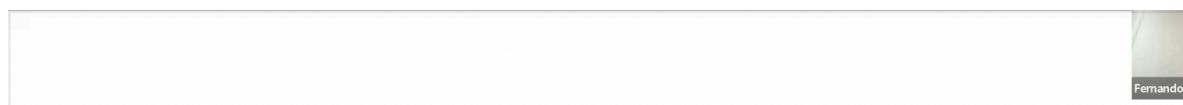


Clase 7 - GNNs : Graph Neural Networks



Deep Learning

Episodio 7: Redes Neuronales en Grafos, Parte 1

Fernando Gama

Escuela de Graduados en Ingeniería Informática y Sistemas, Facultad de Ingeniería, UBA

18 de Agosto de 2022



1 / 53

Redes neuronales para datos que presentan estructuras de grafos

Escenas del capítulo anterior...

GNNs, Pt.
fgama@fi.uba.ar

Fernando

- ▶ Queremos procesar **secuencias de datos**
 - ⇒ Adaptamos la red neuronal para **generalizar** en datos secuenciales
 - ⇒ **Compartir parámetros** ⇒ Mismos valores **durante toda la secuencia**
- ▶ **Redes neuronales recurrentes** ⇒ Aprenden un estado oculto
 - ⇒ Este estado es capaz de **capturar la información temporal relevante**
- ▶ Área activa de investigación ⇒ Muchas extensiones
 - ⇒ Redes neuronales bidireccionales ⇒ El contexto importa
 - ⇒ LSTMs, GRUs ⇒ Poder aprender dependencias de largo alcance
- ▶ **Attention y Transformers** ⇒ Aprender relaciones entre todos los elementos de la (sub)secuencia

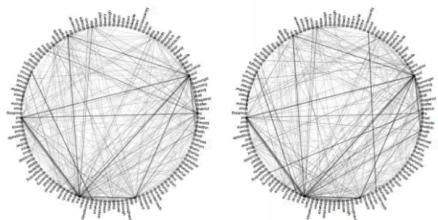


2 / 53

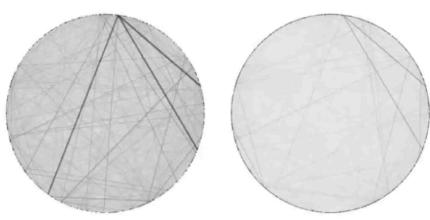
Los grafos nos permiten modelar muchos problemas distintos de forma optima.
Por ejemplo

- ▶ Los grafos son modelos para la estructura de los **datos** y facilitan el aprendizaje en muchas situaciones

Atribución de Autoría



Sistemas de Recomendación

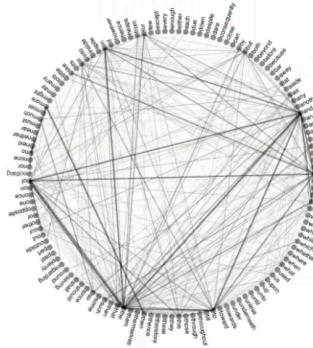


- ▶ En ambos casos existe un grafo que contiene información relevante acerca del problema a resolver

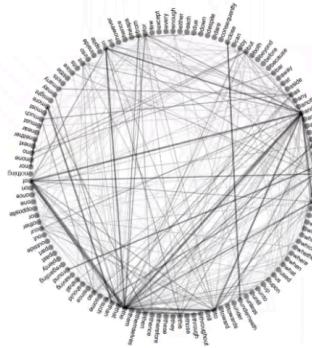
Atribución de Autoría

- ▶ Los **nodos** representan distintas **palabras** y **aristas** qué tan **seguido** dos palabras aparecen juntas
⇒ Representa las distintas formas en que los autores usan el lenguaje

William Shakespeare



Christopher Marlowe



- ▶ Las diferencias en los grafos reflejan diferencias en los estilos de Shakespeare y Marlowe

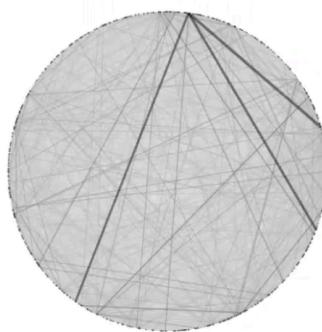
Sistemas de Recomendación

GNNs, Pt.
fgama@fi.uba.ar

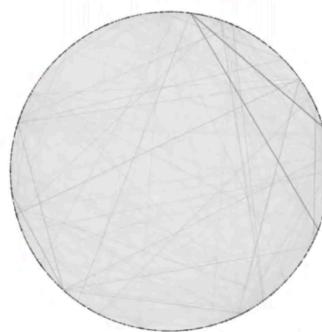
Fernando

- ▶ Los **nodos** representan distintos **productos** y las aristas la **similaridad** entre las puntuaciones
⇒ El grafo informa en la estimación del puntaje para productos no vistos

Puntajes de un cliente



Puntajes de otro cliente



Grafos en Sistemas Físicos

GNNs, Pt.
fgama@fi.uba.ar

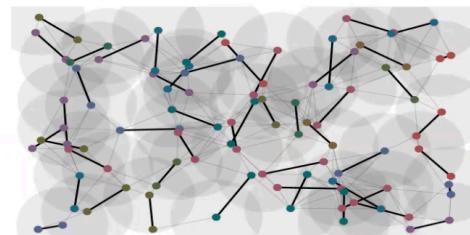
Fernando

- ▶ Los grafos son más que sólo **estructuras de los datos** ⇒ Sirven para modelar **infraestructura de redes**

Control Decentralizado de Sistemas Autónomos



Redes de Comunicación Inalámbrica

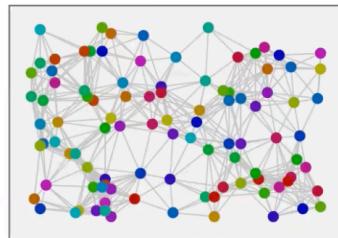


- ▶ El grafo impone una estructura de **información local** ⇒ Pero queremos resolver un **problema global**

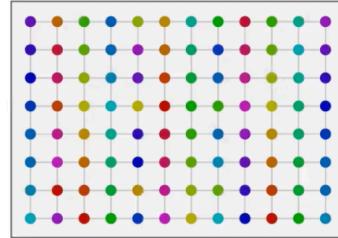
Notar como si bien todos los problemas son muy distintos, se pueden modelar con grafos. **La clave es que los elementos de mis datos tienen elementos de a pares.**

- ▶ Hasta ahora discutimos el *por qué* de pensar en grafos \Rightarrow Pero *cómo* vamos a procesarlos?
- ▶ Las **justificaciones teóricas y la inmensa evidencia empírica** sugieren elegir una red neuronal

Queremos una red neuronal sobre esto



Pero sabemos usar redes neuronales acá

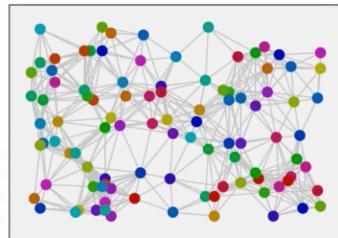


- ▶ Las redes neuronales completas (MLP) no escalan a grandes dimensiones \Rightarrow Pero las CNNs sí

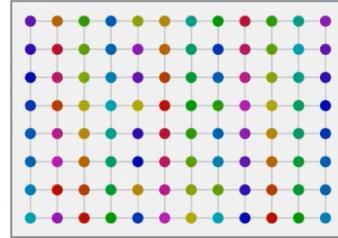
Vamos a arrancar con el MLP, pero agregándole información sobre la estructura de los datos.

- ▶ Las CNNs son una cascada de **capas** \Rightarrow **Convolución** seguido de **activación no-lineal**

Usamos una red neuronal en el grafo



Usamos una red neuronal convolucional



- ▶ Generalizar la **operación de convolución** a grafos y luego aplicar **activación no-lineal**
- ▶ Poner **capas** en cascada para crear una **red neuronal en el grafo (GNN)**

Como vamos a hacer esta operación de "convolución en el grafo"?

- ▶ ¿Cómo generalizamos las convoluciones para ser capaces de operar en grafos?
⇒ Podemos describir la **estructura temporal** usando **grafos que soportan señales en tiempo discreto**

Descripción del **tiempo** con un **grafo de línea dirigido**



- ▶ El **grafo de línea dirigido** representa la adyacencia de los puntos en el **tiempo**

Podemos pensar señales temporales como grafos de linea dirigida.
Donde ademas sabemos que las convocaciones son combinaciones lineales de desplazamientos de la señal

- ▶ Las convoluciones son **combinaciones lineales de desplazamientos** de la señal

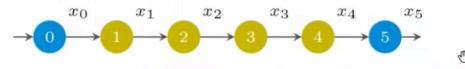
Descripción del **tiempo** con un **grafo de línea dirigido**



Escribimos el desplazamiento de forma genérica ya que si cambio el grafo también quiero cambiar esto:

- Las convoluciones son combinaciones lineales de desplazamientos de la señal

Descripción del tiempo con un grafo de línea dirigido



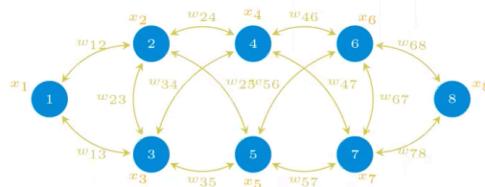
**Noción genérica
de desplazamiento**

- Filtro con coeficientes $h_k \Rightarrow$ Salida $\mathbf{z} = h_0 D_0(\mathbf{x}) + h_1 D_1(\mathbf{x}) + h_2 D_2(\mathbf{x}) + h_3 D_3(\mathbf{x}) + \dots = \sum_{k=0}^{\infty} h_k D_k(\mathbf{x})$

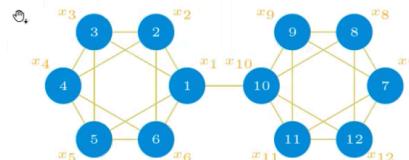
Pero que pasa cuando o tengo un grafo distinto? Con distinta estructura..

- Las señales temporales (y las imágenes) son importantes, pero son una clase limitada de señales
- Usamos grafos como descripciones genéricas de estructuras de la señal

Una señal sobre un grafo



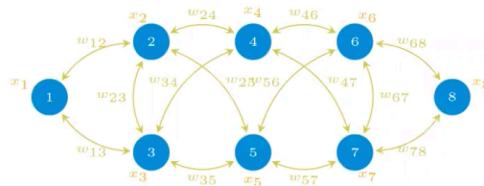
Otra señal sobre otro grafo



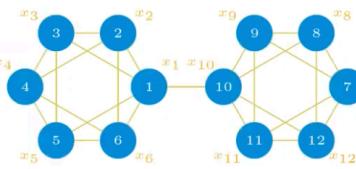
Siempre vamos a tener elementos en un grafo y valores de la señal asociados:

- ▶ Las señales temporales (y las imágenes) son importantes, pero son una clase limitada de señales
- ▶ Usamos grafos como descripciones genéricas de estructuras de la señal
⇒ Con **valores de la señal** asociados a los **nodos** y aristas que determinan similitud

Una señal sobre un grafo



Otra señal sobre otro grafo



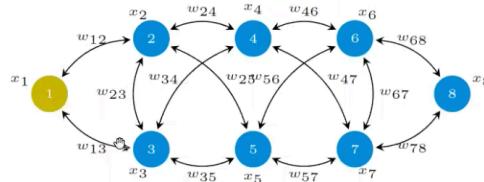
- ▶ Nodos, **señales** y aristas. Productos, **puntajes** y correlaciones. Drones, velocidades y distancias.

La idea es la siguiente, si antes habíamos definido a la convolución como una CL de desplazamientos.. ahora la queremos definir como una CL de desplazamientos *pero sobre el grafo...* el asunto es **como vamos a definir ese desplazamiento**.

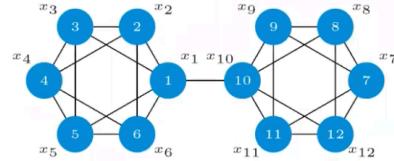
Convoluciones en Grafos

- ▶ Para señales en grafos definimos la **convolución** como una **combinación lineal de desplazamientos**

Una señal sobre un grafo



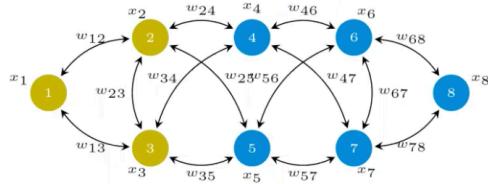
Otra señal sobre otro grafo



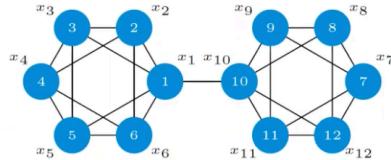
- ▶ Filtro con **coeficientes h_k** ⇒ Salida $\mathbf{z} = h_0 \mathbf{D}_0(\mathbf{x})$

- Para señales en grafos definimos la convolución como una combinación lineal de desplazamientos

Una señal sobre un grafo



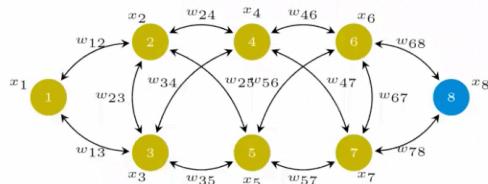
Otra señal sobre otro grafo



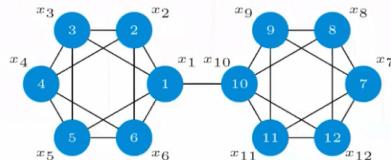
- Filtro con coeficientes $h_k \Rightarrow$ Salida $\mathbf{z} = h_0 \mathbf{D}_0(\mathbf{x}) + h_1 \mathbf{D}_1(\mathbf{x})$

- Para señales en grafos definimos la convolución como una combinación lineal de desplazamientos

Una señal sobre un grafo



Otra señal sobre otro grafo



- Filtro con coeficientes $h_k \Rightarrow$ Salida $\mathbf{z} = h_0 \mathbf{D}_0(\mathbf{x}) + h_1 \mathbf{D}_1(\mathbf{x}) + h_2 \mathbf{D}_2(\mathbf{x}) + h_3 \mathbf{D}_3(\mathbf{x})$

Lo que vamos a hacer es: el valor ed la señal en cada nodo. Mi filtro va a ser h_0 por el valor en ese nodo, luego le pregunto a mis vecinos que valor tienen y peso eso por otro h_1 , luego a los vecinos de mis vecinos por h_2 y así sucesivamente.

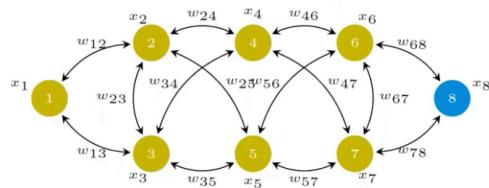
El desplazamiento recupera la noción espacial de la convolución pero ahora va a depender de la estructura del grafo.

h_k van a ser los parametros que yo quiero aprender

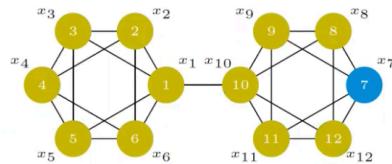
Lo mismo se puede hacer en el grafo de la derecha, lo que cambia es la noción de desplazamiento que utilizamos

- Para señales en grafos definimos la convolución como una combinación lineal de desplazamientos

Una señal sobre un grafo



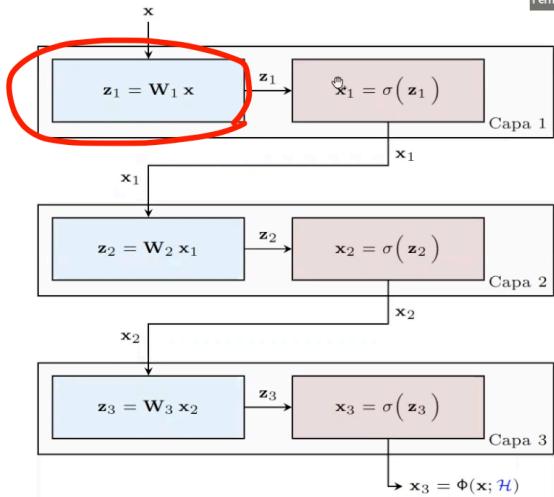
Otra señal sobre otro grafo



- Filtro con coeficientes $h_k \Rightarrow$ Salida $\mathbf{z} = h_0 \mathbf{D}_0(\mathbf{x}) + h_1 \mathbf{D}_1(\mathbf{x}) + h_2 \mathbf{D}_2(\mathbf{x}) + h_3 \mathbf{D}_3(\mathbf{x}) + \dots = \sum_{k=0}^{\infty} h_k \mathbf{D}_k(\mathbf{x})$
- Cómo definimos el desplazamiento para que relacione elementos contiguos de la señal

> Repasando el caminito que fueron tomando nuestras NN desde el MLP hasta ahora:

- Una red neuronal es una cascada de bloques o capas
- Cada una aplica una transformación lineal
 \Rightarrow Y una función de activación no-lineal
- No escala para datos \mathbf{x} de gran dimensión

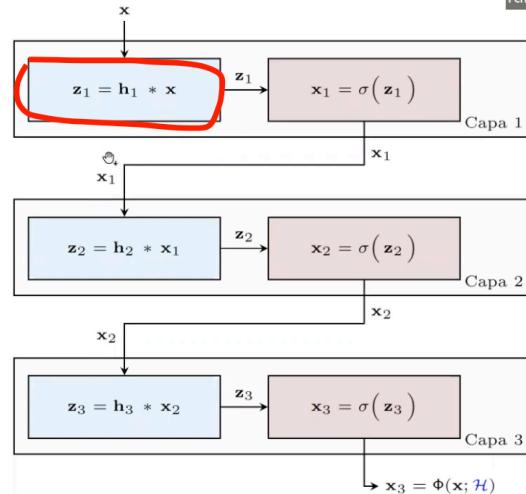


Redes Neuronales Convolucionales (CNNs)

GNNs, Pt.
fgama@fi.uba.ar

Fernando

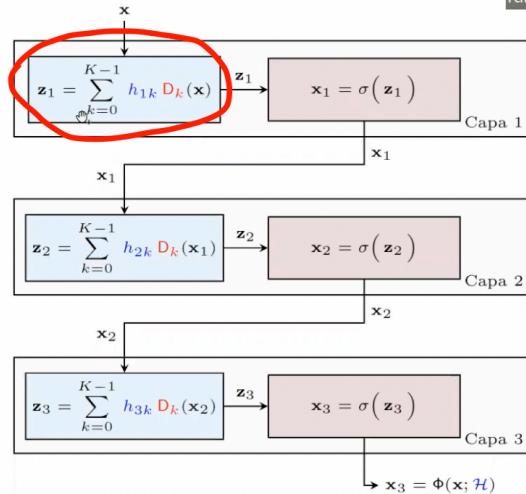
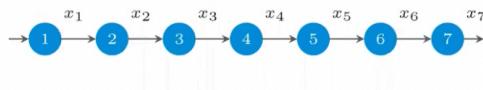
- ▶ Una NN convolucional es una cascada de capas
- ▶ Cada una aplica una convolución
 - ⇒ Y una función de activación no-lineal
- ▶ Escala a datos grandes sin problemas
- ▶ Una CNNs es una variación de un filtro convolucional
 - ⇒ Le agregamos una activación no-lineal y repetimos
 - ⇒ Escalan porque las convoluciones escalan



Pensando Señales Temporales como un Grafo

GNNs, Pt.
fgama@fi.uba.ar

- ▶ Las convoluciones son combinaciones lineales
 - ⇒ De desplazamientos determinados por este grafo
- ▶ Esta una manera de escribir la convolución
 - ⇒ Que permite generalizar a otros grafos



Lo que vamos a hacer ahora es hacer que esta noción de desplazamiento pueda capturar la estructura de cualquier grafo.

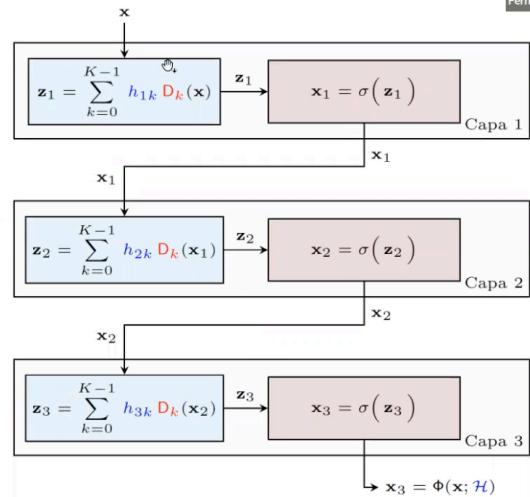
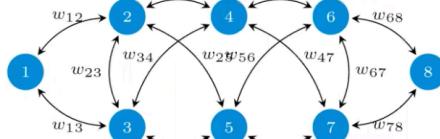
Es a grandes rasgos, reemplazar la operación lineal del MLP por una operación que tenga en cuenta la estructura del grafo.

Redes Neuronales en Grafos (GNNs)

GNNs, Pt.
fgama@fi.uba.ar

Fernando

- El grafo puede ser cualquier grafo
- Hay que definir la noción de desplazamiento

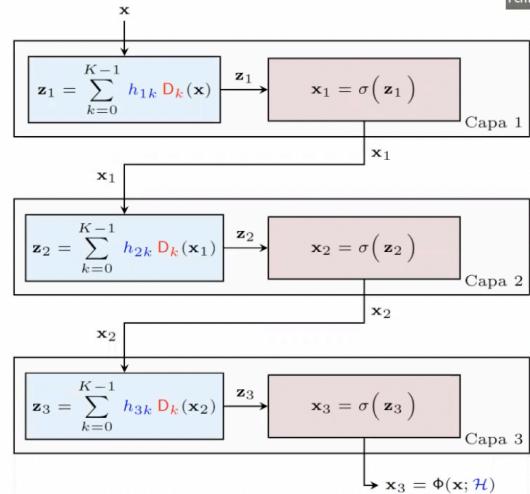


Redes Neuronales en Grafos (GNNs)

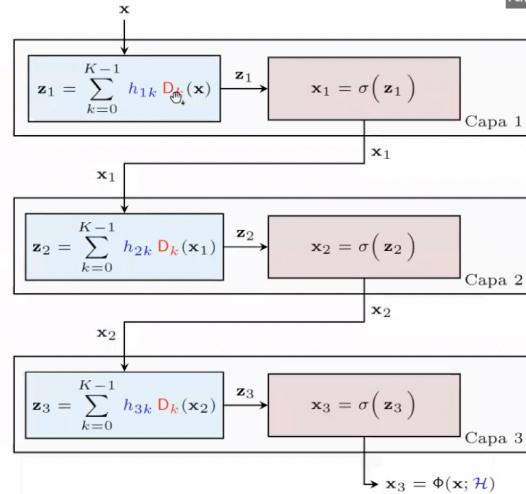
GNNs, Pt.
fgama@fi.uba.ar

Fernando

- Una NN en grafos es una cascada de capas
- Cada capa aplica una convolución en el grafo
 - ⇒ Y una función de activación no-lineal
- Es un MLP con la operación lineal restringida
- Recupera la CNN si es un grafo de línea dirigido



- ▶ Hay evidencia empírica de que las GNNs escalan
- ▶ Una GNN es una variación de un filtro en un grafo
 - ⇒ Se le agrega una activación no-lineal y se repite
- ▶ Escala porque el filtro explota la estructura



Vamos a tener en cuenta el grafo (en el cambio sobre la operación del MLP normal) en la operación de desplazamiento de las cajas celestes.

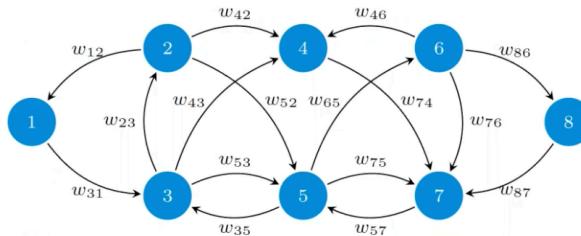
La moraleja acá es independiente del tipo de datos. Si tengo información sobre la estructura de mis datos, voy a buscar una operación que reemplace la TL y explote esa estructura. Hoy son los grafos, antes fueron las convoluciones o las recurrencias, pero en el futuro el concepto será el mismo, cambiara la operación.

Como vamos a hacer esto? Vamos a definir una operación que trabaje en grafos!

> Grafos

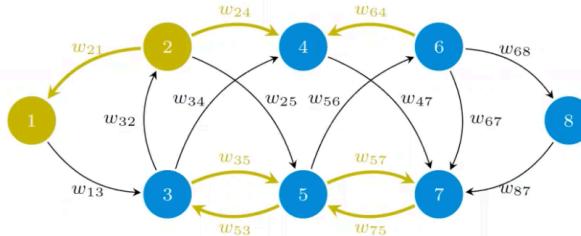
Para esto, primero vamos a definir formalmente lo que es un grafo: un conjunto de tres elementos, **nodos** o vértices (N elementos), **aristas** (pares ordenados de nodos, me dicen que nodos tienen influencia con otros) y **pesos** (en general es optativo, representa el peso de las relaciones entre nodos)

- Un grafo es una tríada $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ de nodos, aristas y pesos
 - ⇒ Los **nodos** son un conjunto de **N** elementos (e.g., $\mathcal{V} = \{1, \dots, N\}$ o $\mathcal{V} = \{v_1, \dots, v_N\}$)
 - ⇒ Las **aristas** son pares ordenados de nodos (i, j) (determinan relaciones de influencia)
 - ⇒ Los **pesos** $w_{ij} \in \mathbb{R}$ son números asociados a las aristas (i, j) (la magnitud de la influencia)



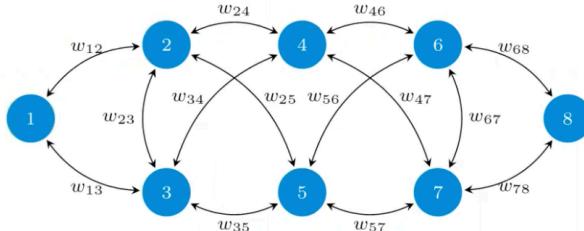
Representación de grafos dirigidos

- La arista (i, j) se representa como una **flecha apuntando de i a j** ⇒ Influencia de i a j
- La arista (i, j) es diferente de la arista (j, i) ⇒ Es posible que $(i, j) \in \mathcal{E}$ y $(j, i) \notin \mathcal{E}$ simultáneamente
- Si las dos aristas existen, puede ser que sus pesos sean distintos ⇒ Es posible que $w_{ij} \neq w_{ji}$



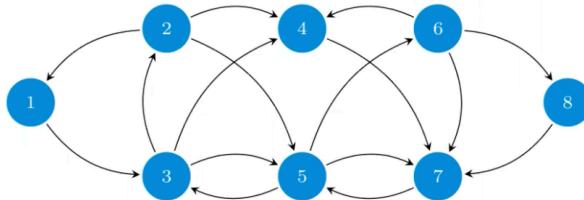
Y en **no dirigidos**, donde todas las aristas vienen de a pares. Las relaciones se van a dar siempre y cuando ambos ejerzan la misma influencia, esto simplifica enormemente el problema porque necesito mucho menos valores para describirlo.

- Un grafo es no dirigido si tanto su conjunto de aristas como su función de pesos son simétricas
 \Rightarrow Las aristas vienen de a pares \Rightarrow Tenemos $(i, j) \in \mathcal{E}$ si y sólo si $(j, i) \in \mathcal{E}$
 \Rightarrow Los pesos son simétricos \Rightarrow W cumple que $W(i, j) = w_{ij} = w_{ji} = W(j, i)$ para todo $(i, j) \in \mathcal{E}$



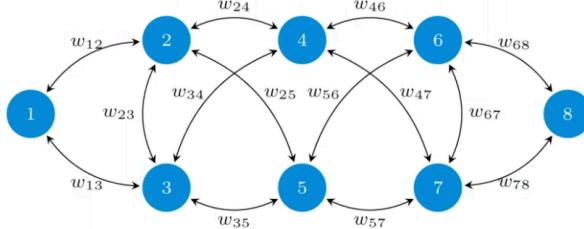
Por ultimo tenemos los **grafos sin peso**. No se tiene en cuenta los pesos, se los asume unitarios. Solo nos importa si la relacion esta o no presente

- Un grafo es sin peso si no tiene una función de peso W
 \Rightarrow En general, se asume entonces, que todos los pesos son unitarios $\Rightarrow w_{ij} = 1$ para todo $(i, j) \in \mathcal{E}$
- Los grafos sin peso pueden ser dirigidos o no dirigidos



—> *En general, lo que mas se suele usar son los grafos NO dirigidos CON peso. Ya que tienen un par de propiedades matemáticas interesantes que simplifican bastante los cálculos.*

- ▶ Los grafos pueden ser dirigidos o no dirigidos, pueden tener peso o no tenerlo
- ▶ En general, vamos a considerar grafos que sean no dirigidos y con pesos



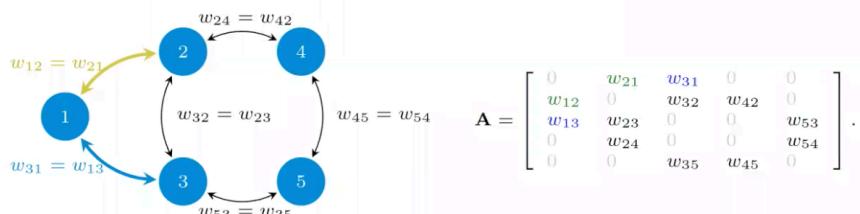
> Matrices de adyacencia

Es una matriz que representa en cada lugar los pesos de las relaciones entre nodos. Cada lugar me esta hablando del peso entre el nodo i y el nodo j .

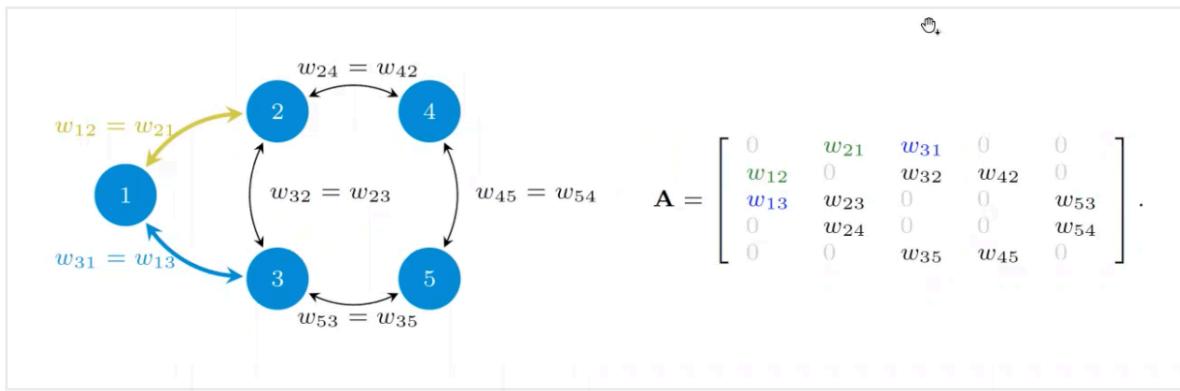
- ▶ La matriz de adyacencia del grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ es una matriz rala $\mathbf{A} \in \mathbb{R}^{N \times N}$ con elementos no-nulos

$$[\mathbf{A}]_{ij} = w_{ji}, \text{ para todo } (j, i) \in \mathcal{E}$$

- ▶ Si el grafo es no dirigido, la matriz de adyacencia es simétrica $\Rightarrow \mathbf{A} = \mathbf{A}^T$ (como en el ejemplo)



En este caso, como tenemos un grafo no dirigido, tenemos una matriz simétrica:



Si el grafo es no dirigido, tenemos una matriz binaria, a todos los pesos se les asigna uno:

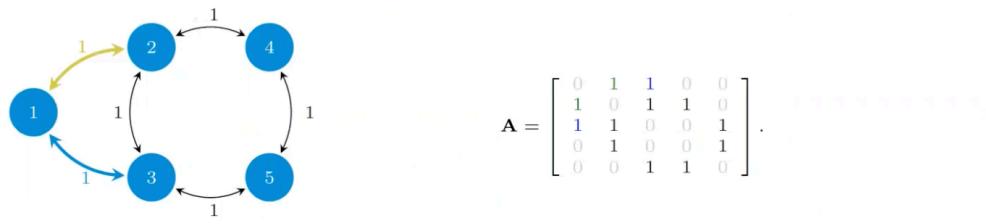
Matrices de Adyacencia

GNNs, Pt.
fgama@fi.uba.ar

Fernando

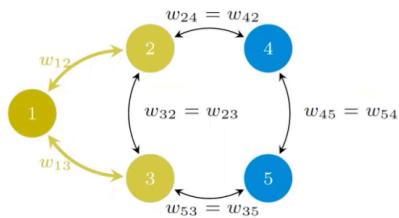
- Para el caso particular en que el **grafo no tiene pesos** \Rightarrow La matriz de adyacencia tiene **pesos unitarios**

$$[\mathbf{A}]_{ij} = 1, \text{ for all } (j, i) \in \mathcal{E}$$



- > Existen otros aspectos de los grafos que son interesantes, como por ejemplo el concepto de
 - **Vecindario:** cuales son los nodos que tienen una determinada relación con un determinado nodo. El vecindario del nodo i es el conjunto de nodos que influencian al nodo i . [conjunto]
 - **Grado:** suma de los pesos de las aristas de cada nodo. [numero]

- El **vecindario** del nodo i es el conjunto de nodos que **influyen** a $i \Rightarrow n(i) := \{j : (j, i) \in \mathcal{E}\}$
- El **grado** d_i del nodo i es la **suma de los pesos** de las aristas incidentes $\Rightarrow d_i = \sum_{j \in n(i)} w_{ij} = \sum_{\{(j, i) \in \mathcal{E}\}} w_{ij}$

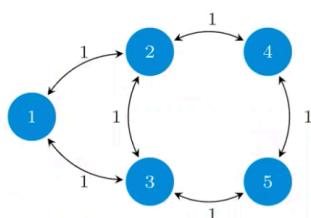


► Vecindario del nodo 1 $\Rightarrow n(1) = \{2, 3\}$

► Grado del nodo 1 $\Rightarrow d_1 = w_{21} + w_{31}$

La **matriz de grado** pone en la diagonal el grado de cada uno de los nodos:

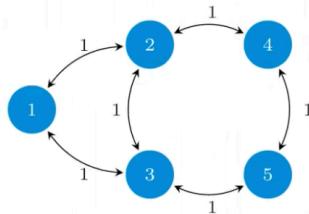
- La matriz de grado es una matriz diagonal \mathbf{D} con los grados en la diagonal $\Rightarrow [\mathbf{D}]_{ii} = d_i$
- Se puede escribir en función de la matriz de adyacencia $\mathbf{D} = \text{diag}(\mathbf{A}1)$



$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}.$$

Por otro lado se usa bastante la **Matriz Laplaciana**:

- La **matriz Laplaciana** dada una matriz de adyacencia \mathbf{A} es $\Rightarrow \mathbf{L} = \mathbf{D} - \mathbf{A} = \text{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$
- Se puede escribir directamente usando los pesos del grafo $[\mathbf{A}]_{ij} = w_{ji}$ (sólo para grafos no dirigidos)
 - \Rightarrow Los elementos **frente a la diagonal** $\Rightarrow [\mathbf{L}]_{ij} = -[\mathbf{A}]_{ij} = -w_{ji}$
 - \Rightarrow Los elementos **en la diagonal** $\Rightarrow [\mathbf{L}]_{ii} = d_i = \sum_{j \in n(i)} w_{ji}$



$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

La misma se calcula como la matriz de grado - la matriz de adyacencia

Importante: la laplaciana solo se puede usar con grafos no dirigidos. Si el grafo es dirigido no vale. Tenemos que usar adaptaciones de la matriz de adyacencia que vamos a ver ahora.

> Normalizaciones

Se suelen aplicar a los pesos por ejemplo. En vez de tener pesos absolutos, se pueden relativizar, en función del peso total del grafo o del peso total del nodo por ej.

- Las **normalizaciones** de la matriz de adyacencia y Laplaciana expresan los pesos relativos al grado
- La matriz de adyacencia normalizada $\Rightarrow \tilde{\mathbf{A}} := \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$ (es simétrica si el grafo es no dirigido)
- La matriz **Laplaciana normalizada** $\Rightarrow \tilde{\mathbf{L}} := \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2}$ (sólo existe en grafos no dirigidos)
- Dadas las definiciones de las normalizaciones de las matrices de adyacencia y Laplaciana $\Rightarrow \tilde{\mathbf{L}} := \mathbf{I} - \tilde{\mathbf{A}}$

> Operador desplazamiento en el Grafo

Va a ser una Matriz S:

Operador de Desplazamientos en el Grafo

GNNs, Pt.
fgama@fi.uba.ar

Fernando

- El operador desplazamiento en el grafo S es un genérico para cualquier matriz del grafo

Matriz de adyacencia

$$S = A$$

Matriz Laplaciana

$$S = L$$

Adyacencia normalizada

$$S = \bar{A}$$

Laplaciana normalizada

$$S = \bar{L}$$

- Si el grafo es no dirigido el operador desplazamiento S es simétrico $\Rightarrow S = S^T$
- La elección específica de ODG importa en la práctica pero la los análisis suelen valer para cualquier S

El elemento que relaciona Nodos que NO tienen arista es CERO.

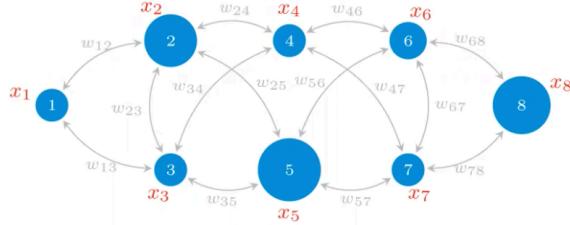
Vamos a usar esta matriz S para definir la noción de desplazamiento de mi GNN.

Esta matriz S que elegimos va a tomar el rol de hiperparámetro de nuestra arquitectura. La performance de la red va a estar sujeta a como elegimos la matriz. Vamos a tener que probar, por ejemplo entre la matriz de adyacencia y la Laplaciana y ver cual funciona mejor.

> Señales en Grafos

Una señal en un grafo, básicamente le asigna un numero o un vector a cada uno de los nodos de mi grafo. Hasta ahora solo tenía números asociados a las aristas (relación) y ahora le asigno también a los nodos.

- Consideremos un grafo \mathcal{G} dado con N nodos y operador de desplazamiento \mathbf{S}
- Una señal en el grafo es un vector $\mathbf{x} \in \mathbb{R}^N$ tal que el elemento x_i se asocia al nodo i
- Para no perder de vista que la estructura de la señal depende del grafo \Rightarrow La señal es el par (\mathbf{S}, \mathbf{x})



- El grafo es una suposición de **similaridad o proximidad** entre los elementos de \mathbf{x} (contiguos o no)

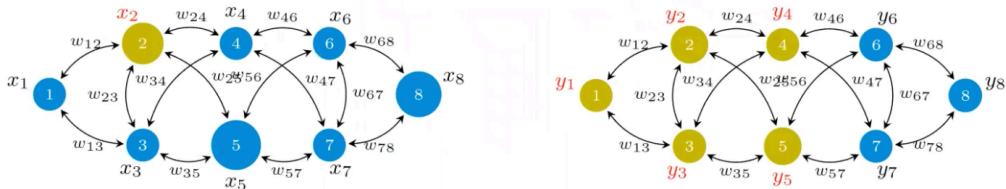
Vamos a tomar que la matriz (grafo) es siempre la misma y que los datos vienen dados por distintas señales. Es decir, uno de mis datos es un valor para cada uno de los nodos, mi siguiente es otro valor y así. PERO la relación que hay entre los nodos es siempre la misma y viene dada por esta matriz S . Podemos tirar una analogía con imágenes acá, donde si bien todas son distintas entre sí potencialmente, la relación entre los píxeles de ellas es la misma, todas tienen el mismo tamaño, etc.

Mi señal la puedo pensar como un vector, donde el elemento 1 del vector corresponde al valor de la señal en el nodo 1, el elemento en el nodo 2, etc. Vamos a ver que pasa mas adelante si en vez de un escalar por nodo asigno un vector ahí también. Por ahora vamos simple.

> Difundir (desplazar) las señales en el Grafo

Recordemos que estamos buscando definir el operacional de desplazamiento para poder definir la operación que metemos en la red neuronal..

- Multiplicar la señal \mathbf{x} con el operador desplazamiento \mathbf{S} difunde la señal
 - La señal difundida $\mathbf{y} = \mathbf{Sx}$ tiene elementos dados por $y_i = \sum_{j \in n(i)} w_{ji} x_j$
- ⇒ Captura una **operación local** donde los elementos se mezclan con los de los vecinos



Tengo mi señal X con cada una de sus componentes indicando el valor de la señal en cada uno de los nodos. $X \in N$ y S (adyacencia) $\in NxN$.

Que pasa? Mi operador desplazamiento S tiene una estructura muy particular: cada elemento de la matriz donde yo NO tenga una arista voy a tener un cero. Luego cuando llegue el momento de multiplicar ahí $Sx = y$ me va a quedar 0. Lo cual me va a llevar a que **solo se multipliquen elementos asociados a los nodos que comparten un vecindario con esa fila**. Es decir, tomemos la fila correspondiente al nodo 2, tomemos el elemento 12 de la matriz = W_{12} . Okey, pongo el W_{12} y voy buscar el valor X_1 .

Vayamos al elemento 3, (el 2 va a estar incluido en la diagonal) y observamos que tenemos una relación 2-3 luego cuando hago la multiplicación de matrices este elemento va a estar multiplicado por x_3 y va a ir para x_2 . Lo mismo va a suceder para el elemento X_4 y X_5 . PERO cuando vaya a multiplicar por 6, voy a ver que como no hay una arista entre 2 y 6, para esa fila voy a tener un 0 y X_6 nunca va a llegar a X_2 , por la estructura que tiene esta Matrix. El resultado de multiplicar por S luego fue que el resultado de la señal y_2 SOLO depende de los valores de mi vecindario y de nada mas.

Por la manera en la que construimos mi matriz S , estoy automáticamente definiendo una operación que tiene en cuenta la estructura del grafo, en particular una operación que es local -> solo incluye elementos en el vecindario.

En el caso de los sistemas de recomendación, S es efectivamente una matriz que yo tengo que guardar en memoria y donde tengo que hacer la multiplicación de matrices. Ahora, para el problema de los drones o comunicaciones, en realidad cada robot tiene poder de computo propio. Entonces cuando voy a multiplicar Sx el robot lo que hace es abrir la antena, recibir las señales, sumarlas y luego actualiza su valor. Nada mas, no tiene ni idea del tamaño de S ni de cualquier cosa que pase mas allá de su vecindario.

Es fundamental que sepamos que la operación $y = Sx$ solo relaciona elementos dados por la estructura del grafo, en el vecindario.

> La secuencia de Difusión

La joda acá es que la operación Sx la vamos a empezar a repetir por el grafo!

La Secuencia de Difusión

GNNs, Pt.
fgama@fi.uba.ar
Fernando

- El ODG se puede **repetir** para producir la **secuencia de difusión** definida recursivamente como

$$\mathbf{x}^{(k+1)} = \mathbf{S}\mathbf{x}^{(k)}, \quad \text{con } \mathbf{x}^{(0)} = \mathbf{x}$$
- Alternativamente, **desdoblamos** la recursión para escribir $\mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x}$

$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0\mathbf{x}$ $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}^{(0)} = \mathbf{S}^1\mathbf{x}$ $\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)} = \mathbf{S}^2\mathbf{x}$ $\mathbf{x}^{(3)} = \mathbf{S}\mathbf{x}^{(2)} = \mathbf{S}^3\mathbf{x}$

- El k -ésimo elemento de la secuencia $\mathbf{x}^{(k)}$ difunde la información al vecindario a k -saltos

.UBAfiuba
FACULTAD DE INGENIERÍA 35 / 53

Cada uno de los nodos actualiza su información con la info de sus vecinos. Cuando ahora vuelvo a multiplicar por S , me llega la información de los vecinos de mis vecinos. Multiplicando por S una segunda vez lo que obtengo es la información de los vecinos que están a dos saltos, la obtengo de mis vecinos a un salto que a su vez traen de mas lejos. Entonces a medida que voy aplicando este operador desplazamiento S , empiezo a incorporar información que esta ubicada cada vez mas lejos. Ahora bien, lo que es muy interesante aca, no la incorpo *hablando directamente* con mis vecinos a dos saltos. Dos nunca se comunica con 6, pero logra obtenerla a través de 4. Dos solo se comunica con 1 3 5 y 4, pero como estos se comunican con sus vecinos, obtiene esa info tambien.

Esta noción de desplazamiento es lo que me permite definir el filtro. Lo único que tengo que hacer es asignarle un coeficiente h_1, h_2 , segun a cuantos saltos este la info.

- Dado un ODG \mathbf{S} y coeficientes h_k , un filtro en un grafo es un polinomio en \mathbf{S}

$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$$

- Aplicar un filtro $\mathbf{H}(\mathbf{S})$ a una señal \mathbf{x} resulta en otra señal

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$$

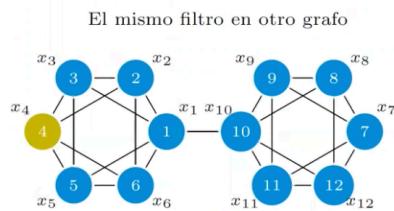
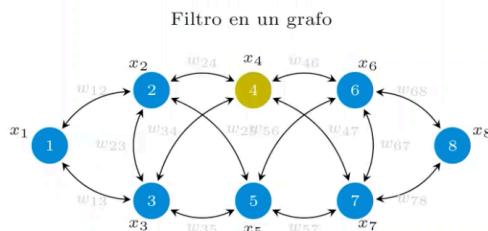
- Decimos que $\mathbf{y} = \mathbf{h} *_{\mathbf{S}} \mathbf{x}$ es la convolución en el grafo del filtro $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ con la señal \mathbf{x}

lo que me queda en rojo, es una combinación lineal de desplazamientos en el grafo.

Ahora, efectivamente, la salida \mathbf{Y} tiene incorporada información de los desplazamientos en el grafo. Lo que vamos a aprender son los valores H_k que le asigna un orden de relevancia a las relaciones con vecinos. ESO es lo que aprendemos con las GNNs.

Vamos a empezar a construir la información con sucesivas interacciones con mis vecinos:

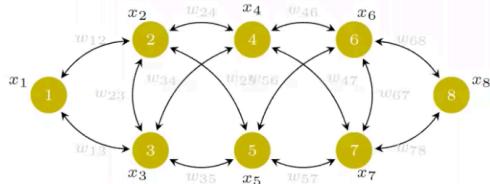
- Las convoluciones en grafos agregan información creciente desde lo local hasta lo global
⇒ Conceptualmente, hacen lo mismo que las convoluciones en el tiempo



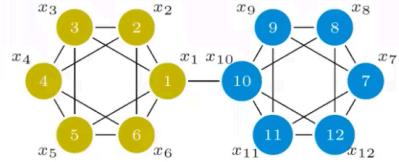
- Filtro con coeficientes $\mathbf{h} = \{h_k\}_{k=0}^{\infty} \Rightarrow \mathbf{z} = h_0 \mathbf{S}^0 \mathbf{x}$

- ▶ Las convoluciones en grafos agregan información creciente desde lo local hasta lo global
 ⇒ Conceptualmente, hacen lo mismo que las convoluciones en el tiempo

Filtro en un grafo



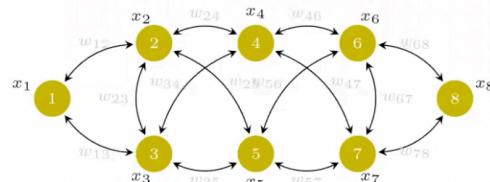
El mismo filtro en otro grafo



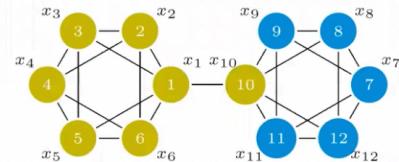
- ▶ Filtro con coeficientes $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ ⇒ $\mathbf{z} = h_0 \mathbf{S}^0 \mathbf{x} + h_1 \mathbf{S}^1 \mathbf{x} + h_2 \mathbf{S}^2 \mathbf{x}$

- ▶ Las convoluciones en grafos agregan información creciente desde lo local hasta lo global
 ⇒ Conceptualmente, hacen lo mismo que las convoluciones en el tiempo

Filtro en un grafo

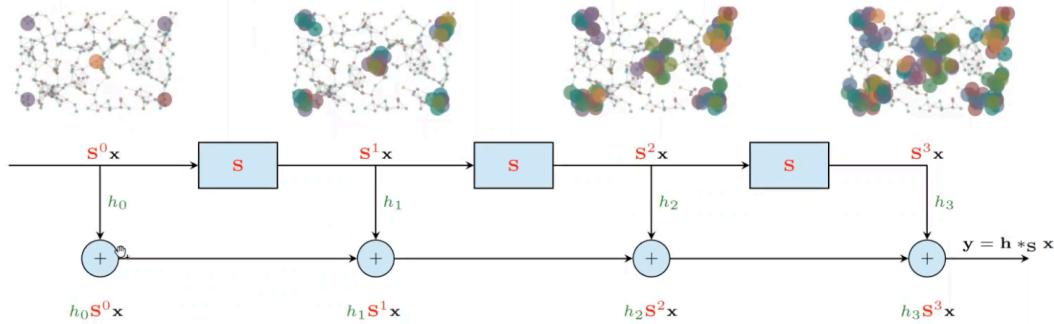


El mismo filtro en otro grafo



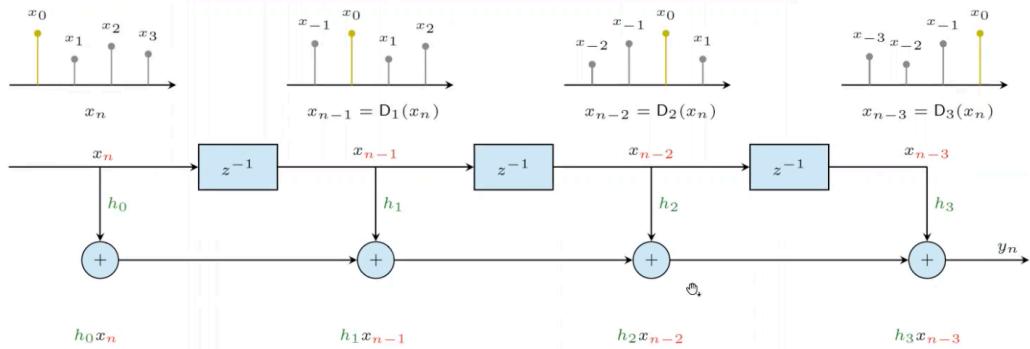
- ▶ Filtro con coeficientes $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ ⇒ $\mathbf{z} = h_0 \mathbf{S}^0 \mathbf{x} + h_1 \mathbf{S}^1 \mathbf{x} + h_2 \mathbf{S}^2 \mathbf{x} + h_3 \mathbf{S}^3 \mathbf{x} + \dots = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$

- ▶ Una convolución en el grafo es una combinación lineal de los elementos de la secuencia de difusión
- ▶ Podemos representar la convolución en el grafo como un diagrama en bloques



Convoluciones en el Tiempo

- ▶ Los **filtros convolucionales** procesan señales **temporales** explotando el **desplazamiento**



- ▶ La convolución **temporal** es una combinación lineal de **desplazamiento** $\Rightarrow y_n = \sum_{k=0}^{K-1} h_k x_{n-k}$

- Las señales temporales se pueden pensar como señales en grafos sobre un grafo de línea \mathbf{S}



- El desplazamiento temporal es equivalente a aplicar el ODG \mathbf{S} del grafo de línea

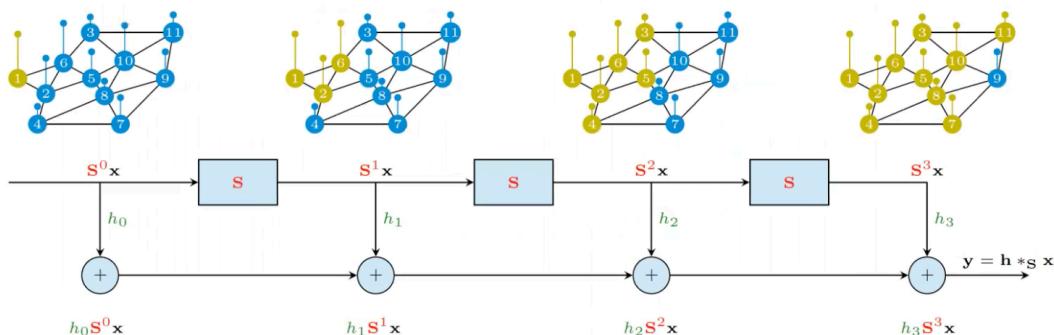
$$\mathbf{S}^3\mathbf{x} = \mathbf{S}(\mathbf{S}^2\mathbf{x}) = \begin{bmatrix} \dots & \vdots & \vdots & \vdots \\ \dots & 0 & 0 & 0 \\ \dots & 1 & 0 & 0 \\ \dots & 0 & 1 & 0 \\ \dots & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x_{-2} \\ x_1 \\ x_0 \\ x_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_{-3} \\ x_{-2} \\ x_{-1} \\ x_0 \\ \vdots \end{bmatrix}$$

- Los desplazamientos son potencias del operador desplazamiento aplicado a la señal
⇒ Por lo tanto, los filtros convolucionales son polinomios en \mathbf{S} ⇒ $D_k(\mathbf{x}) = \mathbf{S}^k\mathbf{x}$

La Convolución en Grafos

- Para un grafo arbitrario con ODG \mathbf{S} la convolución es una combinación lineal de desplazamientos

$$\mathbf{h} * \mathbf{x} = \sum_{k=0}^{K-1} h_k D_k(\mathbf{x}) = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$$



Vamos a la ultima parte, meter todo esto en la RN. Metemos la operacion desplazamiento en el perceptron.

- El objetivo, recordemos, es minimizar el riesgo empírico
- En el problema de minimización del riesgo empírico, tenemos:
 - ⇒ Un conjunto de muestras de entrenamiento $\mathcal{T} = \{\mathbf{x}\}$
 - ⇒ Una función de pérdida $J(\Phi(\mathbf{x}))$ para evaluar el algoritmo Φ
 - ⇒ Una clase de funciones Φ de donde tomamos el algoritmo de aprendizaje $\Phi \in \Phi$
- Encontrar un algoritmo $\Phi^* \in \Phi$ que minimiza la pérdida $J(\Phi(\mathbf{x}))$ promediado sobre las muestras

$$\Phi^* = \arg \min_{\Phi \in \Phi} \sum_{\mathbf{x} \in \mathcal{T}} J(\Phi(\mathbf{x}))$$

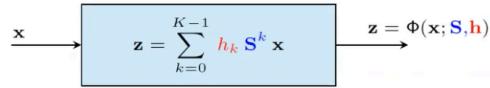
- Usamos $\Phi^*(\mathbf{x})$ para estimar la salida $\hat{y} = \Phi^*(\mathbf{x})$ cuando la entrada \mathbf{x} no pertenecen al entrenamiento

- En estos problemas, la clase de funciones Φ es elección del diseñador

$$\Phi^* = \arg \min_{\Phi \in \Phi} \sum_{\mathbf{x} \in \mathcal{T}} J(\Phi(\mathbf{x}))$$

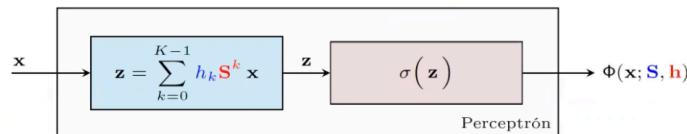
- Diseñar un algoritmo de aprendizaje automático ≡ encontrar la clase de funciones Φ adecuada
- Cuando trabajamos con señales en grafos, elegir filtros convolucionales en grafos parece una buena idea

- ▶ Asumimos que los datos \mathbf{x} son **señales en el grafo** estructuradas mediante un grafo con **ODG \mathbf{S}**
- ▶ La clase de funciones son **filtros en grafos de orden K** sobre el **ODG \mathbf{S}** $\Rightarrow \Phi(\mathbf{x}; \mathbf{S}, \mathbf{h}) = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$



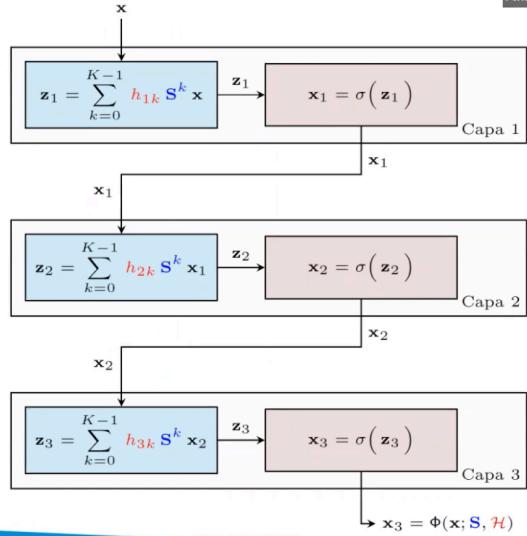
- ▶ Aprender minimizando el riesgo, restringidos a la clase de filtros $\Rightarrow \mathbf{h}^* = \arg \min_{\mathbf{h}} \sum_{\mathbf{x} \in \mathcal{T}} J(\Phi(\mathbf{x}; \mathbf{S}, \mathbf{h}))$
- ⇒ La optimización es sobre los coeficientes \mathbf{h} para el **ODG \mathbf{S}** dado

- ▶ Los filtros tienen **capacidad de representación limitada** \Rightarrow Sólo pueden aprender funciones lineales
- ▶ Elegir Φ como el conjunto de **perceptrones en el grafo** $\Rightarrow \Phi(\mathbf{x}) = \Phi(\mathbf{x}; \mathbf{S}, \mathbf{h}) = \sigma \left(\sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x} \right)$



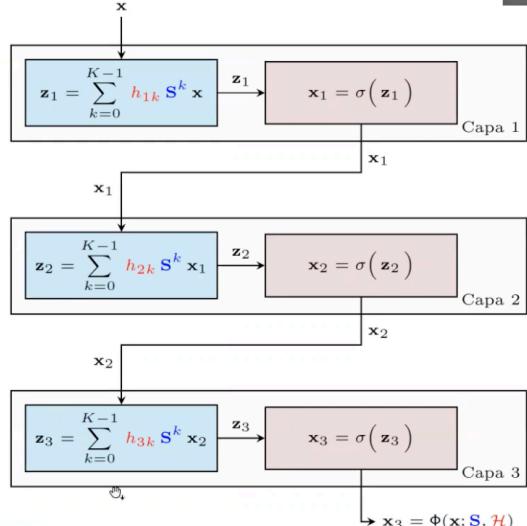
- ▶ El algoritmo óptimo va a ser un perceptrón $\Rightarrow \mathbf{h}^* = \arg \min_{\mathbf{h}} \sum_{\mathbf{x} \in \mathcal{T}} J(\Phi(\mathbf{x}; \mathbf{S}, \mathbf{h}))$
- ⇒ El perceptrón puede aprender relaciones no-lineales \Rightarrow Mayor poder de representación

- ▶ Incrementar la representación con una GNN
- ▶ Cascada de algunos perceptrones en el grafo
 - ⇒ La señal de entrada \mathbf{x} en la capa 1
 - ⇒ La salida de la capa 1 es la entrada de la capa 2
 - ⇒ La salida de la capa 2 es la entrada de la capa 3
- ▶ La última capa es la salida de la GNN ⇒ $\Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$
 - ⇒ Parametrizada por los coeficientes $\mathcal{H} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3]$



Aprendiendo con una Red Neuronal en el Grafo

- ▶ Aprender los filtros $\mathcal{H}^* = (\mathbf{h}_1^*, \mathbf{h}_2^*, \mathbf{h}_3^*)$ de la GNN
- $$\mathcal{H}^* = \arg \min_{\mathcal{H}} \sum_{\mathbf{x} \in \mathcal{T}} J(\Phi(\mathbf{x}; \mathbf{S}, \mathcal{H}))$$
- ▶ Optimizamos sobre el filtro ⇒ El grafo \mathbf{S} viene dado
 - ⇒ Captura información que explota la GNN

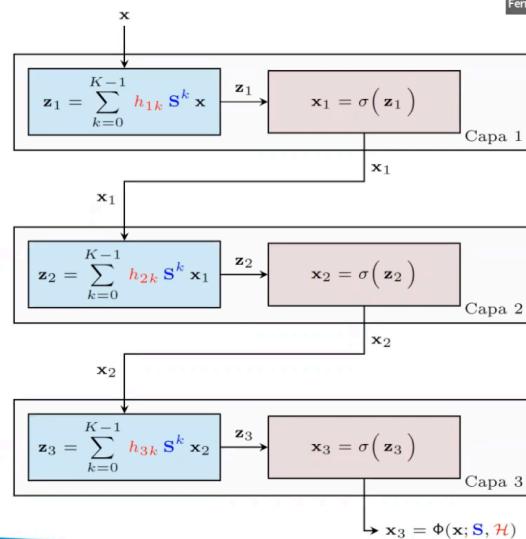


Transferencia de GNNs a Distintos Grafos

GNNs, Pt.
fgama@fi.uba.ar

Fernando

- ▶ La GNN depende del grafo \mathbf{S}
- ▶ Podemos pensar \mathbf{S} como un parámetro
 - ⇒ Captura información sobre la estructura
- ▶ O podemos pensar que \mathbf{S} es la entrada
 - ⇒ Permite transferir a diferentes grafos
 - ⇒ Se aprenden sólo los coeficientes \mathcal{H}^*



> Resumiendo el capítulo

Resumen del Capítulo

GNNs, Pt.
fgama@fi.uba.ar

Fernando

- ▶ La importancia de los datos con **estructuras descriptas por grafos**
 - ⇒ Requieren **tener en cuenta el grafo** para procesar de manera adecuada
- ▶ Señales en grafos, operador de desplazamiento en el grafo
 - ⇒ **Convolución en grafos** ⇒ Combinación lineal de señales en el vecindario
- ▶ **Redes neuronales en grafos** ⇒ Reemplazar la operación lineal por una convolución en el grafo
 - ⇒ **Explota la estructura** de los datos descripta por el grafo

- ▶ Extensión de GNNs a señales en grafos con **múltiples features**
⇒ Se utiliza un banco de filtros para la convolución en el grafo
- ▶ Pooling respetando al **estructura del grafo** ⇒ Resumen local, selección de nodos
- ▶ Sistemas de recomendación ⇒ Las similitudes de los puntajes son un grafo
- ▶ Atribución de autoría ⇒ Las palabras son nodos, y la co-ocurrencia son aristas
⇒ Atribuir textos cortos ⇒ Histogramas de palabras en **textos cortos**
- ▶ Controlar enjambres de robots ⇒ Aprendizaje por imitación, transferencia, escalabilidad

