



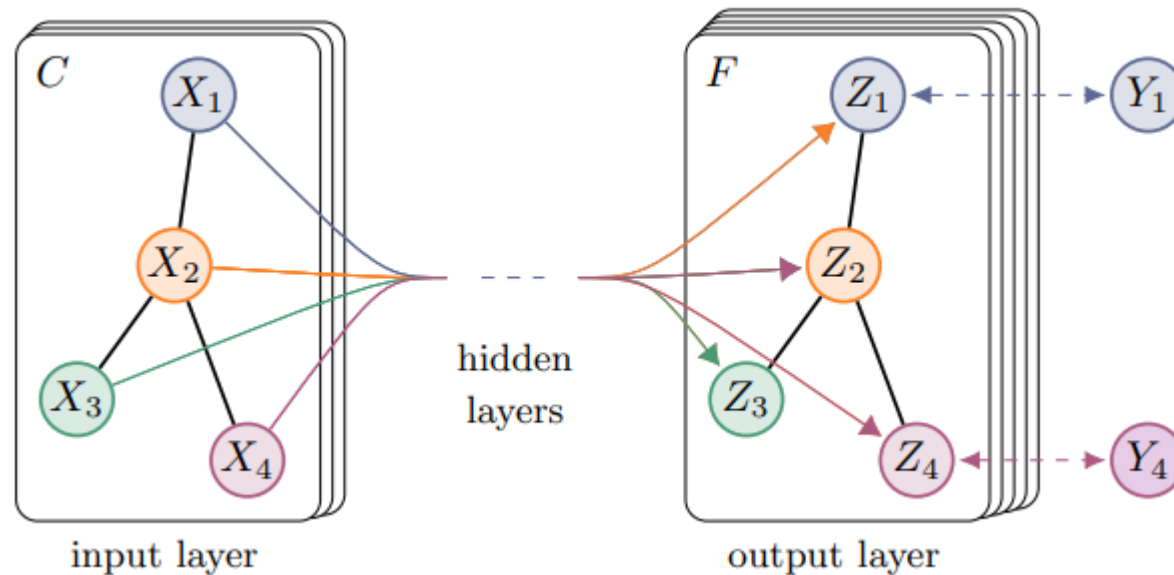
IIC-3641 Aprendizaje Automático basado en Grafos

<https://github.com/marcelomendoza/IIC3641>

- RECAPITULACIÓN -

GNN: Una arquitectura diseñada para grafos

Observamos que los nodos del grafo para los cuales tenemos etiquetas se consideran la partición de entrenamiento. Los otros nodos son parte del entrenamiento pero ingresan sin etiqueta ya que son de testing. Este tipo de entrenamiento se denomina **transductivo**.



Es decir, el framework de GNN define un escenario de entrenamiento semi-supervisado, útil para varias tareas como por ejemplo clasificación de nodos (AF5).

Graph Convolutional Networks y Spectral Graph Convolutions

Las GCN lo que hacen es una simplificación de la operación de convolución, de manera que se pueda realizar directamente sobre la matriz de adyacencia, lo cual la hace eficiente:

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

De hecho la versión definitiva propuesta por Kipf y Welling es aún más simple:

$$g_{\theta} \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x$$

Lo cual se puede reescribir como:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \quad \begin{array}{l} \nearrow \tilde{A} = A + I_N \\ \longrightarrow \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \end{array}$$

con $\Theta \in \mathbb{R}^{C \times F}$ y $Z \in \mathbb{R}^{N \times F}$

Una red de dos capas queda bastante sencilla:



$$Z = f(X, A) = \text{softmax} \left(\hat{A} \text{ReLU} \left(\hat{A} X W^{(0)} \right) W^{(1)} \right)$$

$\nearrow \hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$

- GRAPH ATTENTION NETWORKS -

Graph Attention Networks

Las GCN determinan la importancia de un vecino j en relación con un nodo target i en base a su peso en el enlace A_{ij} (normalizado por sus grados).

Podríamos **parametrizar** estos pesos, de manera que la red los aprenda. Las Graph Attention Networks (GAT) se basan en esta idea y aprenden la importancia de cada nodo en base al mecanismo de atención (Bahdanau et al., 2015).

Graph Attention Layer. Se define un mecanismo de self-attention en los nodos, los cuales miden los coeficientes de atención de todo par de nodos a través de una capa de atención compartida:

$$e_{ij} = a(WH_i^{k-1}, WH_j^{k-1}) \quad \begin{aligned} W &\in \mathbb{R}^{F \times F'} \\ a: \mathbb{R}^{F'} \times \mathbb{R}^{F'} &\rightarrow \mathbb{R} \\ H^k &\in \mathbb{R}^{N \times F'} \\ H^{k-1} &\in \mathbb{R}^{N \times F} \end{aligned}$$

Notamos que a define **atención global** en la capa k en base a las representaciones de la capa $k-1$.



Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In 6th International Conference on Learning Representations (ICLR 2018)

Graph Attention Networks

La GAT de atención global en la práctica se restringe al vecindario del nodo objetivo, incluyendo al nodo target en el vecindario (autorelación). De esta manera, el mecanismo de atención es consistente con A.

Para hacer a los coeficientes comparables entre distintos nodos, se normaliza en torno de cada nodo target i usando una función Softmax:

$$\alpha_{ij} = \text{Softmax}_j(\{e_{ij}\}) = \frac{\exp(e_{ij})}{\sum_{l \in N(i)} \exp(e_{il})}$$

↑
vecindario de i

El mecanismo de atención en las GAT se implementa usando redes feed-forward incluyendo una transformación lineal que colapsa desde $2F'$ a un escalar:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(W_2[WH_i^{k-1} || WH_j^{k-1}]))}{\sum_{l \in N(i)} \exp(\text{LeakyReLU}(W_2[WH_i^{k-1} || WH_l^{k-1}]))}$$

$W_2 \in \mathbb{R}^{1 \times 2F'}$



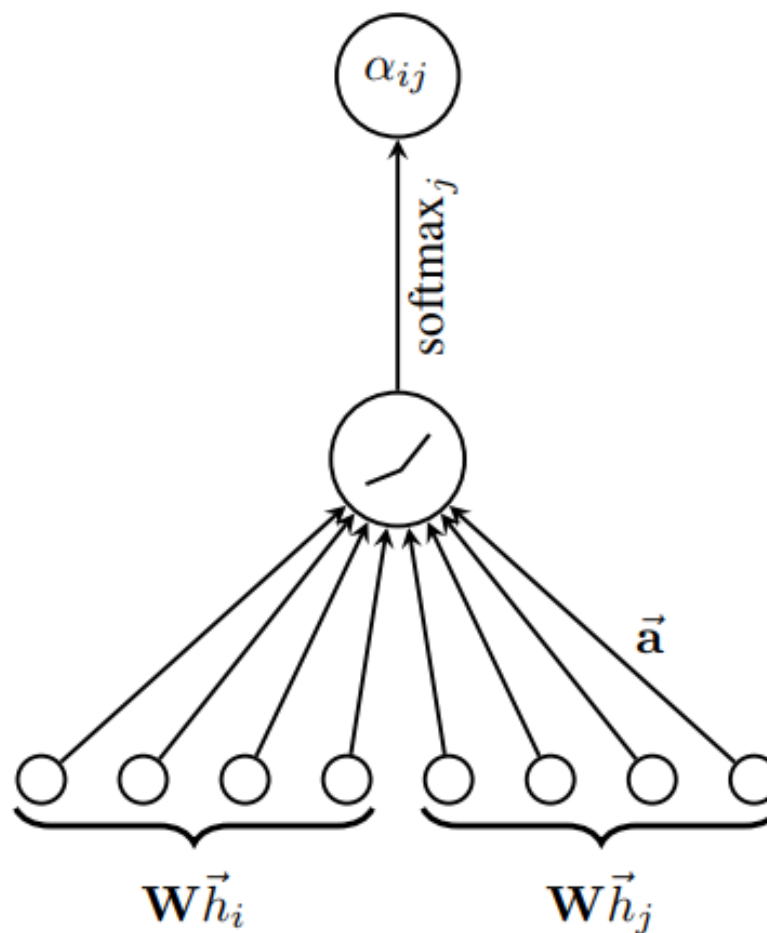
Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In 6th International Conference on Learning Representations (ICLR 2018)

Graph Attention Networks

El mecanismo de atención global define un vector de atención a nivel de features, por eso es de largo $2F'$:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(W_2[WH_i^{k-1} || WH_j^{k-1}]))}{\sum_{l \in N(i)} \exp(\text{LeakyReLU}(W_2[WH_i^{k-1} || WH_l^{k-1}]))}$$

Puedes verse como:



$$W_2 \in \mathbb{R}^{1 \times 2F'}$$

Graph Attention Networks

Finalmente, la GAT obtiene la representación del nodo en la capa k por medio de una combinación lineal de representaciones de la capa $k-1$ ponderada por los coeficientes de atención:

$$H_i^k = \sigma \left(\sum_{j \in N(i)} \alpha_{ij} W H_j^{k-1} \right), \quad W \in \mathbb{R}^{F \times F'}$$

Las GAT muestran inestabilidad durante el proceso de entrenamiento, por lo que incorporan el mecanismo de multi-head attention. Básicamente, se usan K mecanismos independientes de atención y el resultado se concatena, igual que en el transformer:

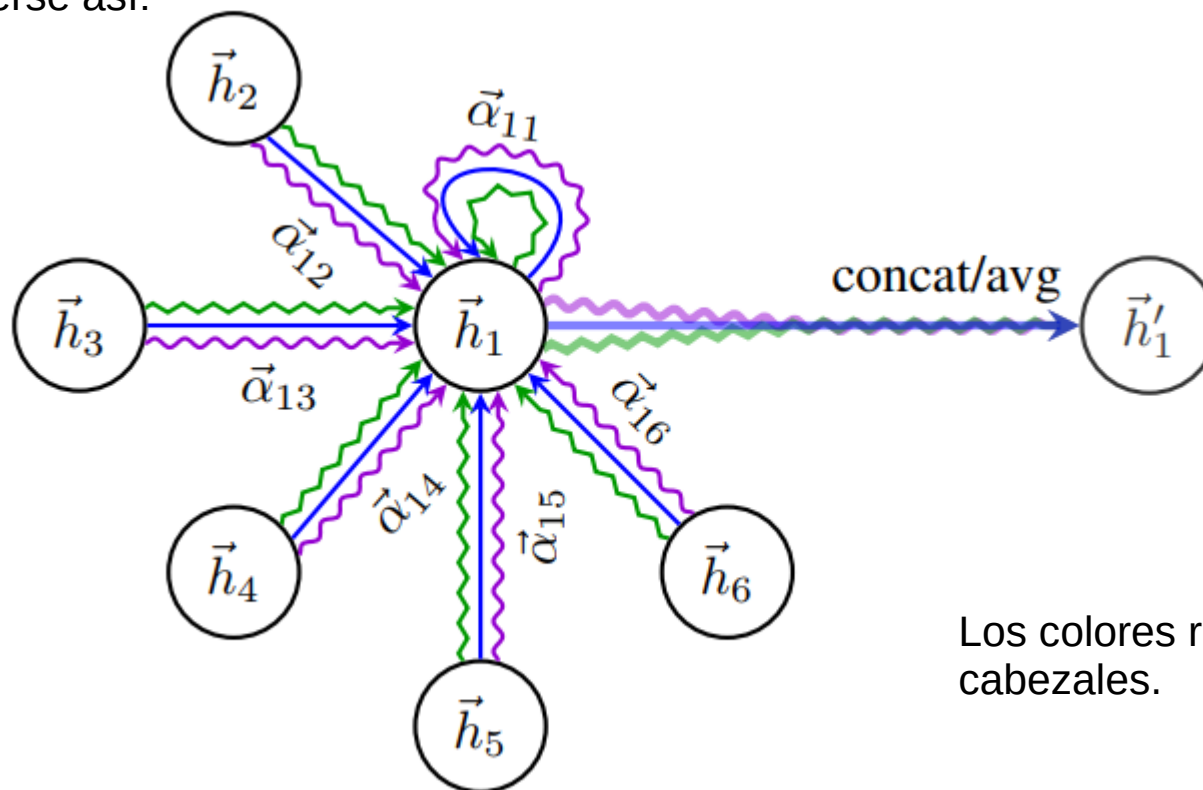
$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

Graph Attention Networks

Con el mecanismo multi-head attention, las redes GAT en la última capa en lugar de concatenar promedian:

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

El proceso puede verse así:



Los colores representan cabezales.

Graph Attention Networks

Ventajas de la GAT:

- Self-attention es paralelizable a nivel de enlaces.
- No se requiere calcular información espectral ya que opera directamente sobre el grafo.
- Analizar los coeficientes de atención beneficia la interpretabilidad del modelo.

Jerga relevante: aprendizaje sobre la misma estructura en modo semi-supervisado se llama **transducción** (el algoritmo tiene acceso a todos los nodos durante el entrenamiento pero sólo algunos tienen etiqueta). Aprendizaje sobre los mismos nodos pero enlaces nuevos se denomina **inducción** (abajo corresponde al ejemplo con redes PPI).

Datasets clásicos para estos escenarios:

	Cora	Citeseer	Pubmed	PPI
Task	Transductive	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
# Edges	5429	4732	44338	818716
# Features/Node	1433	3703	500	50
# Classes	7	6	3	121 (multilabel)
# Training Nodes	140	120	60	44906 (20 graphs)
# Validation Nodes	500	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	1000	5524 (2 graphs)

- RELATIONAL GCN -

Relational GCN

Una limitación de las GAT es que no son fácilmente extendibles a grafos multirelacionales.

Por otro lado, las GCN son extendibles a multirelaciones lo que les permite vectorizar KGs. Se extienden incorporando una idea denominada message-passing.

Message-passing: La idea es usar el framework de las GNNs definiendo las dos funciones AGGREGATE y COMBINE. En el caso de AGGREGATE, esta se define en base a un mecanismo de paso de mensajes a nivel de nodos. Luego se suman todos los mensajes del vecindario:

$$m_i^k = \sum_{i \in N(j)} M_k(H_i^{k-1}, H_j^{k-1}, e_{ij})$$

Features del enlace
↓

↑

mensaje entre el nodo i y j en la capa k , usualmente una transformación lineal

La función COMBINE se encarga de hacer un update de la representación del nodo i en la capa k :

$$H_i^k = U_k(H_i^{k-1}, m_i^k)$$

└─> Función (GRU, activación, u otra)



Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning (pp. 1263–1272). PMLR.

Relational GCN

Fijémonos que las GCN pueden verse como un caso especial de message-passing:

$$h_i^{(l+1)} = \sigma \left(\sum_{m \in \mathcal{M}_i} g_m(h_i^{(l)}, h_j^{(l)}) \right)$$

\searrow $g_m(h_i, h_j) = W h_j$

Para incorporar múltiples relaciones, la R-GCN se extiende de la siguiente forma:

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right)$$

\downarrow
parámetros de la relación

Relational GCN

Si el grafo tiene muchas relaciones, para evitar tener una matriz de pesos por cada relación, se usa un conjunto de matrices base cuya cantidad es menor que num_rels (R). Digamos:

$$V_b \in \mathbb{R}^{d_{in} \times d_{out}}, \quad b = 1, \dots, B$$

donde $B = \text{num_bases} \ll R$.

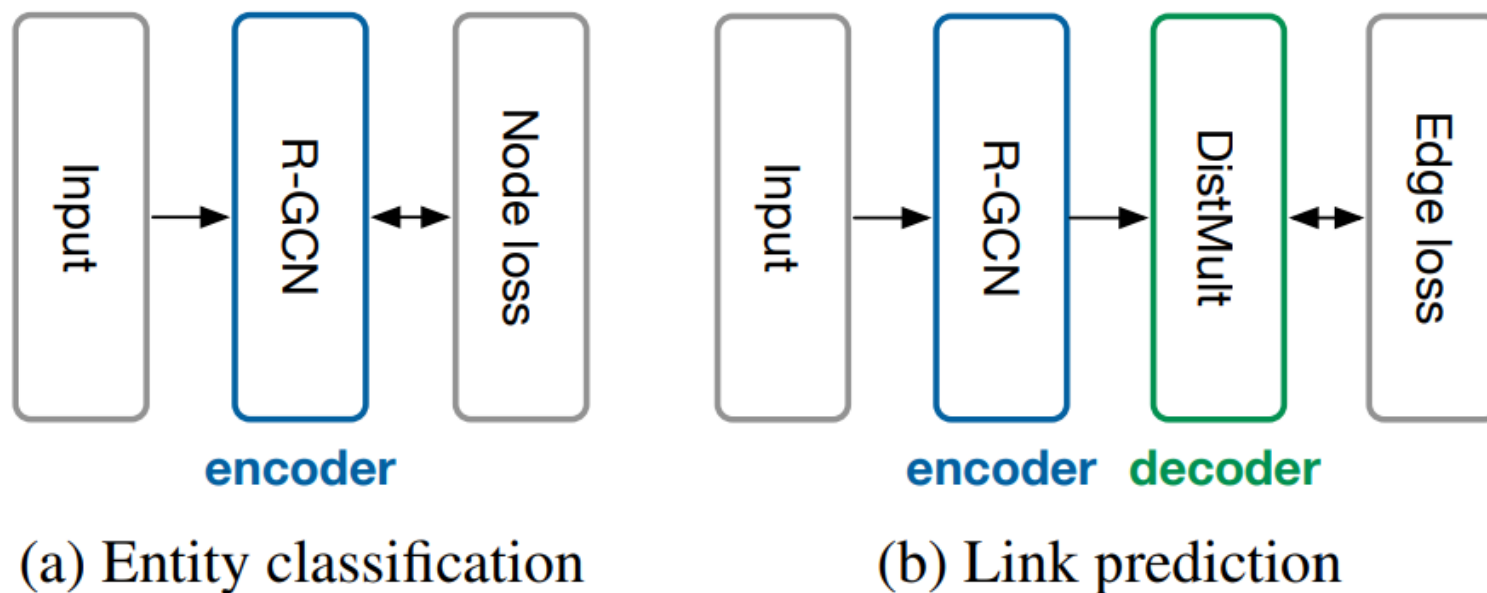
Luego, cada matriz asociada a una relación específica se representa como una combinación lineal de las matrices base:

$$W_r = \sum_{b=1}^B a_{rb} V_b$$

Los coeficientes de la combinación lineal también se aprenden.

Relational GCN

El entrenamiento depende de la tarea. En clasificación de nodos, la función de pérdida se define directamente a nivel de nodos (cross-entropy en transducción). En predicción de enlaces, agregamos un modelo de scoring en base a los embeddings de los nodos, usualmente un producto.



Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. European Semantic Web Conference (pp. 593-607).