Node embeddings

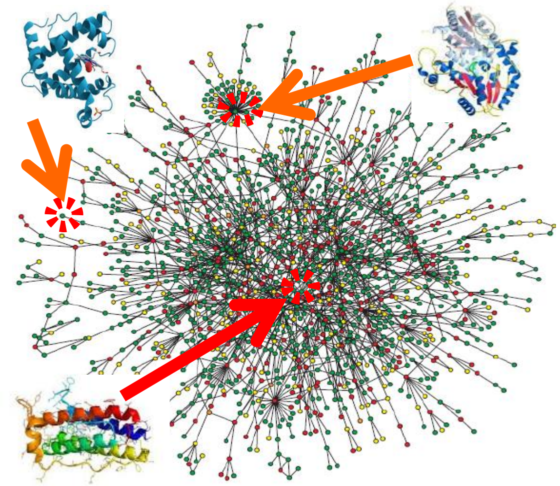
Hamilton et al. 2017. Representation Learning on Graphs: Methods and Applications. IEEE Data Engineering Bulletin on Graph Systems.

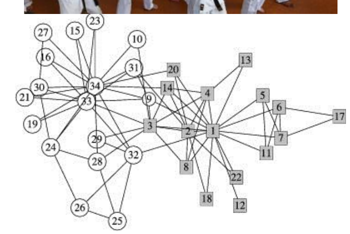
Jure Leskovec Lectures

Redes Complejas

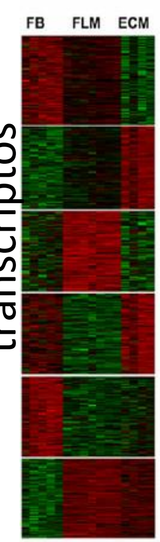
- A lo largo del curso vimos cómo podíamos emplear el lenguaje de redes para analizar la estructura de interacciones presentes en sistemas complejos

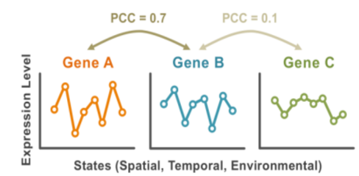
- A partir de G(V,E)
  - Propiedades a escala local, global, mesoscopica
  - Relacion con campos de features (Asortatividad)
  - Centralidad (vertices o enlaces *importantes*)
  - Motifs (conexionados *importantes*)
  - Roles Topologicos
  - Similaridad nodos, enlaces
  - Comunidades (heterogeneidades en escala mesoscopica)
  - Campos sobre grafos: hipótesis de suavidad
    priorizacion, clasificacion, regresion
  - ….

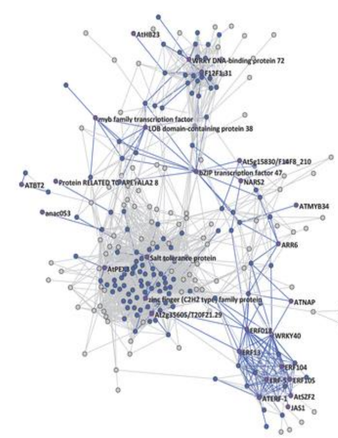




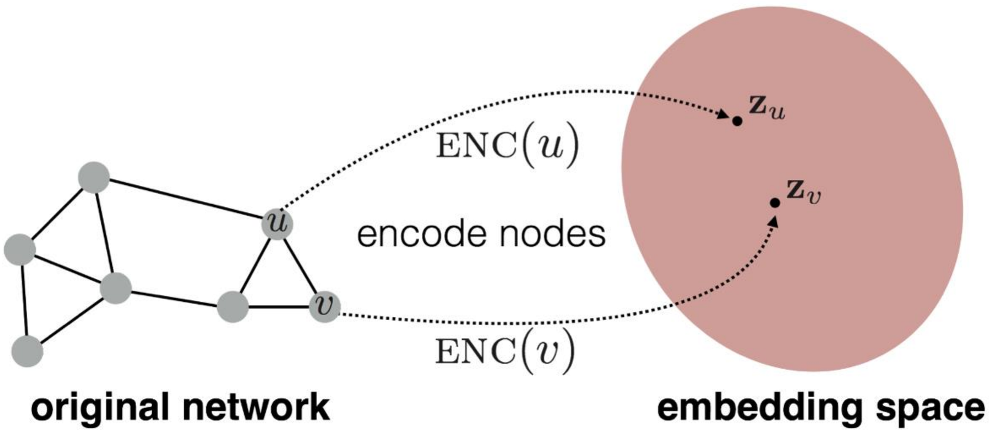
transcriptos







Caracterizando nodos

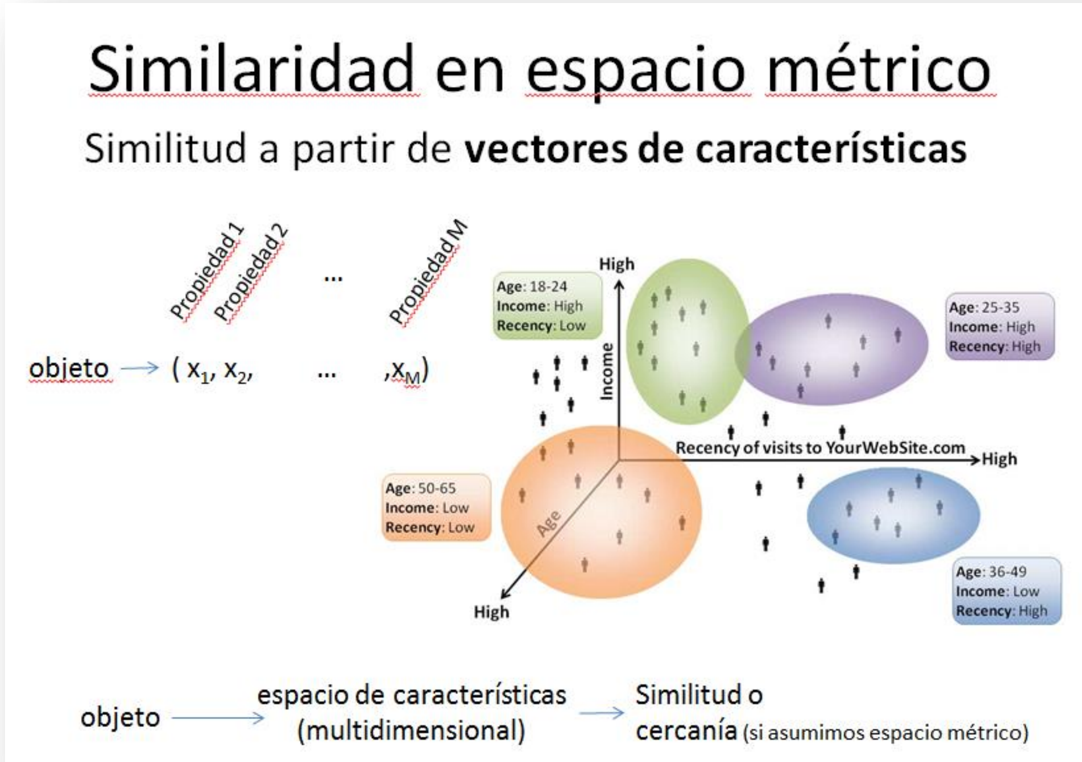


Supongamos que tenemos una red…G(V,E). Podemos conceptualizar a cada nodo como un punto en un espacio de baja dimensionalidad?

Por qué querriamos hacer esto? Muchas técnicas de aprendizaje automático podrían aplicarse: Clasificación, Priorización, Prediccion de nuevos enlaces, etc…

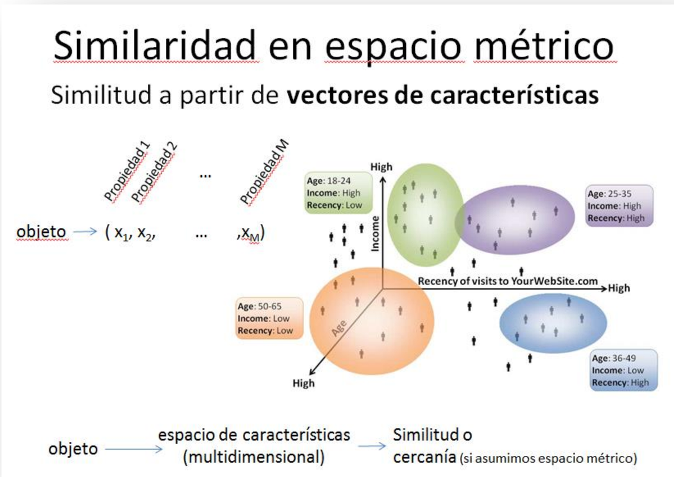
Lo bueno es que si encontramos una manera piola de hacer el embedding, nos podríamos evitar hacer *ingeniería de features* para definir el espacio de ***Z***´s donde trabajar

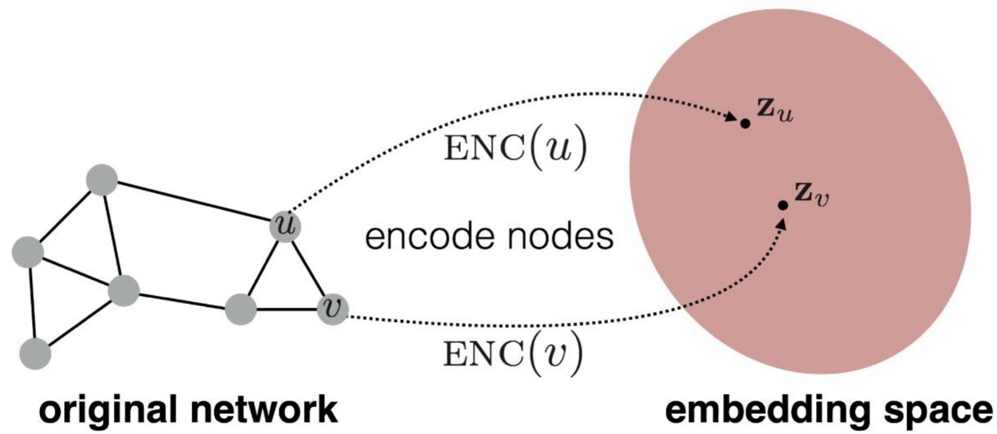
Recordando...



Lo bueno es que si encontramos una manera piola de hacer el embedding, nos podríamos evitar hacer *ingeniería de features* para definir el espacio de ***Z´***s donde trabajar

Caracterizando nodos

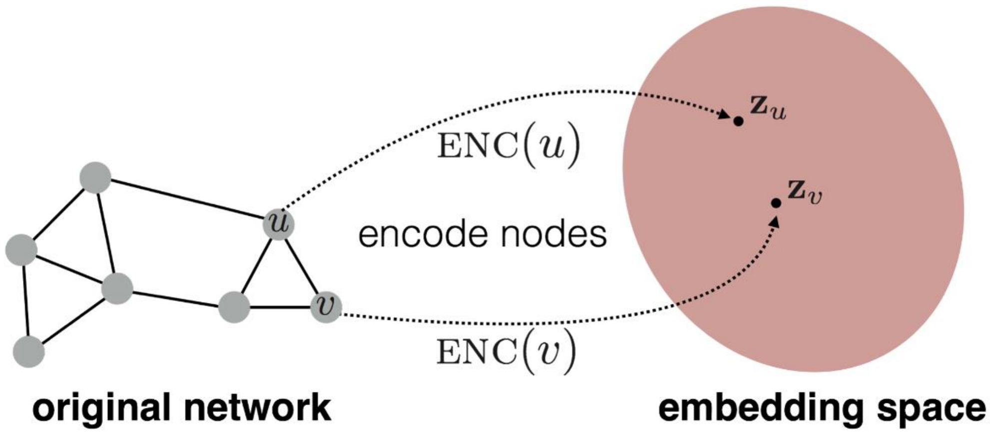




Idea: utilizar la información embebida en G(V,E) para encontrar un buen embeddig.

Lo que vamos a ver es una instancia de **Representation Learning on Networks**

Cómo hacemos la magia?



Idea: Tratamos de acomodar los puntos en el espacio d-dimensional de manera que **nodos similares** en el grafo se correspondan con **puntos cercanos** en el embeding

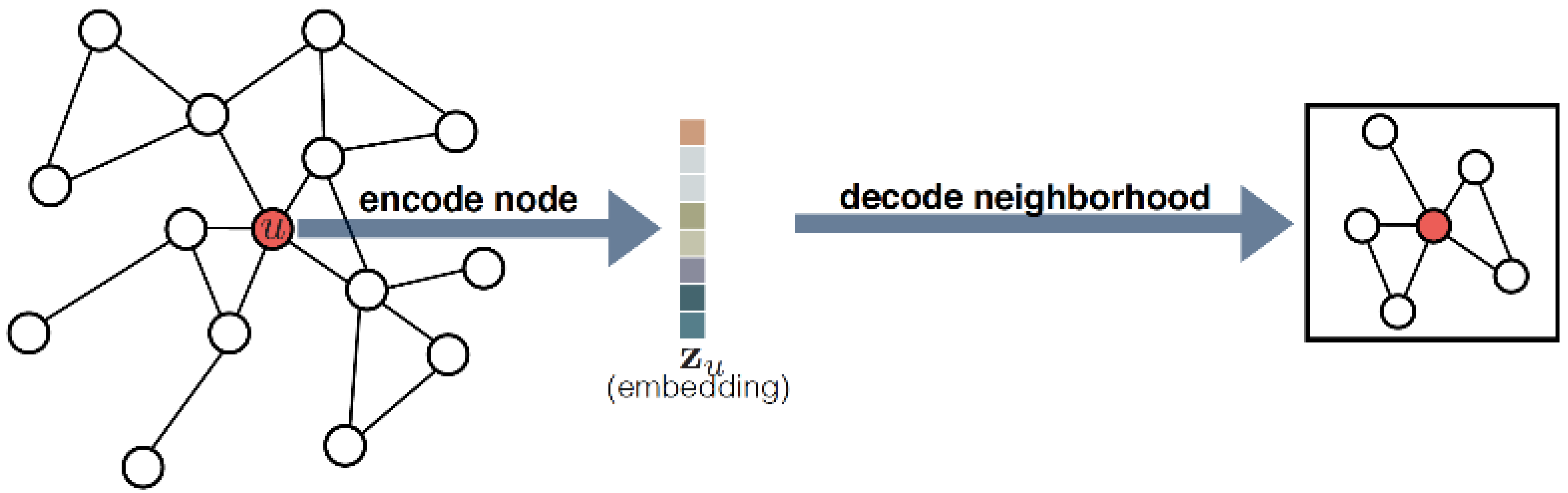
Vamos a llamar *encoder* a la función de mapeo: $Enc: V \rightarrow \mathbb{R}^d$

En nuestro caso $Enc(v) = Z[v]$.

fila de la matriz $Z \epsilon \mathbb{R}^{|V| \times d}$ que define el embedding

Abordaje Encoder-Decoder

Para encontrar un buen *encoder*...



El *encoder* mapea nodos al espacio d-dimensional

$$Enc(v) = Z[v].$$

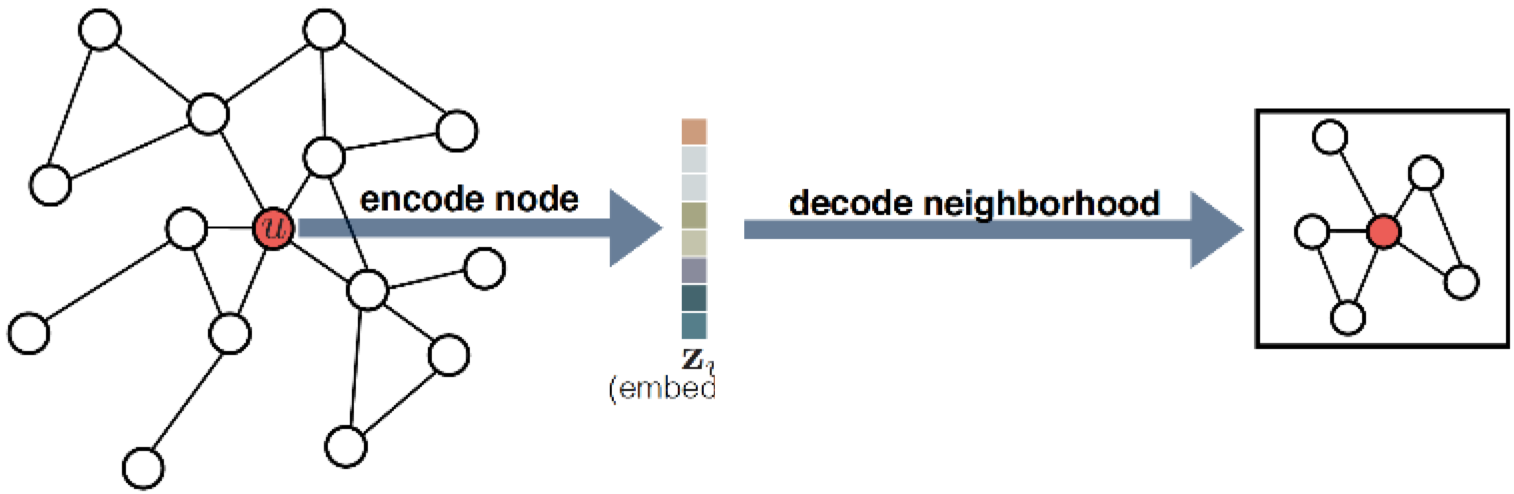
$$Z \epsilon \mathbb{R}^{|V| \times d}$$

El *decoder* reconstruye la vecindad de un nodo a partir de su vector representacion

$$Dec: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$$

Para cada par de ptos $z(u_1), z(u_2)$ estima una relación o similaridad entre los nodos correspondientes u_1 y u_2

Abordaje Encoder-Decoder



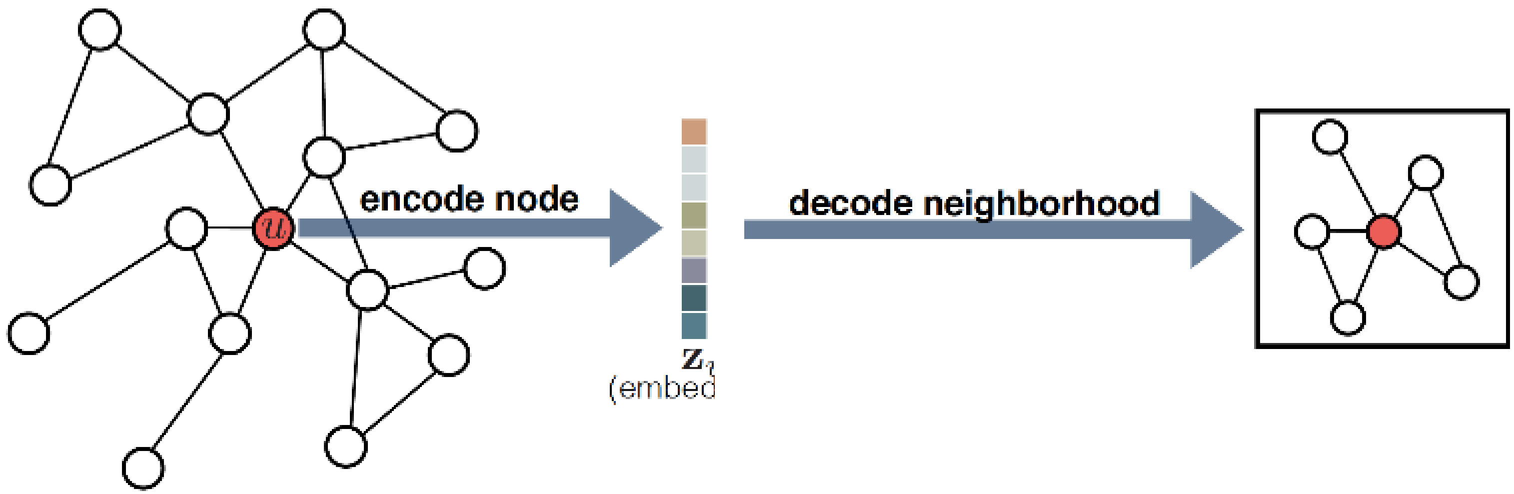
Para un buen *encoder…*

$$\text{DEC}(\text{ENC}(u), \text{ENC}(v) = \quad \text{DEC}(z_u, z_v) \approx S(u, v)$$

Medida de similitud entre puntos del embedding

Medida de similitud entre nodos del grafo

Minimizando perdidas



Para un buen *encoder*…

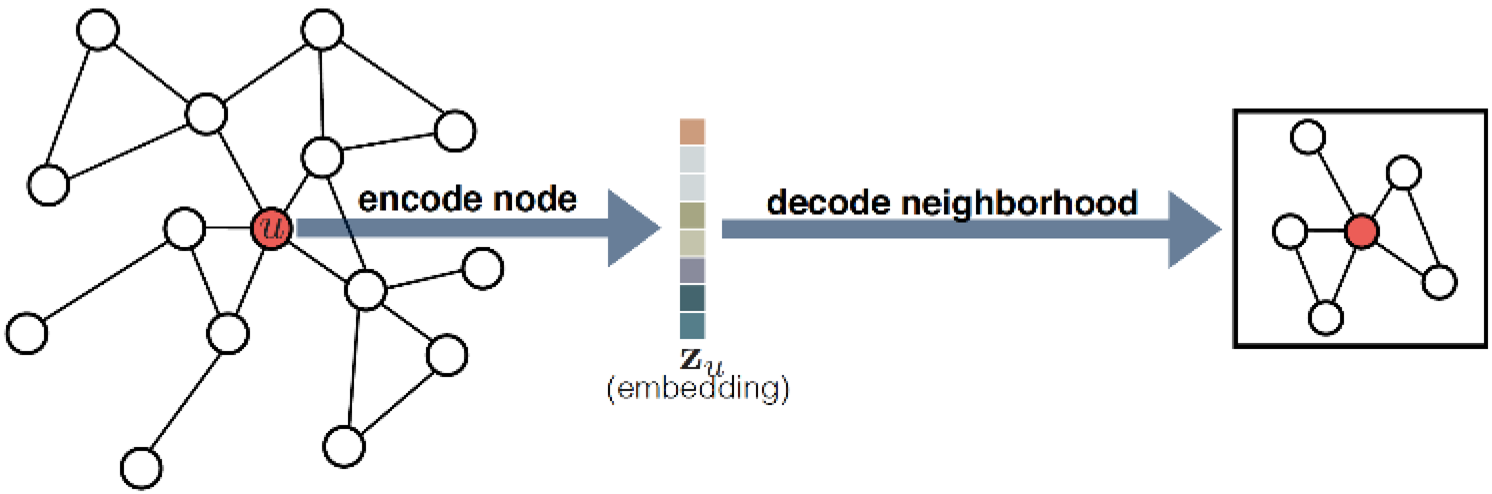
DEC(ENC(u), ENC(v)= DEC(z_u, z_v)$\approx S(u, v)$

Vamos a plantear el problema de encontrar un buen embedding como un problema de minimización de una función de pérdida

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u, v]\right)$$

$\ell: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ mide discrepancia entre similaridades

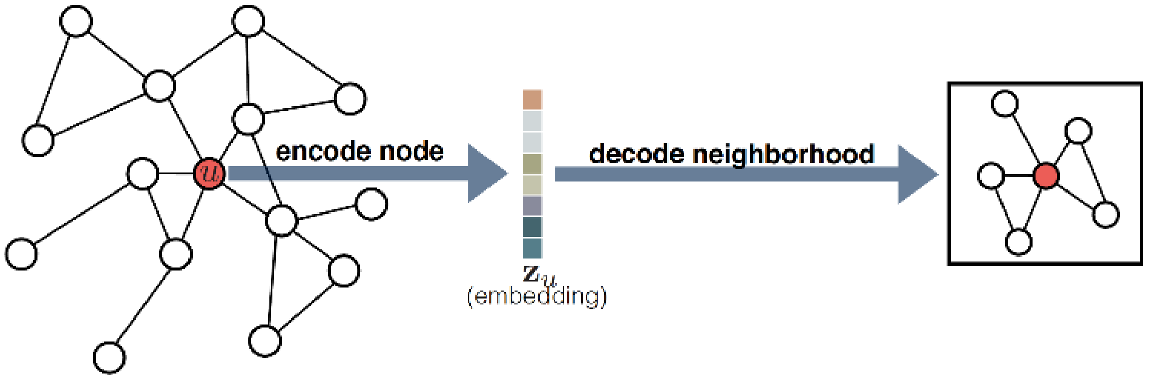
Minimizando perdidas



$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \underbrace{\ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)}$$

Por ejemplo: $\|{z_u}^T . z_v - A_{uv}\|^2$

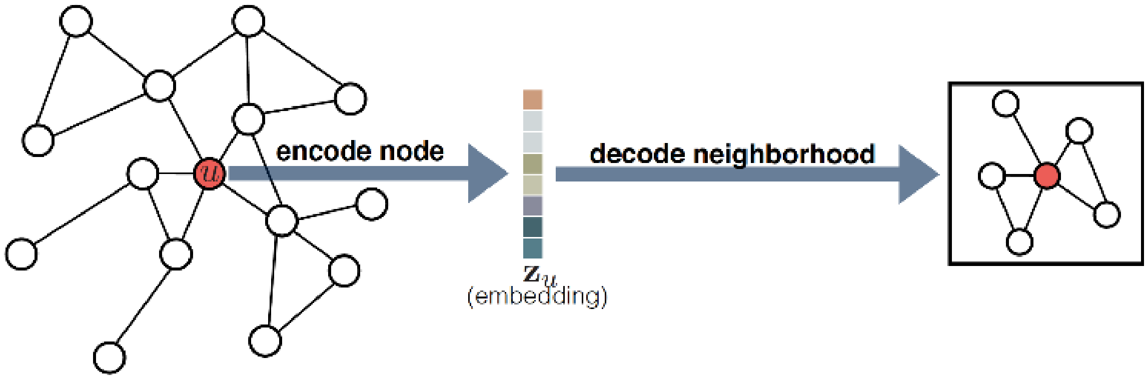
Minimizando perdidas



$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)$$

Method	Decoder	Similarity measure	Loss function
Lap. Eigenmaps	$\lVert\mathbf{z}_u - \mathbf{z}_v\rVert_2^2$	general	$\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) \cdot \mathbf{S}[u,v]$
Graph Fact.	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v]$	$\lVert\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u,v]\rVert_2^2$
GraRep	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v], ..., \mathbf{A}^k[u,v]$	$\lVert\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u,v]\rVert_2^2$
HOPE	$\mathbf{z}_u^\top \mathbf{z}_v$	general	$\lVert\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v) - \mathbf{S}[u,v]\rVert_2^2$
DeepWalk	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_\mathcal{G}(v\|u)$	$-\mathbf{S}[u,v]\log(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v))$
node2vec	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_\mathcal{G}(v\|u)$ (biased)	$-\mathbf{S}[u,v]\log(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v))$

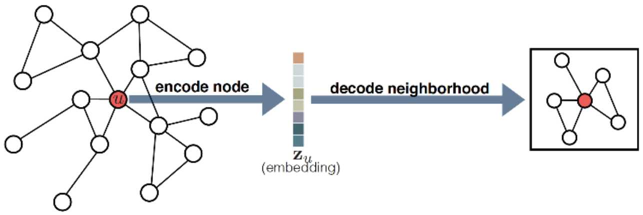
Minimizando perdidas



$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)$$

Method	Decoder	Similarity measure
Lap. Eigenmaps	$\|\mathbf{z}_u - \mathbf{z}_v\|_2^2$	general
Graph Fact.	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v]$
GraRep	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v], ..., \mathbf{A}^k[u,v]$
HOPE	$\mathbf{z}_u^\top \mathbf{z}_v$	general
DeepWalk	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$
node2vec	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$ (biased)

Diferentes medidas S[u,v] van a producir embeddings que reflejen diferentes propiedades del grafo…Ptos cercanos asociados a: nodos conectados? primeros vecinos comunes? roles estructurales similares? …?

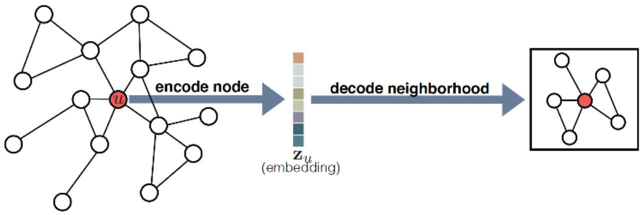


Similaridad de Adyacencia

$$\mathcal{L} = \sum_{(u,v)\epsilon V\times V} \left\| {z_u}^T . z_v - A_{u,v} \right\|^2$$

Idea: la similaridad tipo producto interno debe aproximar la relacion de existencia de enlace

Para encontrar $Z \epsilon \mathbb{R}^{|V|\times d}$ se utiliza Stochastic Gradient Descent



Similaridad multi-salto

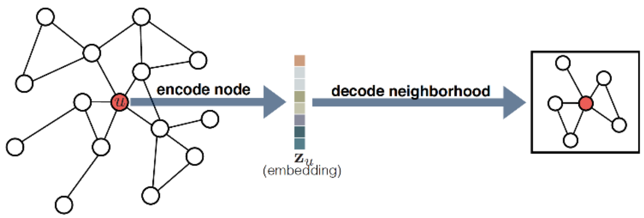
$$\mathcal{L}^{(k)} = \sum_{(u,v)\epsilon V\times V} \left\| {z_u^{(k)}}^T . z_v^{(k)} - [A^k]_{u,v} \right\|^2$$

Idea: la similaridad tipo producto interno debe predecir k-hop vecinos

Se encuentran mappings para diferentes *k* y luego se concatenan

$$z_u = z_u^{k=1} \oplus z_u^{k=2} \oplus \cdots \oplus z_u^{k=K}$$

Cao et al. 2015. GraRep: Learning Graph Representations with Global Structural Information. CIKM



Similaridad Random-Walk

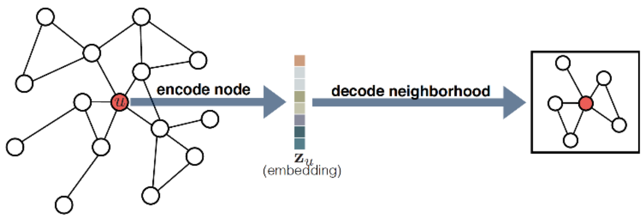
$$z_u^T . z_v \sim \quad P_R(v|u)$$

probabilidad de que *u* y *v* co-ocurran en una caminata aleatoria de tipo *R* sobre el grafo

Idea:

1. Estimar prob de encontrar a *v* en una random-walk que empieza en *u* siguiendo una estrategia *R*
2. Optimizar el embedding para que refleje la estadistica relevada en la caminata

Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations.KDD.
Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. KDD.



Similaridad Random-Walk

Se parametriza a $P_R(v|z_u)$ como

$$\mathcal{L} = \sum_{u \epsilon V} \sum_{v \epsilon N_R(u)} -\log P_R(v|z_u)$$

$$P_R(v|z_u) = \frac{e^{(z_u{}^T . z_v)}}{\sum_{n \epsilon V} e^{(z_u{}^T . z_n)}}$$

Idea:

1. Para cada nodo *u* corro random-walks cortas
2. En cada caso almaceno el *multiset* $N_R(u)$ de nodos visitados siguiendo la estrategia *R*
3. Optimizar embeddings de manera de minimizar $\mathcal{L}$ (o sea maximizar la probabilidad de encontrar cerca de z_u los ENC($N_R(u)$))

Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations.KDD.
Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. KDD.

Problema técnico

$$\mathcal{L} = \sum_{u \epsilon V} \sum_{v \epsilon N_R(u)} -\log \frac{e^{(z_u{}^T . z_v)}}{\sum_{n \epsilon V} e^{(z_u{}^T . z_n)}}$$

Suma anidada sobre nodos o(N^2)!

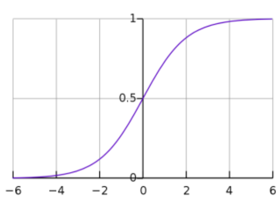
Solución: **Negative sampling**

$$\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n\in V}\exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right) \approx \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{k} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i}))$$

$n_i \sim$ sampling de vertices no incluidos en $N_R(u)$

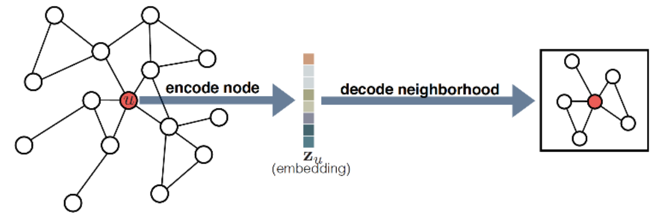
Función sigmoidea

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Comparo similaridad (z_u, z_v) con una estimación (k muestras) de similaridad de z_u respecto vectores asociados a nodos **fuera de la caminata** (de ahí el nombre).
- k altos dan estimaciones mas robustas.
- En la práctica se usa muestreo uniforme (!)

Entonces…



$$\mathcal{L} = -\sum_{u \epsilon V} \sum_{v \epsilon \boxed{N_R(u)}} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{k} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i}))$$

Idea:

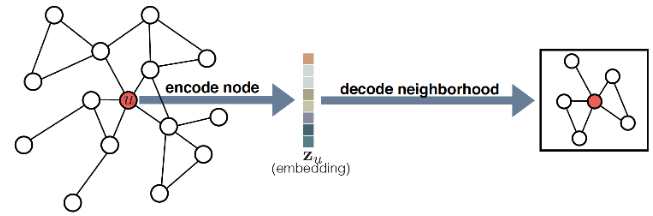
1. Para cada nodo *u* corro random-walks cortas
2. En cada caso almaceno el *multiset* $N_R(u)$ de nodos visitados siguiendo la estrategia *R*
3. Optimizar embeddings de manera de minimizar $\mathcal{L}$ (o sea maximizar la probabilidad de encontrar cerca de z_u los ENC($N_R(u)$))

Posibles estrategias:

- Uniform random walk
- Biased random-walk

Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations.KDD.
Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. KDD.

Entonces…



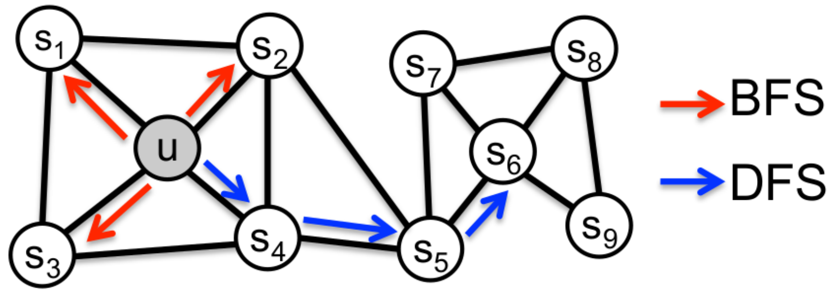
$$\mathcal{L} = -\sum_{u \in V} \sum_{v \in N_R(u)} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{k} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i}))$$

Method	Decoder	Similarity measure
Lap. Eigenmaps	$\|\mathbf{z}_u - \mathbf{z}_v\|_2^2$	general
Graph Fact.	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v]$
GraRep	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v], ..., \mathbf{A}^k[u,v]$
HOPE	$\mathbf{z}_u^\top \mathbf{z}_v$	general
DeepWalk	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$
node2vec	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k \in \mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$ (biased)

Perozzi et al. 2014. DeepWalk: Online Learning of Social Representations.KDD.
Grover et al. 2016. node2vec: Scalable Feature Learning for Networks. KDD.

node2vec: biased random walk

Por que?

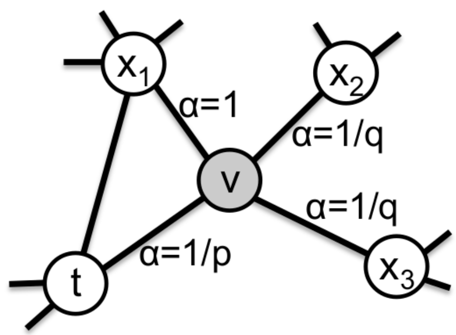


Caminatas aleatorias de diferente tipo resuenan con diferentes regularidades de la red

$$N_{BFS}(u) = \{s_1, s_2, s_3\}$$

$$N_{DFS}(u) = \{s_4, s_5, s_6\}$$

Como?



Se implementa como una caminata aleatoria de **2do orden** (i.e. la prob de transición depende de donde estoy ahora y donde estuvo en el paso anterior)

prob no normalizada $\quad \pi_{vx} = \alpha_{pq}(t,x).\, w_{vx}$

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

p: controla la prob de revisita
q: controla tendencia a DFS

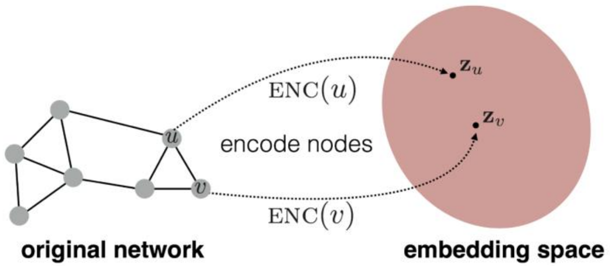
Por ejemplo:
Si p>max(q,1) baja prob de revisitar un nodo de los ultimos dos pasos
Si q>1 tendencia al BFS

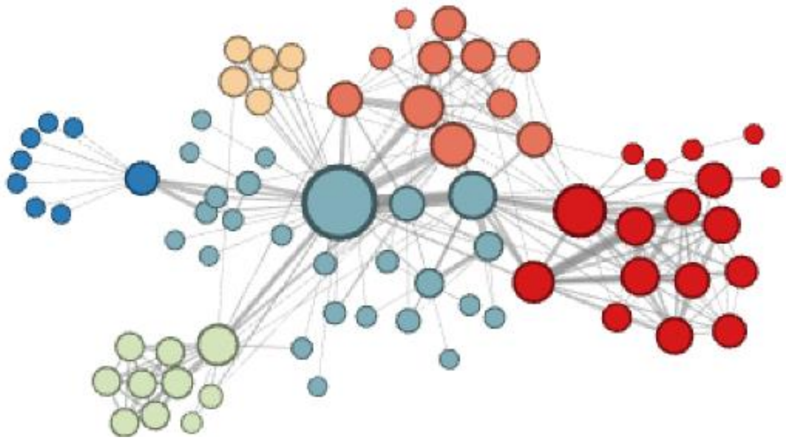
Grover & Leskovec, node2vec: Scalabale Feature Learning for Networks, KDD 2016

node2vec en acción

Red de co-aparición de Les Miserables

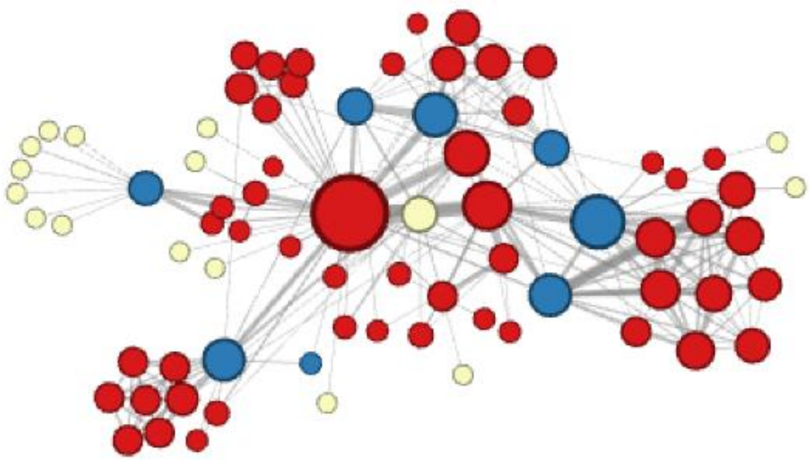
Comunidades detectadas en embedding 16-dim:





p=1 q=0.5

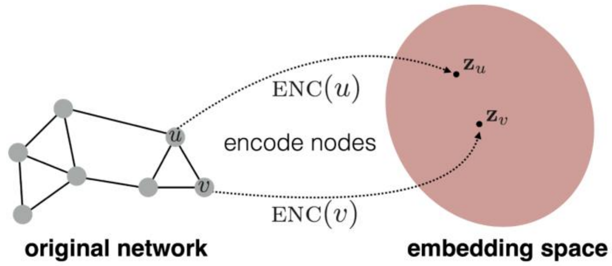
Las comunidades detectadas en el embedding reflejan similaridad relacionada con homofilia.



p=1 q=2

Las comunidades detectadas en el embedding reflejan similaridad de tipo estructural.

O sea



- Estuvimos analizando como embeber vertices de una red en un espacio d-dimensional de manera de reflejar similaridades estructurales de la red
- Existen diferentes maneras de hacerlo

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)$$

Method	Decoder	Similarity measure
Lap. Eigenmaps	$\|\mathbf{z}_u - \mathbf{z}_v\|_2^2$	general
Graph Fact.	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v]$
GraRep	$\mathbf{z}_u^\top \mathbf{z}_v$	$\mathbf{A}[u,v], ..., \mathbf{A}^k[u,v]$
HOPE	$\mathbf{z}_u^\top \mathbf{z}_v$	general
DeepWalk	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$
node2vec	$\frac{e^{\mathbf{z}_u^\top \mathbf{z}_v}}{\sum_{k\in\mathcal{V}} e^{\mathbf{z}_u^\top \mathbf{z}_k}}$	$p_{\mathcal{G}}(v\|u)$ (biased)

Hasta aca