FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

# Aprendizaje Automático para Datos en Grafos

## Graph representation learning

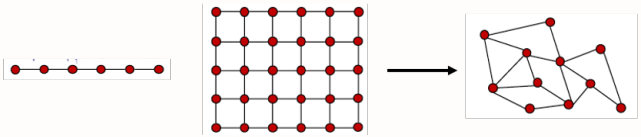
Marcelo Fiori
Muy basado en transparencias de **Gonzalo Mateos**

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

10 de noviembre, 2022

# 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., multiples 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

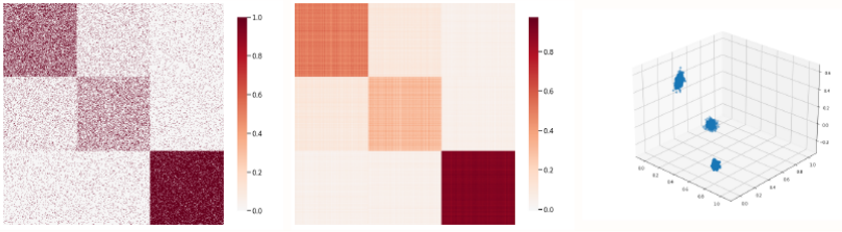
# Embeddings de grafos

- 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 $\{\mathbf{z}_i\}_{i \in \mathcal{V}}$
  $\Rightarrow$ Criterio es preservar propiedades locles y globales del grafo
- Salida es una matriz de embeddings de nodos $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_v}]^\top \in \mathbb{R}^{N_v \times d}$
  $\Rightarrow$ Elegir $d \ll N_v$ para escalabilidad
  $\Rightarrow$ 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

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

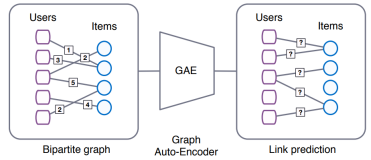
- Sin $\mathbf{X}$, el embedding $\{\mathbf{W}\} \mapsto \mathbf{Z}$ se dice featureless

  $\Rightarrow$ 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: Embeding de grafos dinámicos para clustering temporal
- Típicamente se necesitan señales $\mathbf{X}$ para hacer embedding inductivo

# Embeddings supervisados y no-supervisados

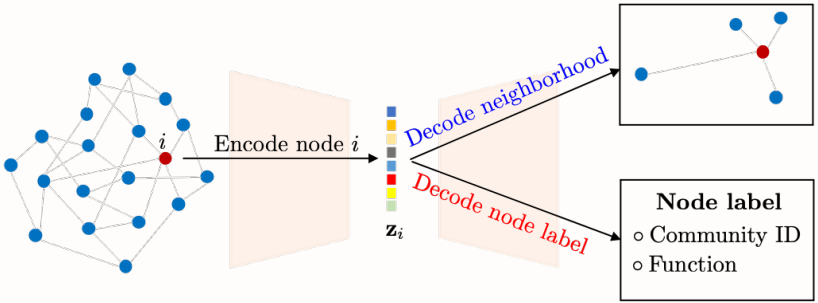
**Unsupervised network embedding**

- 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



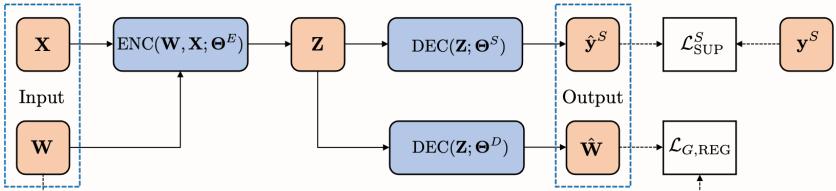

W. L. Hamilton et al, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, 2018

# Roadmap

1. The network embedding problem
2. Una taxonomía de modelos de graph embedding
3. Unsupervised graph embedding

# 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

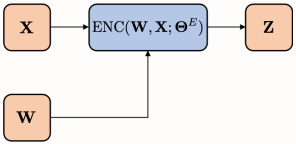
- **Q:** ¿Cuáles son los componentes constitutivos del modelo?

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

# Input

- Grafo no dirigido $G(\mathcal{V}, \mathcal{E})$, con $|\mathcal{V}| = N_v$ y $|\mathcal{E}| = N_e$
  $\Rightarrow$ 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

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

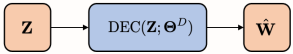
  $\Rightarrow$ Parámetros entrenables $\boldsymbol{\Theta}^E$

- Combina la estructura del grafo con señales para producir un embedding

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

  $\Rightarrow$ Captura diferentes propiedades del grafo en base al tipo de supervisión

# Decoder (I)




- Graph decoder network

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

  $\Rightarrow$ Parámetros entrenables $\boldsymbol{\Theta}^D$

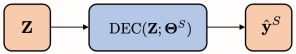
- Usa $\mathbf{Z}$ para producir scores de (di)similitud $\hat{W}_{ij}$ para cada $\{i,j\} \in \mathcal{V}^{(2)}$

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

  $\Rightarrow$ Reconstrucción no supervisada del grafo

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

# Decoder (II)




- Classification network

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

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

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

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

$\Rightarrow$ Aprendizaje (semi)-supervisado para clasificación de nodos/grafos

# Output

- Matriz de similitud reconstruida $\hat{\mathbf{W}} \in \mathbb{R}^{N_v \times N_v}$

  $\Rightarrow$ 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 $\hat{\mathbf{Y}}^N \in [0,1]^{N_v \times |\mathcal{Y}|}$

  $\Rightarrow$ 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 $\mathcal{Y} = \{0,1\}^{\#\text{relation types}}$

  $\Rightarrow$ Cuando #relation types $= 1$ (i.e., predicción de enlaces), salida $\hat{\mathbf{W}}$
- Graph-level: $\hat{y}^G \in \mathcal{Y}$

  $\Rightarrow$ Usando $\mathbf{W}$, convertir $\mathbf{Z}$ a $\hat{y}^G$ via graph pooling

# Funciones de pérdida

- Supervised loss

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

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

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

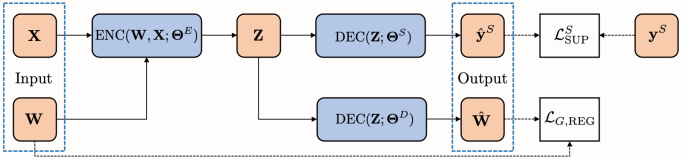
- Graph regularization loss

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

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

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

# Función objetivo




- Weight regularization loss

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

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

- Función objetivo GraphEDM total

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

  $\Rightarrow$ 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 $\mathbf{Z} = \mathrm{ENC}(\mathbf{\Theta}^E) = \mathbf{\Theta}^E$
  - Un simple embedding lookup
- Graph auto-encoding methods $\mathbf{Z} = \mathrm{ENC}(\mathbf{W}; \mathbf{\Theta}^E)$
  - Transductivos como los shallow embeddings, no hay señal $\mathbf{X}$, así que funciona para grafo fijo $G$
- Graph regularization methods $\mathbf{Z} = \mathrm{ENC}(\mathbf{X}; \mathbf{\Theta}^E)$
  - Usamos $\mathbf{W}$ via $\mathcal{L}_{G,\mathrm{REG}}$ para regularizar los embeddings
- Neighborhood aggregation methods $\mathbf{Z} = \mathrm{ENC}(\mathbf{W}, \mathbf{X}; \mathbf{\Theta}^E)$
  - 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

# Unsupervised graph embedding

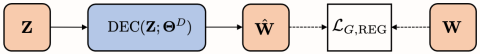
- **Objetivo:** Aprender embeddings de nodos que preserven estructura del grafo
- Optimizamos para reconstruir cierta matriz de (di)similitud entre nodos $s(\mathbf{W})$

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

  - 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 $\{i, j\} \in \mathcal{V}^{(2)}_{obs}$, 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. $[s(\mathbf{W})]_{ij} = \text{P}\left(v_j \,\middle|\, v_i\right)$ que $i, j \in \mathcal{V}$ co-ocurran en paseos al azar (random walks)

# Shallow embeddings




- Shallow embedding methods

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

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

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

  $\Rightarrow$ Enfoques más recientes se basan en random walks (analogías con NLP)

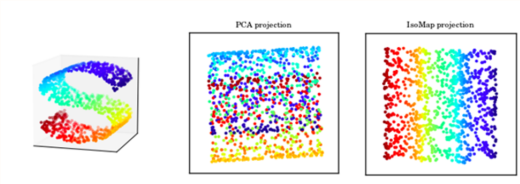
# Métodos basados en distancia

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

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

  $\Rightarrow$ 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) 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 $\mathbf{L} = \mathbf{D} - \mathbf{W}$

  $\Rightarrow$ Esquema no lineal de reducción de dimensionalidad que preserva propiedades locales

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

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

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

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

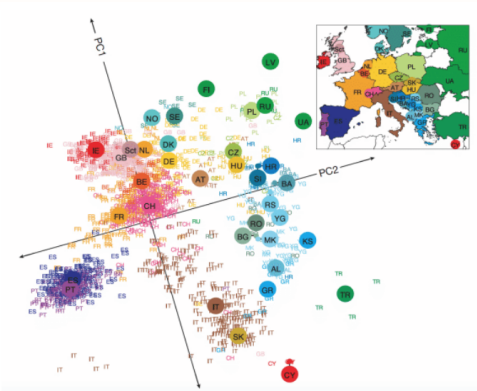
  $\Rightarrow$ Embeddings estarán cerca en $\mathbb{R}^d$ si $i$, $j$ están bien conectados en $G$

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' ($d = 2$)

  $\Rightarrow$ Consistente con el origen de los individuos en Europa




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

# Non-Euclidean embedding spaces

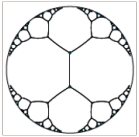
- **Idea:** embedding de grafos con estructura jerárquica en espacio hiperbólico

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

$\Rightarrow$ Poincaré distance decoder:

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

- 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 $s(\mathbf{W})$

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

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

  $\Rightarrow$ 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 $\mathcal{L}_{G,\mathrm{REG}}(\mathbf{W}, \hat{\mathbf{W}}; \mathbf{\Theta})$ 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 sequencias de nodos
  - Aprender distribuciones posicionales de nodos como con palabras [Perozzi et al'14]
- Prob. $\mathrm{P}\left(j \,\middle|\, i\right)$ de visitar $j$ en un random walk de largo $T$ desde $i$

  $\Rightarrow$ 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 sortean pares $\{i, j_1\}, \ldots, \{i, j_N\}$ de $\mathrm{P}\left(j \,\middle|\, i\right)$
  - Largo de cada paseo es $T \in \{2, \ldots, 10\}$
- Término de regularización del grafo: cross-entropy loss

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

$\Rightarrow$ Composición de un softmax y un decodificador de producto externo

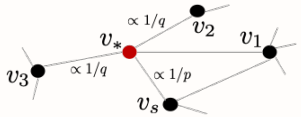
$$\hat{W}_{ij} = \frac{e^{\mathbf{z}_i^\top \mathbf{z}_j}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}}$$

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

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

# DeepWalk y node2vec

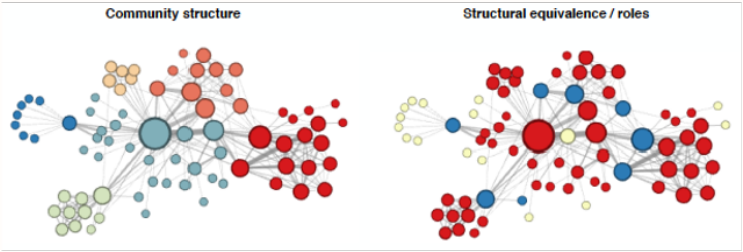
- DeepWalk sortea random walks de forma uniforme (equiprobable en los vecinos)
  - Matriz de probabilidad de transición $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$
  - Usan *softmax jerárquico* para calcular $\sum_{k \in \mathcal{V}} e^{\mathbf{z}_i^\top \mathbf{z}_k}$ usando árboles binarios
- Node2Vec 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 caputar 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 $v_s \to v_*$
  - (i) Controla la probabilidad de revisitar nodos ($v_* \to v_s$); o
  - (ii) Quedarse cerca del nodo predecesor ($v_* \to v_1$); o
  - (iii) Moverse "más lejos" ($v_* \to \{v_2, v_3\}$)

# 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

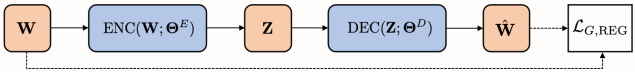
# Limitaciones

- Shallow embeddings: el encoder es muy sencillo

  $\Rightarrow$ 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

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

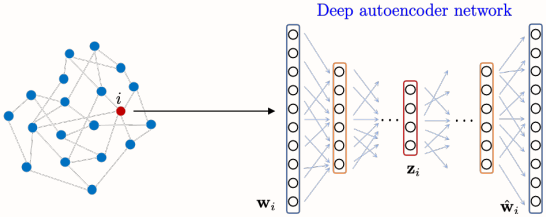
  $\Rightarrow$ Incorporamos $\mathbf{W}$ en el encoder

- Usamos una red neronal profunda en encoder y decoder
  - $\Rightarrow$ Habilidad para modelar no linealidades
  - $\Rightarrow$ 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 $\mathbf{W}$

  $\Rightarrow$ 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

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

  $\Rightarrow$ Incorpora el costo de Laplacian eigenmaps

- Usa deep autoencoders por nodo (parámetros compartidos)

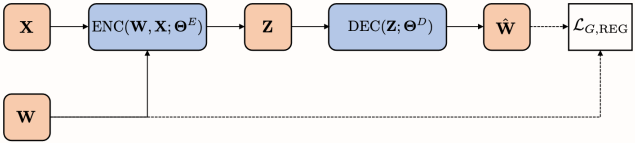
$$\mathbf{z}_i = \mathrm{ENC}(\mathbf{w}_i; \mathbf{\Theta}^E), \quad \hat{\mathbf{w}}_i = \mathrm{DEC}(\mathbf{z}_i; \mathbf{\Theta}^D)$$

  - 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



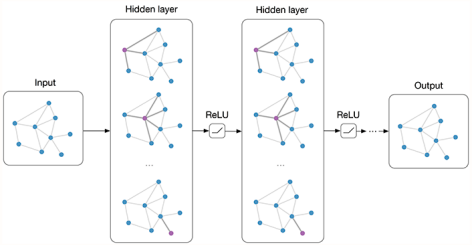

- Graph Neural Networks

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

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

- Genera embedding $\mathbf{z}_i$ por agregación de señales en $\mathcal{N}_i$
  - Convolucional: 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

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

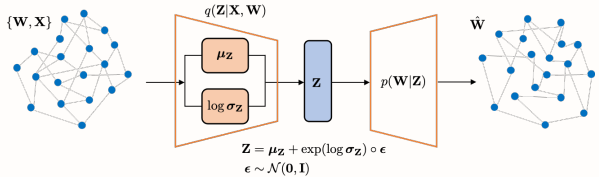
- *Sigmoid cross entropy loss* entre $\mathbf{W}$ y la salida del decoder $\hat{\mathbf{W}}$

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

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

$\Rightarrow$ Modelo no probabilístico, adecuado para grafos sin pesos

# Variational graph autoencoders




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

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

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

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

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

  - 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

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

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

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

# Entrenando VGAE

- Maximizar el evidence likelihood lower bound (ELBO)

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

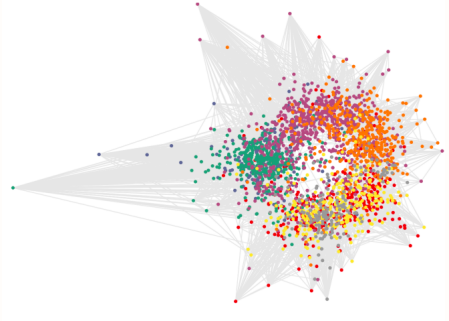
  - $\mathrm{KL}(\cdot,\cdot)$ es la Kullback-Leibler divergence
  - Los parámetros a entrenar $\boldsymbol{\Theta}$ son los filtros de la GCN $\mathbf{H}_k$
  - Requiere un conjunto de grafos de entrenamiento $\{\mathbf{W}_1, \mathbf{X}_1\}, \ldots, \{\mathbf{W}_P, \mathbf{X}_P\}$

- 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

  $\Rightarrow$ Colores indican la etiqueta (disciplina, no usada en entrenamiento)