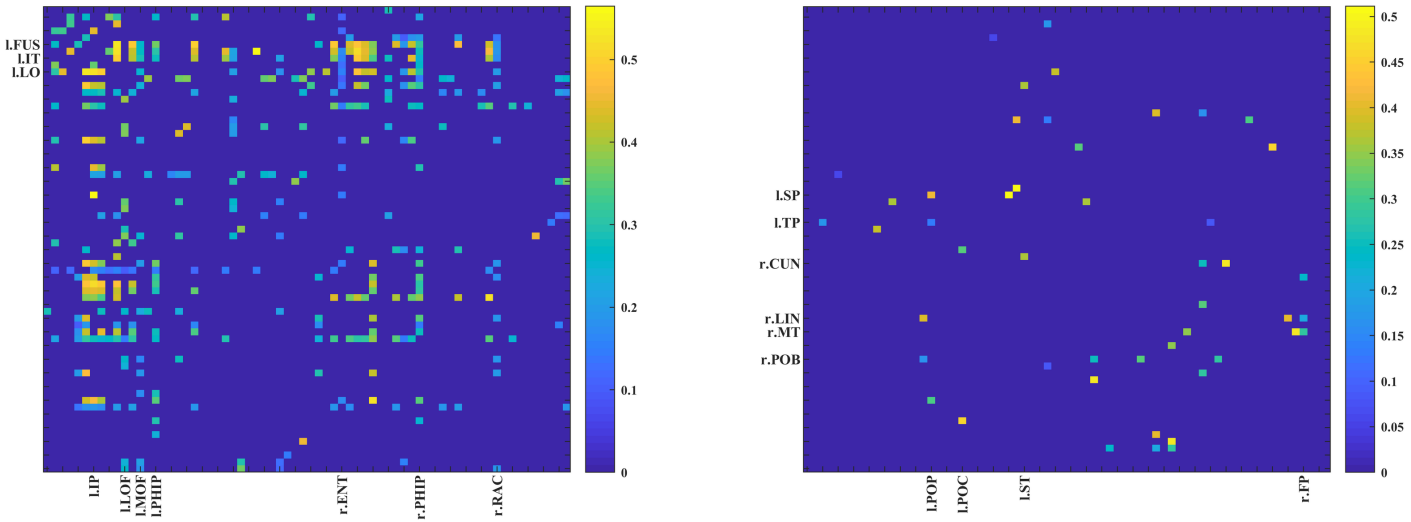
- $P = 466$ subjects from Human Connectome Project (HCP)
  - Two classes: 245 non-drinkers, 221 heavy drinkers
- MSE between reconstructed and empirical FC networks
  - Average test reconstruction error = 0.034174 with std = 0.00208
  - Captured population patterns of SC-FC relationship
- Classification accuracy: $67.4 \pm 2\%$
  - Captured discriminative patterns within each group



  - Reduced dimensional graph embeddings exhibit cluster structure

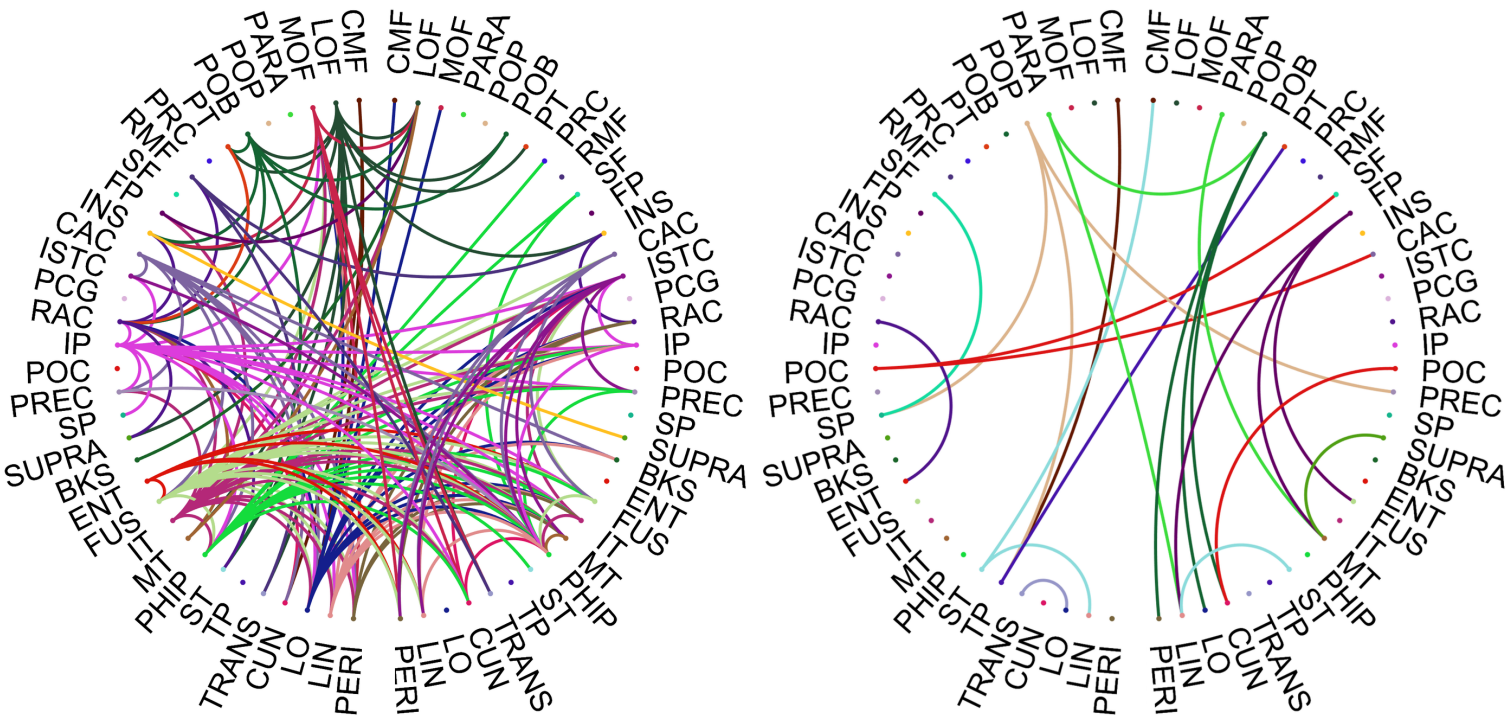
Numerical tests: reconstructed FC

- Investigate group-wise difference within reconstructed FCs
  - Captured difference between subjects in latent representations
- Test for significant group-wise difference in functional connections
  - Edge-wise T-tests ($p < 0.05$) with FDR correction



- Connections weaker (left) & stronger (right) in drinkers

Numerical tests: class differences



- Left: subnetwork of connections weaker in drinkers
  - *Entorhinal, Parahippocampus*, limbic system impaired in drinkers
  - Overall decrease in connection strengths in drinkers
- Right: subnetwork of connections stronger in drinkers
  - Involve regions in multiple cortices $\Rightarrow$ neural compensation
  - Additional connections compensate for alcohol damages

Dynamic social network clustering

- **Goal**: Reveal temporal stages of the evolution of dynamic graphs
  - $\Rightarrow$ Cast as a problem of clustering graph sequences
- Approach: Unsupervised distance-based graph-level RL
- Model: Siamese GRL network + K-means clustering
  - Learn graph-level embedding in unsupervised manner
    - $\Rightarrow$ Preserve network structure and distances between graphs
  - Cluster learned graph embeddings via K-means algorithm
    - $\Rightarrow$ Each cluster represents one temporal stage
- Siamese encoder better for input graphs with divergent structures

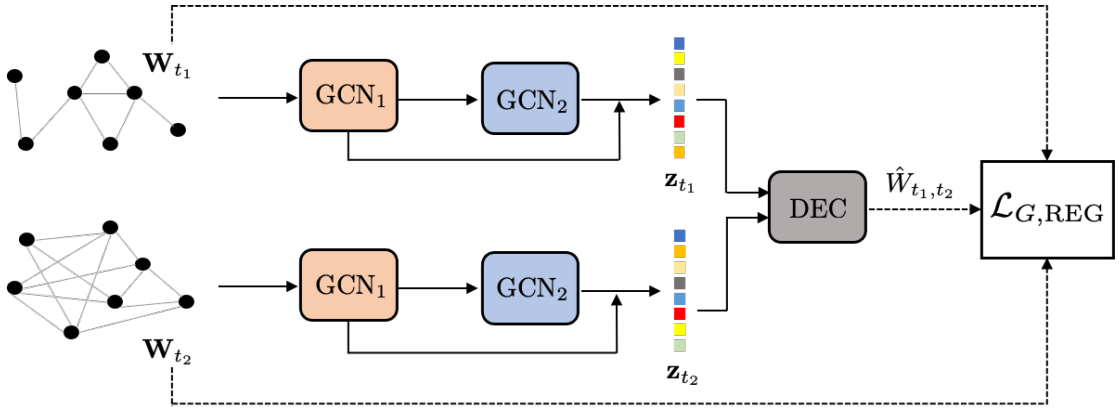
Networks of international football

- $T = 145$ football graphs from year 1872 to 2016
  - Nodes: $N_v = 238$ national teams playing official games
  - Edges: $W_{ij}(t)$ is the number of $\{i, j\}$ games during year t
  - Data comprises 39,052 total games over 145 years

- Expect to unveil various developmental stages in football history

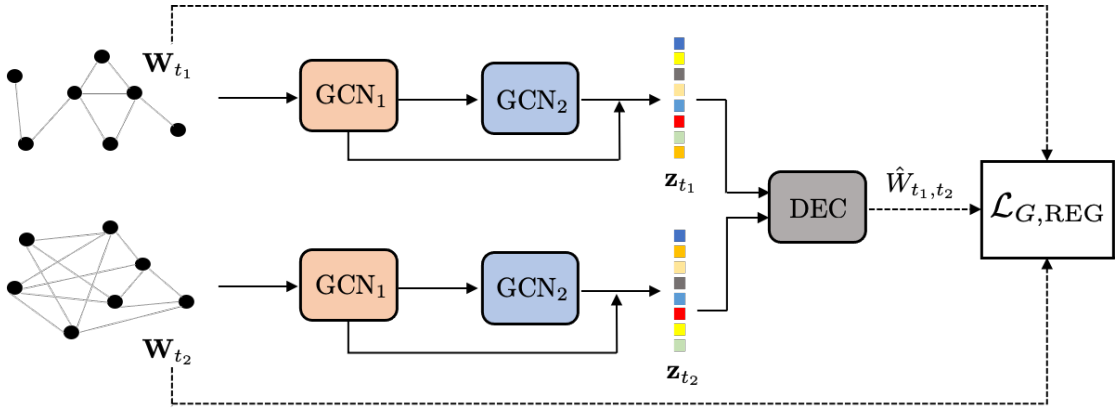
Model architecture: encoder

- Input: $\binom{145}{2} = 10440$ pairs $\{\mathbf{W}_{t_1}, \mathbf{W}_{t_2}\}$ of football graphs



- Model: Siamese encoder network with two GRL pipelines
  - Two-layer GCN with parameters shared across pipelines
- Output: graph embeddings $\{\mathbf{z}_{t_1}, \mathbf{z}_{t_2}\}$ for each input graph
  - Concatenate node embeddings learned at each layer
  - Average graph pooling from node embeddings

Model architecture: decoder



- **Idea:** embeddings preserve distances between graph pairs

$$\mathcal{L}_{G,\text{REG}}(\{\mathbf{W}_t\}, \hat{\mathbf{W}}; \boldsymbol{\Theta}) = \sum_{t_1,t_2} ([\,\text{s}(\{\mathbf{W}_t\})]_{t_1,t_2} - \hat{W}_{t_1,t_2})^2$$

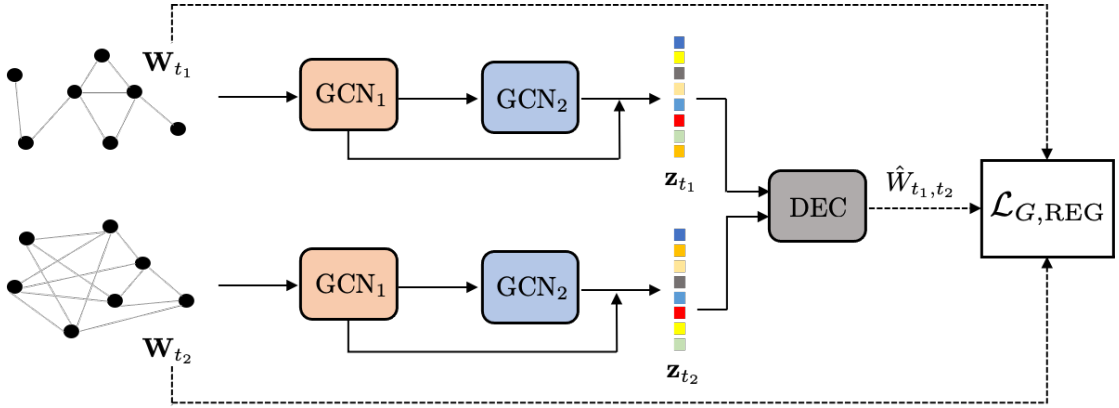
$\Rightarrow$ Euclidean distance decoder: $\hat{W}_{t_1,t_2} = \|\mathbf{z}_{t_1} - \mathbf{z}_{t_2}\|_2$

- Prescribed distances between input graphs encoded in

$$[\,\text{s}(\{\mathbf{W}_t\})]_{t_1,t_2} = d_G(\mathbf{W}_{t_1}, \mathbf{W}_{t_1})$$

$\Rightarrow$ Like MDS and IsoMAP but at graph-(not node-)level

Model architecture: graph distance



- User defined graph distance $d_G(\mathbf{W}_{t_1}, \mathbf{W}_{t_1})$
- Ex: Spectrum distance
  - Distance between spectrum (eigenvalues) of both Laplacian matrices
- Ex: Vertex-edge-overlap (VEO)
  - Measure structural similarity between graphs

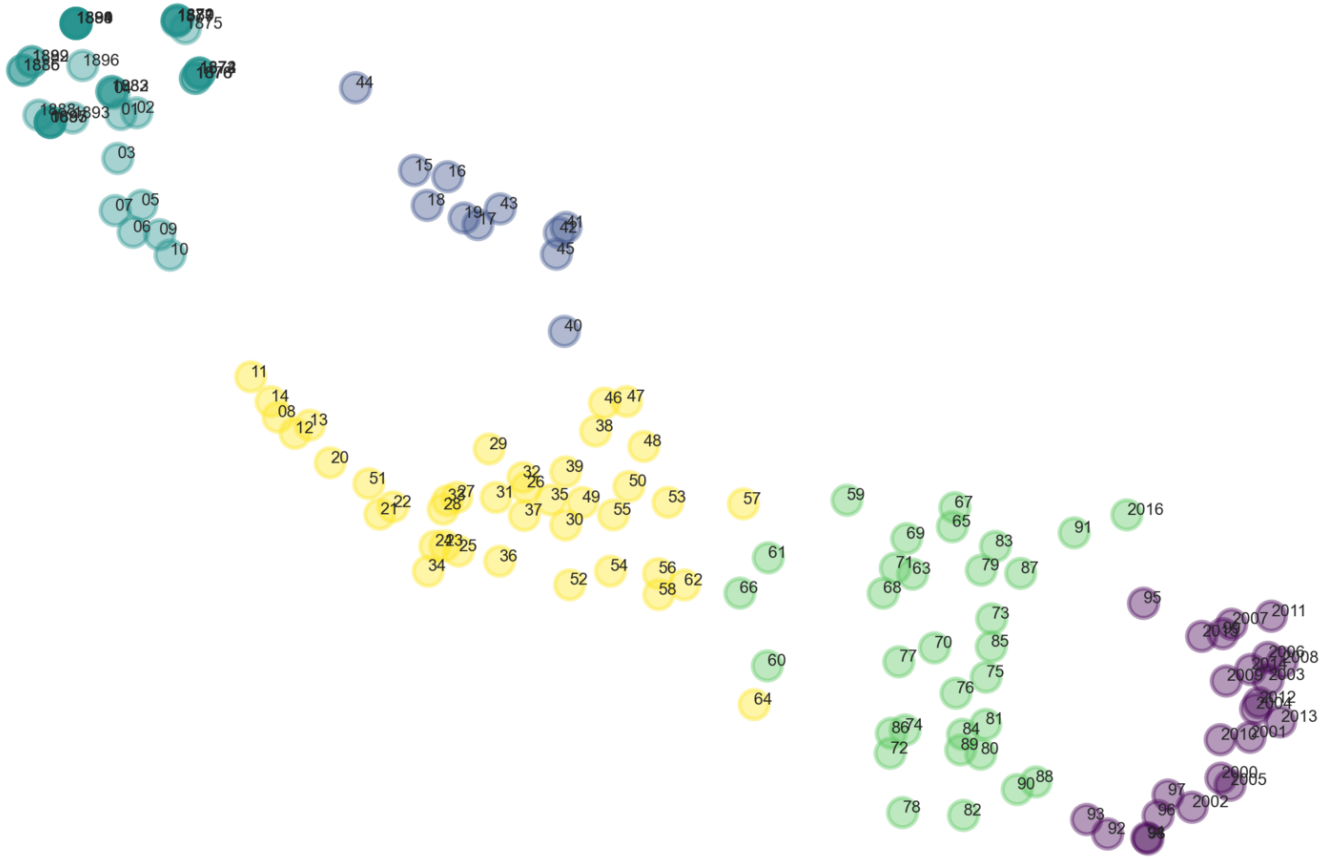
$$\text{VEO}(\mathbf{W}_{t_1}, \mathbf{W}_{t_2}) = 2 \times \frac{|\mathcal{E}_{t_1} \cap \mathcal{E}_{t_2}| + |\mathcal{V}_{t_1} \cap \mathcal{V}_{t_2}|}{|\mathcal{E}_{t_1}| + |\mathcal{E}_{t_2}| + |\mathcal{V}_{t_1}| + |\mathcal{V}_{t_2}|}$$

  - Distance computed as one minus normalized VEO

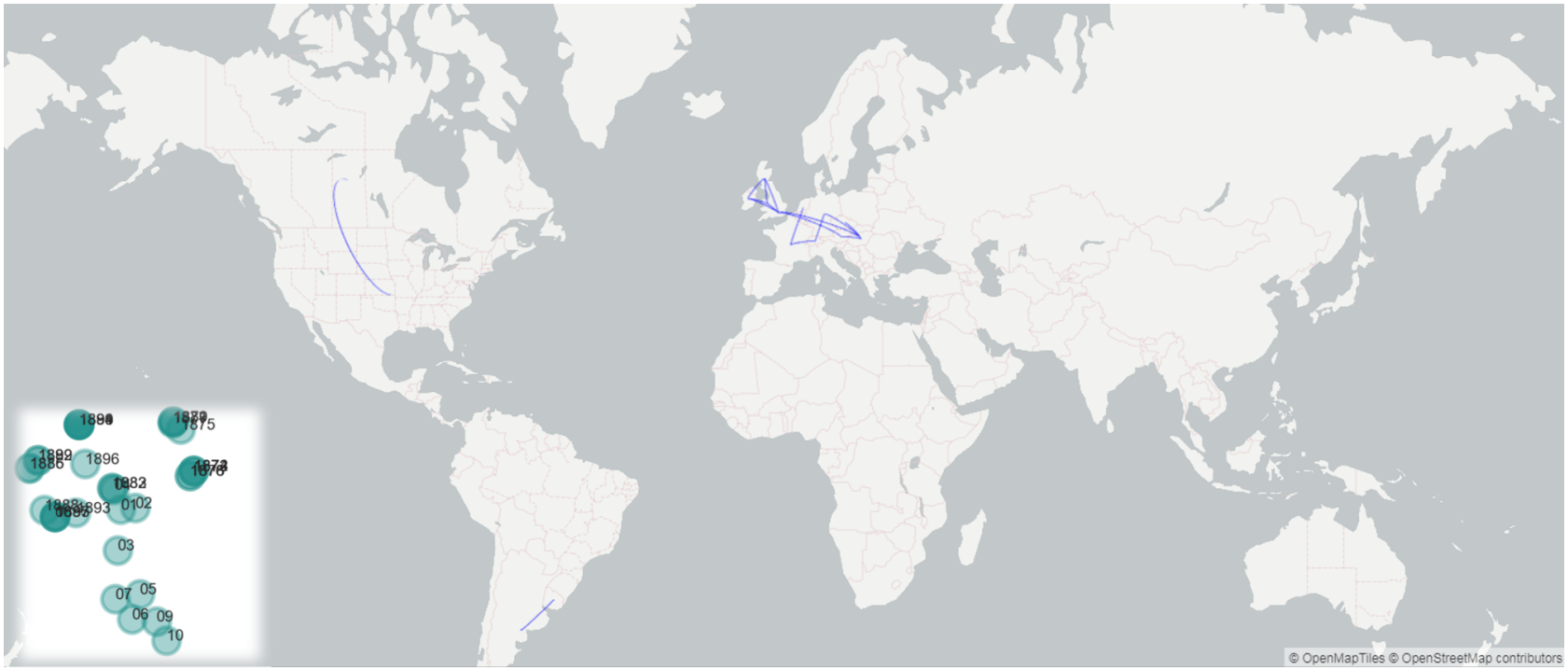
Graph clustering results

- K-means clustering applied to learned graph embeddings in $\mathbb{R}^{48}$

  $\Rightarrow$ Number of clusters $K = 5$ chosen by elbow rule

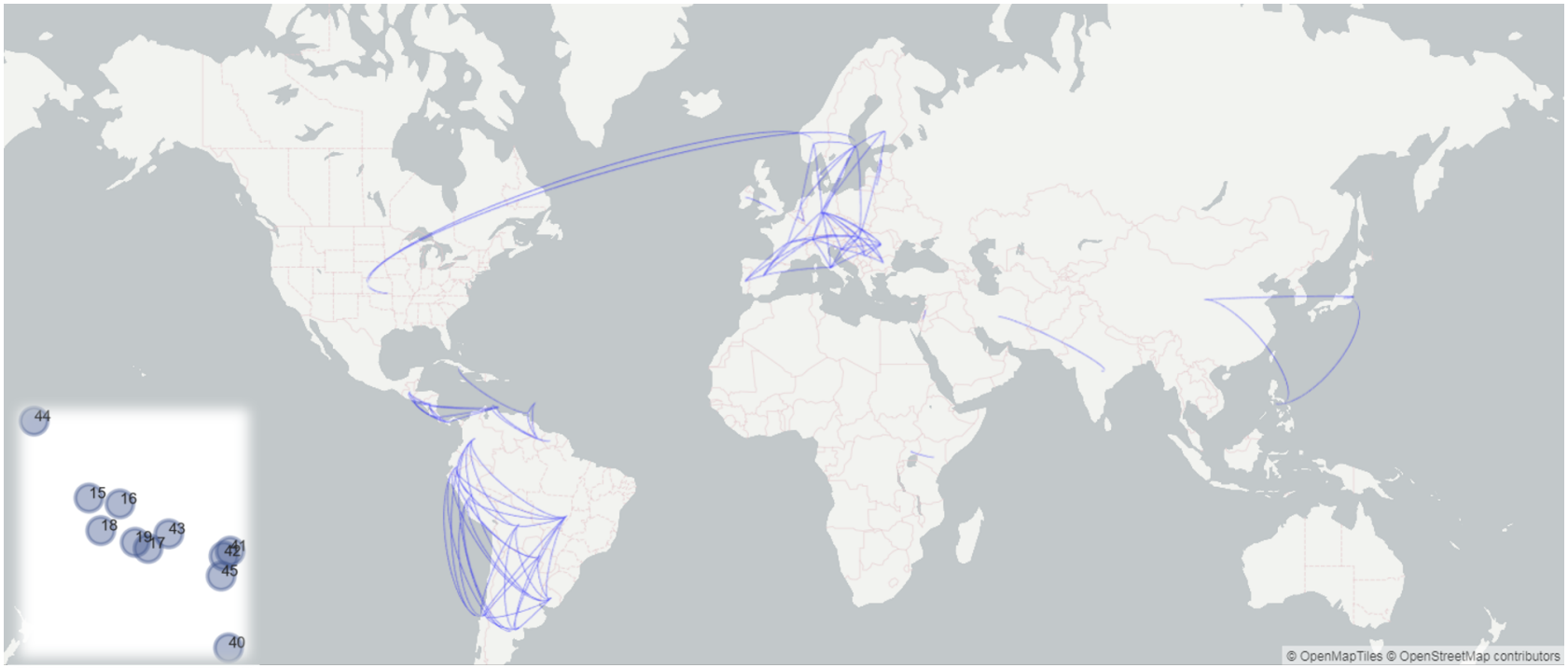


Cluster 1: early 20th century



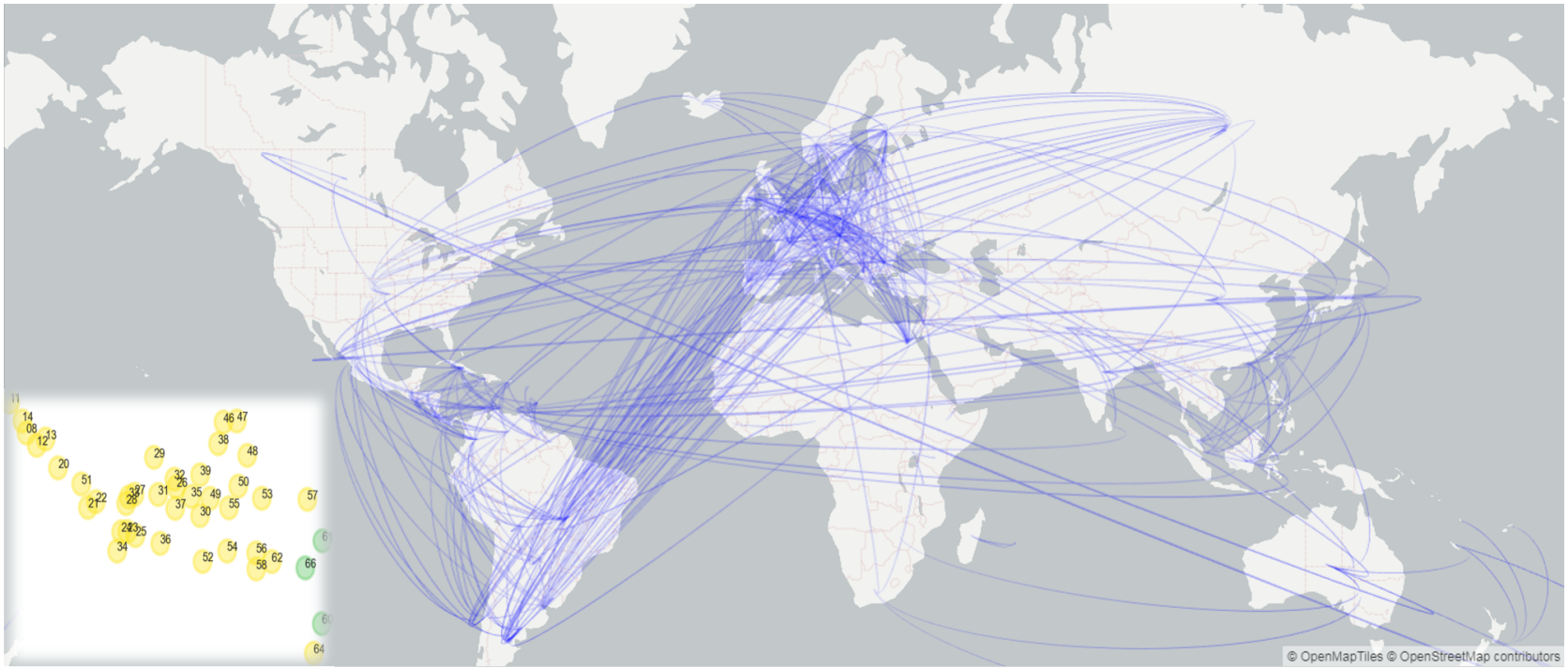
- Mostly regions around UK and the River Plate

Cluster 2: World Wars



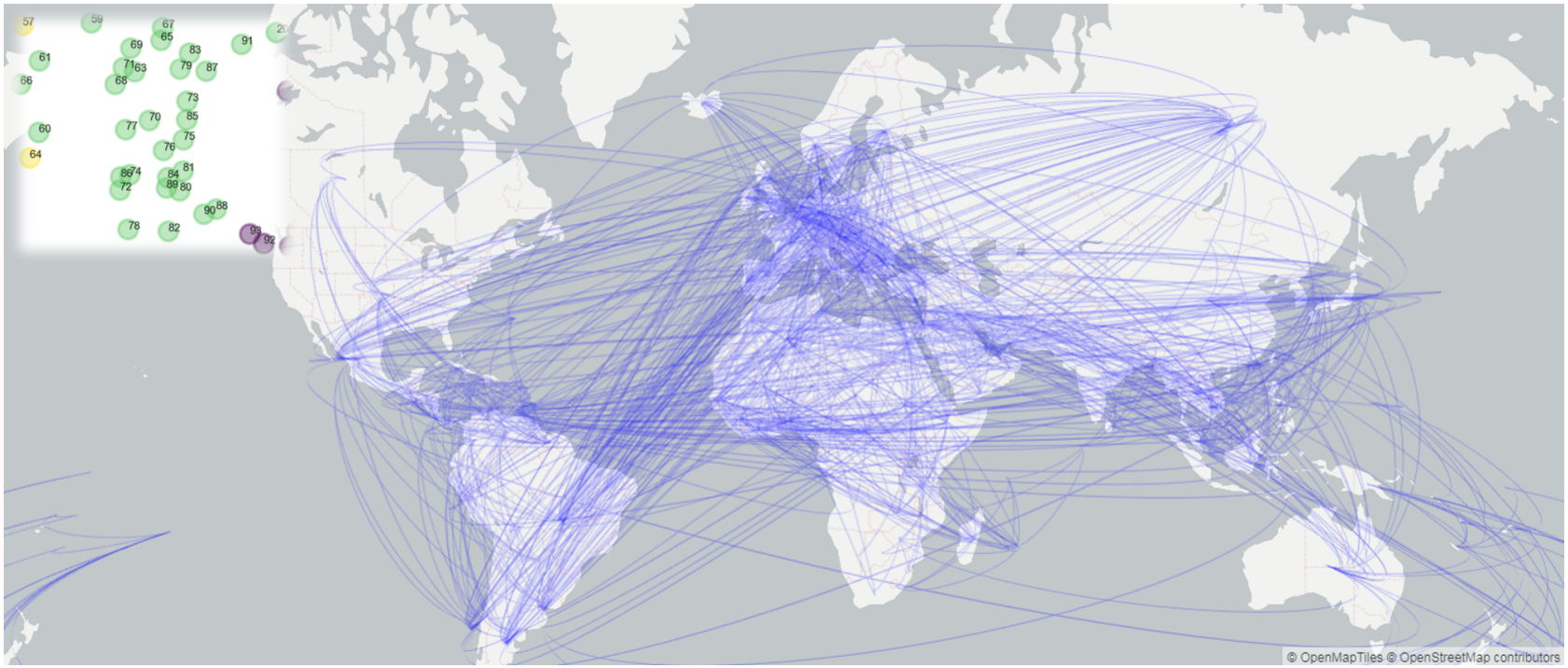
- Decreased activity in Europe and sustained growth in South America

Cluster 3: Post-World War II recovery



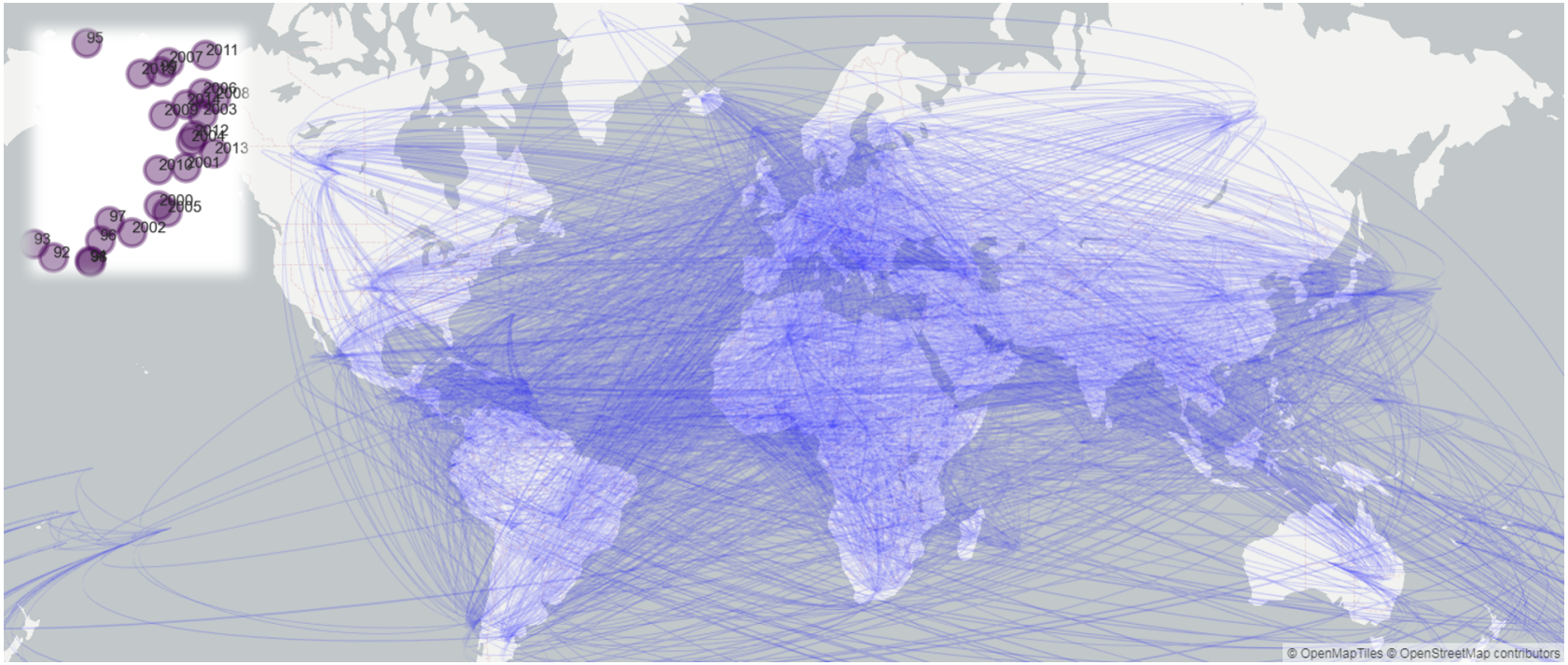
- Noticeable bridging between Europe and America

Cluster 4: Modern development



- Modern expansion of football with Africa and Asia involved

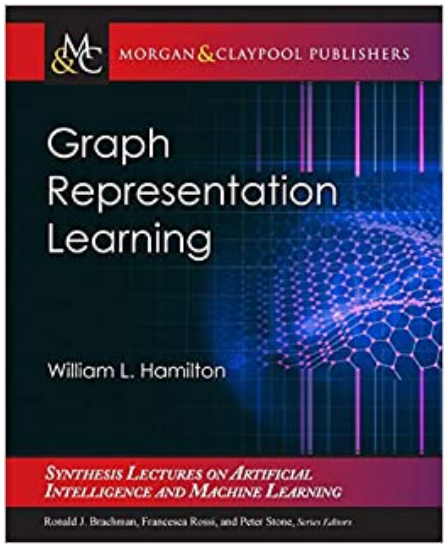
Cluster 5: Current landscape



- Global nature of the game is patently apparent

Open research directions

- GRL is a very active area of research. Many open questions remain:
  - Scalability, interpretability, fairness, theoretical guarantees
  - Robust and unified evaluation protocols and benchmarks
  - Modeling of directed, dynamic and multi-layer graphs
  - Beyond pairwise decoders: decoding higher-order motifs
  - Expressivity via non-Euclidean embeddings?



Glossary

- Convolutional neural network
- Graph representation learning
- Node and graph embedding
- (Un)supervised learning
- Transductive and inductive
- Link prediction
- Encoder-decoder model
- Learnable parameters
- (Dis)similarity scores
- Graph pooling
- Graph regularization loss
- End-to-end learning
- Shallow embedding
- Graph autoencoders
- Neigborhood aggregation
- Higher-order proximity
- Matrix factorization
- Random walks
- Laplacian eigenmaps
- Hyperbolic geometry
- Variational autoencoders
- Brain network analysis
- Graph convolutional network