



FACULTAD DE  
INGENIERÍA



UNIVERSIDAD  
DE LA REPÚBLICA  
URUGUAY

# Aprendizaje Automático para Datos en Grafos

## Graph representation learning

Marcelo Fiori

Muy basado en transparencias de **Gonzalo Mateos**

`mfiori@fing.edu.uy`

`http://www.fing.edu.uy/~mfiori/`

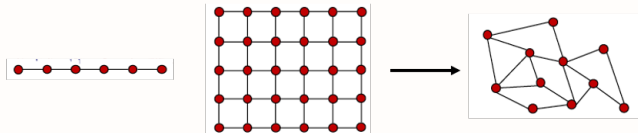
10 de noviembre, 2022



FACULTAD DE  
INGENIERÍA  
UDELAR

# Aprendizaje automático en grafos: Motivación

- Modelos muy exitosos para representación (y) aprendizaje en **datos estructurados**
  - Secuencias (e.g., texto, audio, videos) via **recurrent neural networks (RNNs)**
  - Clasificación de imágenes via **convolutional neural networks (CNNs)**



- Pero los datos no siempre son regulares  $\Rightarrow$  **Estructuras relacionales complejas**
  - **Grafos** en redes sociales, química computacional, biología, ...
- **Desafíos:** aplicar modelos diseñados para datos regulares, en grafos
  - La estructura de los grafos puede ser arbitraria y variar en diferentes escenarios
  - Las convoluciones no generalizan a dominios irregulares

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

# ¿De qué vamos a hablar?

## Graph representation learning (GRL)

- Aprender vectores en baja dimensión (embeddings) para datos en grafos
- Tipos de aprendizaje:
  - **Supervisado**: aprender representaciones para clasificación de nodos o grafos
  - **No-supervisado**: aprender representaciones que preserven estructura de grafos
- Dominio subyacente:
  - **Transductivo**: estructura del grafo fija (e.g., una red social muy grande)
  - **Inductivo**: los grafos de entrada pueden variar (e.g., múltiples moléculas)
- Información en los nodos:
  - **Featureless**: no tenemos información adicional (i.e., graph signals)
  - **With features**: los nodos tienen atributos o características

# Roadmap

- 1 The network embedding problem
- 2 Una taxonomía de modelos de graph embedding
- 3 Unsupervised graph embedding
- 4 Applications



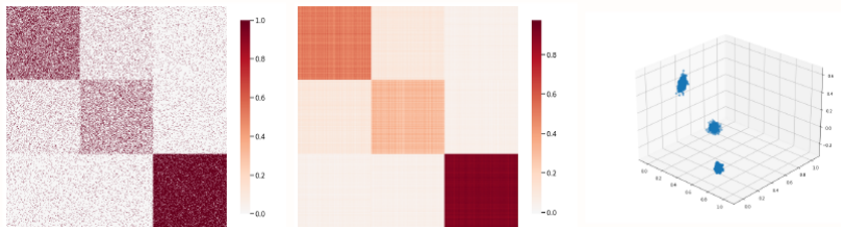
# Embeddings de grafos

- Aprender un mapa de un grafo discreto a un dominio continuo
- Dado  $G(\mathcal{V}, \mathcal{E})$  con matriz de adyacencia (con pesos)  $\mathbf{W} \in \mathbb{R}^{N_v \times N_v}$
- **Objetivo:** aprender una representación en vectores  $d$ -dimensionales  $\{\mathbf{z}_i\}_{i \in \mathcal{V}}$ 
  - ⇒ Criterio es preservar propiedades locales y globales del grafo
- Salida es una matriz de embeddings de nodos  $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_{N_v}]^T \in \mathbb{R}^{N_v \times d}$ 
  - ⇒ Elegir  $d \ll N_v$  para escalabilidad
  - ⇒ Hay reducción de dimensionalidad
- Es posible extender esto a un **embedding del grafo entero** via  $\mathbf{z} \in \mathbb{R}^d$

# Adjacency spectral embedding (revisitado)

- Ex: SBM con  $N_v = 1500$ ,  $Q = 3$  y parámetros:

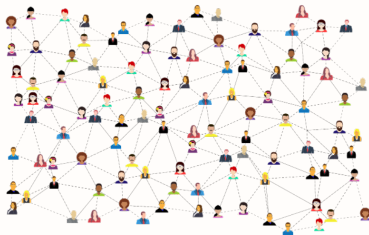
$$\boldsymbol{\alpha} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \quad \boldsymbol{\Pi} = \begin{bmatrix} 0,5 & 0,1 & 0,05 \\ 0,1 & 0,3 & 0,05 \\ 0,05 & 0,05 & 0,9 \end{bmatrix}$$



- Adyacencia muestreada (izq.),  $\mathbf{Z}\mathbf{Z}^\top$  (centro), filas de  $\mathbf{Z}$  (der.)
- Embeddings para poder usar métodos geométricos de análisis

# El rol de las señales en grafos

- Señales en grafos (a.k.a. atributos en nodos o características)  $\mathbf{X} \in \mathbb{R}^{N_v \times F}$



- **Ex:** Edad, género (redes sociales), señales fMRI, ratings de productos
- Los embeddings captura información **estructural** y **semántica** del grafo

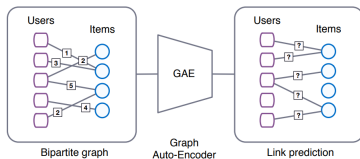
$$\{\mathbf{W}, \mathbf{X}\} \mapsto \mathbf{Z}$$

- Sin  $\mathbf{X}$ , el embedding  $\{\mathbf{W}\} \mapsto \mathbf{Z}$  se dice **featureless**  
 $\Rightarrow$  El mapeo **solo** preserva información **estructural**

# Embeddings transductivos e inductivos

## Transductive network embedding

- Embedding de nodos de un grafo fijo (en general grande)
  - Ex: Recomendación de productos o amigos via predicción de enlaces
  - Ex: Clasificación de nodos en aprendizaje semi-supervisado



- Dados nuevos nodos, hay que actualizar o re-entrenar el modelo

## Inductive network embedding

- Aprender mapas a representaciones que generalizan a grafos no vistos
  - Ex: Embedding de grafos de cerebros para clasificación de sujetos
  - Ex: Embedding de grafos dinámicos para clustering temporal
- Típicamente se necesitan señales **X** para hacer embedding inductivo

# Embeddings supervisados y no-supervisados

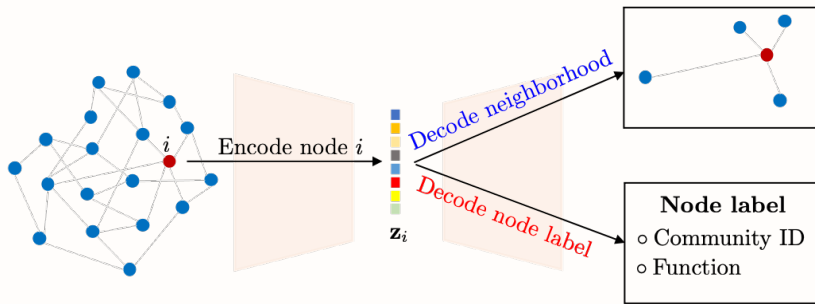
## Unsupervised network embedding

- Solamente tenemos la topología del grafo  $\mathbf{W}$  (y eventualmente  $\mathbf{X}$ )
  - **Preservar estructuras del grafo** optimizando una función de pérdida/reconstrucción
  - Decodificar los embedding  $\mathbf{Z}$  para aproximar bien  $\mathbf{W}$
- **Ex:** compresión, visualización, clustering, predicción de enlaces

## Supervised network embedding

- Además de  $\mathbf{W}$  (y  $\mathbf{X}$ ), tenemos disponibles etiquetas de nodos o grafos  $\mathbf{y}^S$ 
  - Optimizar embeddings para **tareas aguas abajo**
  - Combina reconstrucción y función de pérdida de la tarea específica
- **Ex:** clasificación de nodos, clasificación de grafos

# Una perspectiva encoder-decoder



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

# Roadmap

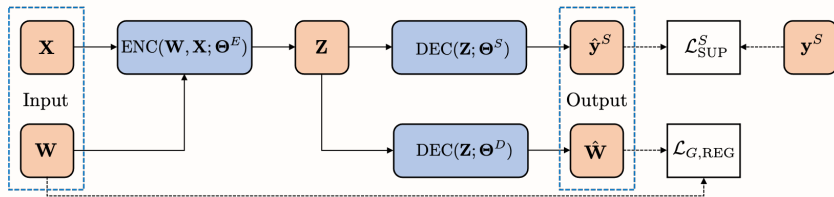
- 1 The network embedding problem
- 2 Una taxonomía de modelos de graph embedding
- 3 Unsupervised graph embedding
- 4 Applications

# Un modelo abarcativo para graph embedding

## ■ Graph Encoder Decoder Model (GraphEDM)

⇒ Marco unificador para revisar y comparar métodos de GRL

⇒ **Biblioteca open-source** con métodos y aplicaciones



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

## ■ Q: ¿Cuáles son los componentes constitutivos del modelo?

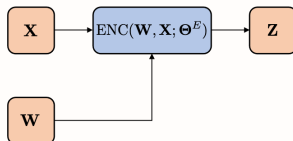
<https://github.com/google/gcnm-survey-paper>



# Input

- Grafo no dirigido  $G(\mathcal{V}, \mathcal{E})$ , con  $|\mathcal{V}| = N_v$  y  $|\mathcal{E}| = N_e$   
⇒ Matriz de adyacencia (con pesos)  $\mathbf{W} \in \mathbb{R}^{N_v \times N_v}$
- Opcionalmente señales en el grafo (características de nodos)  $\mathbf{X} \in \mathbb{R}^{N_v \times F}$
- Para aprendizaje (semi)-supervisado, también necesitamos etiquetas de:
  - Nodos ( $N$ ), para clasificación de nodos y clustering
  - Aristas ( $E$ ), para predicción de enlaces o clasificación de relaciones
  - Grafos ( $G$ ), para clustering y clasificación de grafos
- Las etiquetas de supervisión las denotamos  $\mathbf{y}^S$ , donde  $S \in \{N, E, G\}$

# Encoder



## ■ Graph encoder network

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

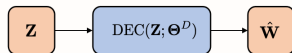
⇒ Parámetros entrenables  $\Theta^E$

## ■ Combina la estructura del grafo con señales para producir un embedding

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

⇒ Captura diferentes propiedades del grafo en base al tipo de supervisión

# Decoder (I)



## ■ Graph decoder network

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

⇒ Parámetros entrenables  $\Theta^D$

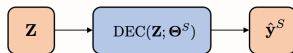
## ■ Usa $\mathbf{Z}$ para producir scores de (di)similitud $\hat{W}_{ij}$ para cada $\{i, j\} \in \mathcal{V}^{(2)}$

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

⇒ Reconstrucción no supervisada del grafo

⇒ Aproxima  $\mathbf{W}$ , o más en general una matriz de (di)similitud  $s(\mathbf{W})$

## Decoder (II)



- Classification network

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

⇒ Parámetros entrenables  $\Theta^S$ , espacio de etiquetas  $\mathcal{Y}$

- Usa  $\mathbf{Z}$  para producir distribuciones de etiquetas para cada nodo

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

⇒ Aprendizaje (semi)-supervisado para clasificación de nodos/grafos

# Output

- Matriz de similitud reconstruida  $\hat{\mathbf{W}} \in \mathbb{R}^{N_v \times N_v}$ 
  - ⇒ Usado para entrenar algoritmos de embedding no supervisados
- Para aprendizaje (semi)-supervisado, las salidas son las etiquetas que se predicen  $\hat{\mathbf{y}}^S$ 
  - El espacio de salida de etiquetas varía dependiendo del tipo de supervisión
- Node-level:  $\hat{\mathbf{y}}^N \in \mathcal{Y}^{N_v}$  or  $\hat{\mathbf{Y}}^N \in [0, 1]^{N_v \times |\mathcal{Y}|}$ 
  - ⇒ Cuando  $|\mathcal{Y}| = d$ , se puede usar activación softmax en las filas de  $\mathbf{Z}$
- Edge-level:  $\hat{\mathbf{Y}}^E \in \mathcal{Y}^{N_v \times N_v}$ , where typically  $\mathcal{Y} = \{0, 1\}^{\#\text{relation types}}$ 
  - ⇒ Cuando  $\#\text{relation types} = 1$  (i.e., predicción de enlaces), salida  $\hat{\mathbf{W}}$
- Graph-level:  $\hat{y}^G \in \mathcal{Y}$ 
  - ⇒ Usando  $\mathbf{W}$ , convertir  $\mathbf{Z}$  a  $\hat{y}^G$  via graph pooling

# Funciones de pérdida

## ■ Supervised loss

$\Rightarrow \mathcal{L}_{\text{SUP}}^S$  compara las etiquetas que se predicen  $\hat{\mathbf{y}}^S$  al ground truth  $\mathbf{y}^S$

Ex: clasificación semi-supervisada de nodos ( $S = N$ ,  $\mathcal{V} = \mathcal{V}_{\text{obs}} \cup \mathcal{V}_{\text{miss}}$ )

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

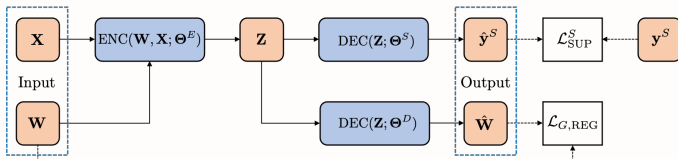
## ■ Graph regularization loss

$\Rightarrow \mathcal{L}_{G, \text{REG}}$  compara  $\hat{\mathbf{W}}$  con matriz de (di)similitud objetivo  $s(\mathbf{W})$

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

- $d_1(\cdot, \cdot)$ : función distancia o de disimilitud
- Aprovechar  $G$  via  $s(\mathbf{W})$  para regularizar los parámetros del modelo  $\Theta$

## Función objetivo



### ■ Weight regularization loss

$\Rightarrow \mathcal{L}_{\text{REG}}$  regulariza los parámetros entrenables  $\Theta$  para reducir overfitting

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

### ■ Función objetivo GraphEDM total

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

$\Rightarrow$  Entrenamiento supervisado ( $\alpha \neq 0$ ) o no supervisado ( $\alpha = 0$ )

■ Q: ¿Aprendizaje supervisado end-to-end o en dos etapas?

# Taxonomía de graph embedding

- Categorizamos los métodos de GRL en base al encoder y la función de pérdida
- **Shallow embedding methods**  $\mathbf{Z} = \text{ENC}(\Theta^E) = \Theta^E$ 
  - Un simple embedding lookup
- **Graph auto-encoding methods**  $\mathbf{Z} = \text{ENC}(\mathbf{W}; \Theta^E)$ 
  - **Transductivos** como los shallow embeddings, no hay señal  $\mathbf{X}$ , así que funciona para grafo fijo  $G$
- **Graph regularization methods**  $\mathbf{Z} = \text{ENC}(\mathbf{X}; \Theta^E)$ 
  - Usamos  $\mathbf{W}$  via  $\mathcal{L}_{G, \text{REG}}$  para regularizar los embeddings
- **Neighborhood aggregation methods**  $\mathbf{Z} = \text{ENC}(\mathbf{W}, \mathbf{X}; \Theta^E)$ 
  - Usamos  $\mathbf{W}$  para propagar información entre nodos y aprender  $\mathbf{Z}$



# Roadmap

- 1 The network embedding problem
- 2 Una taxonomía de modelos de graph embedding
- 3 Unsupervised graph embedding
- 4 Applications

# Unsupervised graph embedding

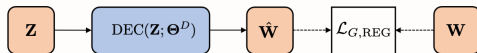
- **Objetivo:** Aprender embeddings de nodos que preserven estructura del grafo
- Optimizamos para reconstruir cierta matriz de (di)similitud entre nodos  $s(\mathbf{W})$

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

$\alpha = 0$

- La salida del decoder es  $\hat{\mathbf{W}}$ , con  $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j)$
  - Regularización  $\mathcal{L}_{G, \text{REG}} = d_1(s(\mathbf{W}), \hat{\mathbf{W}})$
  - Optimizamos sobre conjunto de entrenamiento  $\{i, j\} \in \mathcal{V}_{\text{obs}}^{(2)}$ , usando SGD o métodos espectrales
- La matriz de similitud objetivo  $s(\mathbf{W})$  puede tomar varias formas
    - Ex: Reconstruir proximidad de primer orden via  $[s(\mathbf{W})]_{ij} = W_{ij}$
    - Ex: Proximidad de alto orden  $[s(\mathbf{W})]_{ij} = |\mathcal{N}_i \cap \mathcal{N}_j|$ , Jaccard, Adamic-Adar
    - Ex: Prob.  $[s(\mathbf{W})]_{ij} = P(v_j | v_i)$  que  $i, j \in \mathcal{V}$  co-ocurrán en paseos al azar (random walks)

# Shallow embeddings



## ■ Shallow embedding methods

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

⇒ Un simple embedding lookup, se optimiza directamente  $\mathbf{Z}$

## ■ Dos clases, en base al tipo de decoder $\hat{\mathbf{W}} = \text{DEC}(\mathbf{Z}; \Theta^D)$

- Métodos basados en distancia:  $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j)$
- Métodos basados en productos externos:  $\hat{\mathbf{W}} = \mathbf{Z}\mathbf{Z}^\top \Rightarrow \hat{W}_{ij} = \mathbf{z}_i^\top \mathbf{z}_j$

## ■ Inspirados en reducción de dimensionalidad via low-rank matrix factorization

⇒ Enfoques más recientes se basan en random walks (analogías con NLP)

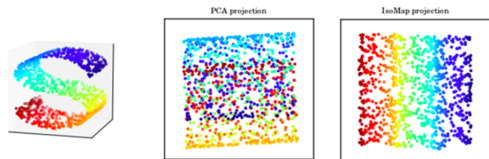
# Métodos basados en distancia

- **Idea:** que los embeddings preserven **distancias** en  $G$  [codificadas en  $s(\mathbf{W})$ ]

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

⇒ 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)** preserva conectividad **local**
  - Seteamos  $[s(\mathbf{W})]_{ij} = 1$  (o  $W_{ij}$ ) si  $W_{ij} > 0$  y 0 si no
- **IsoMAP** preserva distancias geodésicas **globales** en la variedad
  - Seteamos e.g.,  $[s(\mathbf{W})]_{ij} = d_G(i, j)$ , distancia del camino más corto entre  $i, j \in \mathcal{V}$



# Laplacian eigenmaps

- Captura información de  $G$  via propiedades espectrales de  $\mathbf{L} = \mathbf{D} - \mathbf{W}$ 
  - ⇒ Esquema no lineal de reducción de dimensionalidad que preserva propiedades locales

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

- Se puede escribir de forma equivalente como un término de regularización

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

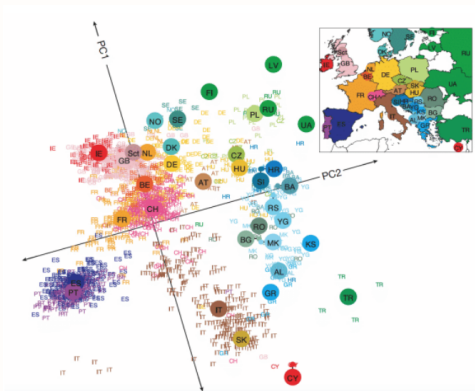
⇒ Euclidean distance decoder:  $\hat{W}_{ij} = d_2(\mathbf{z}_i, \mathbf{z}_j) = \|\mathbf{z}_i - \mathbf{z}_j\|_2$

⇒ Embeddings estarán cerca en  $\mathbb{R}^d$  si  $i, j$  están bien conectados en  $G$

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

# Gene cartography

- **Ex:** Spectral embedding de matriz de 'gene similarity' ( $d = 2$ )  
⇒ Consistente con el origen de los individuos en Europa



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

# Non-Euclidean embedding spaces

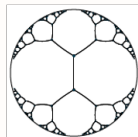
- **Idea:** embedding de grafos con estructura jerárquica en espacio hiperbólico

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

⇒ Poincaré distance decoder:

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

- Captura similitud y jerarquía
- Se usan herramientas de optimización en variedades



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

# Matrix factorization methods

- **Idea:** aprender representaciones de bajo rango de la matriz de similitud  $s(\mathbf{W})$

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

⇒ Decodificador de producto externo:  $\hat{\mathbf{W}} = \text{DEC}(\mathbf{Z}; \Theta^D) = \mathbf{Z}\mathbf{Z}^\top$

⇒ Implica una aproximación via producto interno  $[s(\mathbf{W})]_{ij} \approx \mathbf{z}_i^\top \mathbf{z}_j$

- **Graph factorization (GF)** preserva similitud de primer orden en  $G$ 
  - Seteamos  $[s(\mathbf{W})]_{ij} = W_{ij}$  y evaluamos  $\mathcal{L}_{G,\text{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \Theta)$  en  $(i, j) \in \mathcal{E}$
- **GraRep** preserva similitud de orden mayor en  $G$ 
  - Seteamos e.g.,  $[s(\mathbf{W})]_{ij} = [\mathbf{W}^k]_{ij}$ ,  $k \geq 2$ , cantidad de caminos de largo  $k$
- **HOPE** preserva medidas de similitud general en  $G$  (dirigido)
  - Jaccard, Adamic-Adar y scores relacionados usando vecinos
- Muy relacionado al adjacency spectral embedding (ASE) para RDPGs



# De textos a grafos

- Utilizar herramientas de modelado y aprendizaje de características en NLP
  - **Ex:** Red neuronal *skip-gram* para word2vec embeddings
  - De texto (secuencias de palabras) a grafos (secuencias de nodos)
- **Idea:**  $\mathbf{z}_i$  similares a nodos que tiendan a co-ocurrir en random walks sobre  $G$
- Mirar oraciones en NLP como random walks sobre el vocabulario
  - Generar random walks cortos sobre  $G$  para samplear secuencias de nodos
  - Aprender distribuciones posicionales de nodos como con palabras [Perozzi et al'14]
- Prob.  $P(j | i)$  de visitar  $j$  en un random walk de largo  $T$  desde  $i$   
⇒ Medida de similitud (asimétrica)  $[s(\mathbf{W})]_{ij}$  a decodificar desde  $\mathbf{Z}$

# Random walk approaches

- Pares de entrenamiento  $\{i, j\} \in \mathcal{V}_{obs}^{(2)}$  sampleados de random walks cortos
  - Para cada  $i \in \mathcal{V}$ ,  $N$  se sorteian pares  $\{i, j_1\}, \dots, \{i, j_N\}$  de  $P(j | i)$
  - Largo de cada paseo es  $T \in \{2, \dots, 10\}$
- Término de regularización del grafo: cross-entropy loss

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

⇒ Composición de un softmax y un decodificador de producto externo

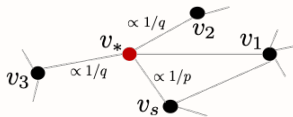
$$\hat{W}_{ij} = \frac{e^{\mathbf{z}_i^\top \mathbf{z}_j}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}}$$

⇒ Implica una aproximación  $\hat{W}_{ij} \approx [s(\mathbf{W})]_{ij} = P(j | i)$

- Evaluar el denominador del softmax es desafiante (complejidad  $\mathcal{O}(N_v)$ )

# DeepWalk y node2vec

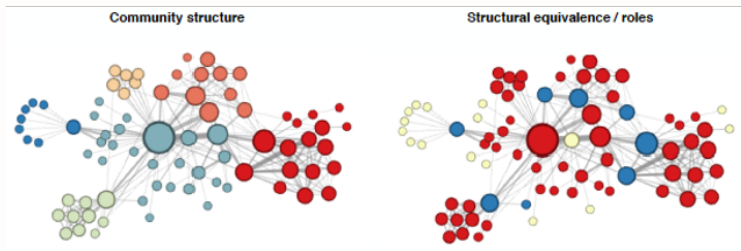
- **DeepWalk** sortea random walks de forma uniforme (equiprobable en los vecinos)
  - Matriz de probabilidad de transición  $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$
  - Usan *softmax jerárquico* para calcular  $\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}$  usando árboles binarios
- **Node2Vec** usa una definición flexible de random walks (sesgados)
  - Interpola suavemente entre paseos del tipo **BFS (Breadth First Search)** o **DFS (Depth First Search)**
  - Efectivo para capturar **roles estructurales** o **estructuras de comunidades**
  - Aproxima  $\sum_{k \in \mathcal{V}^*} e^{\mathbf{z}_i^\top \mathbf{z}_k}$  muestreando  $\mathcal{V}^*$



- Hyperparámetros  $p$  (de retorno) y  $q$  (in-out). Después de hacer  $v_s \rightarrow v_*$ 
  - (i) Controla la probabilidad de visitar nodos ( $v_* \rightarrow v_s$ ); o
  - (ii) Quedarse cerca del nodo predecesor ( $v_* \rightarrow v_1$ ); o
  - (iii) Moverse “más lejos” ( $v_* \rightarrow \{v_2, v_3\}$ )

# Random walks sesgados

- **Ex:** grafo de interacción entre personajes de la novela ‘Les Miserables’



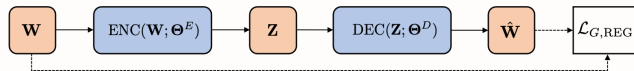
- node2vec interpola entre capturar estructuras **globales** y **locales**
  - El coloreo de la izquierda recupera membresía a comunidades (**global**)
  - El coloreo de la derecha indica roles que juega entre vecinos (**local**)

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

# Limitaciones

- **Shallow embeddings:** el encoder es muy sencillo
  - ⇒ Optimiza directamente un único embedding  $\mathbf{z}_i$  para cada nodo  $i \in \mathcal{V}$
- No se comparten parámetros entre nodos en el encoder
  - Estadísticamente ineficiente, compartir parámetros actúa como un regularizador
  - Computacionalmente ineficiente, el número de parámetros es  $\mathcal{O}(N_v)$
- No permite aprovechar señales
  - Perdemos atributos potencialmente muy informativos respecto a la posición y rol en  $G$
- Inherentemente transductivos
  - Problema para grafos grandes o dinámicos
  - No generalizan a otros grafos más allá de  $G$  (usado en training)

# Autoencoders



## ■ Autoencoders

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

⇒ Incorporamos  $\mathbf{W}$  en el encoder

## ■ Usamos una **red neuronal profunda** en encoder y decoder

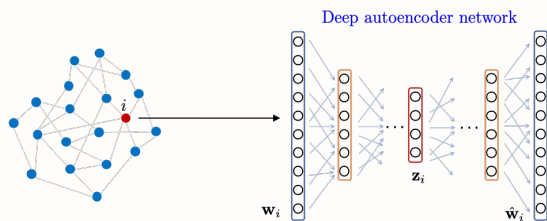
⇒ Habilidad para modelar no linealidades

⇒ Lleva a representaciones más complejas

## ■ Se entrena el modelo end-to-end minimizando una función de pérdida

# Métodos de neighborhood autoencoder

- Sea  $\mathbf{w}_i \in \mathbb{R}^{N_v}$  la  $i$ -ésima columna de  $\mathbf{W}$   
⇒ Captura la información de vecinos de  $i \in \mathcal{V}$



- **Objetivo del autoencoder:** reconstruir  $\mathbf{w}_i$  a partir del embedding  $\mathbf{z}_i$  aprendido

# Structural deep network embedding

- **Structural deep network embedding (SDNE)** minimiza

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

⇒ Incorpora el costo de Laplacian eigenmaps

- Usa deep autoencoders por nodo (parámetros compartidos)

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

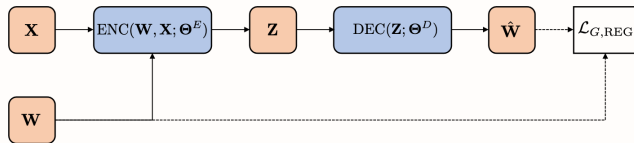
- Via  $\mathbf{w}_i$ , se regulariza el encoder con la topología de  $G$

- **Desventaja:** la dimensión de entrada está fija a  $N_v$ , costoso para grafos  $G$  grandes

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



# Graph neural networks



## ■ Graph Neural Networks

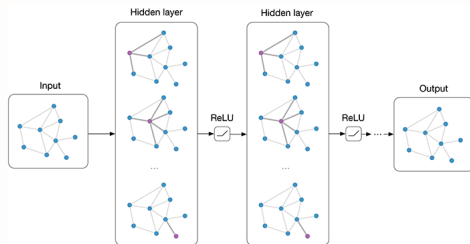
$$\mathbf{Z} = \text{ENC}(\mathbf{W}, \mathbf{X}; \Theta^E)$$

⇒ Usa **señales en el grafo**  $\mathbf{X}$  y topología  $\mathbf{W}$  en el encoder

## ■ Genera embedding $\mathbf{z}_i$ por agregación de señales en $\mathcal{N}_i$

- **Convolutacional**: implementación local y distribuida
- **Eficiencia**: dimensión de parámetros independiente de  $N_v$
- **Regularización**: a través de los parámetros compartidos
- **Inductivos**: genera embeddings para nodos no vistos en entrenamiento

# Convolutional graph autoencoders



- Graph autoencoders (GAE) usa una GCN encoder para aprender embeddings

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

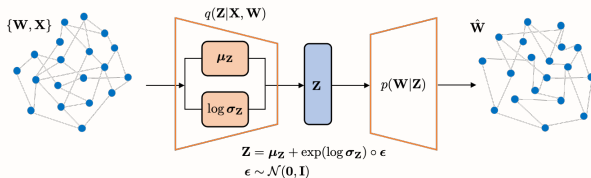
- Sigmoid cross entropy loss entre  $\mathbf{W}$  y la salida del decoder  $\hat{\mathbf{W}}$

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

⇒ Decoder de producto externo:  $\hat{\mathbf{W}} = \text{DEC}(\mathbf{Z}; \Theta^D) = \mathbf{Z}\mathbf{Z}^\top$

⇒ Modelo no probabilístico, adecuado para grafos sin pesos

# Variational graph autoencoders



- **Objetivo:** entrenar un decoder probabilístico para generar grafos realistas

$$\hat{W} \sim p(W | Z)$$

dadas variables latentes de un encoder probabilístico  $Z \sim q(Z | X, W)$

- Minimiza error de reconstrucción dados grafos y señales de entrenamiento
- Post entrenamiento, podemos tirar el encoder y generar grafos  $\hat{W} \sim p(W | Z)$ 
  - Dadas variables latentes  $Z \sim p(Z)$  muestreadas de una distribución a priori

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

# Encoder y decoder probabilístico

- **Encoder:** modelo simple de inferencia parametrizado por GCNs

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

- Dos GCNs separadas para generar media y varianza

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

- Dados  $\boldsymbol{\mu}_{\mathbf{Z}}$  y  $\log \boldsymbol{\sigma}_{\mathbf{Z}}$ , podemos muestrear embeddings latentes via

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

(reparametrization trick)

- **Decoder:** modelo generativo basado en productos externos

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

- $\sigma(\cdot)$  es la función sigmoide logística

■ **Prior:** embeddings latentes de nodos asumidos  $\mathbf{z}_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$

# Entrenando VGAE

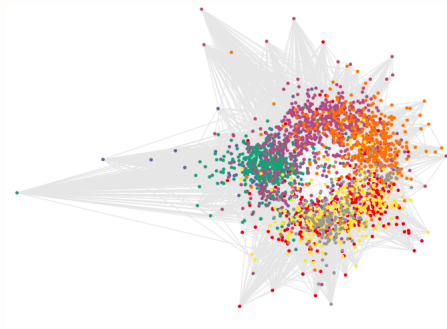
- Maximizar el **evidence likelihood lower bound (ELBO)**

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

- $\text{KL}(\cdot, \cdot)$  es la Kullback-Leibler divergence
  - Los parámetros a entrenar  $\Theta$  son los filtros de la GCN  $\mathbf{H}_k$
  - Requiere un conjunto de grafos de entrenamiento  $\{\mathbf{W}_1, \mathbf{X}_1\}, \dots, \{\mathbf{W}_P, \mathbf{X}_P\}$
- Generar una distribución sobre  $\mathbf{Z}$  que satisfaga dos objetivos (contrapuestos)
    - (a) Los  $\mathbf{Z}$  muestreados sean suficientemente ricos para que el decoder reconstruya  $\mathbf{W}$
    - (b) Que la distribución  $q(\mathbf{Z} | \mathbf{X}, \mathbf{W})$  esté lo más cerca posible del prior  $p(\mathbf{Z})$
  - Objetivo (b) es crítico para **generar nuevos grafos después de entrenar**  
 $\Rightarrow$  Muestrear  $\mathbf{Z} \sim p(\mathbf{Z}) \rightarrow$  Decodificar  $\hat{\mathbf{W}} \sim p(\mathbf{W}_i | \mathbf{Z})$

# VGAE en acción

- **Ex:** Cora dataset con  $N_v = 2708$  papers y  $N_e = 5429$  citas
  - Señales de grafos: presencia/ausencia de 1433 palabras de un diccionario



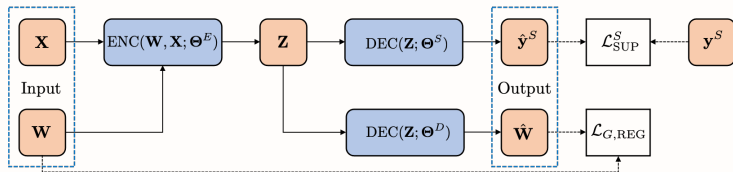
- Espacio latente aprendido con VGAE
  - ⇒ Colores indican la etiqueta (disciplina, no usada en entrenamiento)

# Roadmap

- 1 The network embedding problem
- 2 Una taxonomía de modelos de graph embedding
- 3 Unsupervised graph embedding
- 4 Applications

# Applications

- GRL has been successfully 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



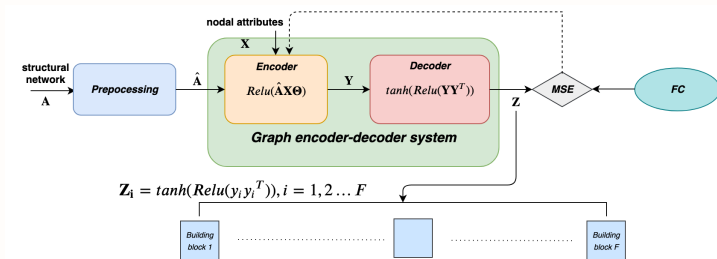
# Networks of the brain

- **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

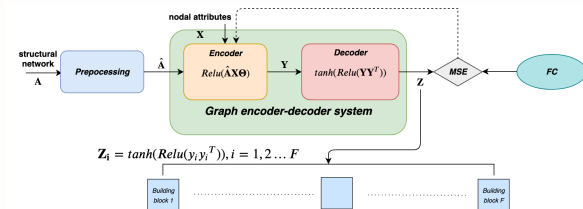
## Model architecture: encoder

- Input SC network  $\mathbf{A} \in \mathbb{R}^{N \times N}$ ,  $N$  regions from brain atlas
  - ⇒ Edge weights represent SC between brain regions
  - ⇒ 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
  - Nodal attributes, e.g. intrinsic properties of brain regions
  - Graph topology information, e.g. regional connection strengths
- A single-layer GCN used for encoder to learn node embeddings

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

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

# Model architecture: decoder

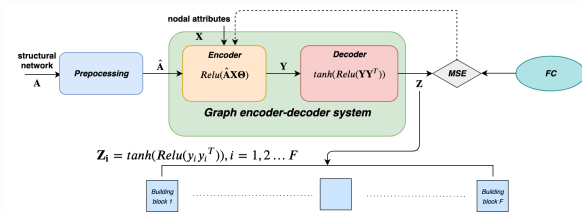


- Node embeddings  $Y \in \mathbb{R}^{N \times F}$  go through the outer-product decoder

$$Z = \tanh(\text{ReLU}(YY^T)) \in \mathbb{R}^{N \times N}$$

- 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  $Z$  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  $\mathbf{y}_i \mathbf{y}_i^T$

$$\mathbf{Z}_i = \tanh(\text{ReLU}(\mathbf{y}_i \mathbf{y}_i^T)), \quad i = 1, \dots, F$$

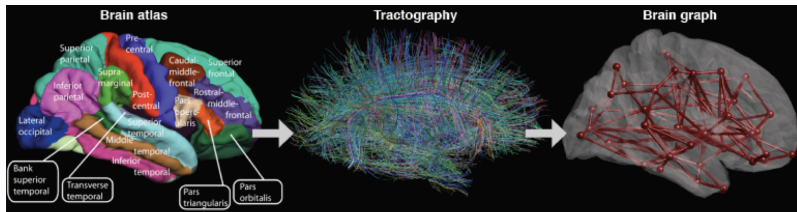
- $\mathbf{Z}_i \Leftrightarrow$  outputs of individual filters in graph convolutional layer

$\Rightarrow$  View as **building blocks** of FC network

- Reveal details about generation of FC patterns from SC networks

## Numerical tests: data

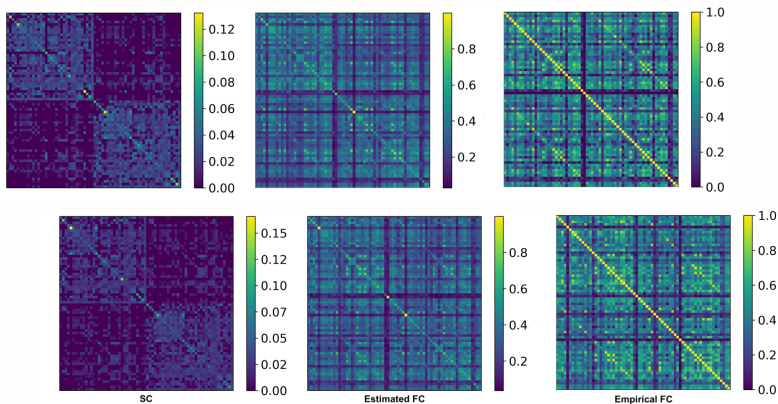
- $P = 1058$  healthy subjects from Human Connectome Project (HCP)
- Preprocessed SC network  $\mathbf{A}$  from diffusion MRI
  - ⇒ Fiber counts between  $N = 68$  cortical surface regions



- Preprocessed FC network from functional MRI
  - ⇒ Blood oxygen-level dependent (BOLD) signals
  - ⇒ Estimated FC  $\Leftrightarrow$  Pearson correlation between BOLD signals
- One-hot encoding as the signal on each graph node ( $\mathbf{X} = \mathbf{I}_N$ )

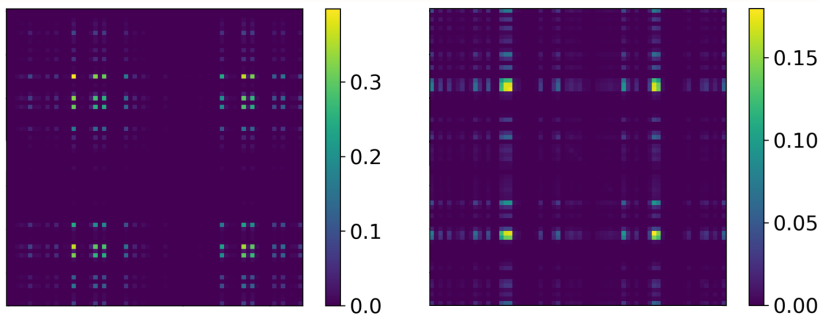
# Numerical tests: FC reconstruction performance

- 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

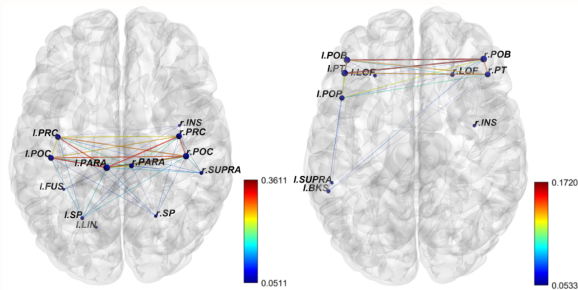
- Investigate the latent variables learnt during model training
  - Output of each graph filter in the graph convolution layer
  - **Building blocks**  $\mathbf{Z}_i$  that generate reconstructed FC graph



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



## Numerical tests: component graphs

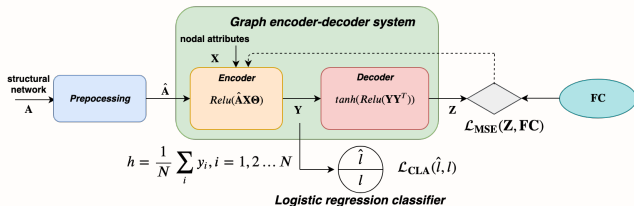


- **Left:** subnetwork of regions in frontal and parietal lobe
  - *Precentral (PRC), Paracentral (PARA)*, motor/sensory functions
  - *Postcentral (POC), Superior Parietal (SP)*, spatial/somatosensory
- **Right:** subnetwork of regions in Inferior Frontal Gyrus
  - *Parsopercularis (POP), Parsorbitalis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

# Supervised GRL for brain network classification

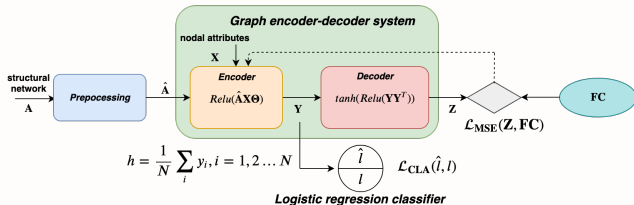
Model the relationship between brain structural and functional network

- **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

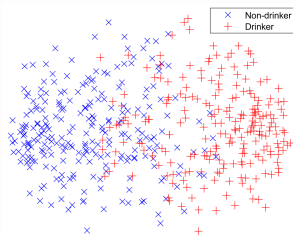
# Model architecture: classifier



- Apply row-wise average-pooling on the encoder output  $Y$ 
  - ⇒ 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:  $\mathcal{L} = \mathcal{L}_{\text{MSE}}(Z, \text{FC}) + \lambda \times \mathcal{L}_{\text{CLA}}(\hat{l}, l)$

# Numerical tests: results

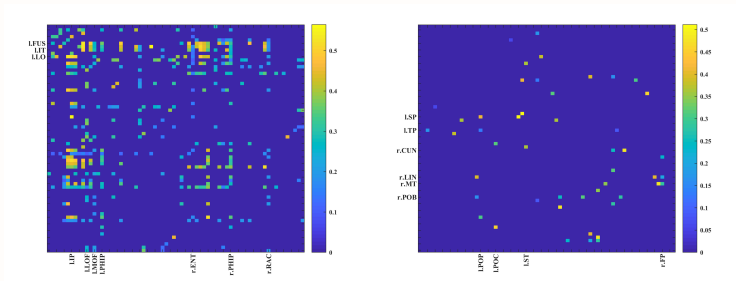
- $P = 466$  subjects from Human Connectome Project (HCP)
  - Two classes: 245 non-drinkers, 221 heavy drinkers
- MSE between reconstructed and empirical FC networks
  - Average test reconstruction error = 0.034174 with std = 0.00208
  - Captured population patterns of SC-FC relationship
- Classification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



- Reduced dimensional graph embeddings exhibit cluster structure

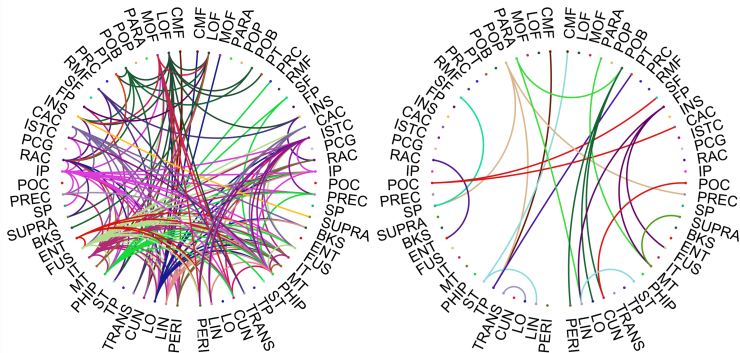
# Numerical tests: reconstructed FC

- 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  
⇒ 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  
⇒ Preserve network structure and **distances between graphs**
  - Cluster learned graph embeddings via K-means algorithm  
⇒ Each cluster represents one temporal stage
- Siamese encoder better for input graphs with divergent structures

# Grafos de fútbol

- $T = 145$  grafos de fútbol desde 1872 a 2016
  - **Nodos:**  $N_v = 238$  selecciones nacionales jugando partidos oficiales
  - **Aristas:**  $W_{ij}(t)$  es la cantidad de partidos entre  $\{i, j\}$  en el año  $t$
  - Los datos incluyen 39,052 partidos en 145 años

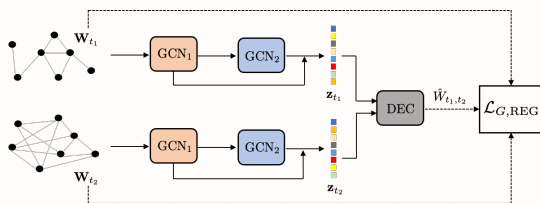


- Esperamos encontrar diferentes etapas en la historia de este deporte



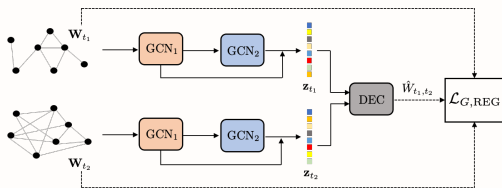
# Arquitectura del modelo: encoder

- **Input:**  $\binom{145}{2} = 10440$  pares  $\{\mathbf{W}_{t_1}, \mathbf{W}_{t_2}\}$  de grafos



- **Modelo:** Siamese encoder network with two GRL pipelines
  - Two-layer GCN with parameters shared across pipelines
- **Output:** embeddings de grafos  $\{\mathbf{z}_{t_1}, \mathbf{z}_{t_2}\}$  para cada grafo de entrada
  - Concatenate node embeddings learned at each layer
  - Average graph pooling from node embeddings

## Model architecture: decoder



- **Idea:** los embeddings preservan **distancias** entre pares de grafos

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

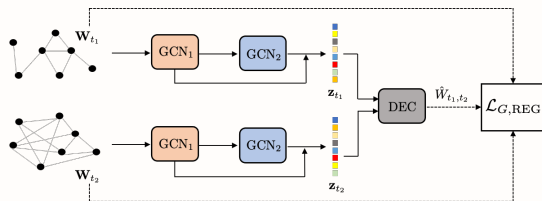
⇒ Euclidean distance decoder:  $\hat{W}_{t_1, t_2} = \|\mathbf{z}_{t_1} - \mathbf{z}_{t_2}\|_2$

- Prescribed distances between input graphs encoded in

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

⇒ Like MDS and IsoMAP but at **graph-**(not node-)level

## Model architecture: graph distance



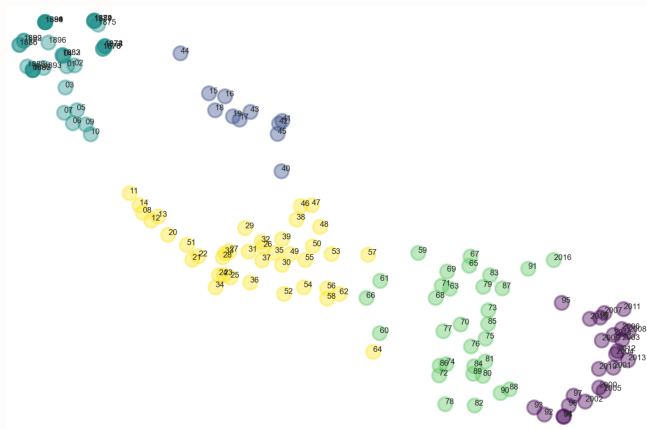
- User defined graph distance  $d_G(\mathbf{W}_{t_1}, \mathbf{W}_{t_1})$
- Ex: **Spectrum distance**
  - Distance between spectrum (eigenvalues) of both Laplacian matrices
- Ex: **Vertex-edge-overlap (VEO)**
  - Measure structural similarity between graphs

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

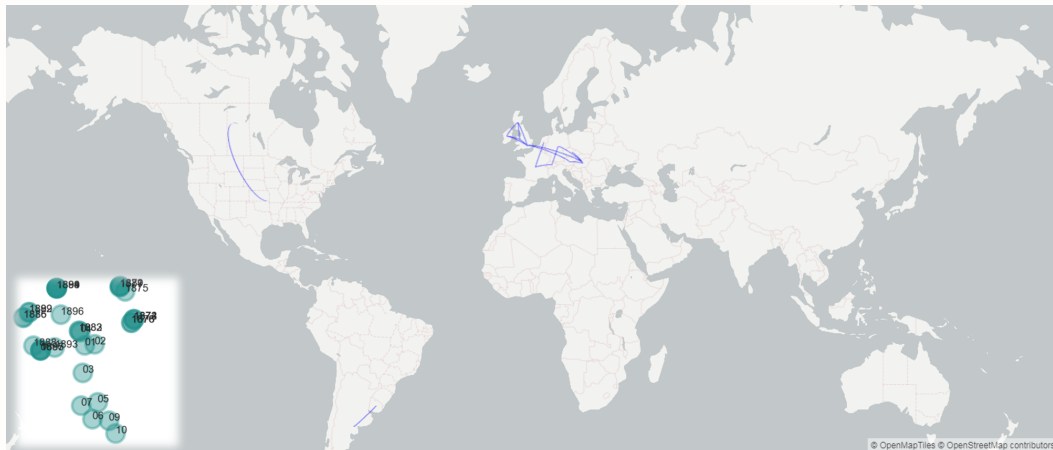
- Distance computed as one minus normalized VEO

# Graph clustering results

- K-means clustering applied to learned graph embeddings in  $\mathbb{R}^{48}$   
⇒ Number of clusters  $K = 5$  chosen by elbow rule

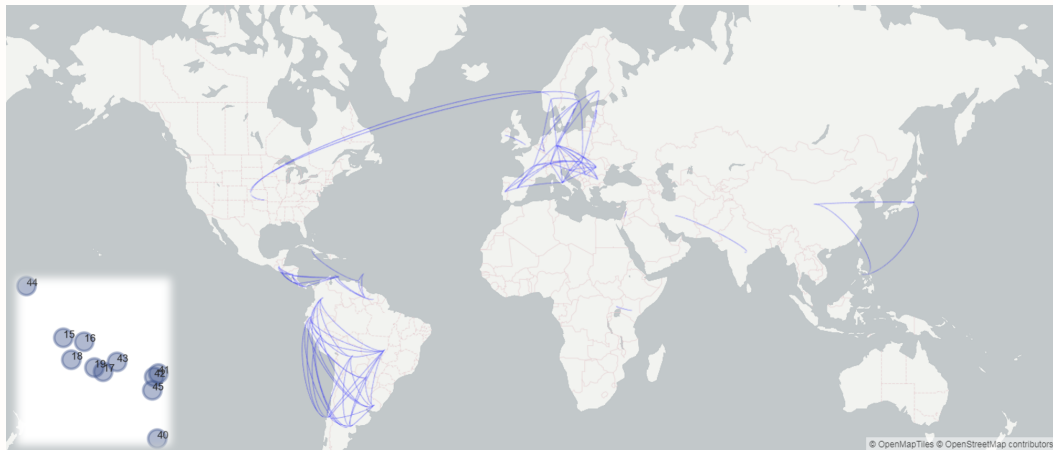


## Cluster 1: early 20th century



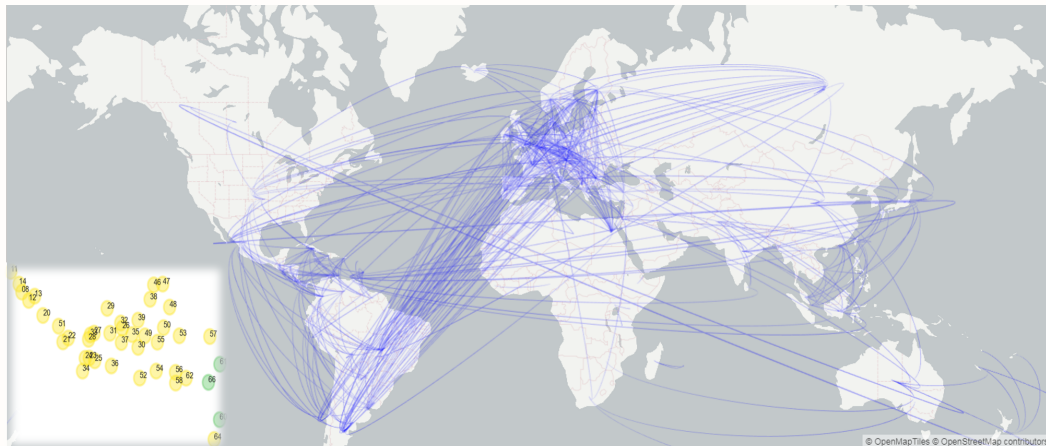
■ Mostly regions around UK and the River Plate

## 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

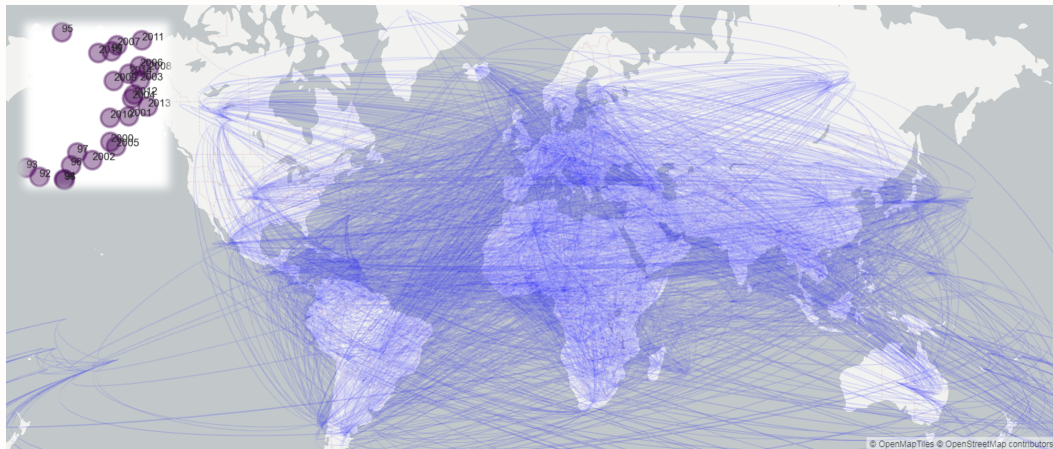
## Cluster 4: Modern development



- Modern expansion of football with Africa and Asia involved



## Cluster 5: Current landscape



- Global nature of the game is patently apparent

# Open research directions

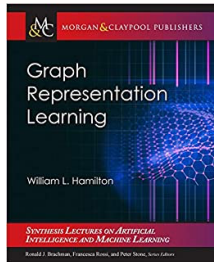
■ GRL is a very active area of research. Many open questions remain:

- 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?

arXiv:2005.0367v1 [cs.LG] 7 May 2020



arXiv:1709.05884v3 [cs.LG] 10 Apr 2018



# 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
- Neighborhood aggregation
- Higher-order proximity
- Matrix factorization
- Random walks
- Laplacian eigenmaps
- Hyperbolic geometry
- Variational autoencoders
- Brain network analysis
- Graph convolutional network