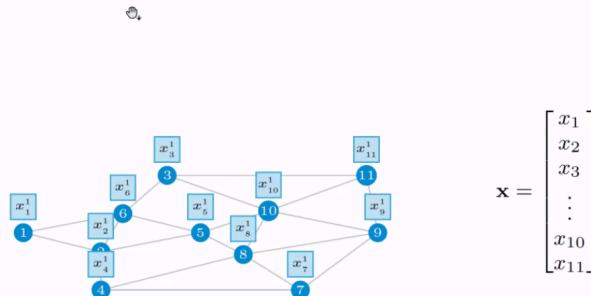


Clase 8 - GNNs 2

Múltiples Features

GNNs, Pt. 2
fgama@fi.uba.ar

- Señal en el grafo \Rightarrow Un sólo escalar asociado a cada nodo $\Rightarrow \mathbf{x} : \text{nodos} \rightarrow \mathbb{R}$

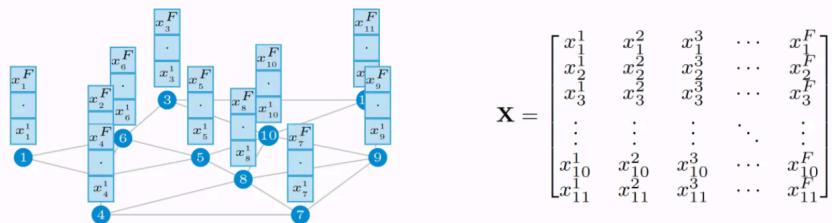


Pasamos a tener muchos features, muchos valores en cada uno de los nodos, en vez de tener un escalar, tenemos un vector.

Múltiples Features

GNNs, Pt. 2
fgama@fi.uba.ar

- Señal en el grafo \Rightarrow Un sólo escalar asociado a cada nodo $\Rightarrow \mathbf{x} : \text{nodos} \rightarrow \mathbb{R}$
- Incrementar la representación \Rightarrow Asignar un vector a cada nodo $\Rightarrow \mathbf{X} : \text{nodos} \rightarrow \mathbb{R}^F$
- Señales en grafos con múltiples features \Rightarrow Matriz \mathbf{X} de tamaño N (nodos) $\times F$ (features)
 - \Rightarrow La fila i recolecta los features en el nodo $i \Rightarrow$ Información local en el nodo i
 - \Rightarrow La columna f representa la feature f en todos los nodos $\Rightarrow \mathbf{x}^f$ es una señal en el grafo



$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^F \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^F \\ x_3^1 & x_3^2 & x_3^3 & \cdots & x_3^F \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{10}^1 & x_{10}^2 & x_{10}^3 & \cdots & x_{10}^F \\ x_{11}^1 & x_{11}^2 & x_{11}^3 & \cdots & x_{11}^F \end{bmatrix}$$

Cada columna corresponde a un feature, cada fila es un nodo. Cada columna corresponde a un feature para cada nodo :)

Cada fila pasa a ser una colección que tiene asociados cada uno de los nodos en el grafo. La señal pasa a ser descripta por una matriz NxF (nodos, features)

En el caso de los drones, esto puede ser por ejemplo, la posición relativa en x e y, la velocidad, la distancia entre robots, etc.

Pero... como extiendo al convolución para trabajar con matrices en vez de vectores? La idea es obtener una operación lineal usando la info local, de implementación distribuida

Extendiendo la Convolución

GNNs, Pt. 2
fgama@fi.uba.ar

- ▶ Convolución \Rightarrow Operación lineal, usando información local, de implementación distribuida

$$\mathbf{Y} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \mathbf{H} \quad \text{esctuctura}$$

- ▶ Multiplicación con **S** en la izquierda \Rightarrow Desplaza cada feature \mathbf{Sx}^f **mantiene grafo**
- ▶ Multiplicación con **H** en la derecha \Rightarrow Combinación lineal de features en cada nodo **estructura arbitraria**

$$\begin{bmatrix} 0 & s_{1,2} & 0 & \cdots & 0 \\ s_{2,1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & s_{3,11} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & s_{10,11} \\ 0 & 0 & s_{3,11} & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^F \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^F \\ x_3^1 & x_3^2 & x_3^3 & \cdots & x_3^F \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{10}^1 & x_{10}^2 & x_{10}^3 & \cdots & x_{10}^F \\ x_{11}^1 & x_{11}^2 & x_{11}^3 & \cdots & x_{11}^F \end{bmatrix} \begin{bmatrix} h^{11} & h^{12} & \cdots & h^{1G} \\ h^{21} & h^{22} & \cdots & h^{2G} \\ h^{31} & h^{32} & \cdots & h^{3G} \\ \vdots & \vdots & \ddots & \vdots \\ h^{F1} & h^{F2} & \cdots & h^{FG} \end{bmatrix}$$

A la izquierda mantiene la topología del grafo, a la derecha puede ser cualquier cosa

Puedo mapear una distinta cantidad de features de entrada con distintos tamaños de salida. La Y va a tener n nodos pero puede tener G features en vez de F. Pueden cambiar.

- ▶ Convolución \Rightarrow Operación lineal, usando información local, de implementación distribuida

$$\mathbf{Y} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X} \mathbf{H}_k$$

- ▶ Multiplicación con **S** en la izquierda \Rightarrow Desplaza cada feature \mathbf{Sx}^f
- ▶ Multiplicación con **H** en la derecha \Rightarrow Combinación lineal de features en cada nodo
- ▶ La convolución es equivalente a la aplicación de un banco de filtros en grafos
- ▶ Para cada feature \mathbf{x}^f aplicamos el filtro (\mathbf{f}, \mathbf{g}) para obtener \mathbf{x}^{fg} \Rightarrow Lineal, local, distribuido

$$\mathbf{x}^{fg} = \sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{x}^f h_k^{fg}$$

\Rightarrow Hay G filtros aplicados a cada entrada \mathbf{x}^f \Rightarrow Hay un total de $F \times G$ filtros (dim. de \mathbf{H}_k)

- ▶ Sumamos la salida del g -ésimo filtro para todas las F features $\Rightarrow \mathbf{y}^g = \sum_{f=1}^F \mathbf{x}^{fg} \Rightarrow \mathbf{Y} = [\mathbf{y}^1, \dots, \mathbf{y}^G]$

> La clase pasada hablamos definido las GNNs para una sola feature, teníamos:

- ▶ Redes neuronales en grafos para señales con una sola feature $\Rightarrow \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H}) = \mathbf{x}_L$

$$\mathbf{x}_\ell = \sigma \left(\sum_{k=0}^{K_\ell-1} \mathbf{S}^k \mathbf{x}_{\ell-1} \mathbf{h}_{\ell k} \right)$$

\mathbf{x}_ℓ señal en la capa $\ell \Rightarrow$ Dimensión $N \times 1$

$\mathbf{h}_{\ell k}$ k -ésimo coef. en la capa $\ell \Rightarrow$ Dim. $1 \times 1 \Rightarrow$ Aprender $\sum_\ell K_\ell$ coef. $\mathcal{H} = \{\mathbf{h}_{\ell k}\}$

S operador desplazamiento (dado en el problema) \Rightarrow Dimensión $N \times N$

- ▶ Hiperparámetros (elecciones del diseñador) \Rightarrow Afectan el número de parámetros a aprender
 - $\Rightarrow L$: cantidad de capas
 - $\Rightarrow K_\ell$: cantidad de coeficientes del filtro en cada capa

Ahora, para múltiples features, reemplazamos la función de convolución por la que vimos en las lides anteriores:

- Redes neuronales en grafos para señales con múltiples features $\Rightarrow \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H}) = \mathbf{X}_L$

$$\mathbf{X}_\ell = \sigma \left(\sum_{k=0}^{K_\ell-1} \mathbf{S}^k \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k} \right)$$

\mathbf{X}_ℓ señal en la capa $\ell \Rightarrow$ Dimensión $N \times F_\ell$

$\mathbf{H}_{\ell k}$ k -ésimo coef. en la capa $\ell \Rightarrow$ Dim. $F_{\ell-1} \times F_\ell \Rightarrow$ Aprender $\sum_\ell K_\ell F_\ell F_{\ell-1}$ coef. $\mathcal{H} = \{\mathbf{H}_{\ell k}\}$

\mathbf{S} operador desplazamiento (dado en el problema) \Rightarrow Dimensión $N \times N$

- Hiperparámetros (elecciones del diseñador) \Rightarrow Afectan el número de parámetros a aprender

$\Rightarrow L$: cantidad de capas

$\Rightarrow K_\ell$: cantidad de coeficientes del filtro en cada capa

$\Rightarrow F_\ell$: cantidad de features en cada capa

Ahora, en cada capa, la señal va a tener dimensión N , pero los features de salida van variando por campo. LA cantidad de coeficientes que uno aprende sigue siendo independiente del tamaño del grafo.

> Por si nos toca trabajar con GNNs, pinta saber que hay varias "arquitecturas" según como elijo mi operador desplazamiento y mi filtro H :

- Redes neuronales en grafos para señales con múltiples features $\Rightarrow \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H}) = \mathbf{X}_L$

$$\mathbf{X}_\ell = \sigma \left(\sum_{k=0}^{K_\ell-1} \mathbf{S}^k \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k} \right)$$

\mathbf{X}_ℓ señal en la capa $\ell \Rightarrow$ Dimensión $N \times F_\ell$

$\mathbf{H}_{\ell k}$ k -ésimo coef. en la capa $\ell \Rightarrow$ Dim. $F_{\ell-1} \times F_\ell \Rightarrow$ Aprender $\sum_\ell K_\ell F_\ell F_{\ell-1}$ coef. $\mathcal{H} = \{\mathbf{H}_{\ell k}\}$

\mathbf{S} operador desplazamiento (dado en el problema) \Rightarrow Dimensión $N \times N$

- Spectral GCNNs: $K_\ell = N - 1$, $\mathbf{S} = \mathbf{L}$ normalizada [Bruna et al, '14] (distintos autovalores)

- ChebNets: $\mathbf{S} = \mathbf{L}$ normalizada, $\mathbf{H}_{\ell k}$ = polinomio de Chebyshev [Defferrard et al, '16]

- Diffusion CNNs: $\mathbf{S} = \mathbf{A}$, $L = 1$, $F_1 = NF_0$, \mathbf{H}_{1k} = único valor para toda la fila [Atwood et al, '16]

- GCNs: $\mathbf{S} = (\mathbf{I} + \mathbf{A})$ normalizada, $K_\ell = 2$, $\mathbf{H}_{\ell 0} = \mathbf{0}$ [Kipf et al, '17]

- SGCs: $\mathbf{S} = (\mathbf{I} + \mathbf{A})$ normalizada, $\mathbf{H}_{\ell k} = \mathbf{0}$ para todo $k < K_\ell$ [Wu et al, '19]

- GINs: $\mathbf{S} = \mathbf{A}$ binaria, $K_\ell = 1$, $\mathbf{H}_{\ell 0} = (1 + \varepsilon_\ell) \mathbf{H}_{\ell 1}$, usar $K_\ell = 0$ para algunos ℓ [Xu et al, '19]

> Pooling

Tenemos el mismo problema que con imágenes, empezamos a tener cada vez mas features capa a capa, crece el costo computacional

- Incrementar la cant. de features en cada capa \Rightarrow Incrementa la dim. de las señales
 \Rightarrow Incrementa el costo computacional de la operación de convolución

- Pooling \mathcal{P} \Rightarrow Construir resúmenes regionales de la información en cada capa

$$\mathbf{x}_\ell = \mathcal{P}_\ell \left\{ \sigma \left[\sum_{k=0}^{K_\ell-1} \mathbf{s}^k \mathbf{x}_{\ell-1} \mathbf{H}_{\ell k} \right] \right\}$$

\Rightarrow Quedarse únicamente con algunos resúmenes $\Rightarrow \mathbf{X}_\ell$ de dim. $N_\ell \times F_\ell$ con $N_\ell \leq N_{\ell-1} \leq N$

El pooling en este caso es poder construir resúmenes regionales en cada una de las capas, un determinado vecindario respecto de cada uno de los nodos.

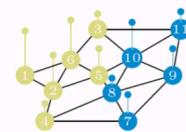
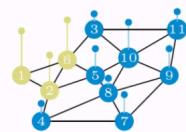
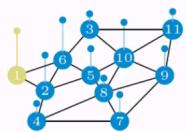
- Pooling \Rightarrow Función de resumen local seguida de un muestreo

$$\mathbf{z}_\ell = \mathcal{P}_\ell \left\{ \mathbf{z}_{\ell-1} \right\} = \mathbf{C}_\ell \rho \left[\mathbf{z}_{\ell-1}; \mathbf{S} \right]$$

$\Rightarrow \rho[\cdot; \mathbf{S}]$ función de resumen \Rightarrow Por ejemplo $[\rho[\mathbf{z}; \mathbf{S}]]_i = \max\{\mathbf{z}_i, [\mathbf{S}\mathbf{z}]_i, \dots, [\mathbf{S}^{K-1}\mathbf{z}]_i\}$

$\Rightarrow \mathbf{C}_\ell$ binaria $N_\ell \times N_{\ell-1}$ matriz de selección $\Rightarrow [\mathbf{C}_\ell]_{ij} \in \{0, 1\}$, $\mathbf{C}\mathbf{1} = \mathbf{1}$, $\mathbf{C}^\top \mathbf{1} \preceq \mathbf{1}$

⊕₄



Una de las maneras de hacer es agarrar cada uno de los nodos y por ejemplo agarrando resúmenes de tamaño dos, ir preguntando recurrentemente nodo a nodo. Tomas por ejemplo para el nodo 1 el máximo o el promedio de la región que lo rodea de tamaño 2 que elegiste.

- Pooling \Rightarrow Función de resumen local seguida de un muestreo

$$\mathbf{z}_\ell = \rho_{\ell} \{ \mathbf{z}_{\ell-1} \} = \mathbf{C}_\ell \rho \left[\mathbf{z}_{\ell-1}; \mathbf{S} \right]$$

$\Rightarrow \rho[\cdot; \mathbf{S}]$ función de resumen \Rightarrow Por ejemplo $[\rho[\mathbf{z}; \mathbf{S}]]_i = \max\{[\mathbf{z}]_i, [\mathbf{S}\mathbf{z}]_i, \dots, [\mathbf{S}^{K-1}\mathbf{z}]_i\}$

$\Rightarrow \mathbf{C}_\ell$ binaria $N_\ell \times N_{\ell-1}$ matriz de selección $\Rightarrow [\mathbf{C}_\ell]_{ij} \in \{0, 1\}$, $\mathbf{C}\mathbf{1} = \mathbf{1}$, $\mathbf{C}^\top \mathbf{1} \leq \mathbf{1}$



Una vez que determinaste el resumen, vamos a elegir alguno de los nodos por algún criterio de influencia por ejemplo y te quedas con esos nodos que van actuar como representantes.

Pero que pasa cuando quiero volver a hacer mi operación de convolución? Mi matriz S de conv tiene NxN y ahora achique ese tamaño, tengo un problema de mis match.

- Convolución en el grafo \Rightarrow Operador desplazamiento tiene la dimensión del grafo $N \times N$

$$\mathbf{C}_{\ell-1} \mathbf{Y}_\ell = \sum_{k=0}^{K-1} \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k}$$

- Luego del pooling (muestreo) la dimensión de la señal $\mathbf{X}_{\ell-1}$ se redujo $N_{\ell-1} \leq N$
 \Rightarrow Desajuste de dimensiones entre S de dim. $N \times N$ y $\mathbf{X}_{\ell-1}$ de dim. $N_{\ell-1} \times F_{\ell-1}$
- Rellenar con ceros \Rightarrow Usar la matriz de selección $\mathbf{C}_{\ell-1}$ para adaptar las dimensiones
 $\Rightarrow \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1}$ tiene dim. $N \times F_{\ell-1}$ donde los nodos no-muestreados ahora valen cero
- La salida \mathbf{Y}_ℓ tiene dim. N \Rightarrow Tiene que ser muestreada para tener dim. $N_{\ell-1}$ (la misma que \mathbf{X}_ℓ)
 $\Rightarrow \mathbf{C}_{\ell-1} \mathbf{Y}_{\ell-1}$ muestrea la salida en los mismos nodos que $\mathbf{X}_{\ell-1}$ $\Rightarrow \mathbf{C}_{\ell-1} \mathbf{Y}_{\ell-1}$ tiene dim. $N_{\ell-1} \times F_\ell$

- Una de las alternativas que tengo es achicar mi matriz S. Esto es conveniente cuando voy a resolver esta operación de manera entera dentro de una computadora. Cuando mi grafo es un grafo abstracto. Si puedo achicar el grafo, lo puedo hacer sin problema.

- Pooling \Rightarrow Reduce el costo computacional de la operación de convolución

$$\mathbf{C}_{\ell-1} \mathbf{Y}_\ell = \sum_{k=0}^{K-1} \mathbf{S}_{\ell-1}^{(k)} \quad \mathbf{X}_{\ell-1} \quad \mathbf{H}_{\ell k}$$

- Computación centralizada $\Rightarrow \mathbf{S}_{\ell-1}^{(k)} = \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top$ reducir el k -ésimo ODG \Rightarrow Dim. $N_{\ell-1} \times N_{\ell-1}$
 \Rightarrow Las operaciones son ahora en dim. $N_{\ell-1} \leq N \Rightarrow$ La matriz $\mathbf{S}_{\ell-1}^{(k)}$ se obtiene antes de entrenar

- Si en cambio tengo una red física, como los drones, no puedo alterar o cambiar el grafo de manera arbitraria. Necesito poder resolver el desajuste sin alterar el tamaño de mi grafo. No puedo apagar robots de manera arbitraria. Lo que se hace es mirar la parte en azul:

- Pooling \Rightarrow Reduce el costo computacional de la operación de convolución

$$\mathbf{C}_{\ell-1} \mathbf{Y}_\ell = \sum_{k=0}^{K-1} \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1} \quad \mathbf{H}_{\ell k}$$

- Computación centralizada $\Rightarrow \mathbf{S}_{\ell-1}^{(k)} = \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top$ reducir el k -ésimo ODG \Rightarrow Dim. $N_{\ell-1} \times N_{\ell-1}$
 \Rightarrow Las operaciones son ahora en dim. $N_{\ell-1} \leq N \Rightarrow$ La matriz $\mathbf{S}_{\ell-1}^{(k)}$ se obtiene antes de entrenar
- Computación decentralizada $\Rightarrow \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1}$ es una señal con ceros en los nodos no muestreados
 \Rightarrow Multiplicación del ODG con la señal $\mathbf{S}^k(\mathbf{C}_{\ell-1}^\top \mathbf{X}) \Rightarrow$ Se hace de manera local y distribuida

- Se le ponen ceros en todos los nodos que NO elegí.

- ▶ Pooling \Rightarrow Reduce el costo computacional de la operación de convolución

$$\mathbf{C}_{\ell-1} \mathbf{Y}_\ell = \sum_{k=0}^{K-1} \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1} \mathbf{H}_{\ell k}$$

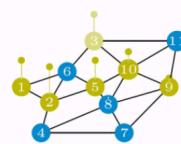
- ▶ Computación centralizada $\Rightarrow \mathbf{S}_{\ell-1}^{(k)} = \mathbf{C}_{\ell-1} \mathbf{S}^k \mathbf{C}_{\ell-1}^\top$ reducir el k -ésimo ODG \Rightarrow Dim. $N_{\ell-1} \times N_{\ell-1}$
 \Rightarrow Las operaciones son ahora en dim. $N_{\ell-1} \leq N$ \Rightarrow La matriz $\mathbf{S}_{\ell-1}^{(k)}$ se obtiene antes de entrenar
- ▶ Computación decentralizada $\Rightarrow \mathbf{C}_{\ell-1}^\top \mathbf{X}_{\ell-1}$ es una señal con ceros en los nodos no muestreados
 \Rightarrow Multiplicación del ODG con la señal $\mathbf{S}^k(\mathbf{C}_{\ell-1}^\top \mathbf{X})$ \Rightarrow Se hace de manera local y distribuida
- ▶ Es una técnica de pooling que reduce el costo computacional y mantiene la estructura del grafo

En general, pooling no se usa porque las operaciones suelen quedar distribuidas en los nodos que ya tengo. El costo computacional sigue dividido entre los N nodos, como se divide el costo no es tan grande.

En grafos prácticamente nunca se usa pooling.

> Ejemplos: Sistemas de recomendación

- ▶ Un conjunto de productos se puntuán por un conjunto de usuarios
 \Rightarrow Algunos pares de productos fueron puntuados por varios usuarios
 \Rightarrow Verosimilitud de dos productos con puntajes similares
 \Rightarrow Estimar la correlación de Pearson entre productos
 \Rightarrow Los productos son los nodos
 \Rightarrow Las similitudes de puntajes son las aristas
- ▶ Un usuario puntuá algunos de los productos \Rightarrow Señal en el grafo



Queremos saber el puntaje que un usuario le daría a un producto específico

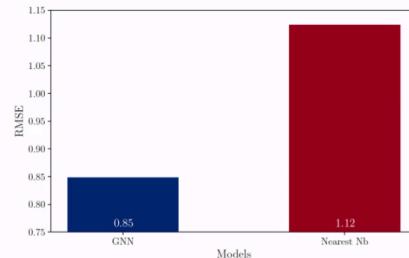
En este ejemplo, un usuario puntuó a 2, 5, 10 y 9, el resto no los puntuó, quiero estimar el puntaje que le daría al producto N3. Dado el grafo que tenemos construido por correlación de Pearson.

- Queremos saber el **puntaje x_t** que un **usuario** le daría a un producto específico $t \Rightarrow$ Estimar $[\mathbf{x}]_t = x_t$
 - \Rightarrow Basado en el puntaje que el usuario le dio a otros productos \Rightarrow Señal en el grafo \mathbf{x}
 - \Rightarrow Explotando las similitudes en los puntajes entre productos \Rightarrow Estructura del grafo \mathbf{S}
- Elemento $[\mathbf{x}]_i = 0$ si el producto i no fue puntuado. $[\mathbf{S}]_{ij}$ correlación Pearson
- Entrenar una GNN $\Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$ para mapear los puntajes \mathbf{x} del usuario y el puntaje x_t del producto t
 - \Rightarrow Seleccionar el cant. de capas L , cant. de features F_ℓ , cant. de coeficientes K_ℓ
 - \Rightarrow La última capa tiene $K_L = 1$ y $F_L = 1 \Rightarrow$ Nos quedamos con el valor $[\mathbf{x}_L]_t = x_t$ en el nodo t

Con el dataset movielens publico:

- **MovieLens-100k** \Rightarrow 100,000 puntajes dados por 943 usuarios a 1,582 películas (puntajes de 1 a 5)
 - \Rightarrow Cada **película** es un **nodo** del grafo \Rightarrow 1,582 nodos
- La **película** es *Star Wars* \Rightarrow 583 usuarios puntuaron la película \Rightarrow Señales
 - \Rightarrow El puntaje de la película específica x_t es extraído como etiqueta $y = x_t \Rightarrow$ Datos $\{(\mathbf{x}, y)\}$
- Usar 90 % de muestras para el entrenamiento y 10 % para la validación
- **Construir el grafo de los datos** \Rightarrow Las aristas son correlaciones \Rightarrow 10 vecinos más cercanos
- Hiperparámetros $\Rightarrow L = 2, F_1 = 64, F_2 = 32, K_1 = K_2 = K = 5$, ReLU como activación de función
- Usar el operador desplazamiento \mathbf{S} obtenido en el dataset \Rightarrow Normalizarlo
- Entrenamos la arquitectura con 40 epochs y 5 de tamaño de batch

- ▶ Resultados obtenidos con las muestras de evaluación ⇒ Comparación con el estándar

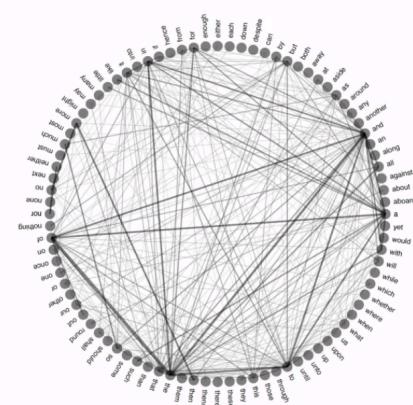


> Problemas de autoría

Todo esto viene de los lingüistas. Hay determinadas palabras como preposiciones o artículos que NO tienen contenido semántico. Parece ser que la manera en la que aparezcan estas palabras es muy particular de cada uno de los autores. Lo que se hace es tomar una ventana para un autor y contar cuantas veces las palabras aparecieron juntas dentro de una ventana de N palabras. Si aparece mucho, la relación es mucho mas grande que si aparece poco.

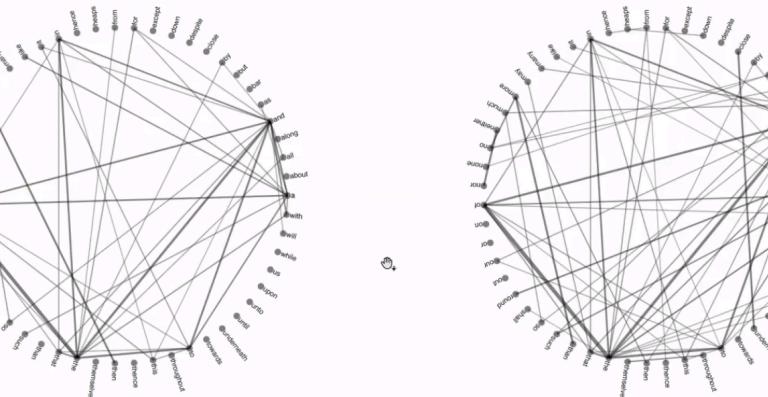
Red de Adyacencia de Palabras (*Word Adjacency Networks*)

- ▶ Palabras funcionales ⇒ Sin contenido semántico
- ▶ Su uso depende de la gramática del lenguaje
- ▶ Diferentes autores usan la gramática de manera distinta
- ▶ Se captura con una **red de palabras adyacentes (WAN)**
⇒ Qué tan seguido pares de palabras aparecen juntas



Resulta que puedo construir esto para varios autores y cada uno va a tener un grafo muy particular propio.. podría usar el grafo para determinar si un texto lo escribió un autor o no

- Las WANs de Shakespeare y Marlowe son distintas ⇒ Confirmar su colaboración en Henry VI



El problema que tiene este metodo es que construir esos grafos requiere de textos sumamente largos:

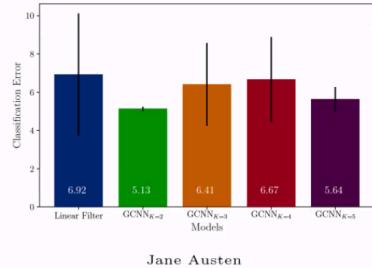
Atribución de Autoría como Señales en el Grafo

- Construir WANs que reflejen el estilo del autor requiere textos largos
⇒ Atribuir textos pequeños es difícil incluso si contamos con una WAN precisa
- La WAN determina un grafo S ⇒ Los nodos son palabras, las aristas son co-ocurrencias
- La señal en el grafo x está definida sobre los nodos ⇒ Histograma de las palabras (los nodos)
- Usar GNNs para determinar el autor de un texto corto
⇒ Explotar una WAN dada S y el histograma x del texto corto

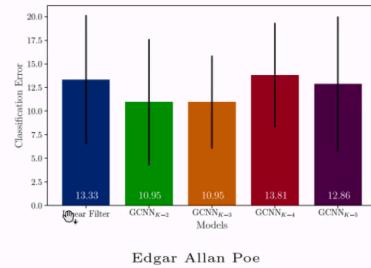
Necesito un montón de palabras para que la ventana me de un grafo lo suficientemente preciso para poder comparar autores.

La GNN va a aprender a mirar: si una palabra aparece mucho pero una palabra que tiene una arista muy fuerte aparece poco, entonces probablemente este texto NO es de este autor. Uno puede usar la red que aprendí con textos largos para usar en textos cortos mediante el uso de histogramas

- ▶ Corpus de textos escritos por Jane Austen y Edgar Allan Poe \Rightarrow Construir WAN S
- ▶ Tomar páginas (1K palabras) escritas por J. Austen o E. A. Poe de un combinado de 22 autores
 \Rightarrow Calcular la frecuencia de palabras funcionales en cada página \Rightarrow Señales en el grafo x
- ▶ Comparar GNNs con distinto valor de K , comparar con un filtro lineal en el grafo



Jane Austen



Edgar Allan Poe

La diferencia con el problema de recomendación (problema de interpolación) es que acá estas rapiendo toda la matriz a un 0 o 1, es un problema de clasificación.

> Robótica

El objetivo es que el enjambre de robots logre coordinar sus velocidades para que todos vuelen juntos sin chocar unos con otros. Si tengo un controlador centralizado desde tierra, esto es una boludez de resolver. Le digo a cada uno la velocidad y listo. El problema es si quiero resolver de forma descentralizada, ahí cada robot solo se puede comunicar con sus vecinos.

- ▶ Queremos que el enjambre **coordine** sus **velocidades** individuales sin chocar



- ▶ Este problema es **fácil** si permitimos coordinación centralizada $\Rightarrow \mathbf{u}_i = \sum_{i=1}^N \mathbf{v}_i$
- ▶ Pero es **muy difícil** si no permitimos la centralización $\Rightarrow \mathbf{u}_i = \dots$

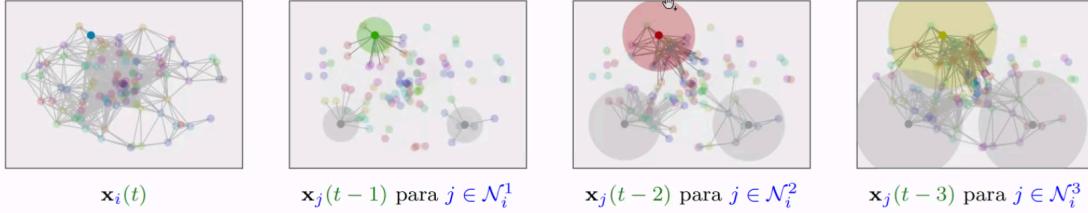
Tenemos una estructura parcial de la información. Tenemos una diferencia de tiempos en la disponibilidad de info. Un Delay. Cuando me llega info capaz ya es vieja y si tomo decisiones en base a eso puede ser equivocada. Tenemos un

objetivo global como equipo pero tenemos que tomar decisiones separados.

Estructura de la Información en Sistemas Distribuidos

GNNs, Pt. 2
fgama@fi.uba.ar

- El desafío de obtener controladores en un sistema distribuido es la **estructura parcial de información**



- El nodo i tiene acceso a su información **local a tiempo t** $\Rightarrow \mathbf{x}_i(t)$
- Y la información de los vecinos a **1 salto a tiempo $t-1$** $\Rightarrow \mathbf{x}_j(t-1)$ para todo $j \in \mathcal{N}_i^1$
- Y la información de los vecinos a **2 saltos a tiempo $t-2$** $\Rightarrow \mathbf{x}_j(t-2)$ para todo $j \in \mathcal{N}_i^2$
- Y la información de los vecinos a **3 saltos a tiempo $t-3$** $\Rightarrow \mathbf{x}_j(t-3)$ para todo $j \in \mathcal{N}_i^3$

La Dinámica del Enjambre

GNNs, Pt. 2
fgama@fi.uba.ar



- Un enjambre de N robots con posiciones $\mathbf{r}_i(t)$, velocidades $\mathbf{v}_i(t)$ y **aceleraciones $\mathbf{u}_i(t)$**
 - Dinámica del sistema \Rightarrow **Aceleración constante** durante cada **intervalo T_s**
- $$\begin{cases} \mathbf{r}_i(t+1) = \mathbf{u}_i(t)T_s^2/2 + \mathbf{v}_i(t)T_s + \mathbf{r}_i(t) \\ \mathbf{v}_i(t+1) = \mathbf{u}_i(t)T_s + \mathbf{v}_i(t) \end{cases}$$

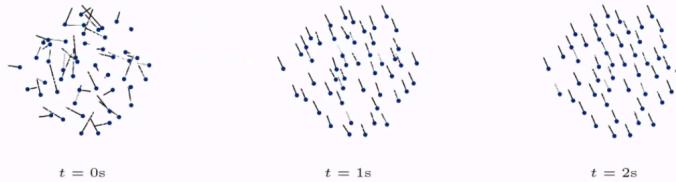
- **Controlamos la aceleración $\mathbf{u}_i(t)$** \Rightarrow Diseñar o aprender una secuencia de valores $\{\mathbf{u}_i(t)\}$

> Variación de la velocidad: voy a comprar la velocidad e cada robot a tiempo t con el promedio de velocidades de todos los robots a tiempo T . Si es baja la diferencia, están coordinados. El objetivo es encontrar el valor de aceleración para poder minimizar esta diferencia.

- ▶ Coordinar las **velocidades** $\mathbf{v}_i(t)$ de todos los robots para que sean la misma evitando choques
- ▶ El costo se mide calculando la **variación de la velocidad** del equipo para toda la trayectoria

$$J[\{\mathbf{v}_i(t)\}] = \frac{1}{N} \sum_t \sum_{i=1}^N \|\mathbf{v}_i(t) - \bar{\mathbf{v}}_j(t)\|^2, \quad \bar{\mathbf{v}}_j = \frac{1}{N} \sum_{j=1}^N \mathbf{v}_j(t)$$

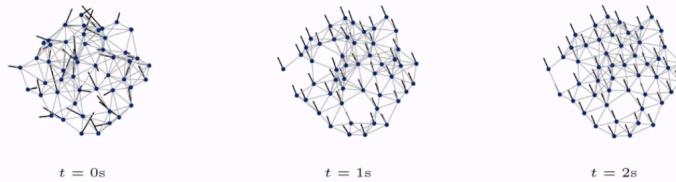
- ▶ Controlar la **aceleración** $\mathbf{u}_i(t)$ tal que $\min_{\mathbf{u}_i(t), t \geq 0} J[\{\mathbf{v}_i(t)\}] \Rightarrow$ Velocidad $\mathbf{v}_i(t+1) = \mathbf{u}_i(t)T_s + \mathbf{v}_i(t)$



Hay un agregado de complejidad que es que tengo el problema distribuido, solo se comunican si estando dentro de un radio de comunicación R . Es lo que voy a usar para armar mi grafo de comunicaciones

- ▶ **Problema distribuido** \Rightarrow Los robots se comunican sólo si $\|\mathbf{r}_i(t) - \mathbf{r}_j(t)\| \leq R$
- ▶ **Grafo de comunicaciones** $\mathbf{S}(t)$ \Rightarrow La arista $[\mathbf{S}(t)]_{ij} = 1$ si los robots se comunican
- ▶ Usar GNNs que respeten el **grafo de comunicación** \Rightarrow Operaciones locales y distribuidas

$$\mathbf{U}(\mathcal{X}_i(t), \mathcal{H}) = \Phi(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H}), \quad \mathbf{H}(\mathcal{X}_i(t)) = \sum_{k=0}^{K-1} h_k \mathbf{S}(t) \cdots \mathbf{S}(t-(k-1)) \mathbf{x}(t-k)$$



> Aprendizaje por imitación

A la hora de entrenar vamos a presumir que existe un controlador experto u estrella que hace exactamente lo que hay que hacer. Lo uso de referencia para imitar su comportamiento

- Entrenar una GNN $\Phi(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H})$ por imitación \Rightarrow Encontrar \mathcal{H} tal que Φ imita $\mathbf{U}^*(t)$

$$\mathcal{H}^* = \arg \min_{\mathcal{H}} \mathbb{E}_{\mathbf{U}^*} \left[\|\Phi(\mathbf{X}(t); \mathbf{S}(t), \mathcal{H}) - \mathbf{U}^*(t)\| \right]$$

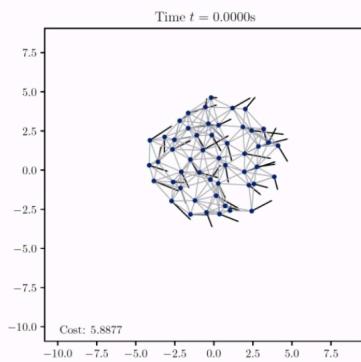
- El controlador experto $\mathbf{U}^*(t)$ que evita colisiones viene dado por

$$\mathbf{u}_i^*(t) = - \sum_{j=1}^N (\mathbf{v}_i(t) - \mathbf{v}_j(t)) - \underbrace{\sum_{j=1}^N \nabla_{\mathbf{r}_i(t)} P(\mathbf{r}_i(t), \mathbf{r}_j(t))}_{\text{collision avoidance}}$$

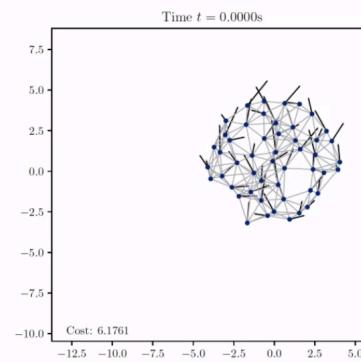
- El estado $\mathbf{x}_i(t)$ de cada robot viene dado por $\nabla_{\mathbf{r}_i(t)} P(\mathbf{r}_i(t), \mathbf{r}_j(t))$ \Rightarrow Se calcula localmente
- El controlador experto \mathbf{U}^* se necesita sólo durante el entrenamiento **pero no** durante la evaluación

> Transferibilidad

A la izquierda tengo el óptimo, el controlador experto. A la derecha tengo lo que aprendí durante la evaluación. Lo primero que observo es que si bien los grafos son parecidos, son distintos. Es fundamental que las GNNs tengan cierto grado de transferencia entre grafos, porque no es de esperar que durante el entrenamiento vea todas las posibles combinaciones de grafos.

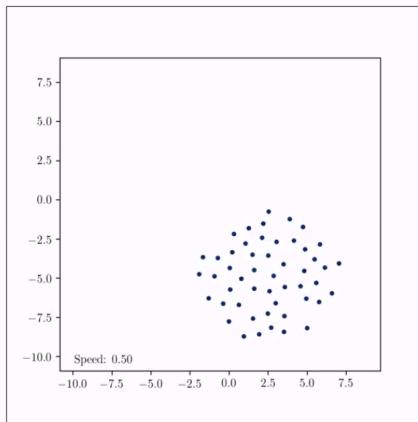


Trayectoria óptima entrenamiento

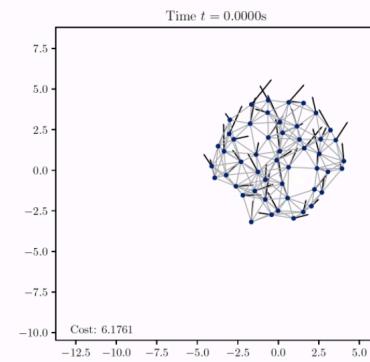


Trayectoria aprendida evaluación

Vemos en el video q tarda un poco mas la GNN pero logra aprender a controlar los robóticos.

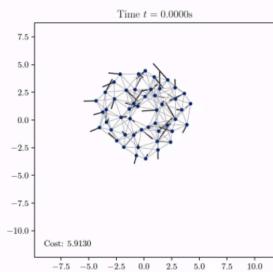


Trayectoria óptima entrenamiento

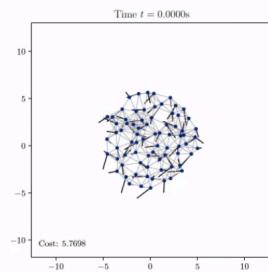


Trayectoria aprendida evaluación

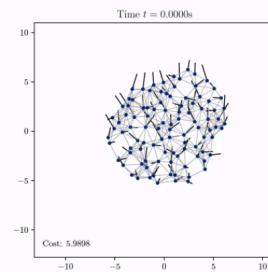
Como pega en la escalabilidad el tamaño de robots.. vamos a ver:



50 robots.



75 robots.

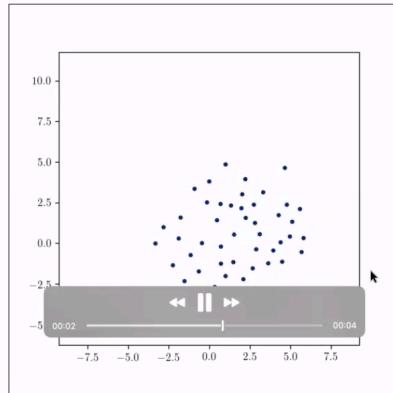


100 robots.

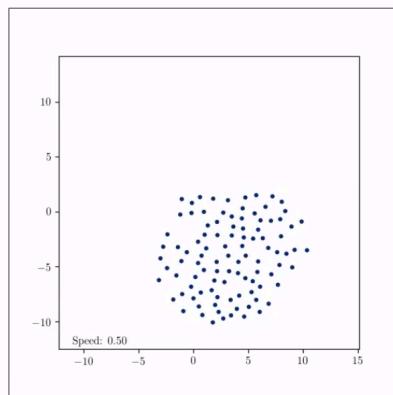
Logro aprender sin problemas, ya que la cantidad de parámetros es independiente de la cantidad de robots.

Un poco mas interesante es ver que pasa si entreno en una cantidad y la uso en otra cantidad mayor..

- Entrenar en 50 robots. Evaluar en 50 robots.



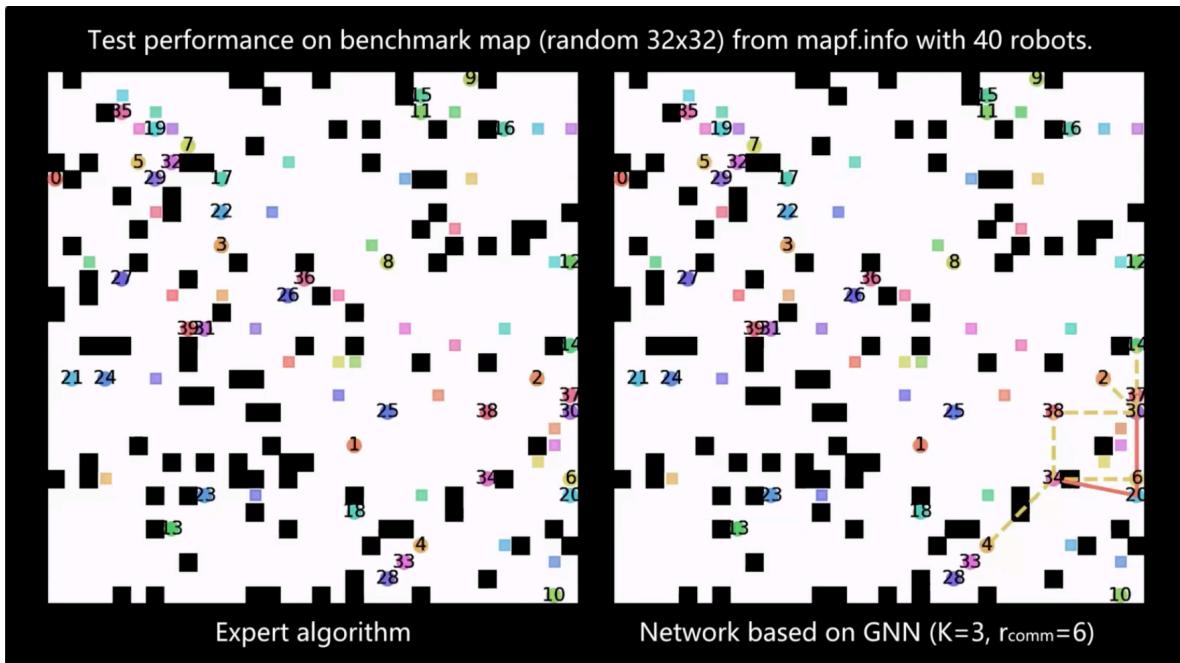
- Entrenar en 50 robots. Evaluar en 100 robots.



Vemos que funciona :). Podemos aprender en poca cantidad de robots y extrapolarlo a mayor cantidad.

Otros Ejemplos:

El objetivo es que cada uno llegue lo mas rápido posible a su destino, evitando su destino, y que evite chocarse con otros robots. A la izquierda vemos como se resuelve de manera experta y a la derecha con GNNs



Fer cuenta que laburo en cambrige con un equipo y lo pusieron a andar en vida real.

> Resumen del capítulo:

Resumen del Capítulo

GNNs, Pt. 2
fgama@fi.uba.ar

- ▶ Extensión de GNNs a señales en grafos con **múltiples features**
⇒ Se utiliza un banco de filtros para la convolución en el grafo
- ▶ Pooling respetando la 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

> Fin del curso :(

- ▶ Aprendizaje automático ⇒ Determinar un **conjunto de operaciones aceptables**
⇒ Usar los **datos** para elegir la mejor ⇒ Optimización, estadística, regularización
- ▶ Redes neuronales ⇒ Conjunto de operaciones aceptables en una gran cantidad de problemas
⇒ Una **transformación** lineal, seguida de una **función de activación no lineal**
- ▶ Redes neuronales **convolucionales** ⇒ Procesar **datos contiguos** ⇒ Imágenes
- ▶ Redes neuronales **recurrentes** ⇒ Procesar **secuencias de datos** ⇒ Lenguaje
- ▶ Redes neuronales en **grafos** ⇒ Procesar grafos ⇒ Redes físicas, redes abstractas

Elegir operaciones adecuadas que exploten la estructura de los datos