FACULTAD DE
INGENIERÍA

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Aprendizaje Automático para Datos en Grafos

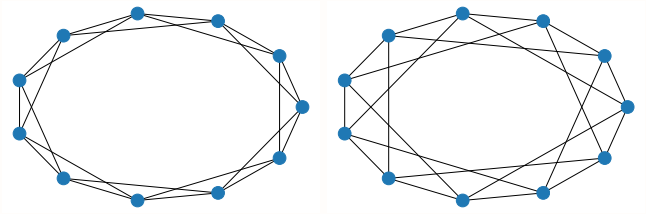
Graph Neural Networks - Cambiando el grafo para aprender en el grafo

Federico 'Larroca' La Rocca

flarroca@fing.edu.uy
http://iie.fing.edu.uy/personal/flarroca

¿Qué puede expresar una GNN?

- Ejemplo
  - Supongamos que queremos clasificar o hacer regresión sobre moléculas
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas en cada grafo?



¿Qué puede expresar una GNN?

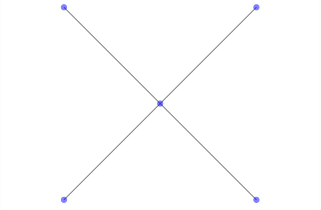
- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro

¿Qué puede expresar una GNN?

- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro

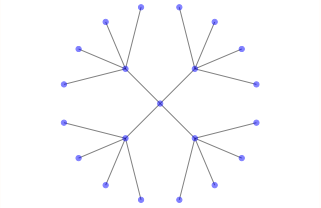
¿Qué puede expresar una GNN?

- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro



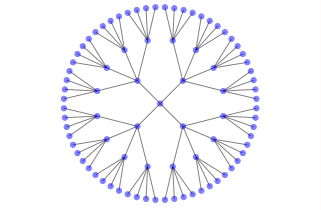
¿Qué puede expresar una GNN?

- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro



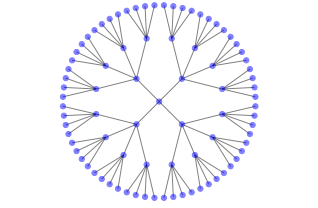
¿Qué puede expresar una GNN?

- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro

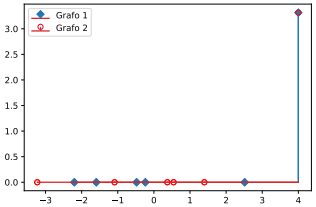


¿Qué puede expresar una GNN?

- Dos formas de ver esta imposibilidad:
  - Grafo computacional: ¿qué señales ve cada nodo a medida que apilo capas? u órdenes del filtro

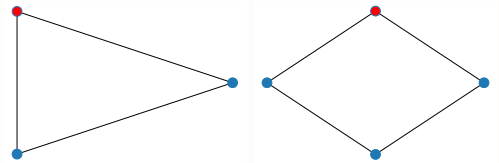


  - ¿Qué espectro tiene esta señal constante sobre ambos grafos?



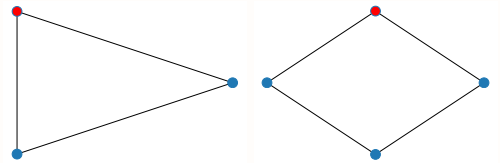
¿Qué puede expresar una GNN?

- Otro ejemplo
  - Quiero clasificar nodos en grafos
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas para los nodos rojos?



¿Qué puede expresar una GNN?

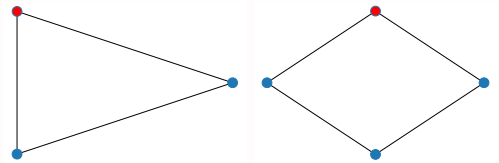
- Otro ejemplo
  - Quiero clasificar nodos en grafos
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas para los nodos rojos?



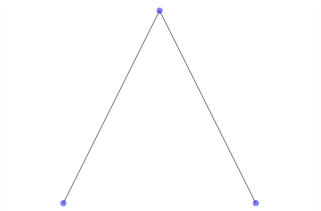
- Usando el grafo computacional u órdenes del filtro

¿Qué puede expresar una GNN?

- Otro ejemplo
  - Quiero clasificar nodos en grafos
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas para los nodos rojos?

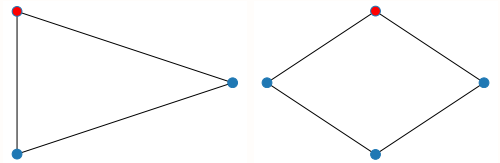


- Usando el grafo computacional u órdenes del filtro

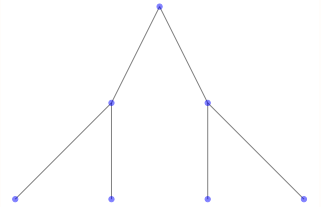


¿Qué puede expresar una GNN?

- Otro ejemplo
  - Quiero clasificar nodos en grafos
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas para los nodos rojos?

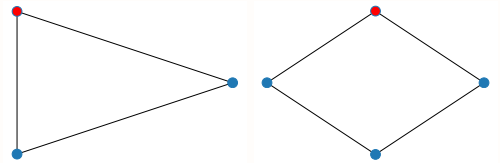


- Usando el grafo computacional u órdenes del filtro

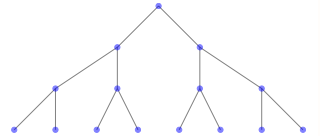


¿Qué puede expresar una GNN?

- Otro ejemplo
  - Quiero clasificar nodos en grafos
  - Mismos features (atómos) en todos los nodos
  - ¿Puede una GNN producir distintas salidas para los nodos rojos?

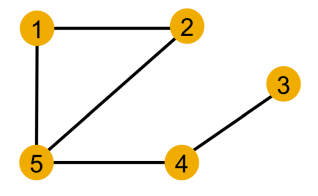


- Usando el grafo computacional u órdenes del filtro



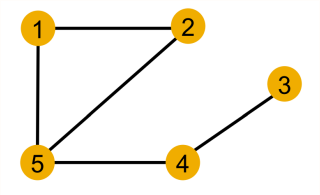
¿Qué puede expresar una GNN?

- Último ejemplo:

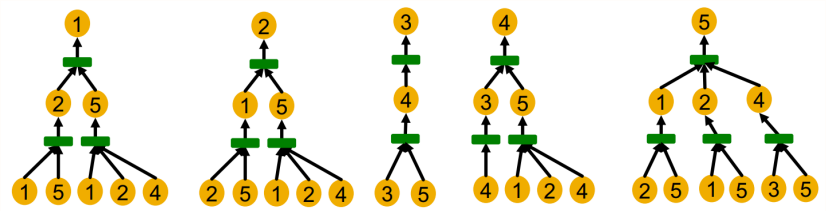


¿Qué puede expresar una GNN?

- Último ejemplo:



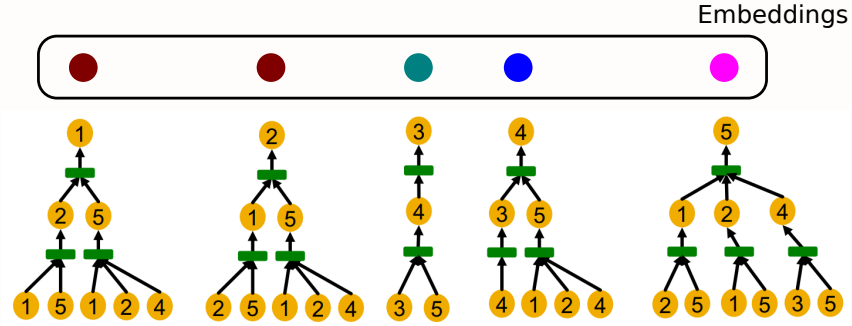
- ¿Cuántos embeddings distintos debería generar?



Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

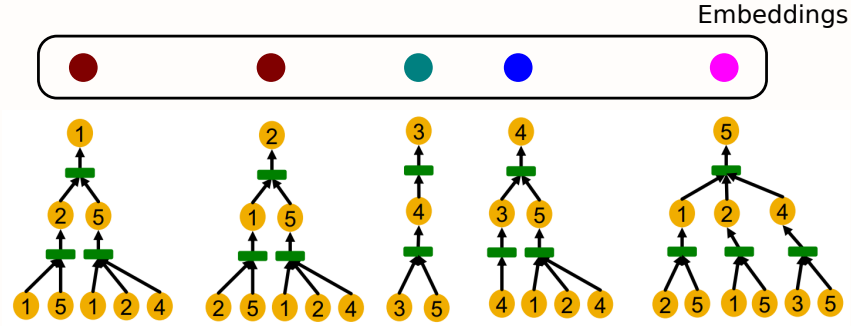
- Idealmente cada grafo computacional debería generar distintos embeddings.



Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

- Idealmente cada grafo computacional debería generar distintos embeddings.



- Queremos funciones inyectivas de los grafos computacionales a los embeddings finales

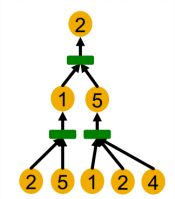
Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

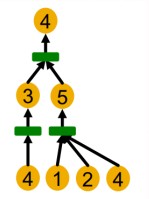
¿La GNN más expresiva?

- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba

¿La GNN más expresiva?

- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba

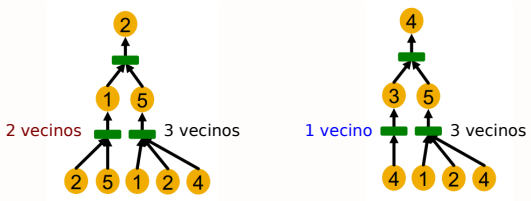




Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

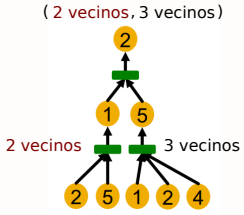
- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba

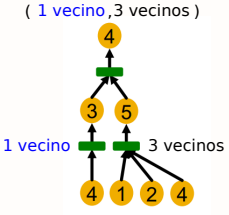


Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba

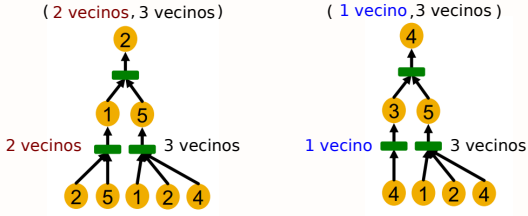




Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba

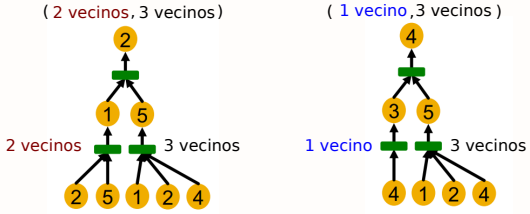


- Si la agregación mantiene la información de vecindad $\Rightarrow$ embeddings distintos

Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

¿La GNN más expresiva?

- Funciones inyectivas de los grafos computacionales a los embeddings finales: ¿cómo?
  - El árbol lo puedo caracterizar de abajo a arriba



- Si la agregación mantiene la información de vecindad ⇒ embeddings distintos
- La GNN más expresiva usa función de agregación inyectiva en la vecindad
  - Ejemplos. ¿Son inyectivos?
    - GCN. Función de agregación: media coordenada-a-coordenada y ReLU.
    - GraphSAGE. Función de agregación: MLP y máx coordenada-a-coordenada.

Ejemplo tomado del curso cs224w, J. Leskovec, Stanford University

Kipf & Welling, "Semi-Supervised Classification with Graph Convolutional Networks", ICLR 2017

Hamilton, Ying, & Leskovec, "Inductive representation learning on large graphs. NeurIPS 2017.

¿La GNN más expresiva?

- ¿Función de agregación inyectiva? Esto se parece mucho al clásico algoritmo de Weisfeiler-Lehman

1-dimensional WL (1-WL) algorithm (a.k.a. color refinement)
Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

1. $c_v^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{X}_v)$ $(\forall v \in V)$
2. $d_u^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{Y}_u)$ $(\forall u \in U)$
3. for $l = 1, 2, \ldots$ (until convergence)
   (a) if $\{\!\{c_v^{(l-1)} \mid v \in V\}\!\} \neq \{\!\{d_u^{(l-1)} \mid u \in U\}\!\}$ then return "non-isomorphic"
   (b) $c_v^{(l)} \leftarrow \textsc{Hash}(c_v^{(l-1)}, \{\!\{c_w^{(l-1)} \mid w \in \mathcal{N}_G(v)\}\!\})$ $(\forall v \in V)$
   (c) $d_u^{(l)} \leftarrow \textsc{Hash}(d_u^{(l-1)}, \{\!\{d_w^{(l-1)} \mid w \in \mathcal{N}_H(u)\}\!\})$ $(\forall u \in U)$
4. return "possibly isomorphic"

Weisfeiler y Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". Nauchno-Technicheskaya Informatsia, 2(9):12–16, 1968

¿La GNN más expresiva?

- ¿Función de agregación inyectiva? Esto se parece mucho al clásico algoritmo de Weisfeiler-Lehman

1-dimensional WL (1-WL) algorithm (a.k.a. color refinement)
Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

1. $c_v^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{X}_v)$ $(\forall v \in V)$
2. $d_u^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{Y}_u)$ $(\forall u \in U)$
3. for $l = 1, 2, \ldots$ (until convergence)
   (a) if $\{\!\{c_v^{(l-1)} \mid v \in V\}\!\} \neq \{\!\{d_u^{(l-1)} \mid u \in U\}\!\}$ then return "non-isomorphic"
   (b) $c_v^{(l)} \leftarrow \textsc{Hash}(c_v^{(l-1)}, \{\!\{c_w^{(l-1)} \mid w \in \mathcal{N}_G(v)\}\!\})$ $(\forall v \in V)$
   (c) $d_u^{(l)} \leftarrow \textsc{Hash}(d_u^{(l-1)}, \{\!\{d_w^{(l-1)} \mid w \in \mathcal{N}_H(u)\}\!\})$ $(\forall u \in U)$
4. return "possibly isomorphic"

- Arquitecturas que tienen el mismo desempeño:
  - GIN: $\mathbf{x}_i^{(k+1)} = \text{MLP}((1+\epsilon)\mathbf{x}_i^{(k)} + \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^{(k)})$

Weisfeiler y Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". Nauchno-Technicheskaya Informatsia, 2(9):12–16, 1968

Xu, Hu, Leskovec, y Jegelka. "How Powerful are Graph Neural Networks?." ICLR 2018.

¿La GNN más expresiva?

- ¿Función de agregación inyectiva? Esto se parece mucho al clásico algoritmo de Weisfeiler-Lehman

1-dimensional WL (1-WL) algorithm (a.k.a. color refinement)
Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

1. $c_v^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{X}_v)$ $(\forall v \in V)$
2. $d_u^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{Y}_u)$ $(\forall u \in U)$
3. for $l = 1, 2, \ldots$ (until convergence)
   (a) if $\{\!\{c_v^{(l-1)} \mid v \in V\}\!\} \neq \{\!\{d_u^{(l-1)} \mid u \in U\}\!\}$ then return "non-isomorphic"
   (b) $c_v^{(l)} \leftarrow \textsc{Hash}(c_v^{(l-1)}, \{\!\{c_w^{(l-1)} \mid w \in \mathcal{N}_G(v)\}\!\})$ $(\forall v \in V)$
   (c) $d_u^{(l)} \leftarrow \textsc{Hash}(d_u^{(l-1)}, \{\!\{d_w^{(l-1)} \mid w \in \mathcal{N}_H(u)\}\!\})$ $(\forall u \in U)$
4. return "possibly isomorphic"

- Arquitecturas que tienen el mismo desempeño:
  - GIN: $\mathbf{x}_i^{(k+1)} = \text{MLP}((1+\epsilon)\mathbf{x}_i^{(k)} + \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^{(k)})$
  - Pero también una GNN "vanilla" (al menos existe alguna parametrización que así lo garantiza)

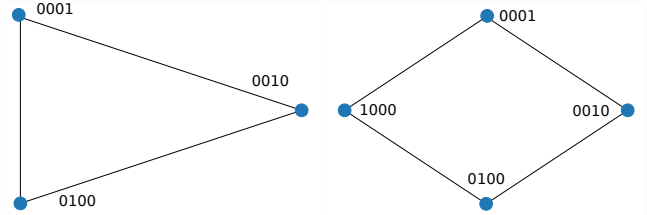
Weisfeiler y Lehman. "A reduction of a graph to a canonical form and an algebra arising during this reduction". Nauchno-Technicheskaya Informatsia, 2(9):12–16, 1968
Xu, Hu, Leskovec, y Jegelka. "How Powerful are Graph Neural Networks?." ICLR 2018.
Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, y Grohe. "Weisfeiler and leman go neural: Higher-order graph neural networks." AAAI 2019.

Más allá del test de WL

- ¿Cómo podemos aumentar la expresividad de la GNN?
  - No usando una señal constante como entrada!

Más allá del test de WL

- ¿Cómo podemos aumentar la expresividad de la GNN?
  - No usando una señal constante como entrada!
- Ejemplo: Identifico cada nodo usando one-hot encoding

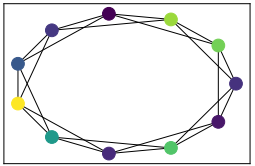


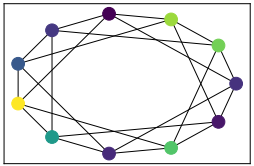
  - El grafo computacional es distinto para cada nodo
  - Pero no escala ni es inductivo

Vignac, Loukas, y Frossard. "Building powerful and equivariant graph neural networks with structural message-passing." NeurIPS 2020.

Más allá del test de WL

- Otro ejemplo: Diferencio cada nodo usando ruido

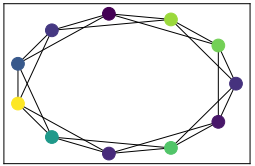


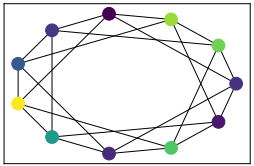


Abboud, Ceylan, Grohe, y Lukasiewicz. "The surprising power of graph neural networks with random node initialization." IJCAI 2021.
Sato, Yamada, y Hisashi Kashima. "Random features strengthen graph neural networks." SDM 2021.

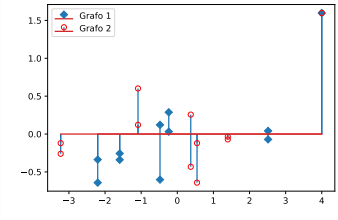
Más allá del test de WL

- Otro ejemplo: Diferencio cada nodo usando ruido





- Fácil de ver en el espectro: aparecen las frecuencias que la señal constante ignoraba



Abboud, Ceylan, Grohe, y Lukasiewicz. "The surprising power of graph neural networks with random node initialization." IJCAI 2021.

Sato, Yamada, y Hisashi Kashima. "Random features strengthen graph neural networks." SDM 2021.

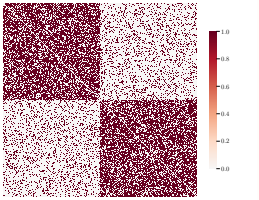
Expresividad en grandes grafos

- Para grafos grandes típicamente el problema es clasificación/regresión a nivel de nodos
- La probabilidad de tener grafos de cómputo iguales va a cero con n
  - El marco del test de WL es útil para grafos pequeños

Expresividad en grandes grafos

- Para grafos grandes típicamente el problema es clasificación/regresión a nivel de nodos
- La probabilidad de tener grafos de cómputo iguales va a cero con n
  - El marco del test de WL es útil para grafos pequeños
- Supongamos que queremos detectar comunidades usando una GNN en el siguiente SBM:

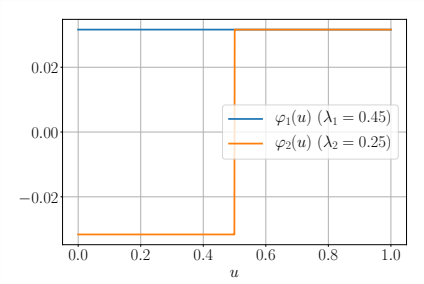
$$\boldsymbol{\alpha} = [0{,}5, 0{,}5] \qquad \boldsymbol{\Pi} = \begin{pmatrix} 0{,}7 & 0{,}2 \\ 0{,}2 & 0{,}7 \end{pmatrix}$$



  - Pregunta: ¿Una GNN puede generar distintas señales para los nodos de las distintas comunidades?

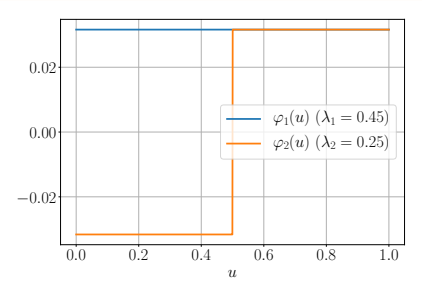
Expresividad en grandes grafos

- Respuesta en el espectro del grafón subyacente



Expresividad en grandes grafos

- Respuesta en el espectro del grafón subyacente



$\Rightarrow$ Entrada $\mathbf{x}$, la salida de cualquier filtro en este grafo será una combinación lineal de:

- $\sum_{i=0}^{n} \mathbf{x}_i$
- $\sum_{i=0}^{n/2-1} \mathbf{x}_i - \sum_{i=n/2}^{n-1} \mathbf{x}_i$

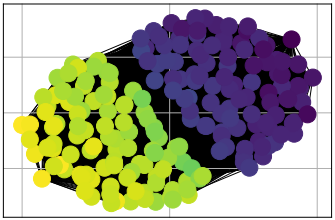
- ¿Qué señales $\mathbf{x}$ generan distintas salidas para los nodos de cada comunidad?

Positional Encodings

- Una GNN no puede generar señales distintas para nodos de distinta comunidad:
  - Positional Encoding $\Rightarrow$ Se lo damos como feature de entrada
  - Da información sobre "dónde" está el nodo en el grafo

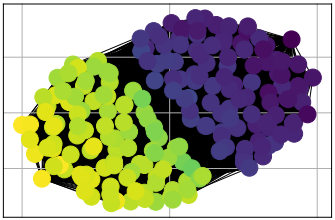
Positional Encodings

- Una GNN no puede generar señales distintas para nodos de distinta comunidad:
  - Positional Encoding ⇒ Se lo damos como feature de entrada
  - Da información sobre "dónde" está el nodo en el grafo
- Posibilidad más intuitiva: valores/vectores propios más significativos del Laplaciano
  - Ejemplo: vector propio del 2º valor propio más pequeño del Laplaciano normalizado



Positional Encodings

- Una GNN no puede generar señales distintas para nodos de distinta comunidad:
  - Positional Encoding ⇒ Se lo damos como feature de entrada
  - Da información sobre "dónde" está el nodo en el grafo
- Posibilidad más intuitiva: valores/vectores propios más significativos del Laplaciano
  - Ejemplo: vector propio del 2º valor propio más pequeño del Laplaciano normalizado



  - Problemas:
    - Valores propios repetidos: rotación
    - ¿Signo de los vectores propios?
  - ¿Cómo aprender bajo esta incertidumbre?

Lim, Robinson, Zhao, Smidt, Sra, Maron, y Jegelka. "Sign and Basis Invariant Networks for Spectral Graph Representation Learning." ICLR 2023.

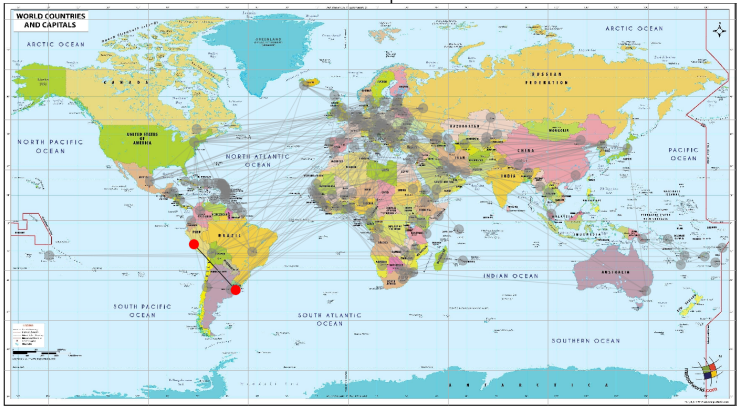
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing

¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)

1994 - Capa 1



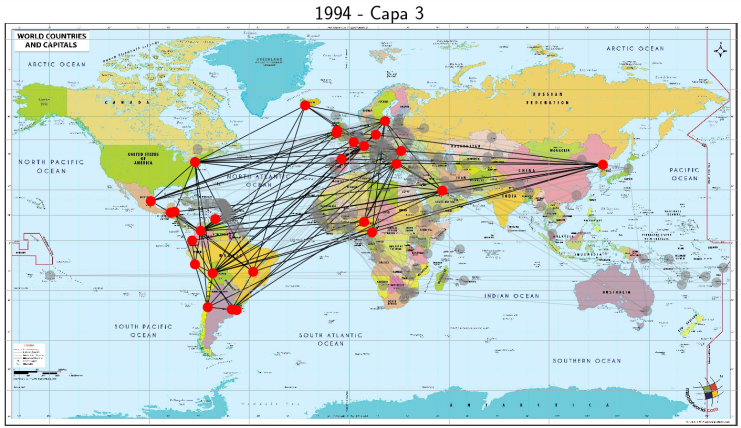
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)

1994 - Capa 2

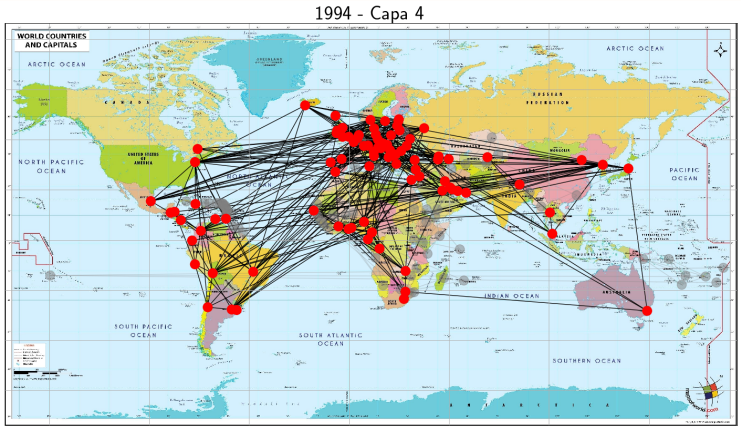
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



¿Modificar la GSO?

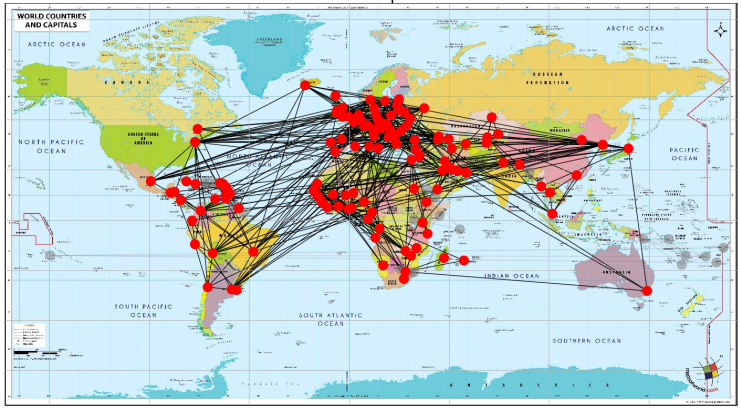
- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



¿Modificar la GSO?

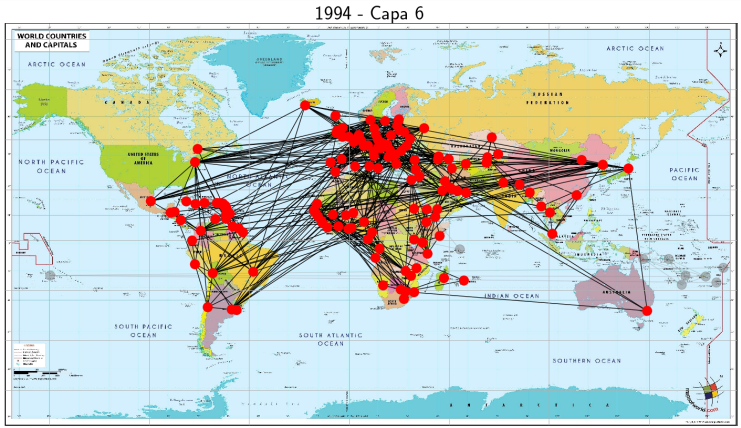
- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)

1994 - Capa 5



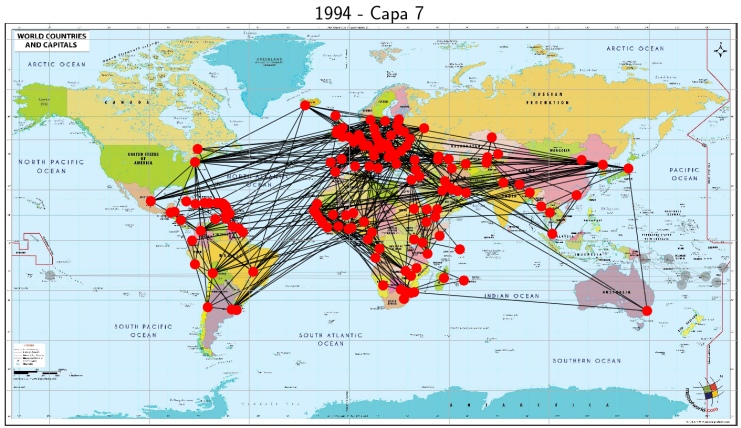
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



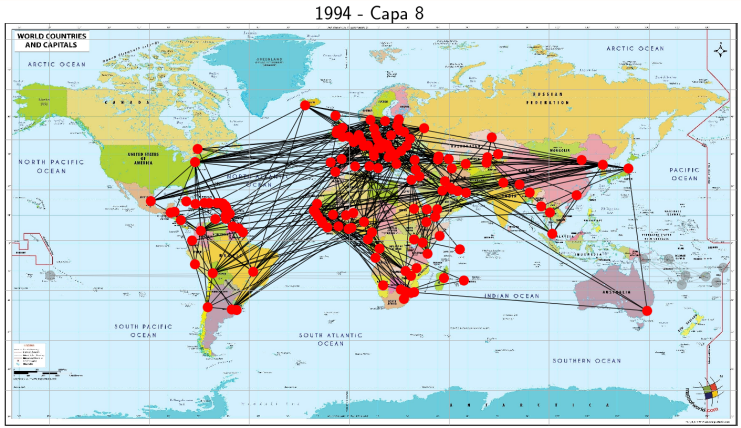
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



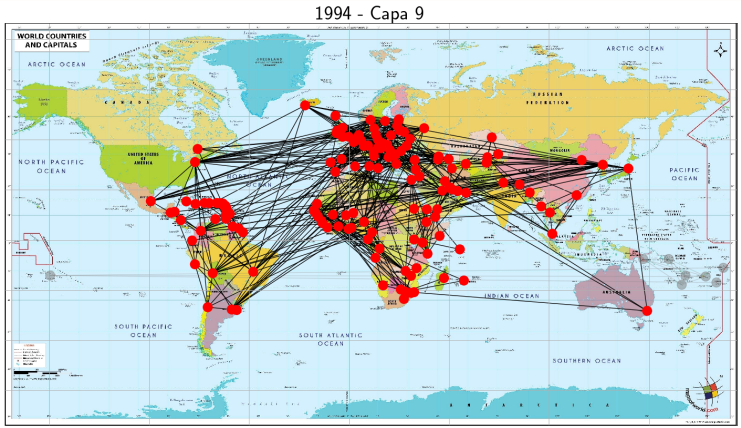
¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



¿Modificar la GSO?

- El problema del grafo anterior estaba en los vectores propios del GSO $\mathbf{S}$
- ¿Porqué no cambiar $\mathbf{S}$?
- Ahora vamos a ver dos problemas que puede producir el GSO usando GNNs:
  - Over-smoothing
  - Over-squashing
- ¿Cuántas capas son muchas capas? Campo receptivo de Uruguay a medida que las apilamos (diám= 8)



¿Modificar la GSO?

- La representación de todos los países en la componente conexa van a tender a lo mismo a medida que apilamos capas: over-smoothing

¿Modificar la GSO?

- La representación de todos los países en la componente conexa van a tender a lo mismo a medida que apilamos capas: over-smoothing
- Ejemplo ilustrativo: $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, filtro de orden 1 y no usamos no-linealidad: $\mathbf{X}^{(L)} = \mathbf{S}^{L}\mathbf{X}$
  - Si el grafo es conexo: $\mathbf{S}$ tiene un valor propio 1 y el el segundo mayor es $\lambda_2 < 1$.
  - A la larga, termina convirgiendo al sub-espacio generado por el vector propio asociado a λ_1

¿Modificar la GSO?

- La representación de todos los países en la componente conexa van a tender a lo mismo a medida que apilamos capas: over-smoothing
- Ejemplo ilustrativo: $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, filtro de orden 1 y no usamos no-linealidad: $\mathbf{X}^{(L)} = \mathbf{S}^{L}\mathbf{X}$
  - Si el grafo es conexo: $\mathbf{S}$ tiene un valor propio 1 y el el segundo mayor es $\lambda_2 < 1$.
  - A la larga, termina convirgiendo al sub-espacio generado por el vector propio asociado a λ_1
  - Qué tan rápido pierde expresividad depende de λ_2, pero su influencia estructural en el grafo no clara
  - Por otro lado, usando $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ se puede probar que la convergencia depende del segundo valor propio más pequeño
    $\Rightarrow$ La pérdida de expresividad es menor si el grafo es más esparso

¿Modificar la GSO?

- La representación de todos los países en la componente conexa van a tender a lo mismo a medida que apilamos capas: over-smoothing
- Ejemplo ilustrativo: $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, filtro de orden 1 y no usamos no-linealidad: $\mathbf{X}^{(L)} = \mathbf{S}^{L}\mathbf{X}$
  - Si el grafo es conexo: $\mathbf{S}$ tiene un valor propio 1 y el el segundo mayor es $\lambda_2 < 1$.
  - A la larga, termina convirgiendo al sub-espacio generado por el vector propio asociado a λ_1
  - Qué tan rápido pierde expresividad depende de λ_2, pero su influencia estructural en el grafo no clara
  - Por otro lado, usando $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ se puede probar que la convergencia depende del segundo valor propio más pequeño
    $\Rightarrow$ La pérdida de expresividad es menor si el grafo es más esparso
  - Justifica el uso de Dropout a nivel de aristas (uno de la familia)

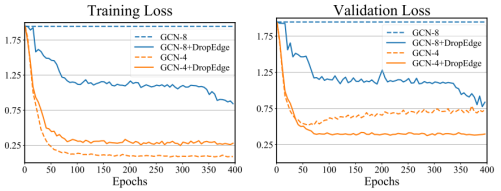


Figure 1: Performance of Multi-layer GCNs on Cora. We implement 4-layer GCN w and w/o DropEdge (in orange), 8-layer GCN w and w/o DropEdge (in blue)[2]. GCN-4 gets stuck in the over-fitting issue attaining low training error but high validation error; the training of GCN-8 fails to converge satisfactorily due to over-smoothing. By applying DropEdge, both GCN-4 and GCN-8 work well for both training and validation.

Giraldo, Fragkiskos, y Bouwmans. "Understanding the Relationship between Over-smoothing and Over-squashing in Graph Neural Networks." CIKM 2023.

Rong, Huang, Xu, y Huang. "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification." ICLR 2019.

¿Modificar la GSO?

- Hipótesis hasta ahora: homofilia
  - La vecindad en el grafo indica influencia
- ¿Y si lo nodos importantes no están cerca?
  - Long-range dependence
  - Ejemplo de juguete

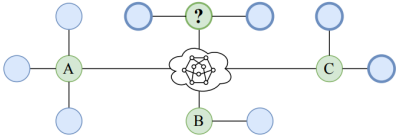


Figure 2: The NEIGHBORSMATCH: green nodes (A, B, C) have blue neighbors ( ) and an alphabetical label. The goal is to predict the label (A, B, or C) of the green node that has the same number of blue neighbors as the target node (?) in the same graph. In this example, the correct label is **C**, because the target node has *two* blue neighbors, like the node marked with C in the same graph.

¿Modificar la GSO?

- Hipótesis hasta ahora: homofilia
  - La vecindad en el grafo indica influencia
- ¿Y si lo nodos importantes no están cerca?
  - Long-range dependence
  - Ejemplo de juguete

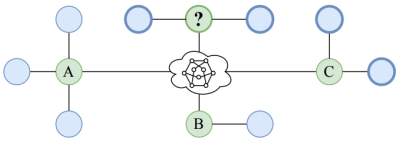


Figure 2: The NEIGHBORSMATCH: green nodes (A, B, C) have blue neighbors ( ) and an alphabetical label. The goal is to predict the label (A, B, or C) of the green node that has the same number of blue neighbors as the target node (?) in the same graph. In this example, the correct label is **C**, because the target node has *two* blue neighbors, like the node marked with C in the same graph.

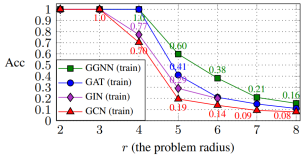


Figure 3: Accuracy across *problem radius* (tree depth) in the NEIGHBORSMATCH problem. Over-squashing starts to affect GCN and GIN even at $r = 4$.

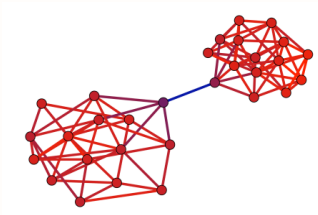
- Ejemplos de verdad en la *Long Range Graph Benchmark*

- ¿Porqué falla la GNN?

Alon, y Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications." ICLR 2021.

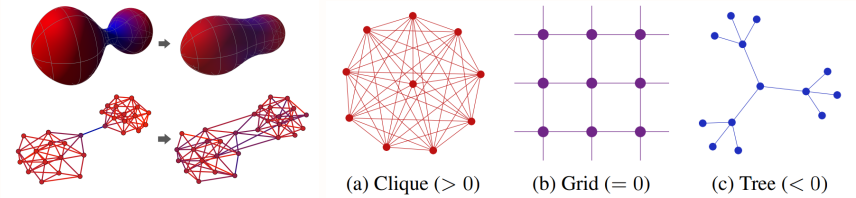
¿Modificar la GSO?

- Intuitivamente: falla si algún nodo actúa de cuello de botella
- Por el enlace azul pasa una cantidad exponencial (en las capas) de representaciones
  - Posible solución: GNN con pocas capas y al final cambiar la GSO por un grafo Fully-connected



¿Modificar la GSO?

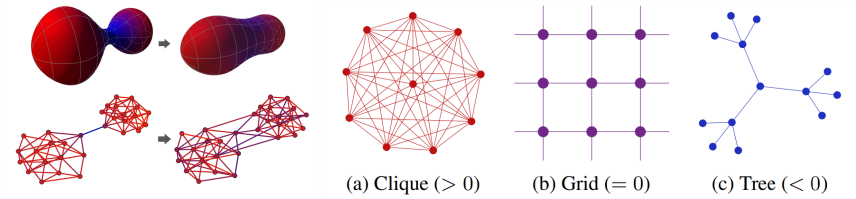
- Intuitivamente: falla si algún nodo actúa de cuello de botella
- Por el enlace azul pasa una cantidad exponencial (en las capas) de representaciones
  - Posible solución: GNN con pocas capas y al final cambiar la GSO por un grafo Fully-connected
  - Otra solución: agregar enlaces que eliminen los cuellos de botella



Topping, Di Giovanni, Chamberlain, Dong, y Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature." ICLR 2022.

¿Modificar la GSO?

- Intuitivamente: falla si algún nodo actúa de cuello de botella
- Por el enlace azul pasa una cantidad exponencial (en las capas) de representaciones
  - Posible solución: GNN con pocas capas y al final cambiar la GSO por un grafo Fully-connected
  - Otra solución: agregar enlaces que eliminen los cuellos de botella



- Compromiso:
  - Agregar enlaces para evitar oversquashing o
  - Sacar enlaces para evitar el oversmoothing

Topping, Di Giovanni, Chamberlain, Dong, y Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature." ICLR 2022.

Karhadkar, Banerjee, y Montufar. "FoSR: First-order spectral rewiring for addressing oversquashing in GNNs." ICLR 2023.

Giraldo, Fragkiskos, y Bouwmans. "Understanding the Relationship between Over-smoothing and Over-squashing in Graph Neural Networks." CIKM 2023.

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más “informativos” que otros

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más "informativos" que otros
- Hasta ahora esa información la tomamos de la GSO y confiamos en las propiedades de estabilidad

$$\mathbf{x}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} S_{j,i} \mathbf{H} \mathbf{x}_j\right)$$

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más "informativos" que otros
- Hasta ahora esa información la tomamos de la GSO y confiamos en las propiedades de estabilidad

$$\mathbf{x}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{j,i} \mathbf{H} \mathbf{x}_j\right)$$

  - ¿Podemos aprender esos pesos?

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más "informativos" que otros
- Hasta ahora esa información la tomamos de la GSO y confiamos en las propiedades de estabilidad

$$\mathbf{x}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{j,i} \mathbf{H} \mathbf{x}_j\right)$$

  - ¿Podemos aprender esos pesos?
- Attention: El peso $\alpha_{i,j}$ depende del vector (embedding) de los nodos i y j
  - Scoring function $e : \mathbb{R}^G \times \mathbb{R}^G \to \mathbb{R}$ da la importancia de j sobre i ($\mathbf{a} \in \mathbb{R}^{2G}$: nuevo parámetro a aprender):

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^{\mathsf{T}} . [\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right)$$

Veličković, Cucurull, Casanova, Romero, Pietrò, Bengio, "Graph Attention Networks", ICLR 2018

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más "informativos" que otros
- Hasta ahora esa información la tomamos de la GSO y confiamos en las propiedades de estabilidad

$$\mathbf{x}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{j,i}\mathbf{H}\mathbf{x}_j\right)$$

  - ¿Podemos aprender esos pesos?
- Attention: El peso $\alpha_{i,j}$ depende del vector (embedding) de los nodos i y j
  - Scoring function $e : \mathbb{R}^G \times \mathbb{R}^G \to \mathbb{R}$ da la importancia de j sobre i ($\mathbf{a} \in \mathbb{R}^{2G}$: nuevo parámetro a aprender):

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^{\mathsf{T}}.[\mathbf{H}\mathbf{x}_i||\mathbf{H}\mathbf{x}_j]\right) = \text{LeakyReLU}\left(\mathbf{a_1}^{\mathsf{T}}\mathbf{H}\mathbf{x}_i + \mathbf{a_2}^{\mathsf{T}}\mathbf{H}\mathbf{x}_j\right)$$

Veličković, Cucurull, Casanova, Romero, Pietrò, Bengio, "Graph Attention Networks", ICLR 2018

Graph Attention Networks

- Una limitante de las GNNs es que les brinda la misma importancia a todos los vecinos
  - Seguramente para clasificar nodos haya vecinos más "informativos" que otros
- Hasta ahora esa información la tomamos de la GSO y confiamos en las propiedades de estabilidad

$$\mathbf{x}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{j,i}\mathbf{H}\mathbf{x}_j\right)$$

  - ¿Podemos aprender esos pesos?
- Attention: El peso $\alpha_{i,j}$ depende del vector (embedding) de los nodos i y j
  - Scoring function $e : \mathbb{R}^G \times \mathbb{R}^G \to \mathbb{R}$ da la importancia de j sobre i ($\mathbf{a} \in \mathbb{R}^{2G}$: nuevo parámetro a aprender):

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^{\mathsf{T}}.\,[\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right) = \text{LeakyReLU}\left(\mathbf{a_1}^{\mathsf{T}}\mathbf{H}\mathbf{x}_i + \mathbf{a_2}^{\mathsf{T}}\mathbf{H}\mathbf{x}_j\right)$$

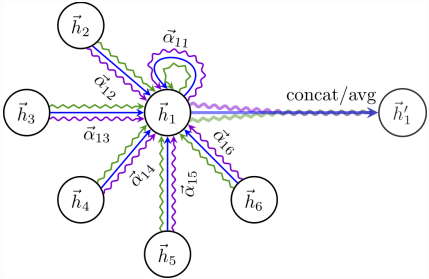
  - La Attention function usa los scores para dar el peso

$$\alpha_{i,j} = \text{softmax}_j(e(\mathbf{x}_i, \mathbf{x}_j)) = \frac{\exp(e(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{j'\in\mathcal{N}_i} \exp(e(\mathbf{x}_i, \mathbf{x}_{j'}))}$$

Veličković, Cucurull, Casanova, Romero, Pietrò, Bengio, "Graph Attention Networks", ICLR 2018

Graph Attention Networks

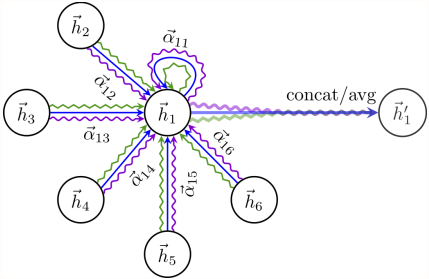
- Se pueden generar varios coeficientes por capa (K valores de $\alpha_{i,j}^k$): Multi-headed attention



Veličković, Cucurull, Casanova, Romero, Pietrìò, Bengio, "Graph Attention Networks", ICLR 2018

Graph Attention Networks

- Se pueden generar varios coeficientes por capa (K valores de $\alpha_{i,j}^k$): Multi-headed attention



- ¿La attention function depende realmente de $\mathbf{x}_i$ y de $\mathbf{x}_j$?

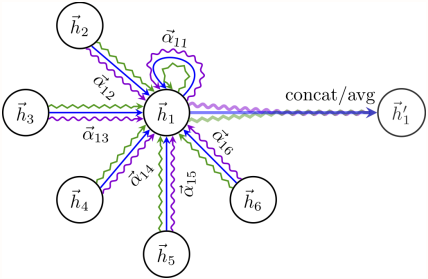
$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a_1}^\top \mathbf{H}\mathbf{x}_i + \mathbf{a_2}^\top \mathbf{H}\mathbf{x}_j\right)$$

$$\alpha_{i,j} = \text{softmax}_j(e(\mathbf{x}_i, \mathbf{x}_j)) = \frac{\exp(e(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{j' \in \mathcal{N}_i} \exp(e(\mathbf{x}_i, \mathbf{x}_{j'}))}$$

Veličković, Cucurull, Casanova, Romero, Pietrìò, Bengio, "Graph Attention Networks", ICLR 2018

Graph Attention Networks

- Se pueden generar varios coeficientes por capa (K valores de $\alpha_{i,j}^k$): Multi-headed attention



- ¿La attention function depende realmente de $\mathbf{x}_i$ y de $\mathbf{x}_j$? ¿Y si algún $\mathbf{a_2}^\top \mathbf{H}\mathbf{x}_j$ es mucho mayor que el resto?

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a_1}^\top \mathbf{H}\mathbf{x}_i + \mathbf{a_2}^\top \mathbf{H}\mathbf{x}_j\right)$$

$$\alpha_{i,j} = \text{softmax}_j(e(\mathbf{x}_i, \mathbf{x}_j)) = \frac{\exp(e(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{j' \in \mathcal{N}_i} \exp(e(\mathbf{x}_i, \mathbf{x}_{j'}))} \approx \mathbf{a_2}^\top \mathbf{H}\mathbf{x}_{j_{\text{máx}}}$$

Veličković, Cucurull, Casanova, Romero, Pietriò, Bengio, "Graph Attention Networks", ICLR 2018

Brody, Alon, Yahav, "How Attentive are Graph Attention Networks?", ICLR 2022

Graph Attention Networks v2

- O sea que el mecanismo de attention "vanilla" identifica los nodos más importantes, pero puede llegar a ignorar el embedding del nodo original

Brody, Alon, Yahav, "How Attentive are Graph Attention Networks?", ICLR 2022

Graph Attention Networks v2

- O sea que el mecanismo de attention "vanilla" identifica los nodos más importantes, pero puede llegar a ignorar el embedding del nodo original
- El problema es la linealidad consecutiva

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^\mathsf{T}.\,[\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right)$$

Brody, Alon, Yahav, "How Attentive are Graph Attention Networks?", ICLR 2022

Graph Attention Networks v2

- O sea que el mecanismo de attention "vanilla" identifica los nodos más importantes, pero puede llegar a ignorar el embedding del nodo original
- El problema es la linealidad consecutiva

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^\mathsf{T}.\,[\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right)$$

- Convirtamos la función de score en un MLP (aproximador universal): GATv2

$$e(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\mathsf{T}\text{LeakyReLU}\left(\mathbf{H}\,[\mathbf{x}_i || \mathbf{x}_j]\right)$$

Brody, Alon, Yahav, "How Attentive are Graph Attention Networks?", ICLR 2022

Graph Attention Networks v2

- O sea que el mecanismo de attention "vanilla" identifica los nodos más importantes, pero puede llegar a ignorar el embedding del nodo original
- El problema es la linealidad consecutiva

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^\top . [\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right)$$

- Convirtamos la función de score en un MLP (aproximador universal): GATv2

$$e(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\top \text{LeakyReLU}\left(\mathbf{H}\,[\mathbf{x}_i || \mathbf{x}_j]\right)$$

- Ejemplo sintético:

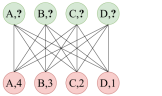


Figure 2: The DICTIONARY-LOOKUP problem of size k=4: every node in the bottom row has an alphabetic *attribute* ($\{A, B, C, ...\}$) and a numeric *value* ($\{1, 2, 3, ...\}$); every node in the upper row has only an attribute; the goal is to predict the value for each node in the upper row, using its attribute.

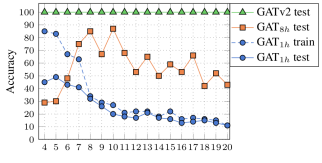


Figure 3: The DICTIONARYLOOKUP problem: GATv2 easily achieves 100% train and test accuracies even for k=100 and using only a single head.

Graph Attention Networks v2

- O sea que el mecanismo de attention "vanilla" identifica los nodos más importantes, pero puede llegar a ignorar el embedding del nodo original
- El problema es la linealidad consecutiva

$$e(\mathbf{x}_i, \mathbf{x}_j) = \text{LeakyReLU}\left(\mathbf{a}^\mathsf{T}.\,[\mathbf{H}\mathbf{x}_i || \mathbf{H}\mathbf{x}_j]\right)$$

- Convirtamos la función de score en un MLP (aproximador universal): GATv2

$$e(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{a}^\mathsf{T}\text{LeakyReLU}\left(\mathbf{H}\,[\mathbf{x}_i || \mathbf{x}_j]\right)$$

- Node prediction real:

Table 2: Average accuracy (Table 2a) and ROC-AUC (Table 2b) in node-prediction datasets (10 runs±std). In all datasets, GATv2 outperforms GAT. † – previously reported by Hu et al. (2020).

(a)

Model	Attn. Heads	**ogbn-arxiv**	**ogbn-products**	**ogbn-mag**
GCN†	0	71.74±0.29	78.97±0.33	30.43±0.25
GraphSAGE†	0	71.49±0.27	78.70±0.36	31.53±0.15
GAT	1	71.59±0.38	79.04±1.54	32.20±1.46
	8	71.54±0.30	77.23±2.37	31.75±1.60
GATv2 (this work)	1	71.78±0.18	**80.63**±0.70	**32.61**±0.44
	8	**71.87**±0.25	78.46±2.45	32.52±0.39

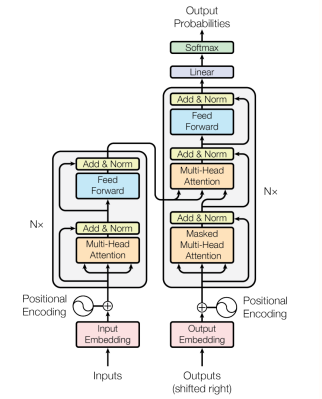
(b)

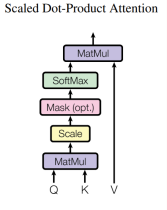
ogbn-proteins
72.51±0.35
77.68±0.20
70.77±5.79
78.63±1.62
77.23±3.32
79.52±0.55

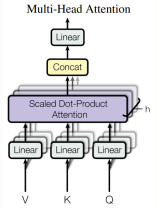
Brody, Alon, Yahav, "How Attentive are Graph Attention Networks?", ICLR 2022

Transformers

- Y a todo esto... ¿qué es un transformer?



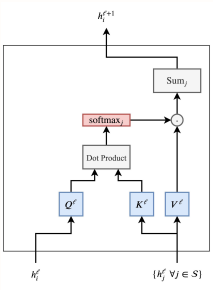




Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, y Polosukhin. "Attention is all you need" NeurIPS 2017.

Transformers

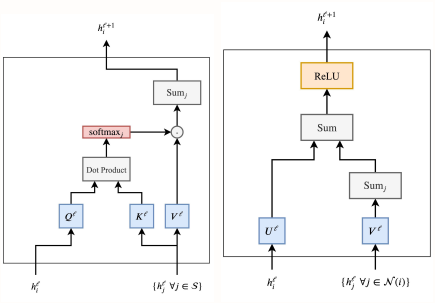
- Pongamos el módulo de attention en un Transformer en lenguaje de GNNs



Joshi, "Transformers are Graph Neural Networks", The Gradient, 2020

Transformers

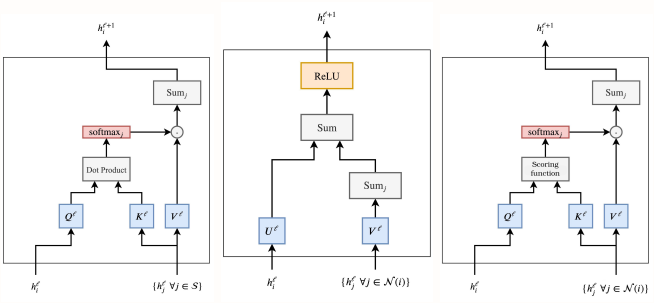
- Pongamos el módulo de attention en un Transformer en lenguaje de GNNs
- Ahora comparémoslo con el de una GNN



Joshi, "Transformers are Graph Neural Networks", The Gradient, 2020

Transformers

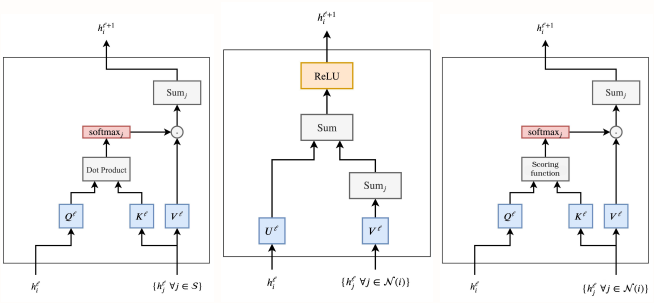
- Pongamos el módulo de attention en un Transformer en lenguaje de GNNs
- Ahora comparémoslo con el de una GNN
- Y finalmente agreguemos attention de GAT



Joshi, "Transformers are Graph Neural Networks", The Gradient, 2020

Transformers

- Pongamos el módulo de attention en un Transformer en lenguaje de GNNs
- Ahora comparémoslo con el de una GNN
- Y finalmente agreguemos attention de GAT



- Mensaje: Un transformer es una GNN con attention sobre un grafo fully connected
  - Incluso se le agregan positional encodings como los que ya vimos

Joshi, "Transformers are Graph Neural Networks", The Gradient, 2020

Rampášek, Galkin, Dwivedi, Luu, Wolf, y Dominique Beaini. "Recipe for a general, powerful, scalable graph transformer." NeurIPS 2022

Preguntín

¿Qué?

¿Porqué?

¿Cómo?

¿Funciona?

¿Debería
funcionar?

Preguntín

¿Qué?

Preguntín

¿Qué? $\Rightarrow$ Machine learning en grafos

Preguntín

¿Qué? $\Rightarrow$ Machine learning en grafos

¿Porqué?

Preguntín

¿Qué? $\Rightarrow$ Machine learning en grafos

¿Porqué? $\Rightarrow$ Modelos genéricos de estructura en la señal

Preguntín

¿Qué? $\Rightarrow$ Machine learning en grafos

¿Porqué? $\Rightarrow$ Modelos genéricos de estructura en la señal

Modelos de infraestructura física

Preguntín

¿Qué? ⇒ Machine learning en grafos

¿Porqué? ⇒ Modelos genéricos de estructura en la señal

Modelos de infraestructura física

¿Cómo?

Preguntín

¿Qué? $\Rightarrow$ Machine learning en grafos

¿Porqué? $\Rightarrow$ Modelos genéricos de estructura en la señal

Modelos de infraestructura física

¿Cómo? $\Rightarrow$ Graph Neural Networks

Preguntín

¿Qué? ⇒ Machine learning en grafos

¿Porqué? ⇒ Modelos genéricos de estructura en la señal

Modelos de infraestructura física

¿Cómo? ⇒ Graph Neural Networks

¿Funciona?

Preguntín

¿Qué?	$\Rightarrow$	Machine learning en grafos
¿Porqué?	$\Rightarrow$	Modelos genéricos de estructura en la señal
		Modelos de infraestructura física
¿Cómo?	$\Rightarrow$	Graph Neural Networks
¿Funciona?	$\Rightarrow$	Sí: Ejemplos

Preguntín

¿Qué?	$\Rightarrow$	Machine learning en grafos
¿Porqué?	$\Rightarrow$	Modelos genéricos de estructura en la señal
		Modelos de infraestructura física
¿Cómo?	$\Rightarrow$	Graph Neural Networks
¿Funciona?	$\Rightarrow$	Sí: Ejemplos
¿Debería funcionar?		

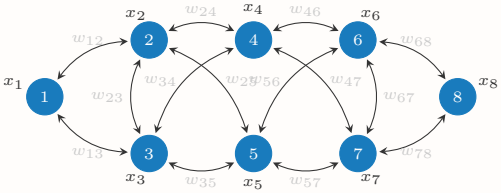
Preguntín

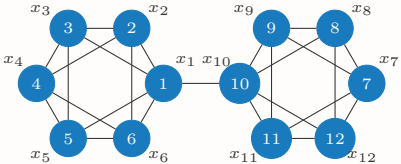
¿Qué?	$\Rightarrow$	Machine learning en grafos
¿Porqué?	$\Rightarrow$	Modelos genéricos de estructura en la señal
		Modelos de infraestructura física

¿Cómo?	$\Rightarrow$	Graph Neural Networks
¿Funciona?	$\Rightarrow$	Sí: Ejemplos
¿Debería funcionar?	$\Rightarrow$	Sí: Equivariance y estabilidad

¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

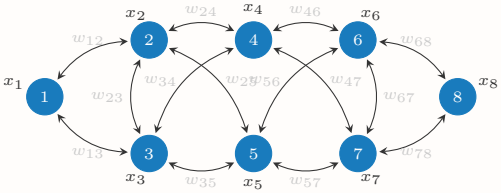
- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$

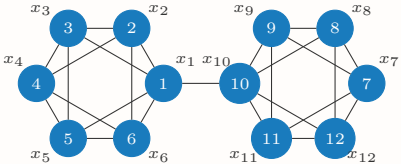




¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

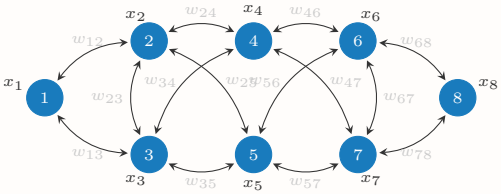
- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

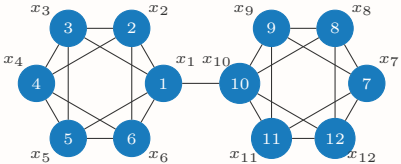




¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

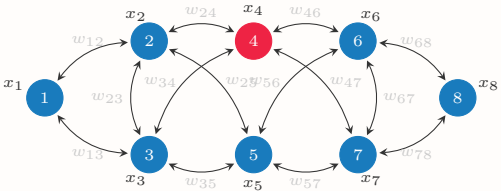


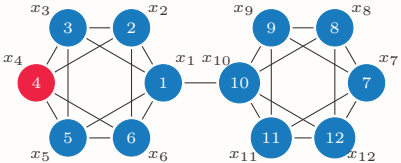


- Comparte/Hereda localidad con convolución temporal $\Rightarrow \mathbf{z} =$

¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

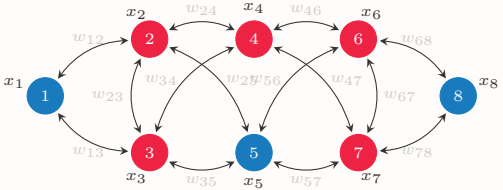


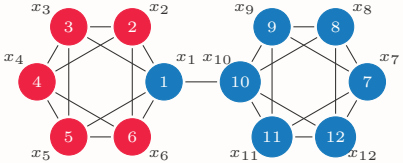


- Comparte/Hereda localidad con convolución temporal $\Rightarrow \mathbf{z} = h_0 \mathbf{S}^0 \mathbf{x}$

¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

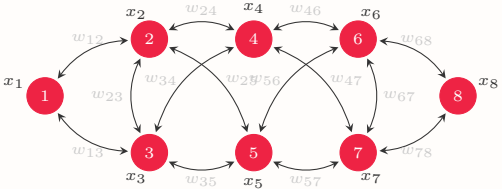


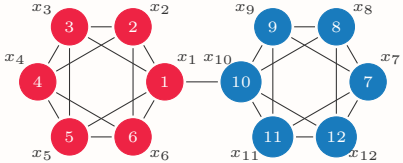


- Comparte/Hereda localidad con convolución temporal $\Rightarrow \mathbf{z} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x}$

¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

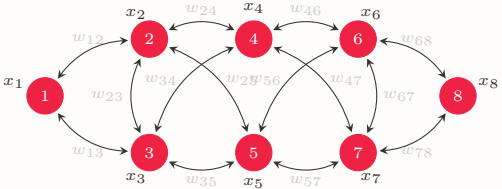


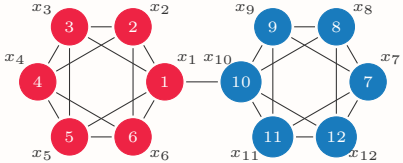


- Comparte/Hereda localidad con convolución temporal $\Rightarrow \mathbf{z} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x}$

¿Cómo? Filtros en grafos ≡ Convoluciones en grafos

- Filtrar usando el GSO $\mathbf{S}$ con coeficientes h_k es un polinomio en $\mathbf{S}$ $\Rightarrow \mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$
- Convolución es aplicar el filtro $\Rightarrow \mathbf{z} = \mathbf{h} *_{\mathbf{S}} \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$

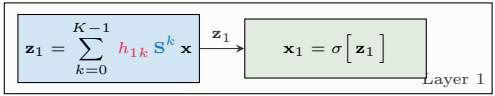




- Comparte/Hereda localidad con convolución temporal $\Rightarrow \mathbf{z} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$
- La salida del filtro depende de los coeficientes del filtro $\mathbf{h}$ y del GSO $\mathbf{S}$

¿Cómo? Graph Neural Networks (GNNs)

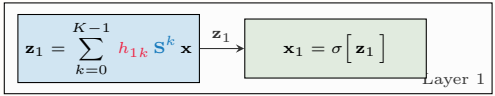
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

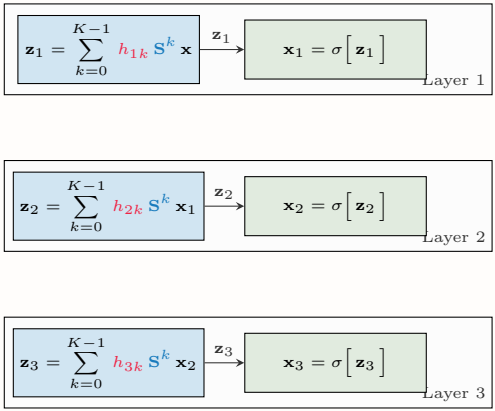
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)

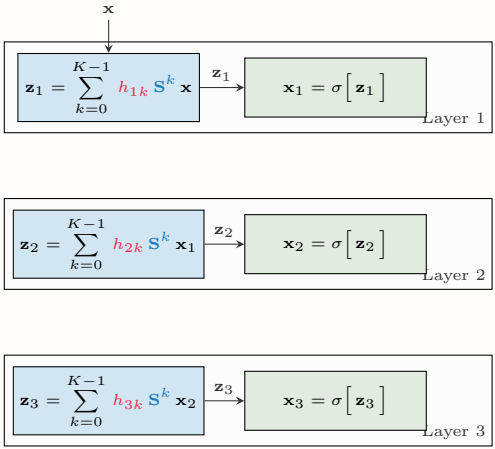


Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)

  $\Rightarrow$ Entramos con $\mathbf{x}$ a la Capa 1

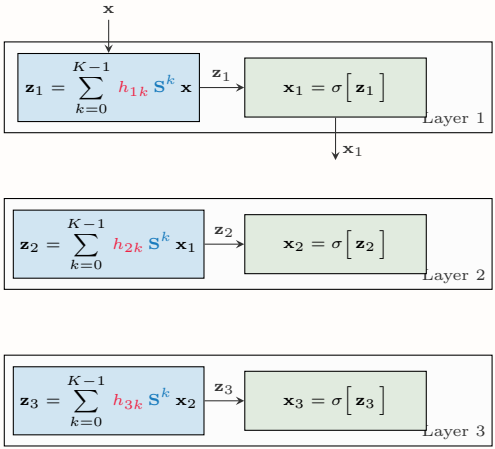


Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)

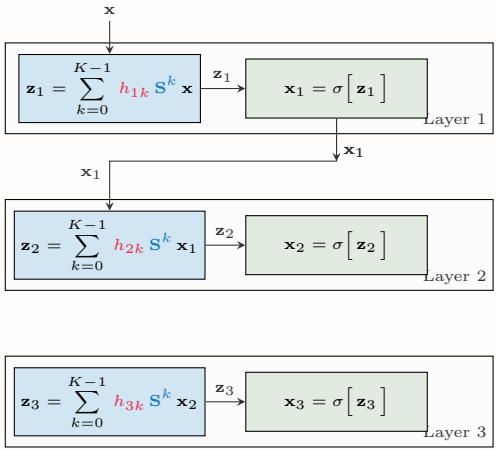
  $\Rightarrow$ Entramos con $\mathbf{x}$ a la Capa 1



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

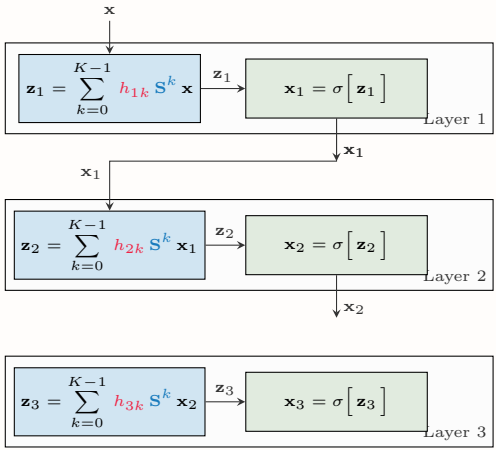
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)
  - ⇒ Entramos con $\mathbf{x}$ a la Capa 1
  - ⇒ Conectamos salida de la Capa 1 con entrada de 2



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

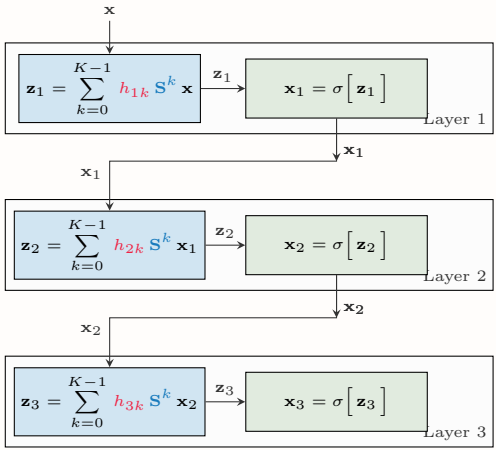
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)
  - ⇒ Entramos con $\mathbf{x}$ a la Capa 1
  - ⇒ Conectamos salida de la Capa 1 con entrada de 2



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

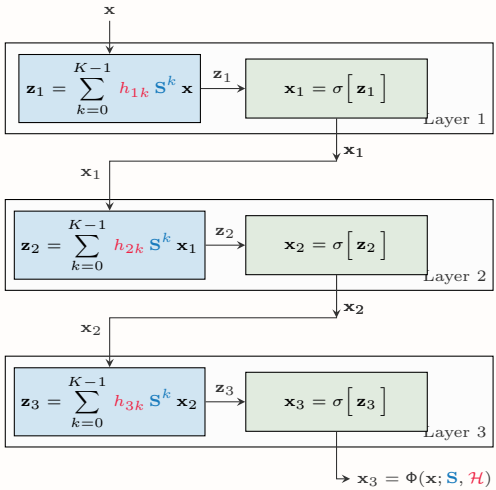
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)
  - ⇒ Entramos con $\mathbf{x}$ a la Capa 1
  - ⇒ Conectamos salida de la Capa 1 con entrada de 2
  - ⇒ y la salida de la Capa 2 con la entrada de la 3



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

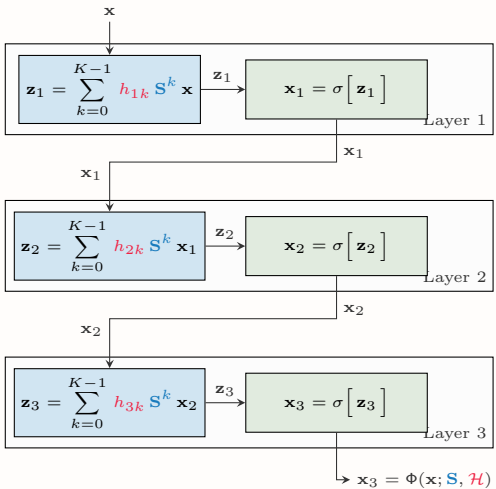
- Un graph perceptron es la composición de un graph filter con una no-linealidad puntual
- Apilar algunos graph perceptrons (3 en la Fig.)
  - ⇒ Entramos con $\mathbf{x}$ a la Capa 1
  - ⇒ Conectamos salida de la Capa 1 con entrada de 2
  - ⇒ y la salida de la Capa 2 con la entrada de la 3
- La última capa es la salida de la GNN ⇒ $\Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP 2018

¿Cómo? Graph Neural Networks (GNNs)

- Arquitectura básica $\Rightarrow$ Mucho más se usa en la práctica



Ruiz, Gama, Ribeiro, "Invariance-Preserving Localized Activation Functions for Graph Neural Networks", IEEE TSP 2019

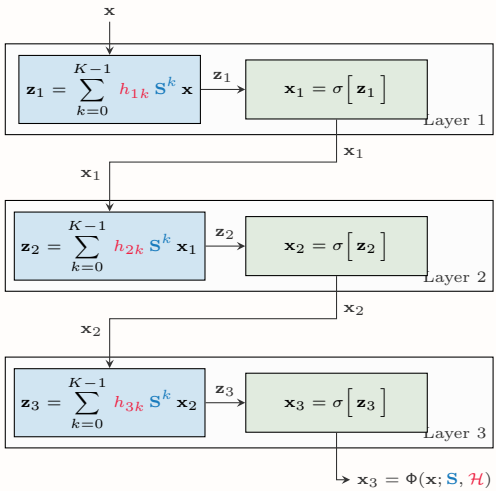
Isufi, Gama, Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks", IEEE TPAMI 2021

Ruiz, Gama, Ribeiro, "Gated Graph Recurrent Neural Networks, IEEE TSP 2020

¿Cómo? Graph Neural Networks (GNNs)

- Arquitectura básica $\Rightarrow$ Mucho más se usa en la práctica

  $\Rightarrow$ Múltiples features.



Ruiz, Gama, Ribeiro, "Invariance-Preserving Localized Activation Functions for Graph Neural Networks", IEEE TSP 2019

Isufi, Gama, Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks", IEEE TPAMI 2021

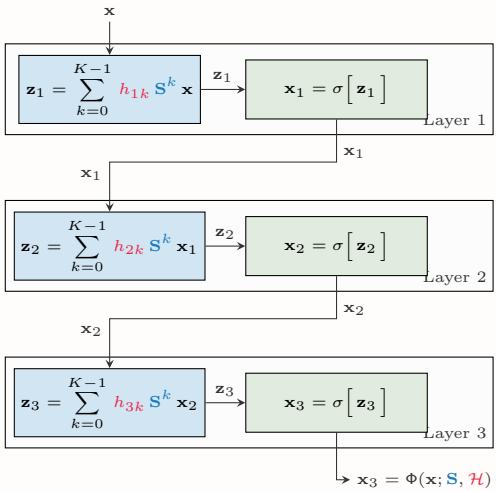
Ruiz, Gama, Ribeiro, "Gated Graph Recurrent Neural Networks, IEEE TSP 2020

¿Cómo? Graph Neural Networks (GNNs)

- Arquitectura básica $\Rightarrow$ Mucho más se usa en la práctica

  $\Rightarrow$ Múltiples features.

  $\Rightarrow$ Pooling.



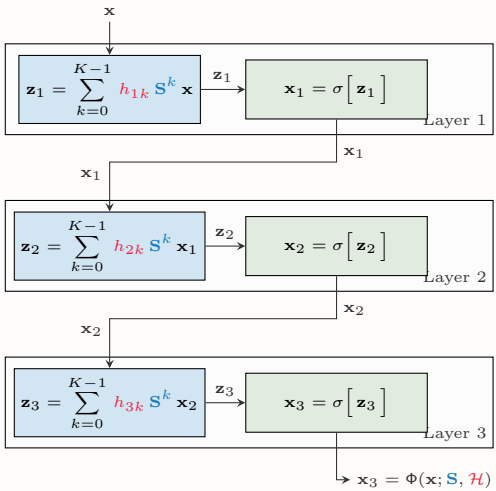
Ruiz, Gama, Ribeiro, "Invariance-Preserving Localized Activation Functions for Graph Neural Networks", IEEE TSP 2019

Isufi, Gama, Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks", IEEE TPAMI 2021

Ruiz, Gama, Ribeiro, "Gated Graph Recurrent Neural Networks, IEEE TSP 2020

¿Cómo? Graph Neural Networks (GNNs)

- Arquitectura básica $\Rightarrow$ Mucho más se usa en la práctica
  - $\Rightarrow$ Múltiples features.
  - $\Rightarrow$ Pooling.
- Además de varias cosas que no cubrimos
  - $\Rightarrow$ Local Nonlinear Activation.
  - $\Rightarrow$ Edge Varying Graph Filters.
  - $\Rightarrow$ Recurrent GNNs y Gating



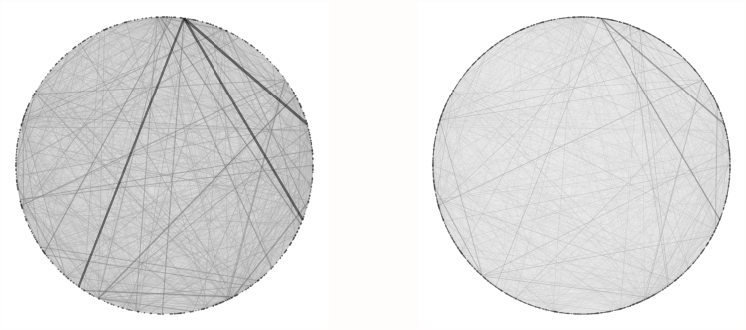
Ruiz, Gama, Ribeiro, "Invariance-Preserving Localized Activation Functions for Graph Neural Networks", IEEE TSP 2019

Isufi, Gama, Ribeiro, "EdgeNets: Edge Varying Graph Neural Networks", IEEE TPAMI 2021

Ruiz, Gama, Ribeiro, "Gated Graph Recurrent Neural Networks, IEEE TSP 2020

¿Funciona? Recommendation Systems

- Las GNNs son el SoA prediciendo ratings $\Rightarrow$ Aprovechan similaridades entre ratings anteriores
- Usar la estructura es necesario para aprendizaje escalable. El model free learning son los padres



Ruiz, Gama, Ribeiro, "Invariance-Preserving Localized Activation Functions for Graph Neural Networks", IEEE TSP 2019

¿Debería funcionar? Permutation Equivariance de GNNs

- GNNs son equivariantes a permutaciones.

Teorema (Gama, Bruna, Ribeiro)

Tenemos un graph signal $(\mathbf{S}, \mathbf{x})$ *y otro graph signal* $(\hat{\mathbf{S}}, \hat{\mathbf{x}})$ *permutado con* $\hat{\mathbf{S}} = \mathbf{P}^{\top}\mathbf{S}\mathbf{P}$ *y* $\hat{\mathbf{x}} = \mathbf{P}^{\top}\mathbf{x}$.

Ejecutamos GNNs con los mismos coeficientes $\mathcal{H}$ *produciendo las salidas* $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ *y* $\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H})$.

Las salidas están relacionadas por sus permutaciones respectivas

$$\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H}) = \mathbf{P}^{\top}\, \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$$

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", IEEE TSP 2020

¿Debería funcionar? Permutation Equivariance de GNNs

- GNNs son equivariantes a permutaciones.
  $\Rightarrow$ Procesamiento de Señales usando GNNs es independiente del etiquetado de los nodos

Teorema (Gama, Bruna, Ribeiro)

Tenemos un graph signal $(\mathbf{S}, \mathbf{x})$ *y otro graph signal* $(\hat{\mathbf{S}}, \hat{\mathbf{x}})$ *permutado con* $\hat{\mathbf{S}} = \mathbf{P}^{\top}\mathbf{S}\mathbf{P}$ *y* $\hat{\mathbf{x}} = \mathbf{P}^{\top}\mathbf{x}$.

Ejecutamos GNNs con los mismos coeficientes $\mathcal{H}$ *produciendo las salidas* $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ *y* $\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H})$.

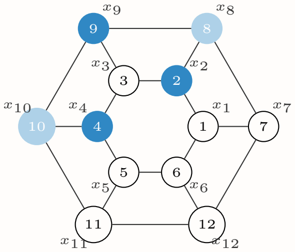
Las salidas están relacionadas por sus permutaciones respectivas

$$\mathbf{\Phi}(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H}) = \mathbf{P}^{\top}\, \mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$$

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", IEEE TSP 2020

¿Debería funcionar? Permutation Equivariance de GNNs

- Permutation equivariance explica la capacidad de las GNNS de aprovechar la estructura de la señal para generalizar
  - ⇒ Una red neuronal generaliza sólo de la izquierda al medio
  - ⇒ Graph Neural Networks generaliza también de la izquierda a la derecha



¿Debería funcionar? Estabilidad a Perturbaciones en el Grafo

- GNNs pueden ser estables respecto a perturbaciones relativas del grafo

Teorema (Gama, Bruna, Ribeiro)

Sean $\mathbf{S}$ *y* $\hat{\mathbf{S}}$ *GSOs en grafos con distancia relativa* $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$. *Sea* $\mathbf{\Phi}(\cdot; \mathbf{S}, \mathcal{H})$ *una GNN con* L *capas y* F *features por capa y además filtros integral Lipschiz con constante* C. *Entonces,*

$$\left\| \mathbf{\Phi}(\cdot; \mathbf{S}, \mathcal{H}) - \mathbf{\Phi}(\cdot; \hat{\mathbf{S}}, \mathcal{H}) \right\|_{\mathcal{P}} \leq 2C \left(1 + \delta\sqrt{N}\right) LF^{L-1} \varepsilon + \mathcal{O}(\varepsilon^2)$$

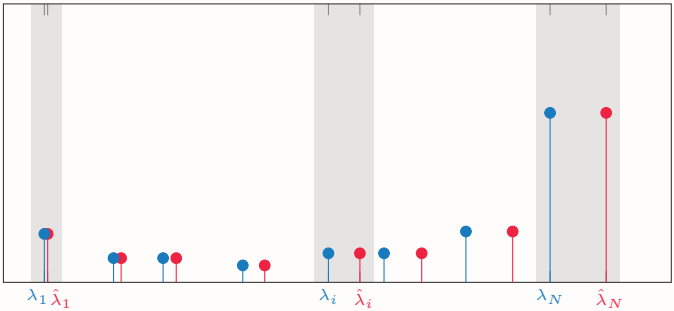
donde $\delta = (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1 \leq 8$ *es el eigenvector misalignment constant entre los vectores propios* $\mathbf{V}$ *del grafo* $\mathbf{S}$ *y los vectores propios* $\mathbf{U}$ *of the error matrix* $\mathbf{E}$.

Gama, Bruna, Ribeiro, "Stability Properties of Graph Neural Networks", IEEE TSP 2020

¿Debería funcionar? Estabilidad a Perturbaciones en el Grafo

- Filtros Integral Lipschitz son estables, pero no pueden discriminar señales de alta frecuencia en el grafo
- No-linealidades punto-a-punto actúan como demoduladores generando componentes en frecuencias bajas

  $\Rightarrow$ Pueden ser separadas por filtros estables en la siguiente capa



Preguntín

¿Qué? ⇒ Machine learning en grafos

¿Porqué? ⇒ Modelos genéricos de estructura en la señal Modelos de infraestuctura física

¿Cómo? ⇒ Graph Signals. Graph Convolutions. Graph Neural Networks.

¿Funciona? ⇒ Recommendation systems.

Indoor Localization

¿Deberían funcionar? ⇒ Permutation equivariance. Estabilidad a perturbaciones en el grafo.

Imposible para los grafos si son demasiado selectivos