FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Aprendizaje Automático para Datos en Grafos

Graph Neural Networks - Parte II

Federico 'Larroca' La Rocca
Muy basado en transparencias de **Fernando Gama**

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

Octubre 2022

Graph Neural Networks: ¿Porqué?

- Tenemos intuición sobre convoluciones temporales. De las graph convolutions no tanto.

  $\Rightarrow$ Información local, implementación eficiente (distribuida)

- Cuando la intuición falla tenemos que recurrir a propiedades matemáticas $\Rightarrow$ ¿Qué sabemos sobre CNNs?
- CNNs también aprovechan la información local y tienen una implementación eficiente

  $\Rightarrow$ Equivariancia (equivariance) a traslaciones y estabilidad [Mallat '12] $\Rightarrow$ explican su gran desempeño

  Equivarianza $\neq$ Invarianza

- Permutation equivariance $\Rightarrow$ Aprovechar las simetrías internas del grafo

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

Permutation Equivariance y Reordenamiento de Nodos

- Una permutación $\mathbf{P}$ es una matriz binaria que cumple

$$\{\mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P1} = \mathbf{1}, \mathbf{P}^{\mathsf{T}}\mathbf{1} = \mathbf{1}\}$$

O sea, un único 1 por fila y columna. Permite definir reordenamientos de índices $\sigma(i) = j$. Ejemplo:

$$\mathbf{P}^{\mathsf{T}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{x}$ reordena las entradas del vector (tomar $\mathbf{P}^{\mathsf{T}}$ como un GSO)

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_1 \end{pmatrix} = \begin{pmatrix} x_{\sigma(1)} \\ x_{\sigma(2)} \\ x_{\sigma(3)} \end{pmatrix}$$

- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{S}$ reordena las filas de la matriz (pensar en $\mathbf{S}$ como N columnas concatenadas)
- El producto $\mathbf{SP}$ reordena las columnas de la matriz con el mismo orden que $\mathbf{P}^{\mathsf{T}}$

$\Rightarrow$ El producto $\mathbf{P}^{\mathsf{T}}\mathbf{SP}$ reordena las entradas de la matriz

Permutation Equivariance y Reordenamiento de Nodos

- Una permutación $\mathbf{P}$ es una matriz binaria que cumple

$$\{\mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P1} = \mathbf{1}, \mathbf{P}^{\mathsf{T}}\mathbf{1} = \mathbf{1}\}$$

O sea, un único 1 por fila y columna. Permite definir reordenamientos de índices $\sigma(i) = j$. Ejemplo:

$$\mathbf{P}^{\mathsf{T}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{x}$ reordena las entradas del vector (tomar $\mathbf{P}^{\mathsf{T}}$ como un GSO)
- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{S}$ reordena las filas de la matriz (pensar en $\mathbf{S}$ como N columnas concatenadas)

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \mathbf{x}_3^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_2^{\mathsf{T}} \\ \mathbf{x}_3^{\mathsf{T}} \\ \mathbf{x}_1^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{\sigma(1)}^{\mathsf{T}} \\ \mathbf{x}_{\sigma(2)}^{\mathsf{T}} \\ \mathbf{x}_{\sigma(3)}^{\mathsf{T}} \end{pmatrix}$$

- El producto $\mathbf{SP}$ reordena las columnas de la matriz con el mismo orden que $\mathbf{P}^{\mathsf{T}}$

$\Rightarrow$ El producto $\mathbf{P}^{\mathsf{T}}\mathbf{SP}$ reordena las entradas de la matriz

Permutation Equivariance y Reordenamiento de Nodos

- Una permutación $\mathbf{P}$ es una matriz binaria que cumple

$$\{\mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P1} = \mathbf{1}, \mathbf{P}^{\mathsf{T}}\mathbf{1} = \mathbf{1}\}$$

O sea, un único 1 por fila y columna. Permite definir reordenamientos de índices $\sigma(i) = j$. Ejemplo:

$$\mathbf{P}^{\mathsf{T}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{x}$ reordena las entradas del vector (tomar $\mathbf{P}^{\mathsf{T}}$ como un GSO)
- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{S}$ reordena las filas de la matriz (pensar en $\mathbf{S}$ como N columnas concatenadas)
- El producto $\mathbf{SP}$ reordena las columnas de la matriz con el mismo orden que $\mathbf{P}^{\mathsf{T}}$

$$\begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{\sigma(1)} & \mathbf{x}_{\sigma(2)} & \mathbf{x}_{\sigma(3)} \end{pmatrix}$$

$\Rightarrow$ El producto $\mathbf{P}^{\mathsf{T}}\mathbf{SP}$ reordena las entradas de la matriz

Permutation Equivariance y Reordenamiento de Nodos

- Una permutación $\mathbf{P}$ es una matriz binaria que cumple

$$\{\mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P1} = \mathbf{1}, \mathbf{P}^{\mathsf{T}}\mathbf{1} = \mathbf{1}\}$$

O sea, un único 1 por fila y columna. Permite definir reordenamientos de índices $\sigma(i) = j$. Ejemplo:

$$\mathbf{P}^{\mathsf{T}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{x}$ reordena las entradas del vector (tomar $\mathbf{P}^{\mathsf{T}}$ como un GSO)
- El producto $\mathbf{P}^{\mathsf{T}}\mathbf{S}$ reordena las filas de la matriz (pensar en $\mathbf{S}$ como N columnas concatenadas)
- El producto $\mathbf{SP}$ reordena las columnas de la matriz con el mismo orden que $\mathbf{P}^{\mathsf{T}}$

$\Rightarrow$ El producto $\mathbf{P}^{\mathsf{T}}\mathbf{SP}$ reordena las entradas de la matriz

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} x_{22} & x_{23} & x_{21} \\ x_{32} & x_{33} & x_{31} \\ x_{12} & x_{13} & x_{11} \end{pmatrix} = \begin{pmatrix} x_{\sigma(1)\sigma(1)} & x_{\sigma(1)\sigma(2)} & x_{\sigma(1)\sigma(3)} \\ x_{\sigma(2)\sigma(1)} & x_{\sigma(2)\sigma(2)} & x_{\sigma(2)\sigma(3)} \\ x_{\sigma(3)\sigma(1)} & x_{\sigma(3)\sigma(2)} & x_{\sigma(3)\sigma(3)} \end{pmatrix}$$

Permutation Equivariance y Reordenamiento de Nodos

- Una matriz de permutación es ortonormal ¿Cuánto es la identidad reordenada?

$$\mathbf{P}^{\mathsf{T}}\mathbf{I}\mathbf{P} = \mathbf{I} = \mathbf{P}^{\mathsf{T}}\mathbf{P}$$

$\Rightarrow$ Podemos reordenar primero y operar después u operar y luego reordenar $(\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P})(\mathbf{P}^{\mathsf{T}}\mathbf{x}) = \mathbf{P}^{\mathsf{T}}(\mathbf{S}\mathbf{x})$

- La permutación es equivalente a reordenar los nodos del grafo
- Elegir un GSO $\mathbf{S}$ para describir un grafo fuerza un ordenamiento de los nodos
  - $\Rightarrow$ Este ordenamiento es necesario para operar. Pero es arbitrario
- Queremos algoritmos de procesamiento de señales que sean independientes de ordenamientos arbitrarios

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

Permutation Equivariance

- Consideremos la convolución en grafos $\mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \, \mathbf{x}$
- Depende de los parámetros del filtro $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ y del shift operator $\mathbf{S}$; aplicado a la señal de entrada $\mathbf{x}$

Teorema (Gama, Bruna, Ribeiro)

Las convoluciones en grafos son equivariantes a permutaciones. Para un grafo con shift operator $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ *y una señal en el grafo* $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ *permutados se cumple que*

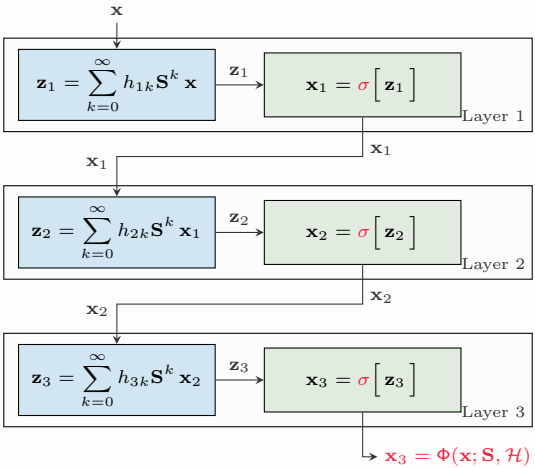
$$\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{x}$$

Prueba $\Rightarrow$ $\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, \hat{\mathbf{S}}^k \hat{\mathbf{x}} = \sum_{k=0}^{\infty} h_k \, (\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P})^k \mathbf{P}^{\mathsf{T}}\mathbf{x} = \mathbf{P}^{\mathsf{T}} \left(\sum_{k=0}^{\infty} h_k \, \mathbf{S}^k \mathbf{x} \right) = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})\mathbf{x}$

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

Las GNNs heredan la Permutation Equivariance de los grafos

- Una GNN es una composición de capas
  $\Rightarrow$ Filtros en grafos y no-linealidades punto-a-punto
- Una operación punto-a-punto no mezcla los valores en los nodos
  $\Rightarrow$ Independiente del grafo
- La GNN conserva la permutation equivariance



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

Permutation Equivariance en las GNNs

Teorema (Gama, Bruna, Ribeiro)

Las GNNs son equivariantes a permutaciones. Para un grafo con shift operator $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ *y una señal en el grafo* $\hat{\mathbf{x}} = \mathbf{P}^{\mathsf{T}}\mathbf{x}$ *permutados se cumple que*

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

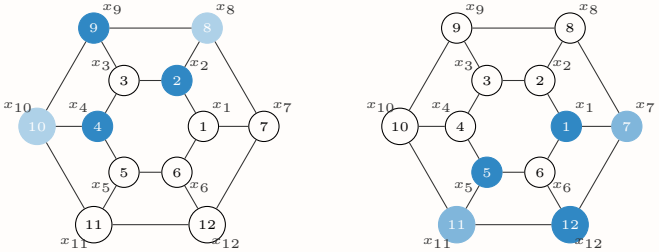
donde $\Phi(\hat{\mathbf{x}}; \hat{\mathbf{S}}, \mathcal{H})$ *es la salida de procesar* $\hat{\mathbf{x}}$ *en* $\hat{\mathbf{S}}$ *con la GNN* $\mathcal{H}$ *y* $\Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$ *es la salida de procesar* $\mathbf{x}$ *con* $\mathbf{S}$ *usando la misma GNN* $\mathcal{H}$.

- El procesamiento de señales usando GNNs es independiente del orden y el etiquetado

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

La Equivarianza a Permutaciones es muy Valiosa

- La invarianza al re-etiquetado de los nodos permite a las GNNs aprovechar la simetría de la señal



- Aunque diferentes, ambas señales con permutaciones una de la otra

  $\Rightarrow$ Permutation equivariance $\Rightarrow$ La GNN puede aprender a predecir la situación de la derecha

- Permutation Equivariance no es siempre una buena idea
  - Edge-Variant GNNs
  - Feature Augmentation
  - Attention

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

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

You, Gomes-Selman, Ying, Leskovec, "Identity-aware Graph Neural Networks", IEEE AAAI 2021

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

Cómo Medir Distancias Módulo Permutaciones

- Consideremos un grafo con GSO $\mathbf{S}$ y otro grafo con GSO $\hat{\mathbf{S}}$ (mismo tamaño N)
- Tenemos los coeficientes $\{h_k\}$ y los filtros $\mathbf{H}(\mathbf{S})$ y $\mathbf{H}(\hat{\mathbf{S}})$ que usan los mismos taps sobre los dos GSOs
- Queremos caracterizar qué tan distintos son $\mathbf{H}(\mathbf{S})$ y $\mathbf{H}(\hat{\mathbf{S}})$
- Sabemos que cuando $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$, entonces $\mathbf{H}(\hat{\mathbf{S}}) = \mathbf{P}^{\mathsf{T}}\mathbf{H}(\mathbf{S})$ para todas las permutaciones $\mathbf{P} \in \mathcal{P}$

  $\Rightarrow$ Buscamos una distancia entre los operadores $\mathbf{H}(\mathbf{S})$ y $\mathbf{H}(\hat{\mathbf{S}})$ que sea módulo permutaciones

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = \min_{\mathbf{P}\in\mathcal{P}} \|\mathbf{H}(\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}) - \mathbf{H}(\hat{\mathbf{S}})\| = \min_{\mathbf{P}\in\mathcal{P}} \max_{\mathbf{x}:\|\mathbf{x}\|=1} \|\mathbf{H}(\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P})\mathbf{x} - \mathbf{H}(\hat{\mathbf{S}})\mathbf{x}\|$$

$\Rightarrow$ Si $\hat{\mathbf{S}} = \mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}$ entonces $\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = 0$. Otra forma de interpretar la equivarianza a permutaciones

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

Perturbaciones

- Podemos medir distancias entre filtros en grafos módulo permutaciones $\Rightarrow \|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}}$
- Quisiéramos caracterizarla como una función de alguna distancia entre $\mathbf{S}$ y $\hat{\mathbf{S}}$ (los mismos taps)

  $\Rightarrow$ Queremos saber cómo cambiar el soporte del grafo afecta la salida del filtro
- Medir la distancia entre $\mathbf{S}$ y $\hat{\mathbf{S}}$ $\Rightarrow$ Conjunto de matrices de error absoluto módulo permutaciones

$$\mathcal{E}(\mathbf{S}, \hat{\mathbf{S}}) = \{\mathbf{E} \in \mathbb{R}^{N \times N} : \mathbf{P}^{\top} \hat{\mathbf{S}} \mathbf{P} = \mathbf{S} + \mathbf{E} \ , \ \mathbf{P} \in \mathcal{P}\}$$

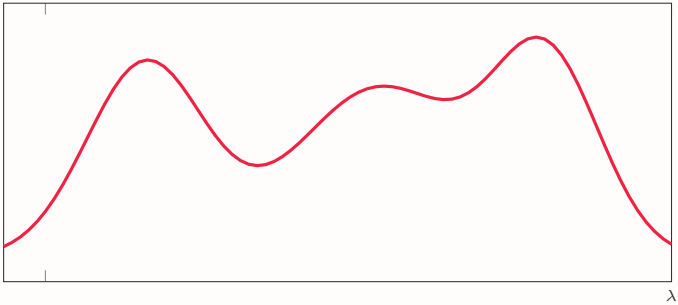
  $\Rightarrow$ Definamos la distancia de perturbaciones absolutas como $\mathsf{d}(\mathbf{S}, \hat{\mathbf{S}}) = \min_{\mathbf{E} \in \mathcal{E}(\mathbf{S}, \hat{\mathbf{S}})} \|\mathbf{E}\| \leq \|\hat{\mathbf{S}} - \mathbf{S}\|$
- El efecto del GSO en la salida del graph filter $\Rightarrow$ Graph frequency

$$\tilde{y}_i = \tilde{\mathsf{h}}(\lambda_i)\tilde{x}_i$$

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

Restricción en los Filtros: Lipschitz

- La respuesta en frecuencia $\tilde{\mathsf{h}}(\lambda)$ del filtro $\mathbf{H}$ cumple $|\mathsf{h}'(\lambda)| \leq C$

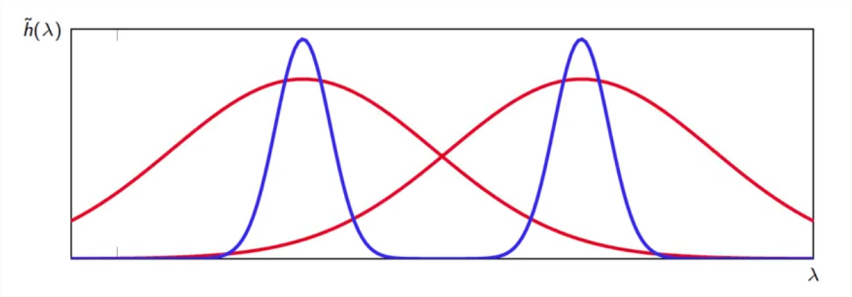


- Efecto de la constante de Lipschitz $\Rightarrow$ C Pequeño / Grande = Baja / Alta discrimabilidad

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

Restricción en los Filtros: Lipschitz

- La respuesta en frecuencia $\tilde{\mathsf{h}}(\lambda)$ del filtro $\mathbf{H}$ cumple $|\mathsf{h}'(\lambda)| \leq C$



- Efecto de la constante de Lipschitz $\Rightarrow$ C Pequeño / Grande = Baja / Alta discrimabilidad

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

Estabilidad de Filtros en grafos a Perturbaciones Absolutas

Teorema (Gama, Bruna, Ribeiro)

Las convoluciones en grafos son estables a perturbaciones absolutas. Dados dos grafos con GSOs $\mathbf{S}$ *y* $\hat{\mathbf{S}}$ *respectivamente tal que su distancia de perturbaciones absolutas es* $\mathbf{d}(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$*, entonces los filtros Lipschitz con constante* C *cumplen*

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \leq C\ (1 + \delta\sqrt{N})\ \varepsilon + \mathcal{O}(\varepsilon^2)$$

donde $\delta = (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1 \leq 8$ *es la eigenvector misalignment constant entre los vectores propios* $\mathbf{V}$ *y* $\mathbf{U}$ *de* $\mathbf{S}$ *y de la matriz de error* $\mathbf{E}$ *respectivamente.*

- La diferencia en la salida está acotada linealmente por la distancia entre los GSOs
- La cota depende del filtro y el modelo de perturbación
- La cota es universal para todos los grafos con la misma cantidad de nodos N

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

Limitaciones del Modelo de Perturbación Absoluta

- Aunque el teorema sea correcto y cierto para todos los grafos, no es muy útil

  $\Rightarrow$ El modelo de perturbación absoluto es demasiado arbitrario $\Rightarrow$ No mide cambios estructurales en el grafo

- Perturbaciones absolutas relacionan $\hat{\mathbf{S}}$ con $\mathbf{S}$ mediante $\mathbf{P}^{\top}\hat{\mathbf{S}}\mathbf{P} = \mathbf{S} + \mathbf{E}$ $\Rightarrow$ Matriz de error $\mathbf{E}$ arbitraria

$$\mathbf{P}^{\top}\begin{bmatrix} \\ \hat{\mathbf{S}} \\ \\ \end{bmatrix}\mathbf{P} = \begin{bmatrix} \\ \mathbf{S} \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \mathbf{E} \\ \\ \end{bmatrix}$$

  $\Rightarrow$ Podemos alterar la topología del grafo de forma arbitraria sin cambiar el valor de $\|\mathbf{E}\|$

- Necesitamos un mejor modelo de perturbación $\Rightarrow$ Uno que tome en cuenta el grafo a ser perturbado

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

Perturbaciones Relativas

- Vamos a seguir midiendo la distancia entre filtros ódulo permutaciones

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = \min_{\mathbf{P}\in\mathcal{P}} \|\mathbf{H}(\mathbf{P}^{\mathsf{T}}\mathbf{S}\mathbf{P}) - \mathbf{H}(\hat{\mathbf{S}})\|$$

- Pero ahora mediremos la distancia entre $\mathbf{S}$ y $\hat{\mathbf{S}}$ usando matrices con errores relativos módulo permutaciones

$$\mathcal{E}(\mathbf{S}, \hat{\mathbf{S}}) = \{\mathbf{E} \in \mathbb{R}^{N\times N} : \mathbf{P}^{\mathsf{T}}\hat{\mathbf{S}}\mathbf{P} = \mathbf{S} + \mathbf{E}^{\mathsf{T}}\mathbf{S} + \mathbf{S}\mathbf{E} \ , \ \mathbf{P} \in \mathcal{P}\}$$

$\Rightarrow$ Definamos la distancia de perturbaciones relativas como $\mathsf{d}(\mathbf{S}, \hat{\mathbf{S}}) = \min_{\mathbf{E}\in\mathcal{E}(\mathbf{S},\hat{\mathbf{S}})} \|\mathbf{E}\| \leq \dfrac{\|\hat{\mathbf{S}} - \mathbf{S}\|}{\|\mathbf{S}\|}$

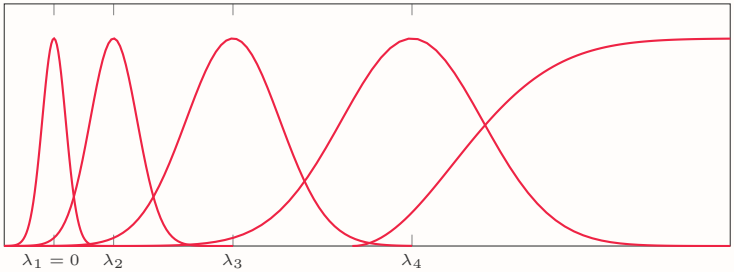
- El efecto del GSO en la salida del grafo la vemos en el dominio de la frecuencia

$$\tilde{y}_i = \tilde{\mathsf{h}}(\lambda_i)\tilde{x}_i$$

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

Restricción en los Filtros: Integral Lipschitz

- La respuesta en frecuencia $\mathsf{h}(\lambda)$ del filtro $\mathbf{H}$ cumple $|\lambda \mathsf{h}'(\lambda)| \leq C \Rightarrow$ Lipschitz con constante decreciente en λ



- Los filtros Integral Lipschitz son anchos para valores altos de λ
- Pueden ser arbitrariamente selectivos a bajos λ

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

Estabilidad de Filtros en Grafos a Perturbaciones Relativas

Teorema (Gama, Bruna, Ribeiro)

Las convoluciones en grafos son estables a perturbaciones relativas. Dados dos grafos con GSOs $\mathbf{S}$ *y* $\hat{\mathbf{S}}$ *respectivamente tal que su distancia de perturbaciones relativas es* $\mathsf{d}(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$*, entonces los filtros integral Lipschitz con constante* C *cumplen*

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} \leq 2C\ (1 + \delta\sqrt{N})\ \varepsilon + \mathsf{O}(\varepsilon^2)$$

donde $\delta = (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1 \leq 8$ *es la eigenvector misalignment constant entre los vectores propios* $\mathbf{V}$ *y* $\mathbf{U}$ *de* $\mathbf{S}$ *y de la matriz de error* $\mathbf{E}$ *respectivamente.*

- La diferencia en la salida está acotado linealmente por la distancia entre GSOs
- La cota depende del filtro y el modelo de perturbación
- La cota es universal para todos los grafos con el mismo número de nodos N

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

Estabilidad de GNNs a Perturbaciones Relativas

Teorema (Gama, Bruna, Ribeiro)

Las GNNs son estables ante perturbaciones relativas. Sean $\mathbf{S}$ *y* $\hat{\mathbf{S}}$ *los GSO de dos grafos tales que su distancia relativa es* $\mathsf{d}(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$. *Sea* $\mathbf{\Phi}(\cdot; \mathbf{S}, \mathcal{H})$ *una GNN con* L *capas y* F *features por capa y con filtros integral Lipschitz de constante* C. *Entonces, se cumple que*

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

donde $\delta = (\|\mathbf{U} - \mathbf{V}\|_2 + 1)^2 - 1 \leq 8$ *es la eigenvector misalignment constant entre los vectores propios* $\mathbf{V}$ *y* $\mathbf{U}$ *de* $\mathbf{S}$ *y de la matriz de error* $\mathbf{E}$ *respectivamente.*

- La diferencia en la salida está acotado linealmente por la distancia entre los GSOs
- La cota depende del filtro, la arquitectura elegida y el modelo de perturbación
- La cota es universal para todos los grafos con el mismo número de nodos N

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

¿Graph Filters o GNNs?

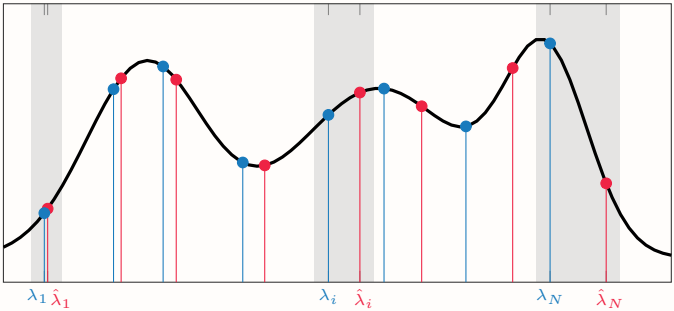
- Permutation equivariance y estabilidad son propiedades de las convoluciones en grafos y las GNNs
- ¿Porqué elegir GNNs en vez de simplemente grafos?
  - $\Rightarrow$ Q1: ¿Qué aportan las no-linealidades punto-a-punto?
  - $\Rightarrow$ Q2: ¿Qué tienen de malo las convoluciones lineales?
- A2: Pueden ser inestables a perturbaciones del grafo si son demasiado discriminativos (selectivos)
- A1: Hacen que las GNNs sean estables a perturbaciones y a la vez conserven la selectividad
- Esto se puede ver en el **dominio espectral**

$$\tilde{y}_i = \tilde{\mathsf{h}}(\lambda_i)\tilde{x}_i$$

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

Demostración del Teorema de Estabilidad

- El teorema de estabilidad de GNNs se prueba de forma elemental para el caso de dilatación de aristas $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- En este caso, se produce una dilatación espectral $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$

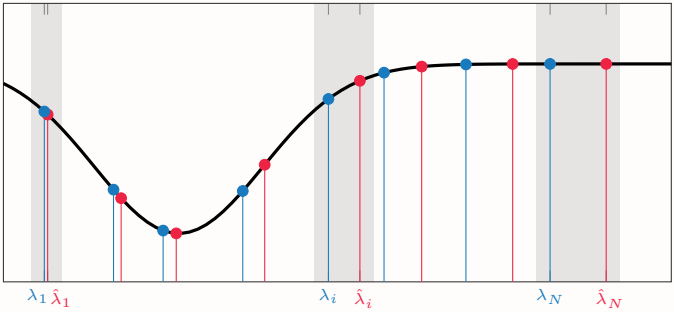


- Pequeñas deformaciones pueden resultar en grandes variaciones a la salida del filtro para valores altos de λ si no es integral Lipschitz

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

Demostración del Teorema de Estabilidad

- El teorema de estabilidad de GNNs se prueba de forma elemental para el caso de dilatación de aristas $\Rightarrow \hat{\mathbf{S}} = (1+\varepsilon)\mathbf{S}$
- En este caso, se produce una dilatación espectral $\Rightarrow \hat{\lambda}_i = (1+\varepsilon)\lambda_i$, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$

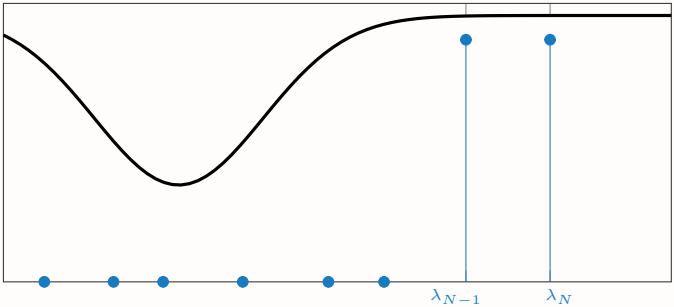


- Integral Lipschitz siempre es estable $\Rightarrow$ los valores propios no se mueven o el filtro no cambia

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

Banco de Filtros Selectivos son Inestables

- Q2: ¿Qué tienen de malo las convoluciones lineales?
- No pueden ser a la vez estables a deformaciones y discriminar features a frecuencias altas

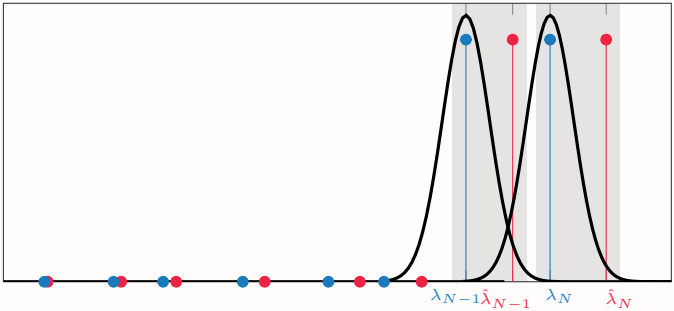


- Limita su utilidad en problemas de aprendizaje automático cuando los features importantes están en frecuencias (valores propios) altos

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

Banco de Filtros Selectivos son Inestables

- Q2: ¿Qué tienen de malo las convoluciones lineales?
- No pueden ser a la vez estables a deformaciones y discriminar features a frecuencias altas

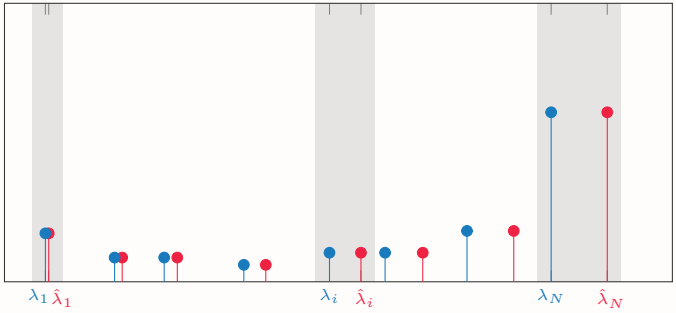


- Limita su utilidad en problemas de aprendizaje automático cuando los features importantes están en frecuencias (valores propios) altos

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

Las No-Linealidades Cambian el Soporte del Espectro

- Q1: ¿Qué aportan las no-linealidades punto-a-punto?
- Siguen siendo permutation equivariant pero a la vez generan componentes en otras frecuencias (bajas)
  $\Rightarrow$ Que podemos discriminar con filtros estables



Spectrum of rectified graph signal

$$\mathbf{x}_{\text{relu}} = \text{máx}(\mathbf{x}, 0)$$

- La no-linealidad demodula. Crea nuevos componentes en en frecuencias bajas

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

Las No-Linealidades Cambian el Soporte del Espectro

- Q1: ¿Qué aportan las no-linealidades punto-a-punto?
- Siguen siendo permutation equivariant pero a la vez generan componentes en otras frecuencias (bajas)
  $\Rightarrow$ Que podemos discriminar con filtros estables

Las GNNs son arquitecturas de procesamiento de información **estables** y **selectivas**

- La no-linealidad demodula. Crea nuevos componentes en en frecuencias bajas

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