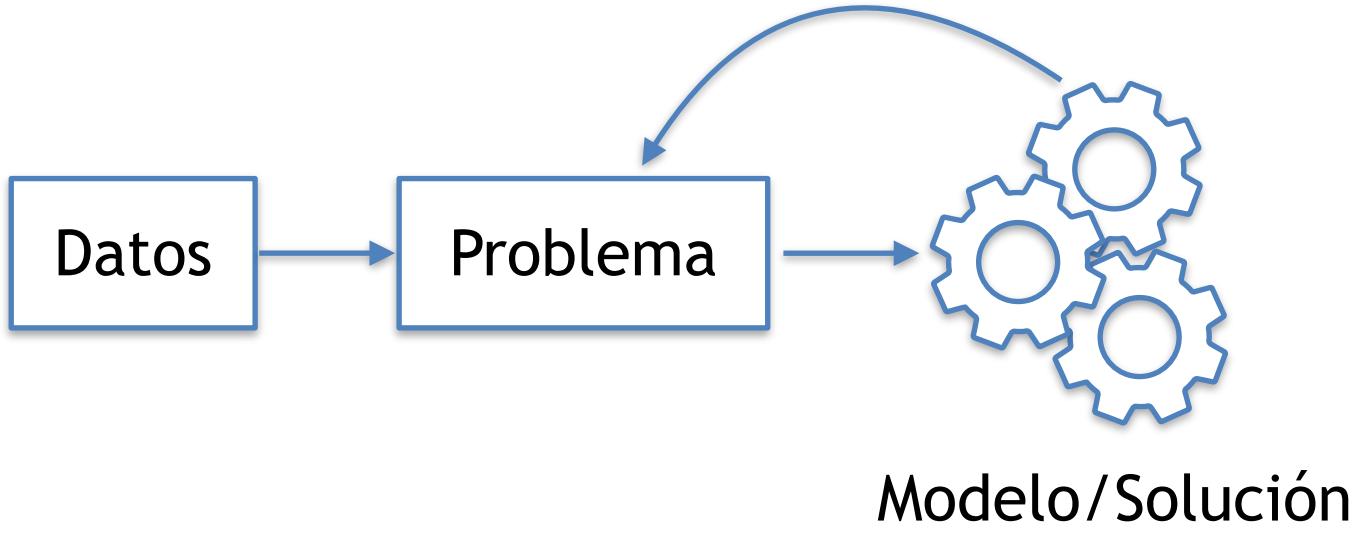
Redes basadas en grafos

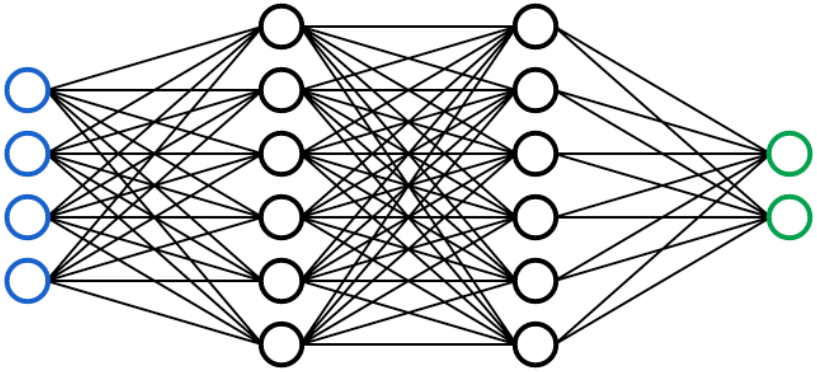
Contexto y motivación

A lo largo del curso hemos visto que una de las principales motivaciones para el nacimiento de un nuevo paradigma de arquitectura es el aumento en el uso de nuevos tipos de datos

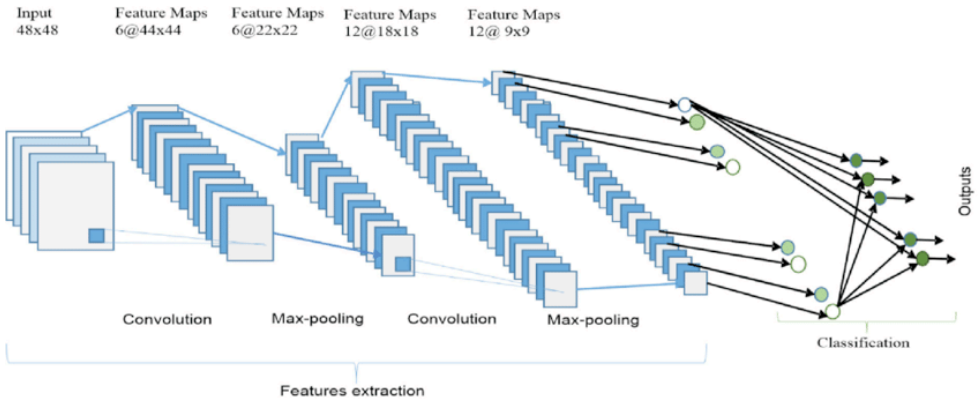


Contexto y motivación

Owner	Country	File_Date	IPC_Class
Company A	US	6/18/2008	H05H13
Company A	EP	1/30/1998	A61N5
Company A	EP	1/30/1998	A61N5
Company A	EP	1/30/1998	A61N5
Company A	JP	8/28/1997	A61N5
Company A	JP	10/4/2002	A61N5
Company A	JP	1/27/2003	A61N5
Company A	JP	4/14/2003	A61N5
Company A	JP	5/13/2011	A61N5
Company B	JP	4/2/1998	G12B13
Company B	JP	4/2/1998	G12B13
Company B	JP	5/28/1997	A61N5
Company B	JP	11/12/1997	A61N5
Company B	JP	2/29/2000	A61N5
Company B	JP	4/30/2002	A61N5

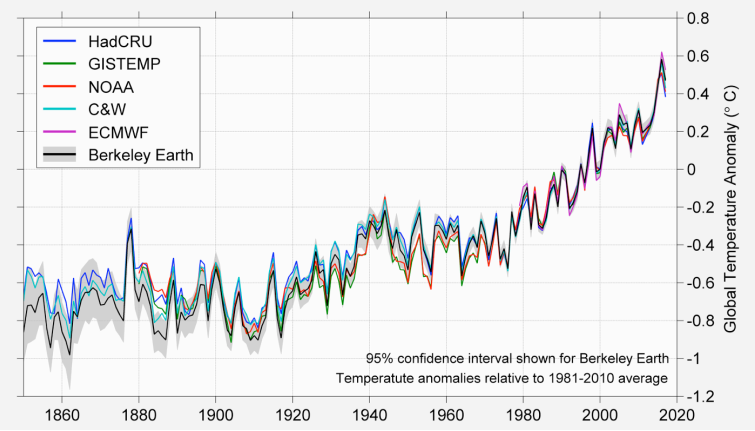


Contexto y motivación



Contexto y motivación

```
(S (NP The quick brown fox)
   (VP jumped
       (PP over
           (NP the lazy dog))))
```



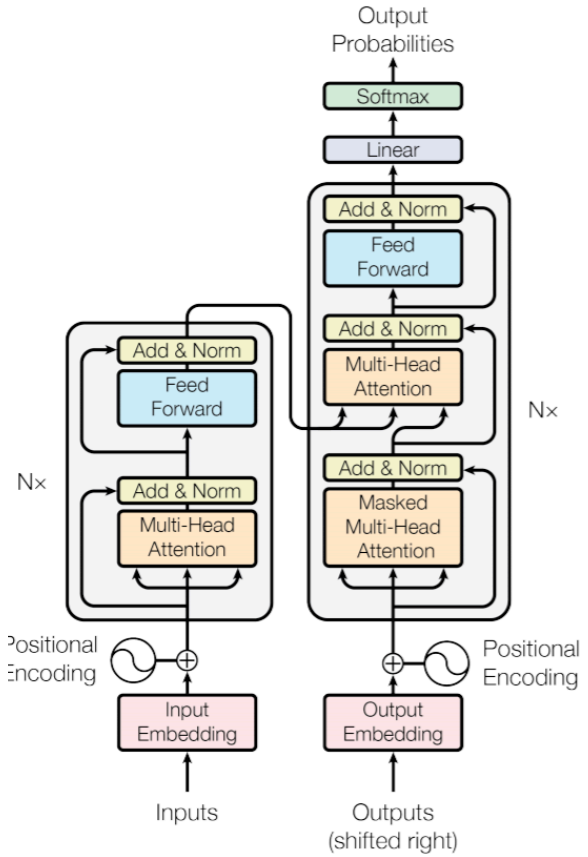
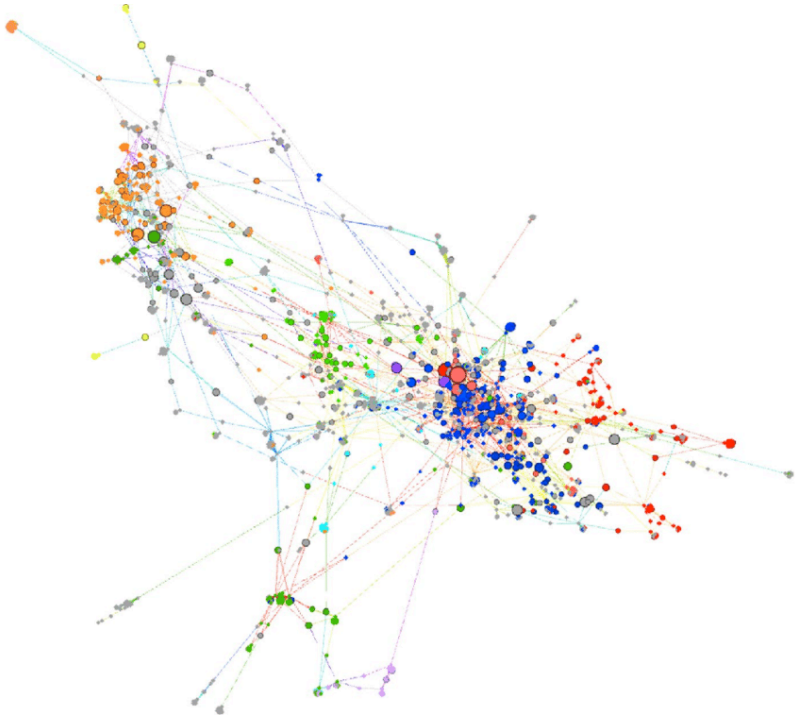


Figure 1: The Transformer - model architecture.

Contexto y motivación

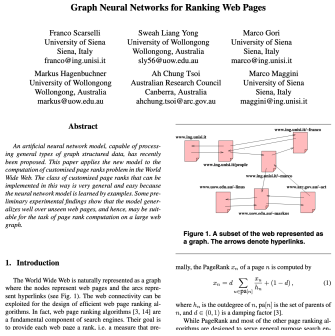
Un tipo de dato que aún no hemos discutido y que ha motivado muchísimo estudio en la última década son los grafos… y están en muchas partes



- Redes de citas entre autores.
- Relaciones humanas en un grupo.
- Redes de computadores.
- Inter-dependencia de genes.
- Conexiones eléctricas de ciudades.
- Etc.

Contexto y motivación

El trabajo donde originalmente se propone usar un modelo neuronal sobre un grafo con una estructura que aproveche las relaciones entre los elementos es de Scarcelli et al (2005) [1] en un trabajo que proponía usarlo para calcularan una versión de modificada de pagerank



Graph Neural Networks for Ranking Web Pages

Franco Scarselli, University of Siena, Siena, Italy, franco@ing.unisi.it
Sweah Liang Yong, University of Wollongong, Wollongong, Australia, sly56@uow.edu.au
Marco Gori, University of Siena, Siena, Italy, marco@ing.unisi.it
Markus Hagenbuchner, University of Wollongong, Wollongong, Australia, markus@uow.edu.au
Ah Chung Tsoi, Australian Research Council, Canberra, Australia, ahchung.tsoi@arc.gov.au
Marco Maggini, University of Siena, Siena, Italy, maggini@ing.unisi.it

Abstract

An artificial neural network model, capable of processing general types of graph structured data, has recently been proposed. This paper applies the new model to the computation of customised page ranks problem in the World Wide Web. The class of customised page ranks that can be implemented in this way is very general and easy because the neural network model is learned by examples. Some preliminary experimental findings show that the model generalizes well over unseen web pages, and hence, may be suitable for the task of page rank computation on a large web graph.

1. Introduction

The World Wide Web is naturally represented as a graph where the nodes represent web pages and the arcs represent hyperlinks (see Fig. 1). The web connectivity can be exploited for the design of efficient web page ranking algorithms. In fact, web page ranking algorithms [3, 14] are a fundamental component of search engines. Their goal is to provide each web page a rank, i.e. a measure that rep-

Figure 1. A subset of the web represented as a graph. The arrows denote hyperlinks.

mally, the PageRank x_n, of a page n is computed by

$$x_n = d \sum_{v \in pa[n]} \frac{x_v}{l_v} + (1-d), \qquad (1)$$

where l_v is the outdegree of v, $pa[n]$ is the set of parents of n, and $d \in [0,1)$ is a damping factor [3].

While PageRank and most of the other page ranking al-

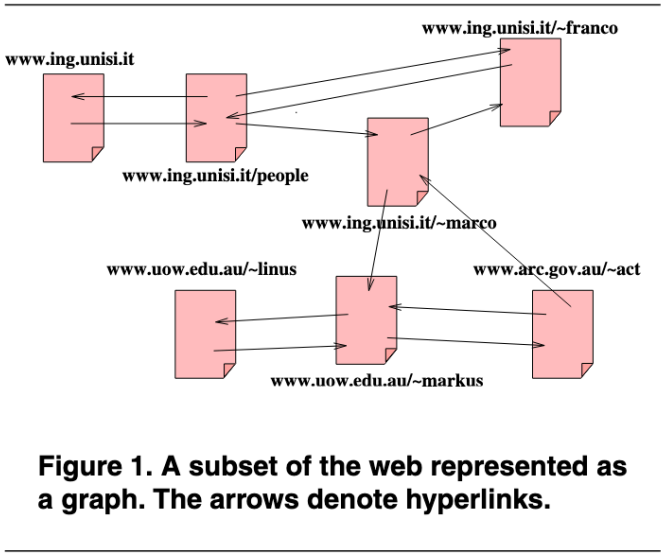


Figure 1. A subset of the web represented as a graph. The arrows denote hyperlinks.

[1] Graph Neural Networks for Ranking Web Pages, Scarcelli et al (2015)

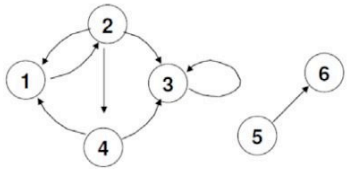
Contexto y motivación

A pesar de lo anterior el modelo fue realmente generalizado, nuevamente por Scarcelli pero con otro coautores, en el trabajo titulado "*The Graph Neural Model*" en 2008, por lo que para efectos prácticos podemos considerar este último como el nacimiento de las redes neuronales de grafos como las conocemos hoy en día.

¿Por qué necesitamos un modelo nuevo?

Una idea inicial con la que podríamos vernos tentados a aplicar nuestros modelos ya estudiados sobre un grafo podría ser la de alimentar a una red FF clásica con la Matriz de adyacencia.

Por ejemplo, podríamos aplanar la matriz de adyacencia a un solo vector de dimensión $M_{flatten} \in \mathbb{R}^{2\|V\|}$ y alimentarlo a un MLP.



	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	1	1	0	0
3	0	0	1	0	0	0
4	1	0	1	0	0	0
5	0	0	0	0	0	1
6	0	0	0	0	0	0

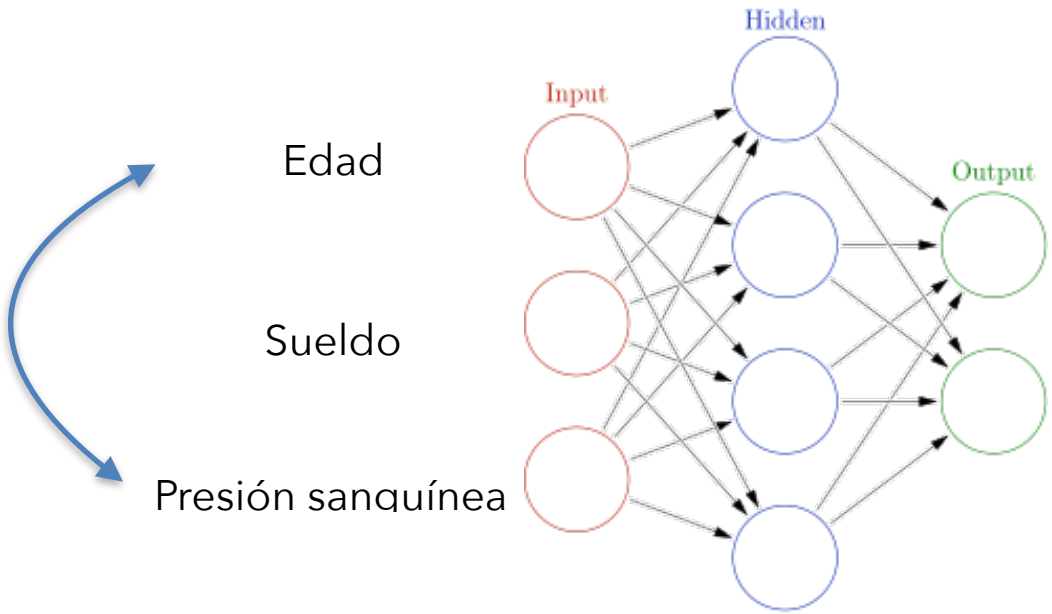
Flatten(M)

$$z_G = MLP(M[1] \oplus M[2] \oplus \ldots \oplus M[\|V\|])$$

Donde $M[i] \in \mathbb{R}^{\|V\|}$ representa una fila (o columna, dependiendo de como se quiera ver) de la matriz de adyacencia y $\oplus$ es concatenación vectorial.

¿Por qué necesitamos un modelo nuevo?

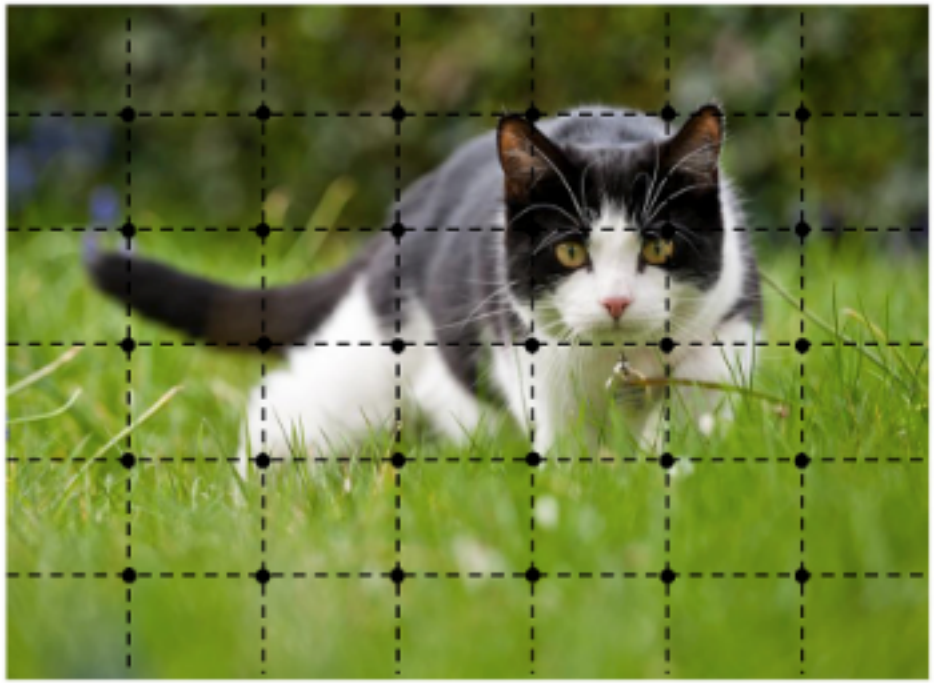
El problema clave con esta aproximación es que **no es invariante a permutaciones**.



La invarianza a permutaciones es una característica clave dentro del modelo que hagamos ya que sin importar el orden de las filas o columnas en una matriz de adyacencia, si esta codifica al mismo grafo, el resultado debe ser el mismo por nuestro modelo

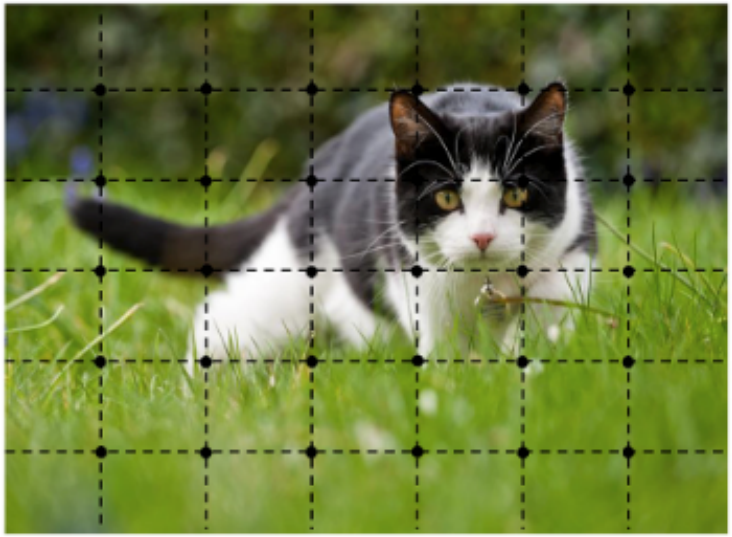
¿Por qué necesitamos un modelo nuevo?

Otra alternativa que podríamos pensar es usar nuestras capas convolucionales sobre el grafo…

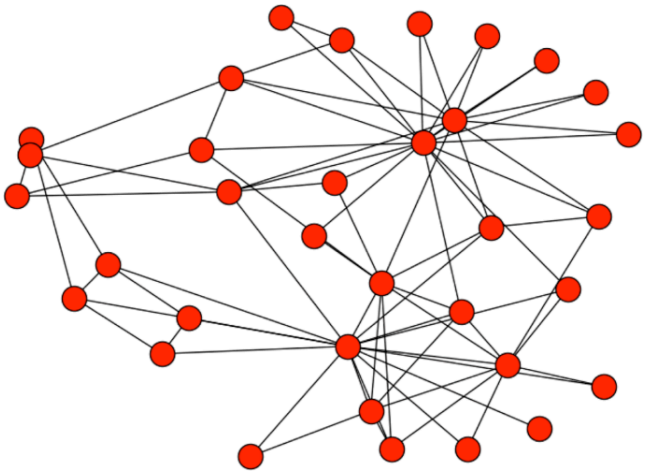


¿Por qué necesitamos un modelo nuevo?

Las CNNs están bien definidas solo en espacios euclidianos, como una imagen o un vector 1D, no lo están en espacios no euclideanos como un grafo donde los patrones de conexión entre los nodos son dispersos.



$\neq$



Construyendo un modelo inicial

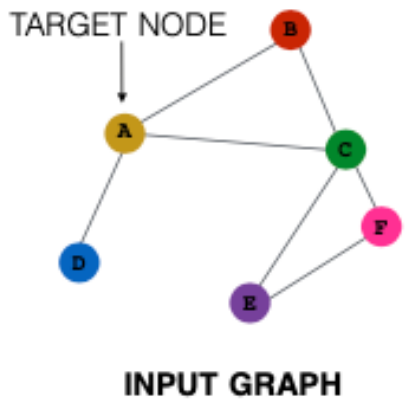
Vamos a definir un grafo como una tupla de valores $\mathcal{G} = (V, \mathcal{E}, X, E)$ donde:

- V es el conjunto de vértices (nodos).
- $\mathcal{E} \subseteq V \times V$ es el conjunto de aristas (conexiones).
- $X \in \mathbb{R}^{N \times D}$ es la matriz de atributos de los vértices.
- $E \in \mathbb{R}^{N \times N \times F}$ es la matriz de atributos de las aristas.

Donde D y F son la dimensionalidad usada para la representación de los atributos de los vértices y las aristas respectivamente.

Paso de Mensajes Neuronal

Aunque hay varios trabajos que han llegado a derivar la estructura básica de una GNN desde distintos puntos de vista, la mayoría coincide en que un modelo neuronal para grafos es una forma de pasos de mensajes (*neural message passing*) donde un vector de mensaje es transmitido de un nodo a otro donde haya una conexión y usado para actualizar la representación del correspondiente nodo, el primero en proponer esta modalidad fue Gilmer et al [2] y se le llamó **Message Passing Neural Network** (MPNN).



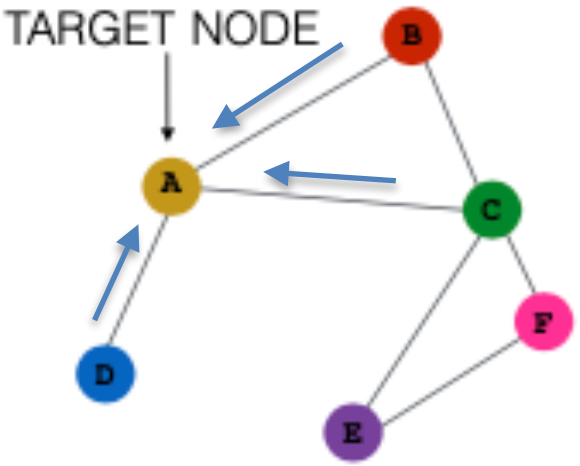
Fuente Imagen: Graph Representation Learning (William L. Hamilton), Capitulo 5.

[1] Neural Message Passing for Quantum Chemistry .- Gilmer et al (2017)

Paso de Mensajes Neuronal

El algoritmo de forma general se va a componer de dos pasos que se van a repetir por varias iteraciones de manera cíclica:

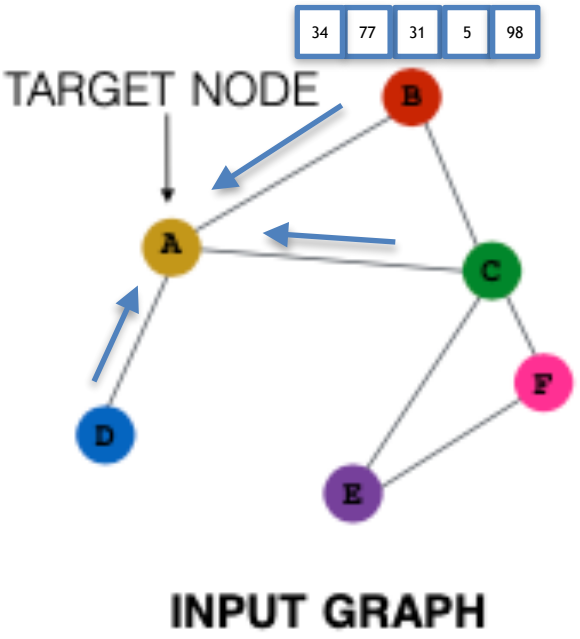
1. Paso de mensajes
2. Actualización de los estados de los nodos



INPUT GRAPH

Paso de Mensajes Neuronal

En cada iteración de paso de mensajes el embedding latente con que el se representa cada nodo $h_v^{(k)}$ (para un nodo $v \in V$ durante la iteración k) es actualizado agregando la información de cada nodo en el vecindario de v: $\mathcal{N}(v)$.

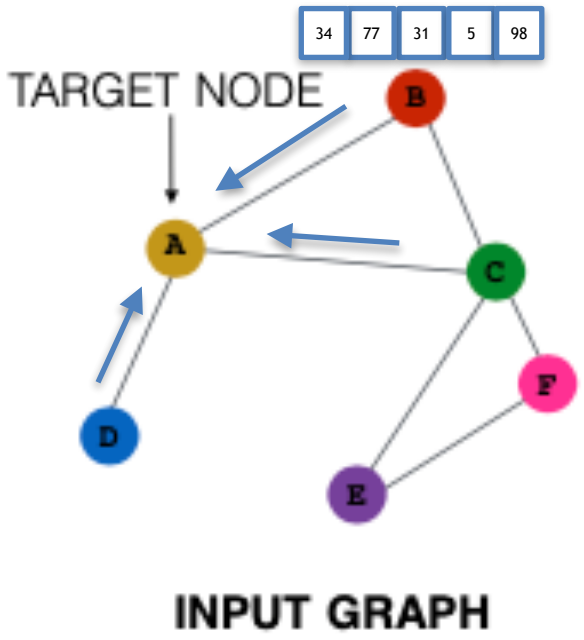


$$h_v^{(k+1)} = f_{update}^{(k)}(h_v, g_{aggregate}^{(k)}(\{h_u^{(k)}, \forall u \in \mathcal{N}(v)\}))$$
$$= f_{update}^{(k)}(h_v^{(k)}, m_{\mathcal{N}(v)}^{(k)})$$

$m_{\mathcal{N}(v)}^{k}$ es el mensaje resultante de la agregación de los vectores de estado de los nodos en el vecindario; y pasado como mensaje "*entrante*" al nodo v

Paso de Mensajes Neuronal

Notar que tanto la función $f_{update}(\cdot)$ como $g_{aggregate}(\cdot)$ tienen que ser diferenciables por lo que podríamos implementarlas como una red FF clásica.

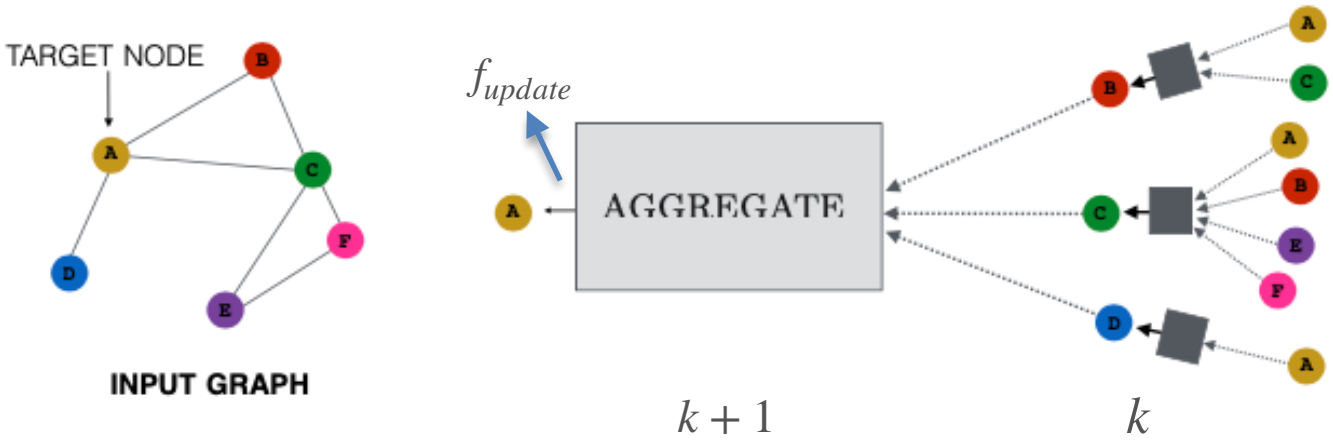


$$h_v^{(k+1)} = f_{update}^{(k)}(h_v, g_{aggregate}^{(k)}(\{h_v, \forall v \in \mathcal{N}(v)\}))$$
$$= f_{update}^{(k)}(h_v^{(k)}, m_{\mathcal{N}(v)}^{(k)})$$

Paso de Mensajes Neuronal

Al final de K iteraciones el embedding final de un nodo es simplemente el último embedding calculado:

$$z_v = h_v^{(K)}$$



La GNN básica - Aterrizando la matemática

Hasta ahora lo que hemos definido es solo un "*marco de paso de mensajes*" pero no realmente un modelo (donde están los parámetros?).

Para poder llegar a una representación que nos permita aprender necesitamos parámetros en el modelo; y no solo eso, necesitamos parámetros que influencien significativamente la salida del modelo, o en nuestro caso, las representaciones de los nodos.

Lo que necesitamos ahora es definir las funciones f_{update} y $f_{aggregate}$ de manera tal que tengan parámetros.

La GNN básica - Aterrizando la matemática

Sea entonces $h_v^{(k)}$ el embedding del nodo v en la iteración k, esté estará determinado por:

$$h_v^k = \sigma\left(W_{self}^{(k)} h_v^{(k-1)} + W_{neigh}^k \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} + b^{(k)}\right)$$

Donde $\sigma(\cdot)$ es la función de actualización (f_{update}) que será escogida como una función no-lineal element-wise y las matrices $W_{self}^{(k)}, W_{neigh}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ son matrices de parámetros entrenables.

Notar que $d^{(k)}$ es la dimensionalidad del vector de embedding del nodo v en la iteración k, además recordando que previamente dijimos que $m_{\mathcal{N}(v)} = \sum_{u \in \mathcal{N}(v)} h_u$ podemos reescribir la ecuación anterior como:

$$h_v^{(u)} = \sigma\left(W_{self} h_v + W_{neigh} m_{\mathcal{N}(v)}\right)$$

Self-loops

Es posible simplificar la ecuación anterior mediante la adición de self-loops en cada nodo de manera que ahora $v \in \mathcal{N}(v)$ y por lo tanto de la ecuación:

$$h_v^{(u)} = \sigma\left(W_{self} h_v + W_{neigh} m_{\mathcal{N}(v)}\right)$$

Puede ser omitido el paso explicito de actualización (aplicación de la función σ), quedando:

$$h_v^k = g_{aggegrate}(\{h_u^{k-1}, \forall u \in \mathcal{N}(v) \cup \{v\}\})$$

La principal ventaja de esto es que ya no se requiere una función explícita de actualización ya que esto es realizado de forma implícita en el paso de agregación, sin embargo, esto tiene algunas desventajas principalmente que resta expresividad al modelo debido a que ahora las matrices W_{self} y W_{neigh} son una sola.

Generalización sobre $g_{aggregate}$

Generalización sobre $g_{aggregate}$

Es posible mejorar nuestro modelo base trabajando sobre las funciones de $g_{aggregate}$ y f_{update}, ahora discutiremos algunas de estas mejoras, comenzando por la función de agregación.

$$h_v^{k+1} = f_{update}^k(h_v, g_{aggregate}^k(\{h_v, \forall v \in \mathcal{N}(v)\}))$$

Normalización de Vecindario

Hasta ahora hemos considerado como función de agregación la función más simple que podríamos usar: una suma de los embeddings de los vecinos. El problema con esta aproximación es que es inestable, altamente dependiente y sensible al grado de los nodos.

Una solución inicial podría ser normalizar la operación implementada por $g_{aggregate}$ de acuerdo a la cantidad de miembros del vecindario:

$$m_{\mathcal{N}(v)} = \frac{\sum_{u \in \mathcal{N}(v)} h_u}{|\mathcal{N}(v)|}$$

Normalización de Vecindario

Se ha visto experimentalmente que otros tipos de normalización también son efectivos, por ejemplo, Kipf y Welling (2016) [3] implementan una normalización que toma en cuenta tanto el grado del nodo objetivo como el de cada nodo en su vecindario:

$$m_{\mathcal{N}(v)} = \sum_{u \in \mathcal{N}(v)} \frac{h_u}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}}$$

Esta aproximación fue titulada por sus autores como "*Symmetric Normalization*", presenta ventajas principalmente en grafos/redes donde la distribución de grados es altamente sesgada positiva.

Graph Convolutional Networks (GCNs)

Esta aproximación también fue propuesta por Kipf y Welling (2016) y hace uso tanto de la normalización simétrica de vecindario como de la aproximación de self-loops. En este modelo se define la función de paso de mensajes como:

$$h_i^k = \sigma\left(W^{(k)} \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{h_v}{\sqrt{|\mathcal{N}(v)||\mathcal{N}(u)|}} \right)$$

Esta aproximación es una de las más populares dentro de lo que es la implementación de GNNs.

Atención sobre el vecindario - GAT

El primer trabajo que aplica la idea de atención (de Bahdanau et al.) para mejorar la operación de agregación de nuestras GNNs fue el de Velickovic et al (2018) [4] formando la denominada Graph Attention Network (GAT).

La idea principal es asignar un puntaje de atención a cada vecino, este puntaje será utilizado para ponderar la influencia de este vecino durante la fase de agregación

$$m_{\mathcal{N}(v)} = \sum_{u \in \mathcal{N}(v)} \alpha_{v,u} h_u,$$

Donde $\alpha_{v,u}$ denota la atención sobre el vecino $u \in \mathcal{N}(v)$ en la fase de agregación para el nodo v.

Atención sobre el vecindario - GAT

En el paper original de GAT, los puntajes de atención $\alpha_{v,u}$ son definidos como:

$$\alpha_{v,u} = \frac{exp(a^T[Wh_v \oplus Wh_u])}{\sum_{u' \in \mathcal{N}(v)} exp(a^T[Wh_v \oplus Wh_{u'}])},$$

Donde a es el vector de puntajes de atención entrenable, W es una matriz de pesos entrenable y $\oplus$ denota concatenación.

La popularidad de las GAT no quiere decir, sin embargo, que esta sea la única forma efectiva de implementar atención sobre nuestras GNNs, veamos algunas variaciones.

Atención sobre el vecindario - Atención Bilineal y atención general vía MLP

En el modelo de atención bilineal se expresa el vector de puntajes de atención como:

$$\alpha_{v,u} = \frac{exp(h_v^T W h_u)}{\sum_{u' \in \mathcal{N}(v)} exp(h_v^T W h_{u'})}.$$

Mientras que si queremos, podemos usar un MLP con output escalar:

$$\alpha_{v,u} = \frac{exp(MLP(h_v, h_u))}{\sum_{u' \in \mathcal{N}(v)} exp(MLP(h_v, h_{u'}))}.$$

Atención sobre el vecindario - Múltiples cabezas de atención

Es posible también implementar el tipo de atención Multi-cabezal que propusieron Vaswani et al en su modelo Transformer. La idea es computar K pesos de atención distintos entre un nodo v y un determinado nodo de su vecindario $u \in \mathcal{N}(v)$: $\alpha_{v,u,k'}$ usando capas de atención independientes (cada una con sus matrices W).

Los mensajes se agregan y combinan en la fase de agregación:

$$m_{\mathcal{N}(v)} = [a_1 \oplus a_2 \oplus \ldots \oplus a_K]$$
$$a_k = W_k \sum_{u \in \mathcal{N}(v)} \alpha_{v,u,k} h_u,$$

Donde los puntajes de atención $\alpha_{v,u,k}$ para cada cabeza de atención k es computada con cualquiera de las formas que vimos anteriormente.

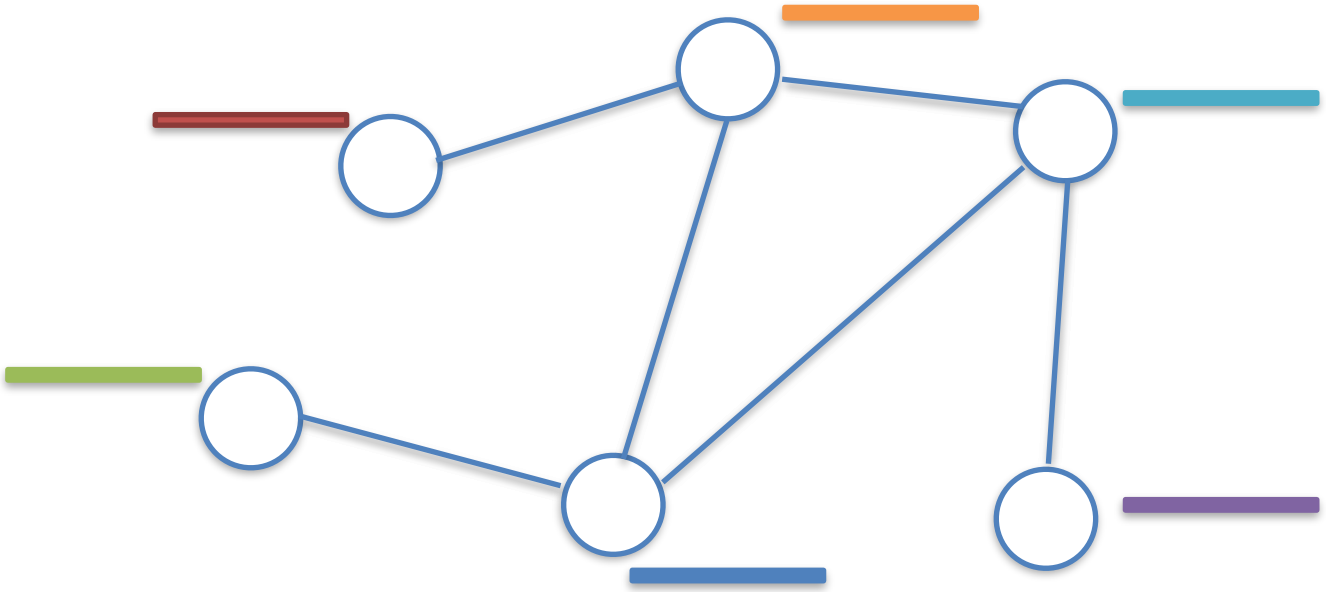
Sobre-Suavizamiento

(Over-smoothing)

Sobre-Suavizamiento

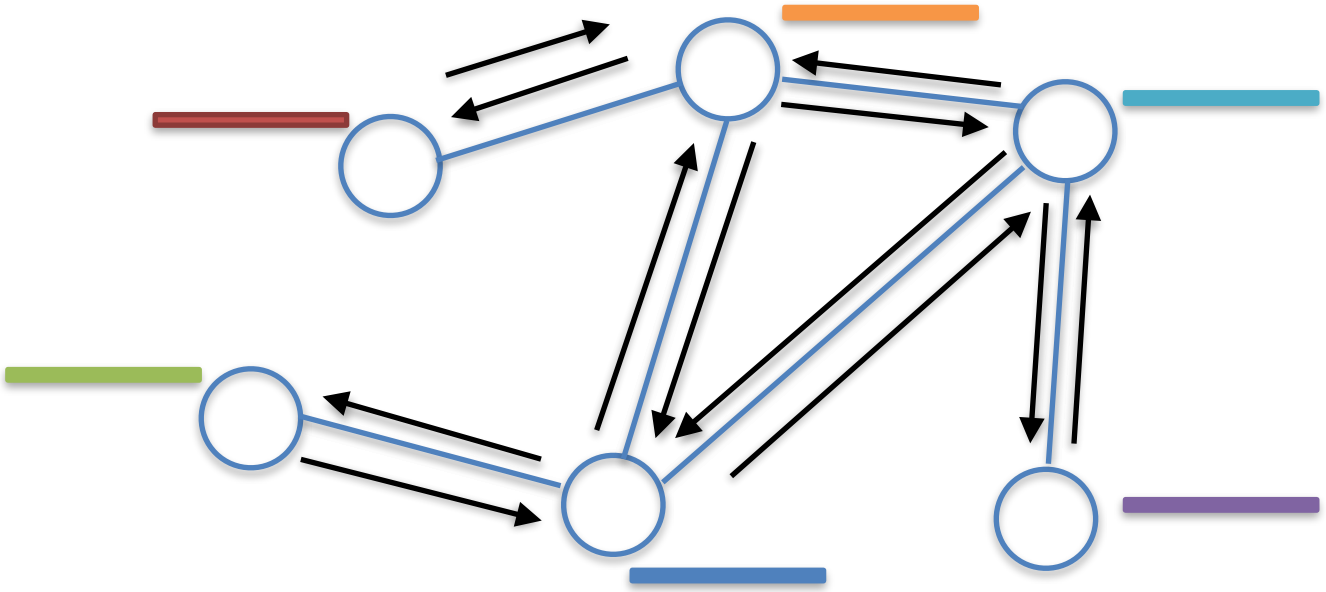
Hay un problema bastante importante que afecta a nuestro modelo de paso de mensajes y tiene que ver con la cantidad de información que estamos transmitiendo a cada nodo sobre el resto de los nodos en la red.

Para ejemplificar el problema supongamos un grafo simple con sus correspondientes embeddings iniciales marcados de color:



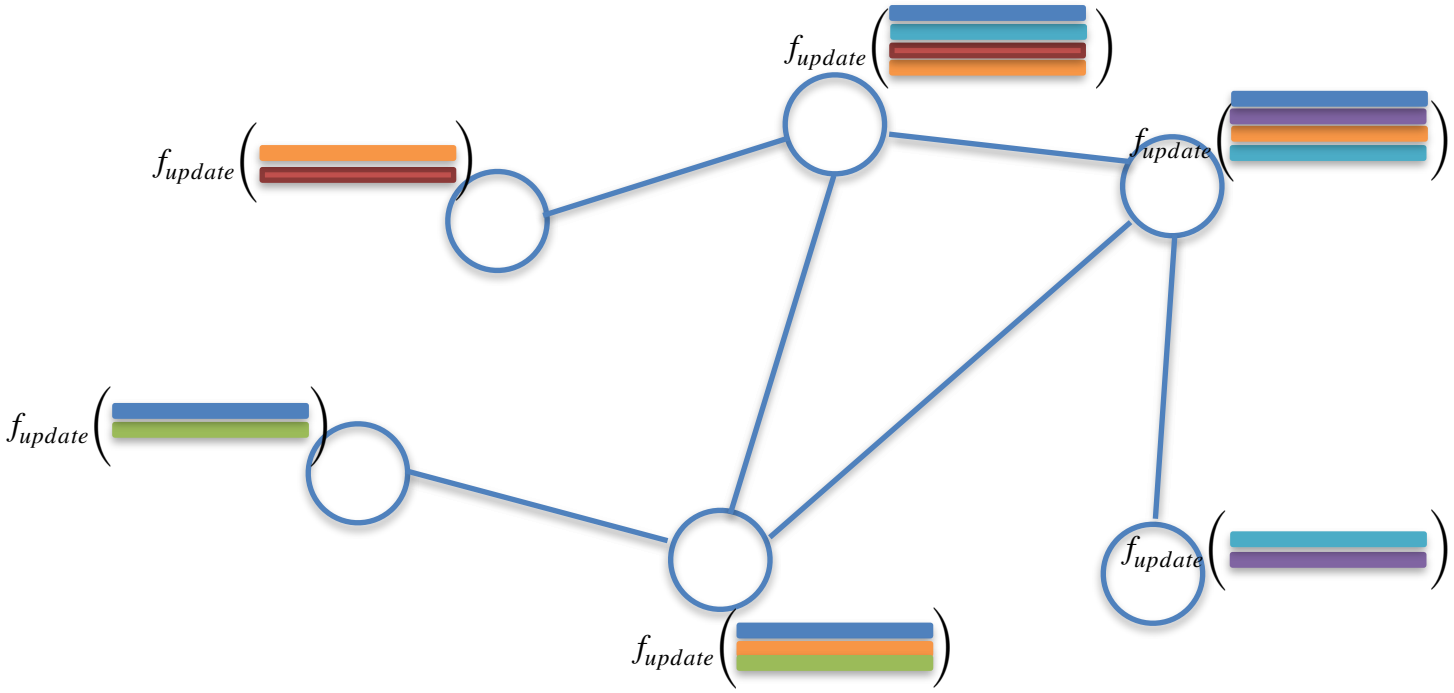
Sobre-Suavizamiento

Ahora supongamos que entrenamos la red con una sola capa y por lo tanto hacemos un paso de mensajes y actualizamos:



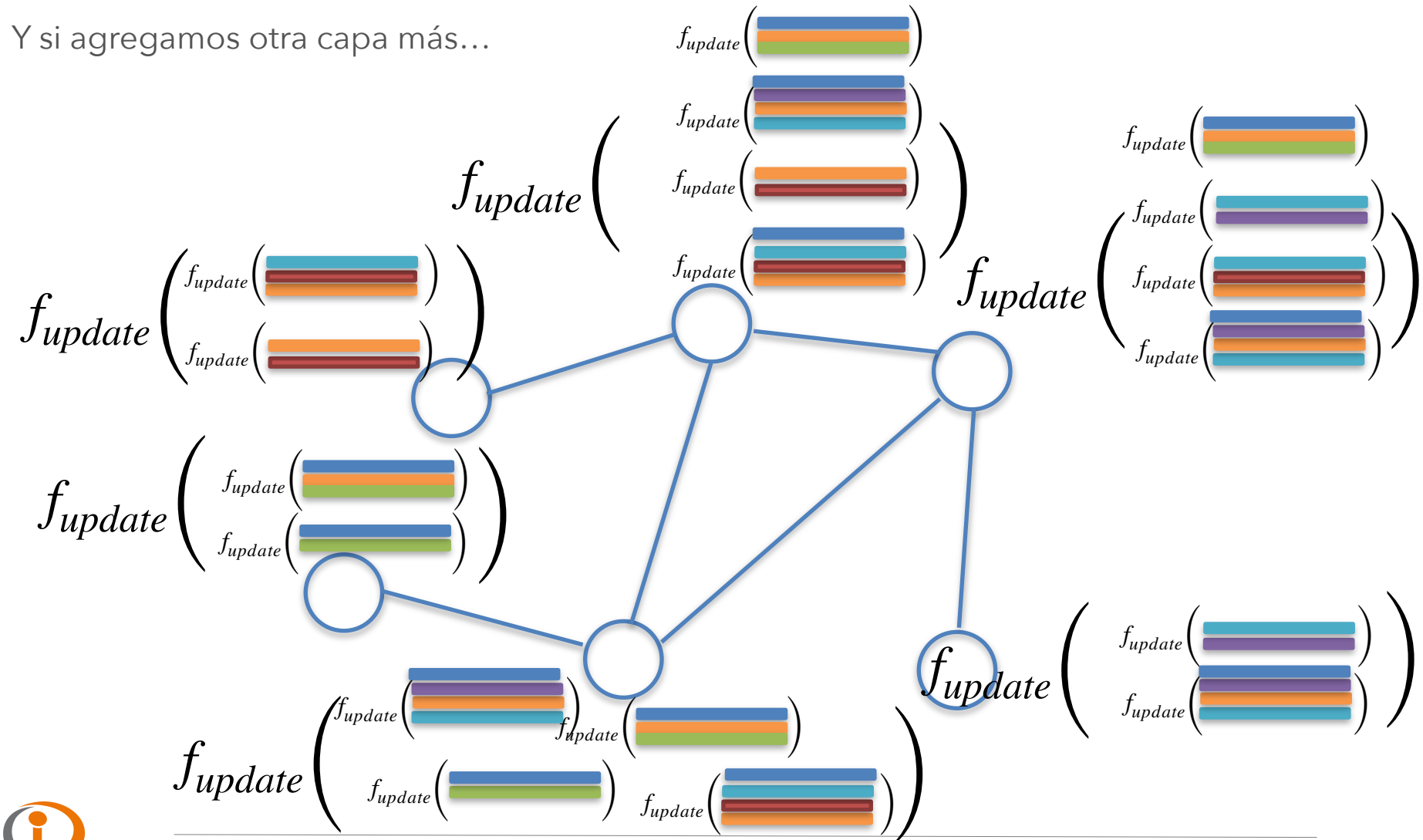
Sobre-Suavizamiento

Ahora supongamos que entrenamos la red con una sola capa y por lo tanto hacemos un paso de mensajes y actualizamos:



Sobre-Suavizamiento

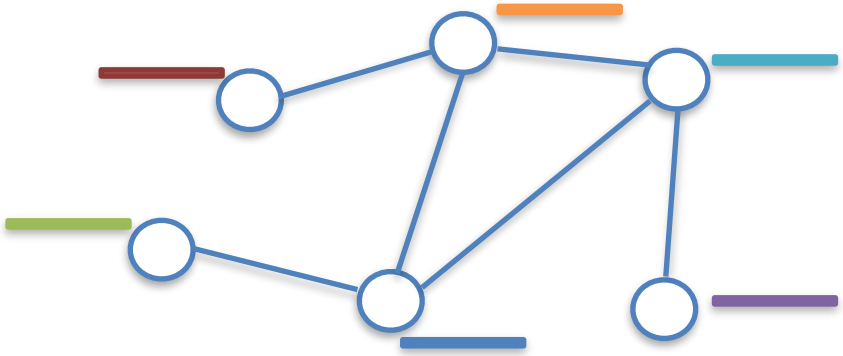
Y si agregamos otra capa más...



Sobre-Suavizamiento

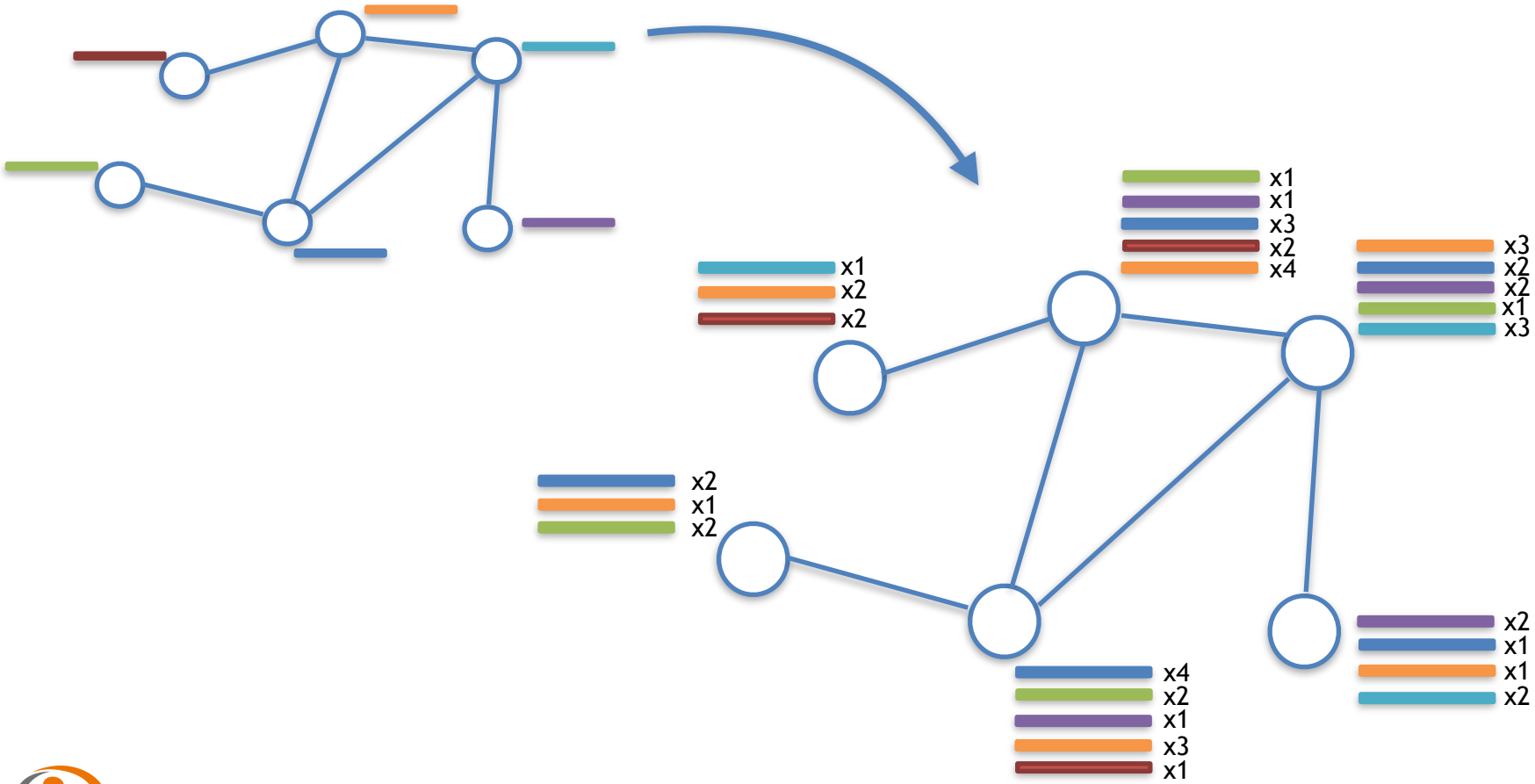
En el dibujo anterior se puede ver parte de uno de los problemas anteriores que discutimos cuando mencionamos la necesidad de normalización por el tamaño del vecindario: Nodos con mayor grado reciben mayor información que aquellos con menor grado, sin embargo, hay otro problema extremadamente importante que se puede ver mejor si hacemos una especie de gráfica con la cantidad de color de cada tipo que recibe finalmente cada nodo para actualizar su embedding.

La distribución de colores partió de esta manera:



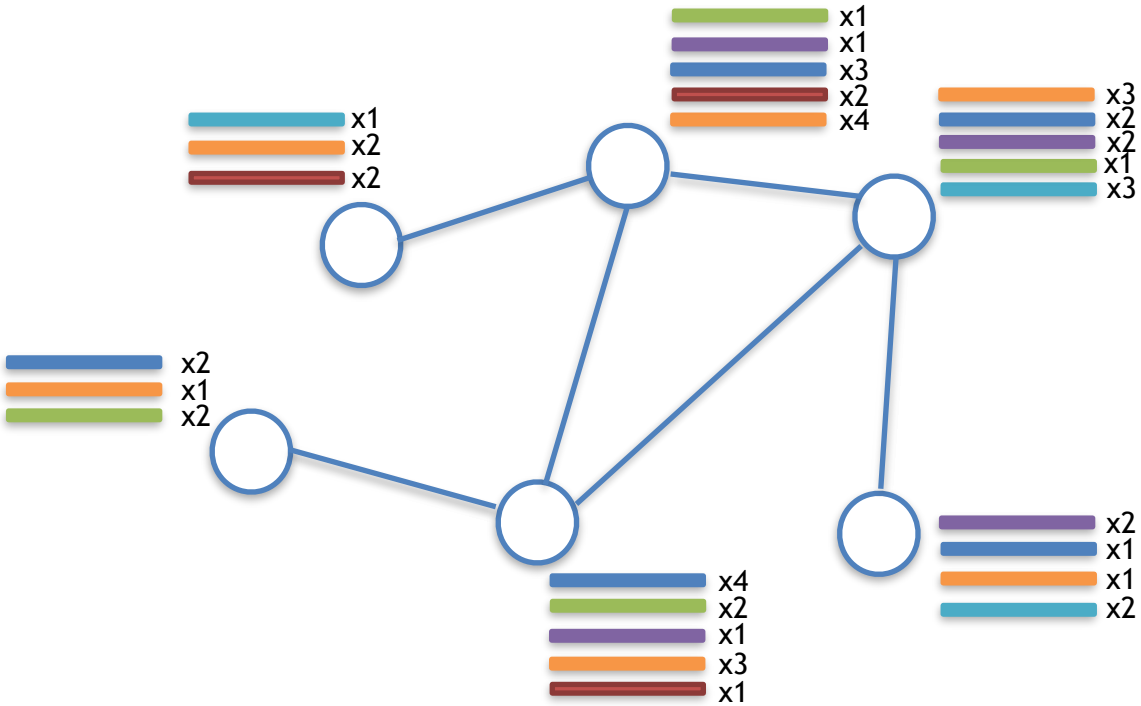
Sobre-Suavizamiento

Y luego de haber realizado dos pasos de mensajes tenemos una distribución de colores en cada nodo bastante más variada.



Sobre-Suavizamiento

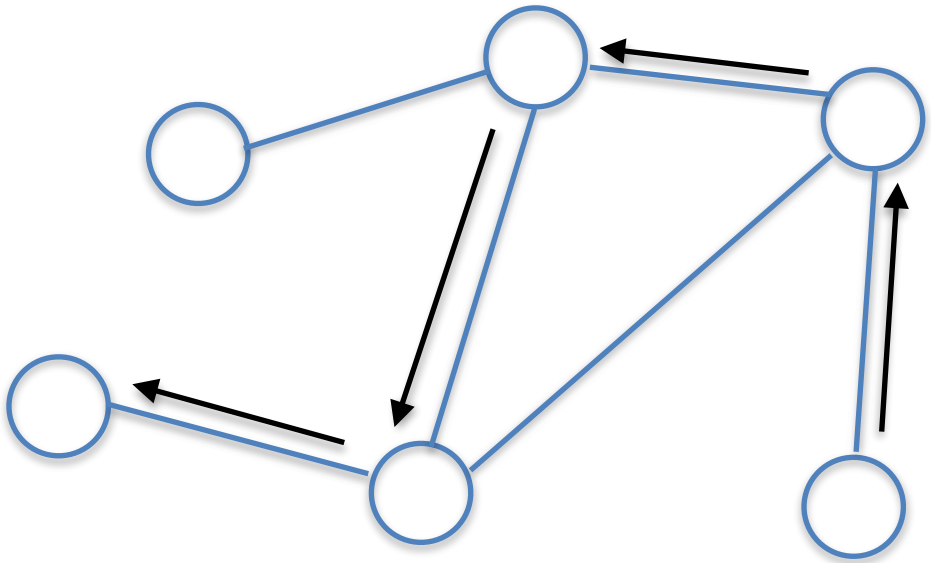
El problema con esto es que al apilar varias capas de nuestra GCNN (hacer al red más profunda) la distribución de colores se va a ir haciendo cada vez más homogénea, lo que implica que ahora va a ser cada vez más difícil distinguir el embedding de un nodo con el de otro porque la información va a estar tan mezclada que todo se va a ver muy similar.



Sobre-Suavizamiento

El problema anterior se le llama sobre-suavizamiento (Over-Smoothing) y es un problema crítico en redes de grafos profundas.

Más formalmente, lo que está ocurriendo es que los vectores de embedding de cada nodo se vuelven muy similares entre sí, sobre todo cuando hay self-loops, esta es una problemática porque queremos hacer GNN más profundas para poder aprender a codificar en las representaciones de los nodos representaciones de largo plazo de la estructura del grafo.



Sobre-Suavizamiento

Xu et al. [2018] comenta sobre esta influencia y la define en términos de la matriz jacobiana:

$$I_K(u, v) = 1^T\left(\frac{\partial g_v^{(K)}}{\partial h_u^{(0)}}\right)1,$$

Donde 1 es un vector de solo 1's. El valor anterior usa la suma de las entradas de la matriz jacobiana para medir la cantidad de influencia que tiene la representación inicial de un nodo u en la representación final de otro nodo v.

Sobre-Suavizamiento

Los autores también proponen el siguiente teorema:

Teorema

Para cualquier GNN donde se tengan auto-conexiones y una función de agregación de la forma:

$$AGGREGATE(\{h_v, \forall v \in \mathcal{N}(u) \cup \{u\}\}) = \frac{1}{f_{(|\mathcal{N}(u) \cup \{u\}|)}} \sum_{v \in \mathcal{N}(u) \cup \{u\}} h_v,$$

Donde $f : \mathbb{R}^+ \to \mathbb{R}^+$ *es una función de normalización cualquier diferenciable, se tiene que:*

$$I_K(u, v) \propto p_{\mathcal{G},K}(u \mid v),$$

donde $p_{\mathcal{G},K}(u \mid v)$ *denota la probabilidad de visitar el nodo* v *en un paseo aleatorio de largo* K *comenzando desde el nodo* u *.*

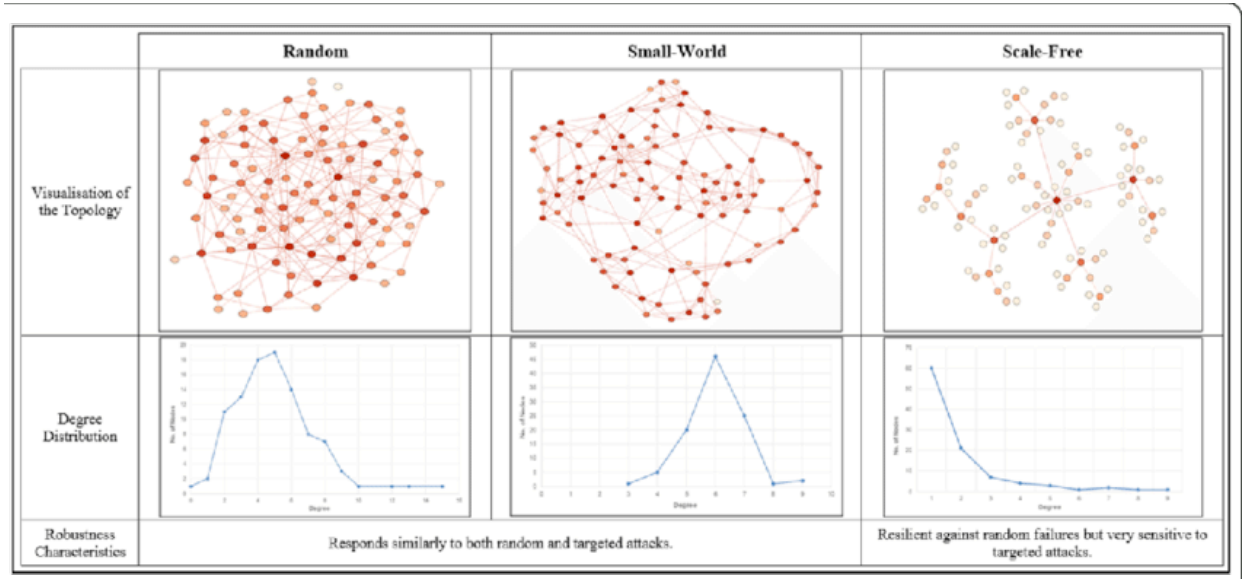
Sobre-Suavizamiento

Esto es un gran problema porque al hacer $K \to \infty$ la influencia de cada nodo se aproxima a una distribución estacionaria dada por un paseo aleatorio sobre el grafo y casi uniforme, lo que implica que toda la información del vecindario local (como información sobre las comunidades) se pierde

Mientras más conectada esté la red más se nota el fenómeno de sobre-suavizamiento por lo fácil que se transmite la información por los hubs (nodos con alto grado).

Sobre-Suavizamiento

Por si lo anterior no fuese lo suficientemente preocupante, en la actualidad se ha descubierto que gran (GRAN) parte de los grafos que observamos en una inmensa cantidad de fenómenos tienen propiedades ya sea de mundo pequeño (la distancia entre dos nodos cualquiera en el grafo es pequeña si uno sabe por "qué nodos irse", tan pequeña como incluso $O(log(|V|))$ o libres de escala.



Fuente imagen: Network Science approach to modeling the topology and robustness of supply chain networks: a review and perspective, Perera et al.

Generalización sobre f_{update}

Concatenación - Skip-Connections

Una forma de paliar el sobre-suavizamiento es extrapolar una técnica que ya nos debería parecer familiar: Skip-Connections!

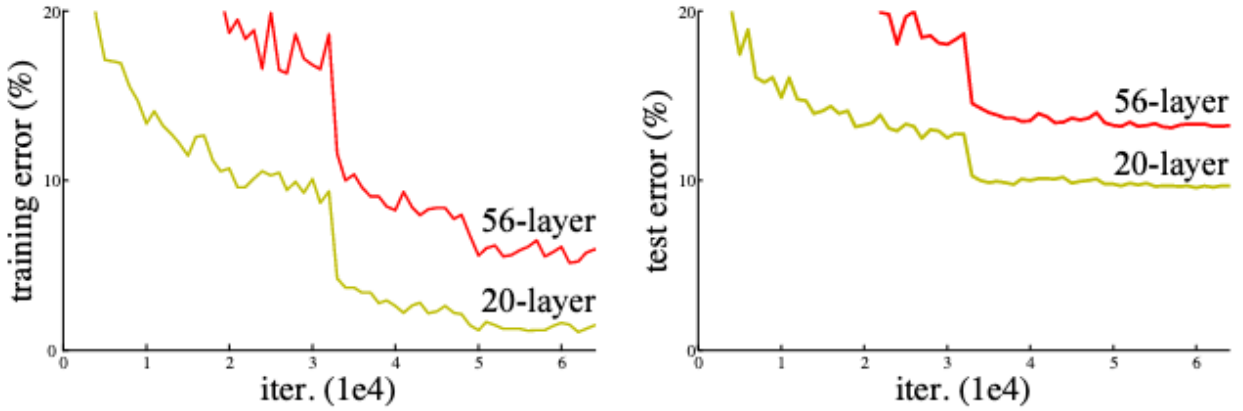


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Fuente imagen: Deep Residual Learning for Image Recognition, He et al. (2015)

Concatenación - Skip-Connections

Recordemos que las skip-connections que implementaba la arquitectura ResNet trataban de preservar explícita y directamente la información de capas anteriores, en nuestro caso para GNNs la idea es que haga lo mismo.

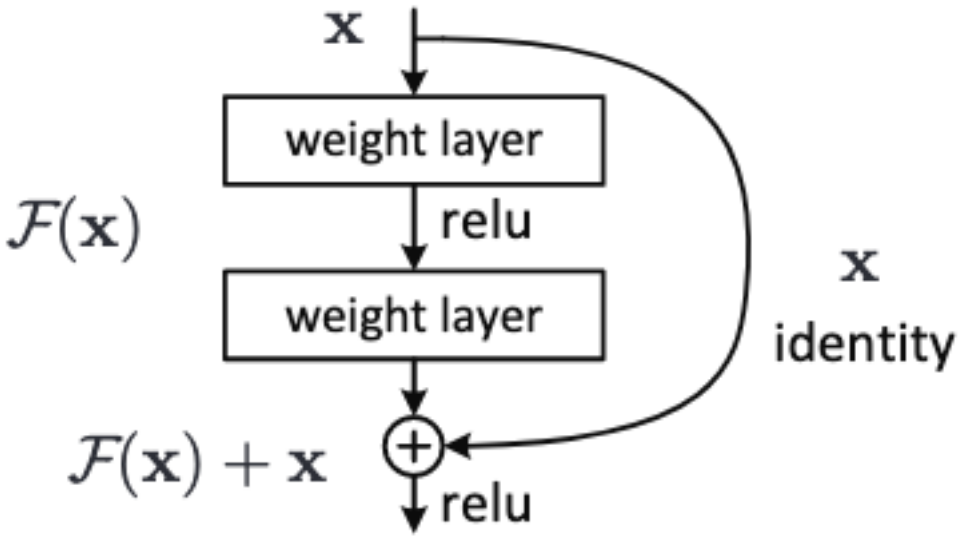


Figure 2. Residual learning: a building block.

Fuente imagen: Deep Residual Learning for Image Recognition, He et al. (2015)

Concatenación - Skip-Connections

Una de las skip-connection más simples que podemos implementar es la concatenación para preservar la información original de una arista durante los pasos de mensaje:

$$f_{update}^{(concat)}(h_v, m_{\mathcal{N}(v)}) = (f_{update}(h_v, m_{\mathcal{N}(v)}) \oplus h_v)$$

Técnica usada por primera vez por Hamilton et al. (2017)[1] en su aproximación llamada GraphSage para generar embeddings de nodos de manera inductiva.

[1] Inductive Representation Learning on Large Graphs, Hamilton et al. (2017).

Concatenación - Interpolación Lineal

Otra técnica que podemos aplicar dentro del la "familia" de las concatenaciones es la interpolación lineal propuesta inicialmente por Pham et al. (2017)[1]

$$f_{update}^{(linearInt)}(h_v, m_{\mathcal{N}(v)}) = \alpha_1 \circ f_{update}(h_v, m_{\mathcal{N}(v)}) + (1 - \alpha_1) \odot h_v,$$

Donde $\alpha_1 \in [0,1]^d$es un vector que regula la influencia del valor de obtenido del embedding luego de la actualización junto con los mensajes del vecindario, con respecto al embedding actual y puede ser aprendido durante entrenamiento. La operación $\circ$ denota multiplicación elemento a elemento.

La representación final en esta forma de función de actualización es una interpolación lineal entre la información actualizada del vecindario y la representación previa.

[1] Column networks for collective classification, Pham et al. (2017)

Gated Updates

Otra forma popular de ver el proceso de paso de mensajes es verlo como un proceso de actualización con base en un índice temporal que en nuestro caso son las distintas operaciones de paso de mensajes.

Aunque las aproximaciones más tempranas de la utilización de redes recurrentes comienza alrededor del 2015 con un trabajo de Li et al. [1] con la propuesta de usar GRU, no es sino hasta 2019 que se ve el uso de LSTM por Selsam et al. 2019 [2]:

$$h_v^{(k)} = LSTM(h_v^{(k-1)}, m_{\mathcal{N}(v)}^{(k)}),$$

En general se ha visto que la utilización de este tipo de funciones de actualización basadas en RNNs facilitan la posibilidad de hacer más profunda a la red y aliviar el sobre-suavizamiento.

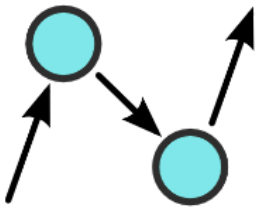
[1] Gated graph sequence neural networks, Li et al. (2015)

[2] Learning a SAAT solver from single-bit supervision, Selsam et al. (2019)

GNNs Multi-Relacionales

Grafos Multi-Relacionales

Existe gran cantidad de grafos donde las relaciones entre aristas pueden ser de más de un tipo, tomemos como ejemplo ConceptNet que es una "*red semántica diseñada para ayudar a computadores a entender los significados de las palabras usadas por las personas*".

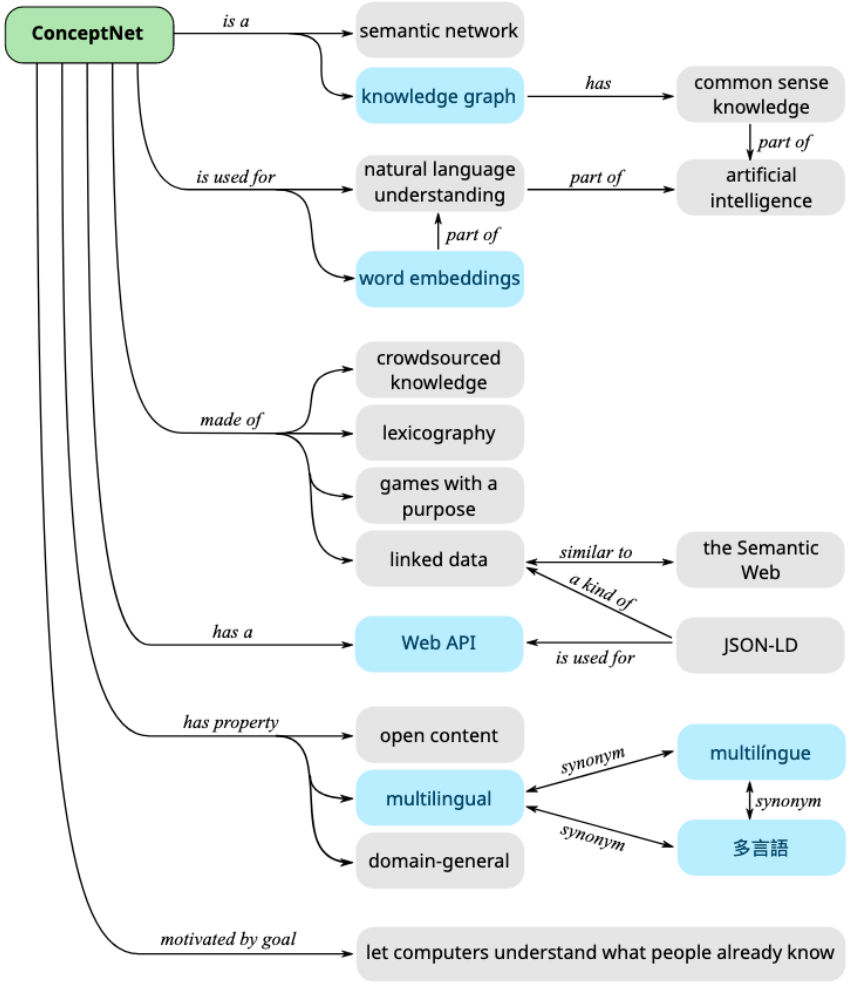


ConceptNet

An open, multilingual knowledge graph

Podríamos tomar muchos otros ejemplos: Relaciones/tipos de asociación entre personas, tipo de dependencia entre proteínas y funciones, etc.

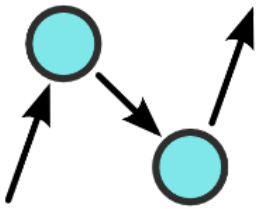
Grafos Multi-Relacionales



Fuente imagen: Página web ConceptNet: conceptnet.io

Grafos Multi-Relacionales

Existe gran cantidad de grafos donde las relaciones entre aristas pueden ser de más de un tipo, tomemos como ejemplo ConceptNet que es una "*red semántica diseñada para ayudar a computadores a entender los significados de las palabras usadas por las personas*".



ConceptNet

An open, multilingual knowledge graph

Podríamos tomar muchos otros ejemplos: Relaciones/tipos de asociación entre personas, tipo de dependencia entre proteínas y funciones, etc.

Grafos Multi-Relacionales

Sclichtkrull et al (2017) [1] propone modificar la función de agregación para permitir múltiples tipos de relación entre aristas mediante la inclusión de una matriz de transformación por tipo de relación:

$$m_{\mathcal{N}(v)} = \sum_{\tau \in \mathcal{R}} \sum_{u \in \mathcal{N}_\tau(v)} \frac{W_\tau h_u}{f_n(\mathcal{N}(v), \mathcal{N}(u))}$$

Donde f_n es una función de normalización que puede depender de los vecindarios tanto de un nodo v como de su vecino u. La aproximación de los autores es similar a la que discutimos antes con normalización sobre el vecindario pero con la adición de la posibilidad de agregar información a lo largo de diferentes tipos de aristas (τ).

[1] Modeling relational data with graph convolutional networks.- Schlichtkrull et al. (2017)

Grafos Multi-Relacionales

Un problema con la aproximación anterior es el elevado incremento en la cantidad de parámetros de nuestro modelo ya que ahora debemos tener una matriz de parámetros por cada tipo de conexión, lo que fácilmente nos puede llevar a sobreajustar. Para solucionar esto, Schlichtkrull et al. proponen en su lugar utilizar con conjunto de bases matriciales para formar la matriz de parámetros de manera que esta siempre sea formada a partir de una combinación lineal de esta base de matrices.

$$W_\tau = \sum_{i=1}^{b} \alpha_{i,\tau} B_i \,.$$

De esta manera, todas las matrices de relaciones son tomadas a partir de una combinación lineal de las mismas b bases matriciales: $B_1, \ldots, B_b$, dejando como únicos parámetros a encontrar el vector de pesos $\alpha_{i,\tau}$ que regula la "cantidad" de cada base en la matriz de un determinado tipo de conexión.

Grafos Multi-Relacionales

Para terminar, otra forma bastante natural de incluir información de las aristas en el modelo es modificando también la función de agregación de manera de concatenar esta información de las aristas a la representación latente del vector de embedding de un determinado vecino:

$$m_{\mathcal{N}(v)} = f_{aggregate}(\{h_u \oplus e_{(v,\tau,u)}, \forall u \in \mathcal{N}(v)\})$$

Fin