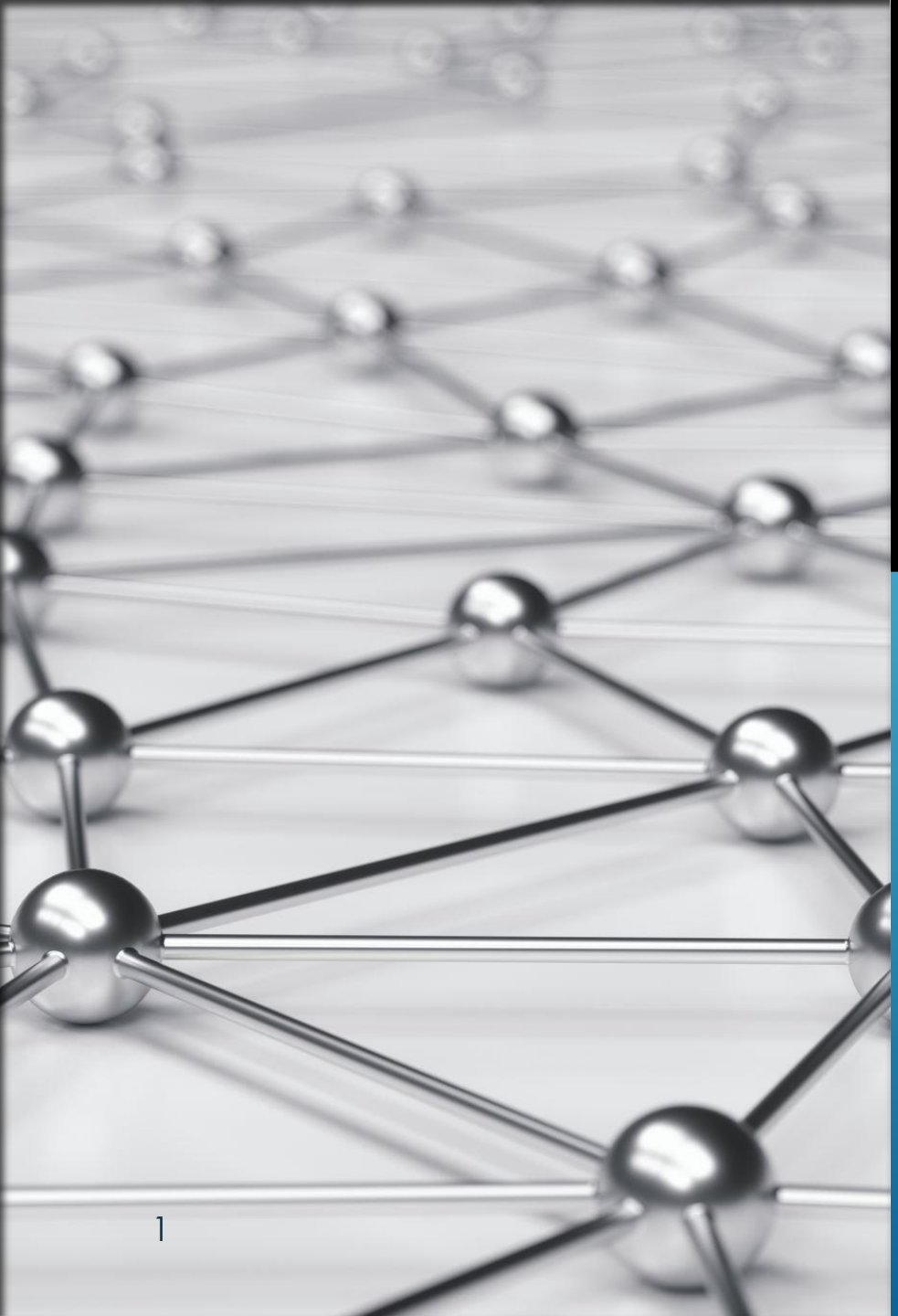


Graph Convolutional Network - GCN

Uma Introdução

- Emap - Redes Neurais e Deep Learning
- Prof. Renato Rocha
- Aluno: Gilberto Ramos
- Data: 29.10.2021

AGENDA

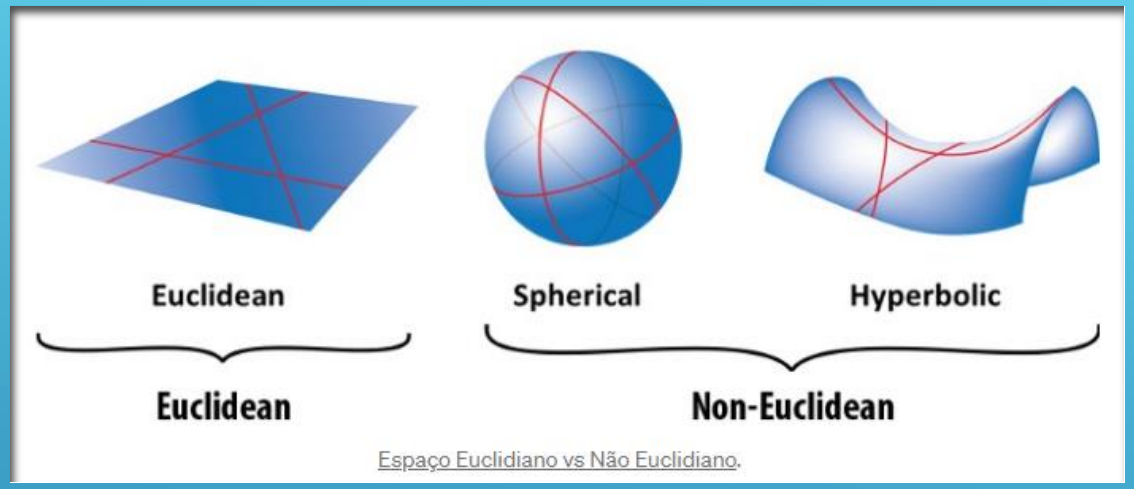
01 Introdução

02 Introdução a teoria de Grafos

03 Introdução as GNNs e GCNs

04 GCN- Estado da arte

05 Aplicação GCN em Python



Espaço Euclidiano vs Não Euclidiano.

dados euclidianos x não euclidianos

a não regularidade das estruturas de dados

INTRODUÇÃO

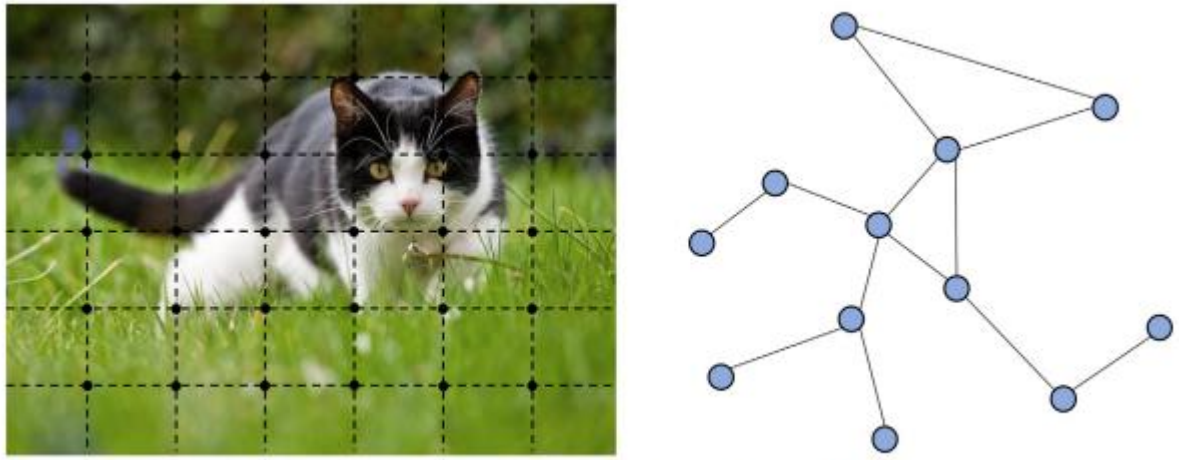
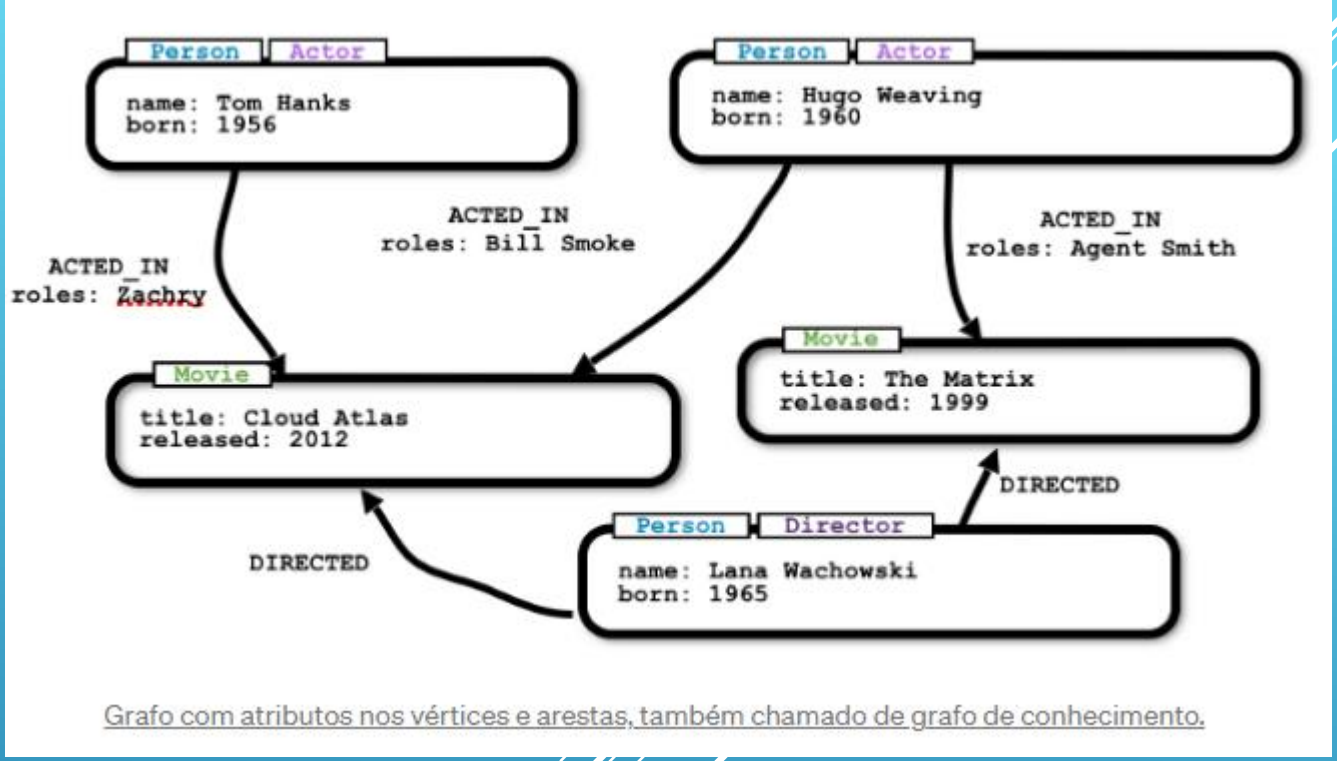


Fig. 1. Left: image in Euclidean space. Right: graph in non-Euclidean space.

INTRODUÇÃO

- **Como a IA aprende a estrutura?**



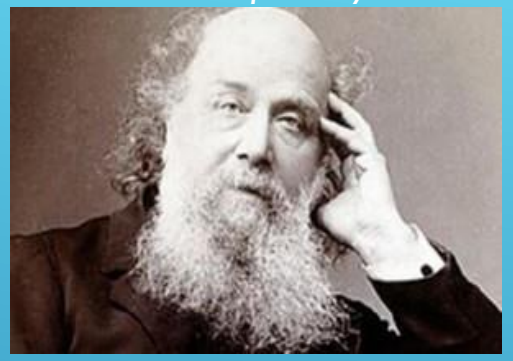
Grafo com atributos nos vértices e arestas, também chamado de grafo de conhecimento.

- *Além dos vértices e arestas, os grafos também podem conter um conjunto de atributos que descrevem cada vértice. Nesse caso, o grafo pode ser definido como:* $G = (V, E, X)$, *onde* X *é uma matriz de propriedades por vértices.*

INTRODUÇÃO A TEORIA DE GRAFOS

Matemáticos que introduziram o tema:

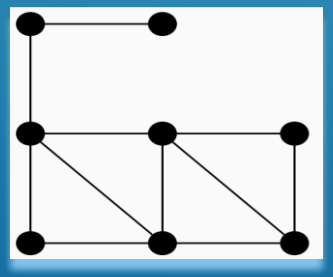
James Josheph Sylvester (1814 – 1897).



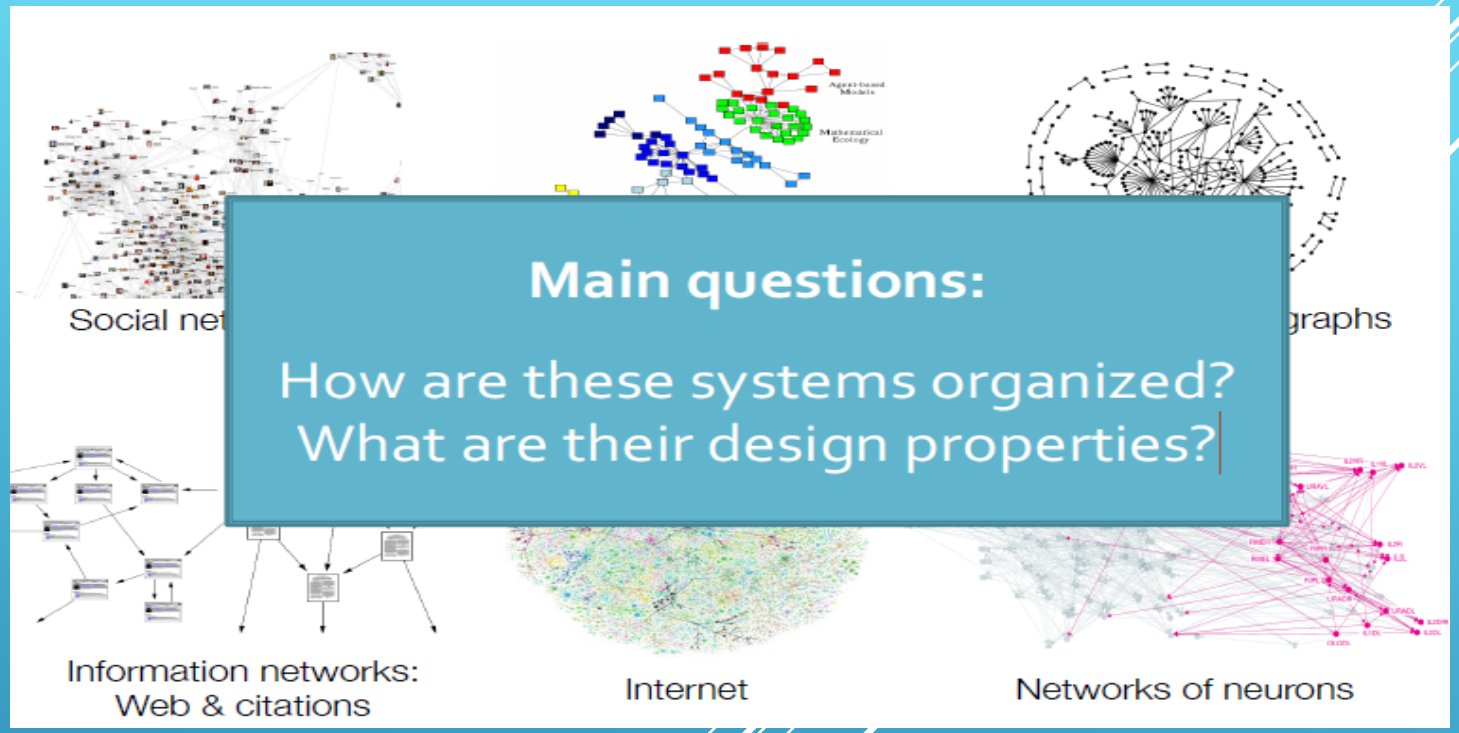
Leonhard Euler (1707-1783)

Um grafo G é composto por um conjunto não-vazio de vértices (nós) V(G), um conjunto de arestas E(G) e uma função de incidência Ψ_G , a qual relaciona arestas de E(G) com pares de vértices (não necessariamente distintos) de V(G) [14].

Exemplo de um Grafo:



INTRODUÇÃO A TEORIA DE GRAFOS

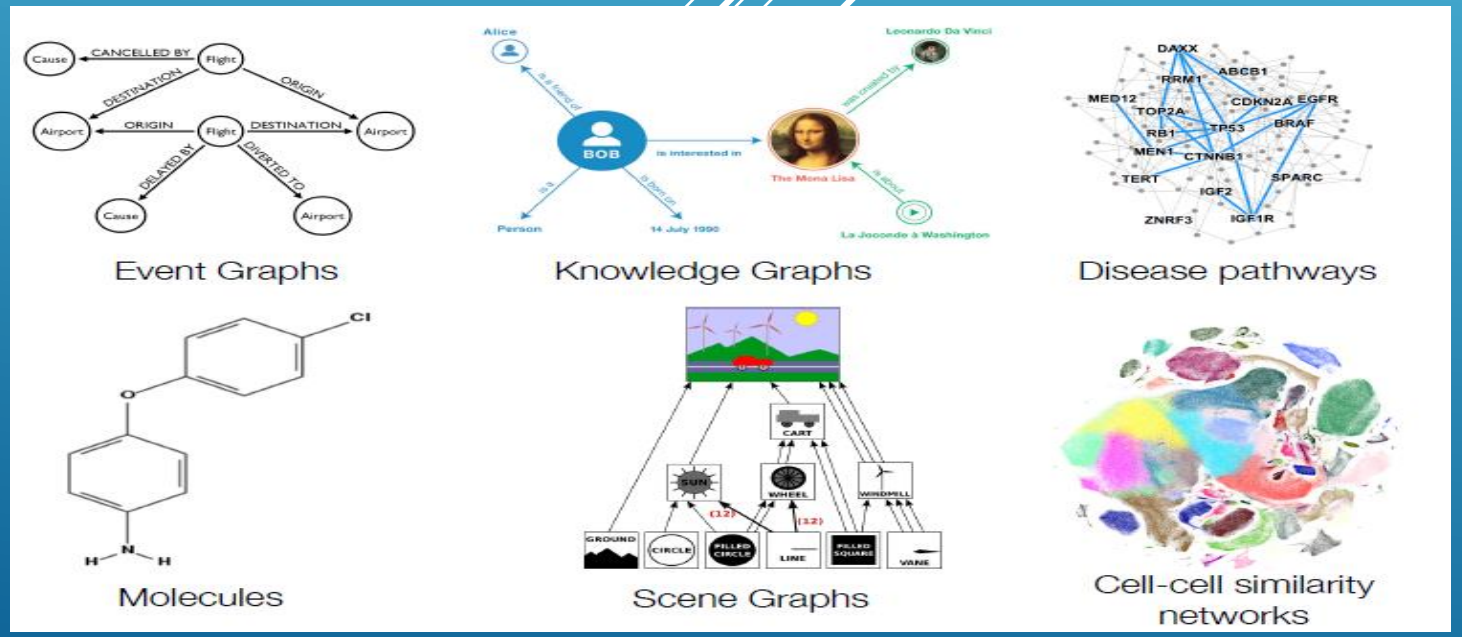


Main questions:

How are these systems organized?
What are their design properties?

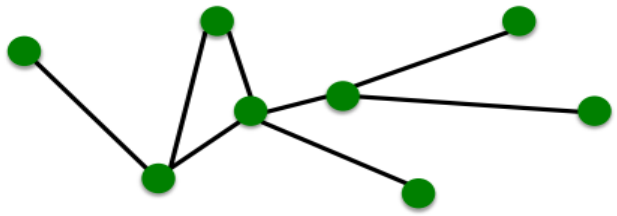
Main questions:

How do we take advantage of relational structure for better prediction?



INTRODUÇÃO A TEORIA DE GRAFOS

Complex domains (knowledge, text, images, etc.) have rich relational structure, which can be represented as a **relational graph.**



- **Objects:** nodes, vertices — N
- **Interactions:** links, edges — E
- **System:** network, graph — $G(N,E)$

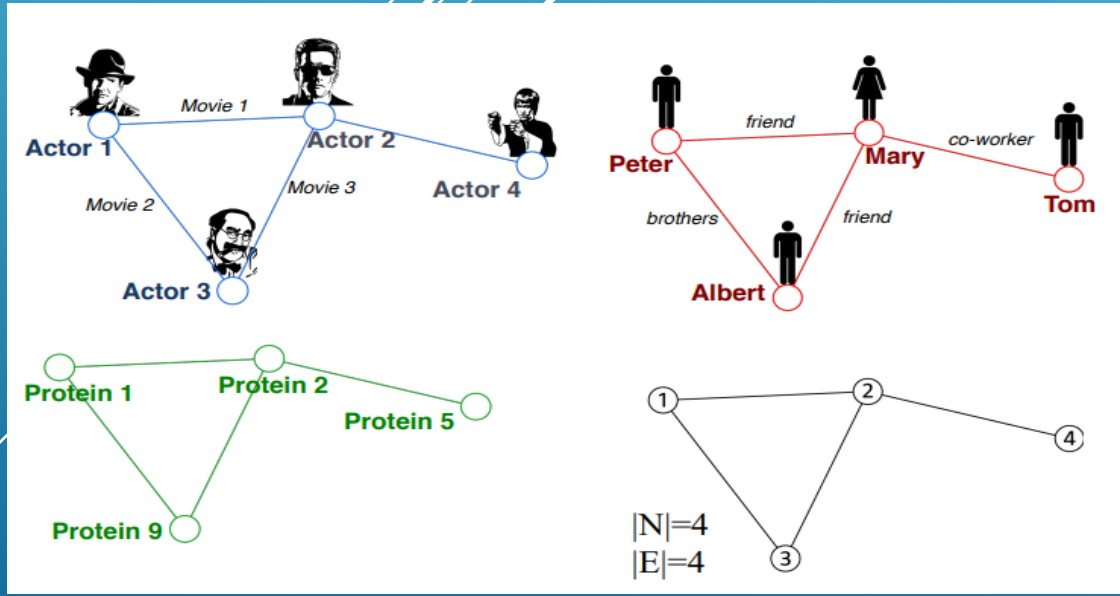
Networks or Graphs?

Network often refers to real systems
- Web, Social network, Metabolic network

Language: Network, node, link

Graph is a mathematical representation of a network
- Web graph, Social graph, Knowledge Graph

Language: Graph, vertex, edge

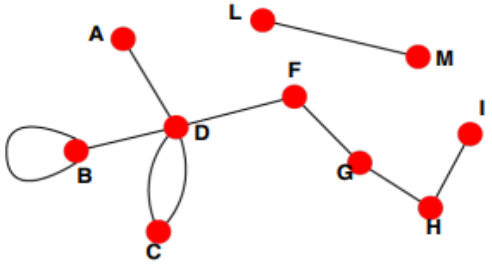


INTRODUÇÃO A TEORIA DE GRAFOS

Directed vs Undirected Graphs

Undirected

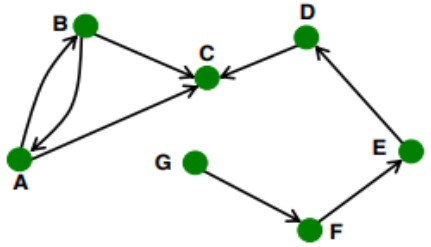
- **Links:** undirected (symmetrical, reciprocal)



- **Examples:**
  - Collaborations
  - Friendship on Facebook

Directed

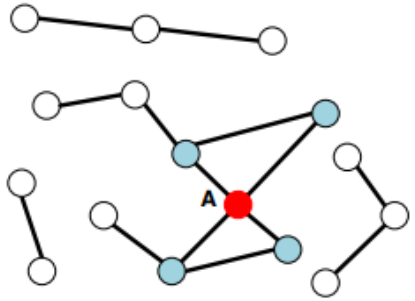
- **Links:** directed (arcs)



- **Examples:**
  - Phone calls
  - Following on Twitter

Nodes Degrees

Undirected

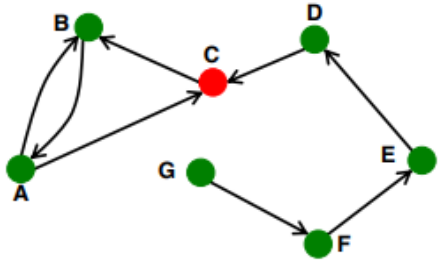


Node degree, k_i: the number of edges adjacent to node i

$$k_A = 4$$

Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

Directed



In directed networks we define an **in-degree** and **out-degree.**

The (total) degree of a node is the sum of in- and out-degrees.

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

$$\bar{k} = \frac{E}{N} \qquad \overline{k^{in}} = \overline{k^{out}}$$

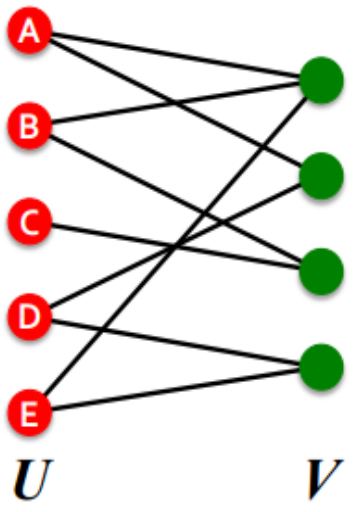
Source: Node with $k^{in} = 0$

Sink: Node with $k^{out} = 0$

INTRODUÇÃO A TEORIA DE GRAFOS

Bipartite Graph

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V; that is, U and V are **independent sets**

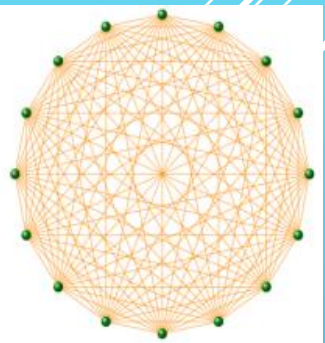


- **Examples:**
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)
- **"Folded" networks:**
  - Author collaboration networks
  - Movie co-rating networks

Complete Graph

The **maximum number of edges** in an undirected graph on N nodes is

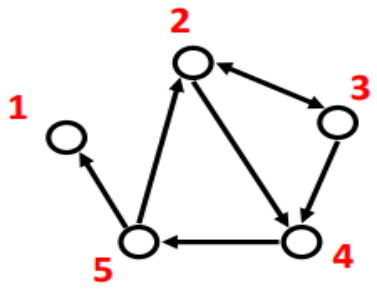
$$E_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$$



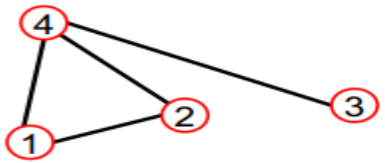
An undirected graph with the number of edges $E = E_{max}$ is called a **complete graph**, and its average degree is N-1

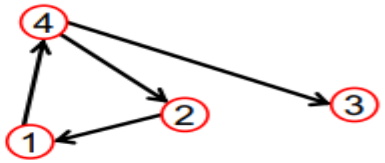
INTRODUÇÃO A TEORIA DE GRAFOS

- **Represent graph as a set of edges:**
  - (2, 3)
  - (2, 4)
  - (3, 2)
  - (3, 4)
  - (4, 5)
  - (5, 2)
  - (5, 1)



Representing Graphs: Adjacency Matrix





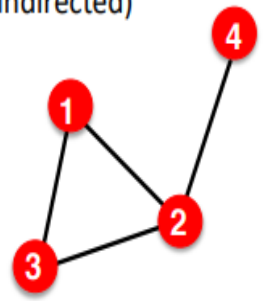
$A_{ij} = 1$ if there is a link from node i to node j

$A_{ij} = 0$ otherwise

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

More types of graphs

- **Unweighted**
  (undirected)



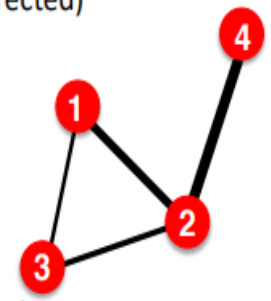
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0 \qquad A_{ij} = A_{ji}$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Examples: Friendship, Hyperlink

- **Weighted**
  (undirected)



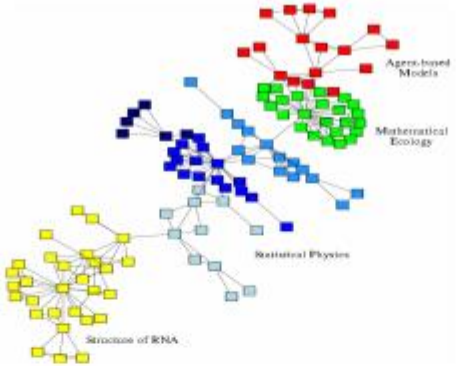
$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0 \qquad A_{ij} = A_{ji}$

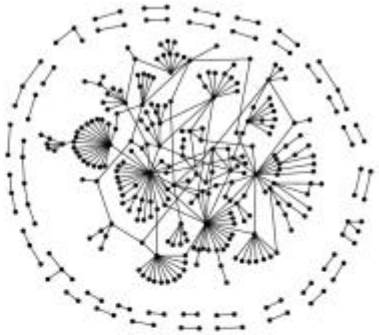
$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

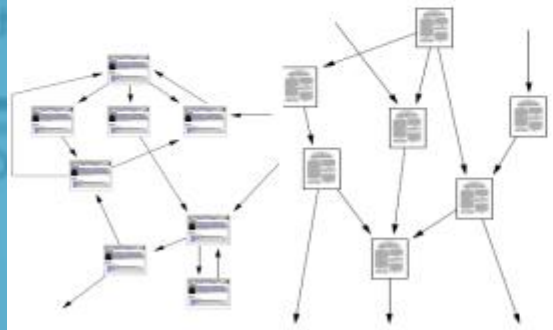
Social networks



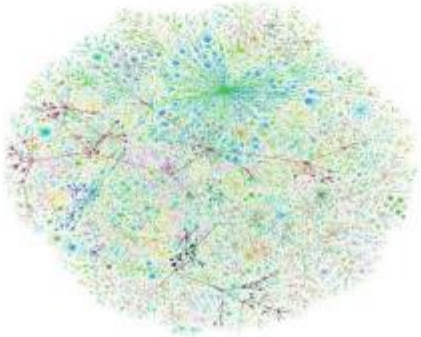
Economic networks



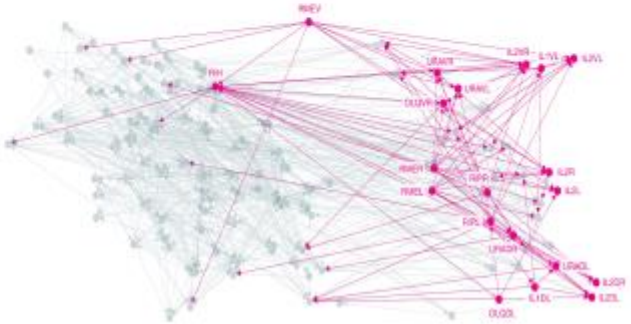
Communication networks



Information networks:
Web & citations

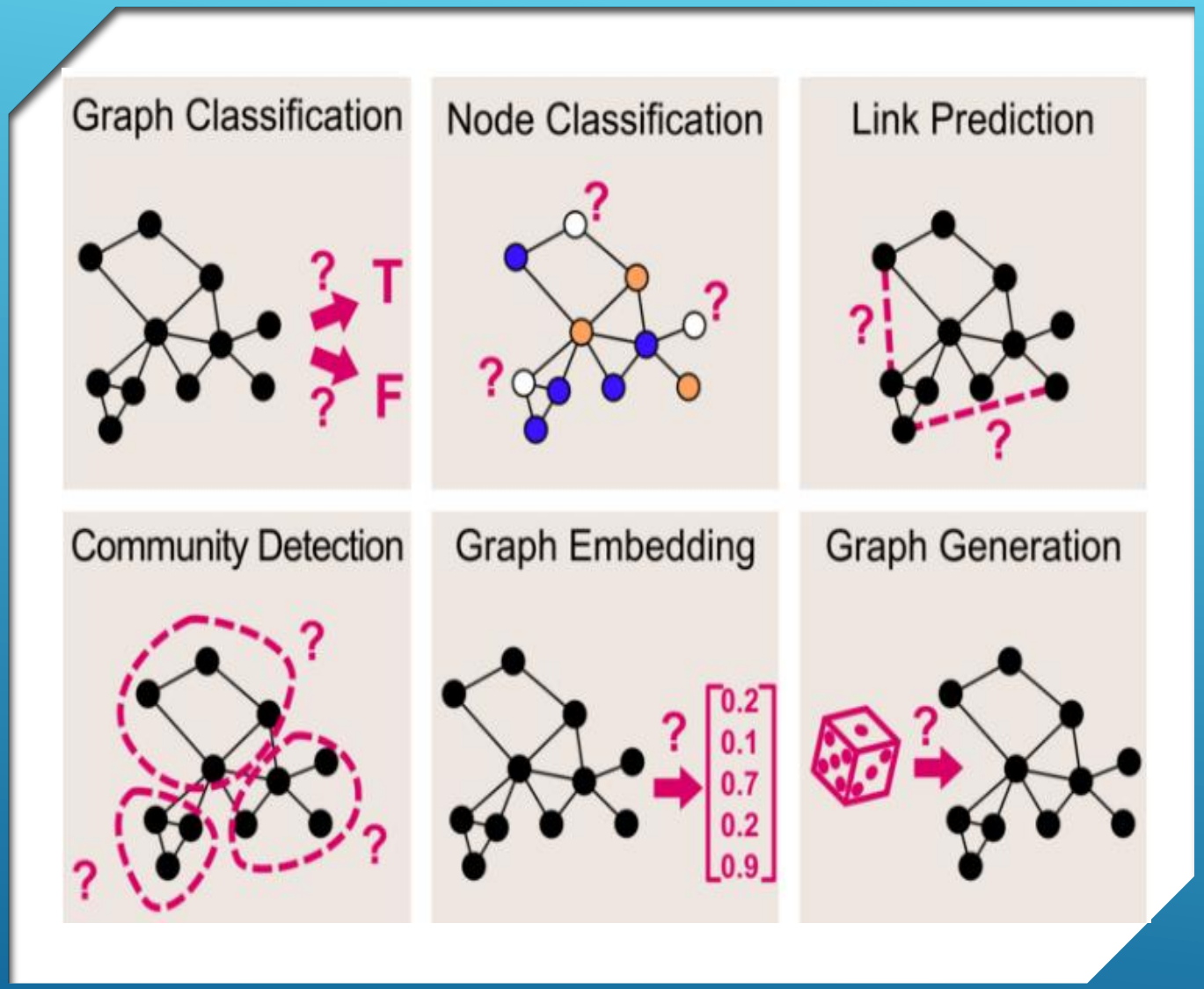


Internet



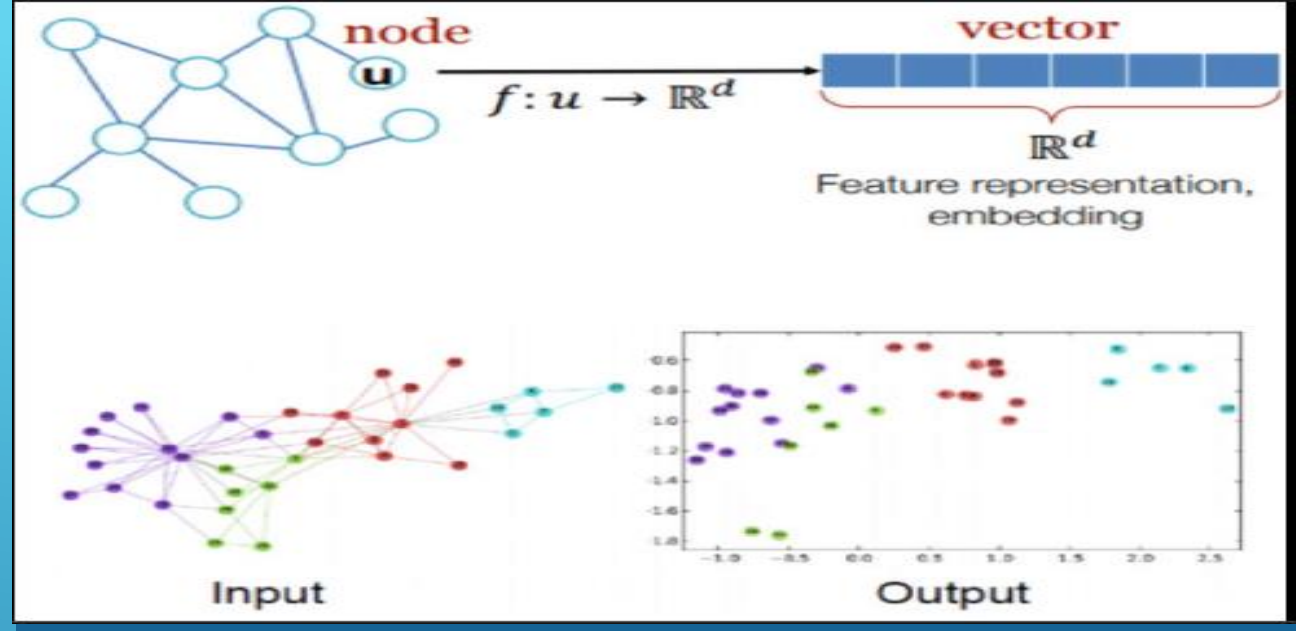
Networks of neurons

MODELAGEM USANDO GRAFOS



TAREFAS DE APRENDIZADO DE MÁQUINA EM GRAFOS

REDES NEURAIS DE GRAFOS: GNN



CNN ⟶ GNN

Objetivo: substituir imagens por grafos

- Numa CNN cada **pixel** é representado por um vetor n-dimensional (3D-RGB no input)
- Numa **GNN** cada **nó(vértice)** do grafo de entrada é representado por um vetor n-dim

Propostas em [GMS05; Sca+09], com o objetivo de aprender, através de exemplos, uma função que mapeasse tanto um grafo G a um vetor de números reais, como um nó v.

- $\tau(G) = \mathbb{R}^m$
- $\tau(G, v) = \mathbb{R}^m$

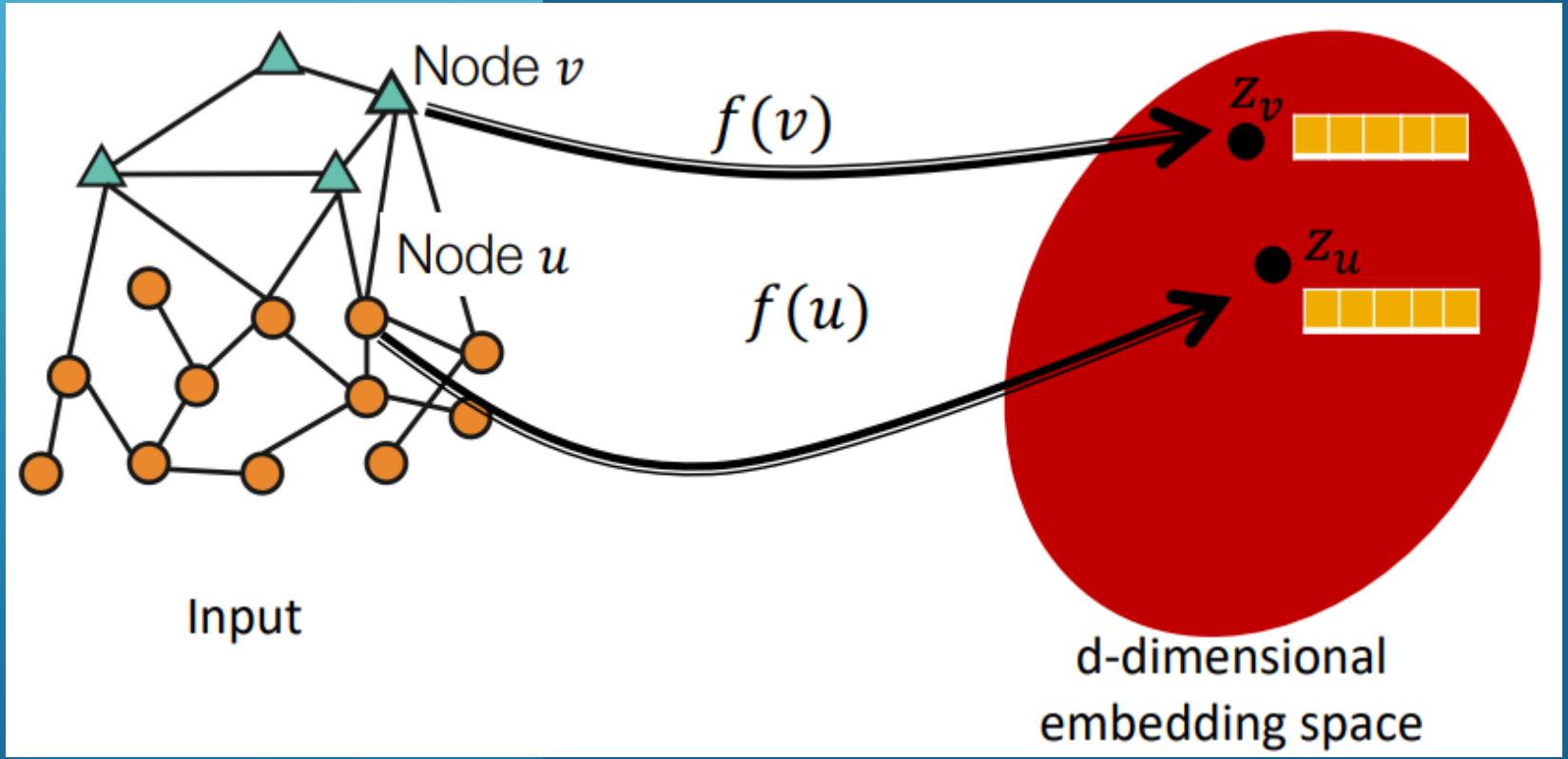
Cada nó $v \in V$ possui um vetor de características $\boldsymbol{l}_v$.

$\boldsymbol{x}_v = f_{\boldsymbol{w}}(\boldsymbol{l}_v, \boldsymbol{x}_{ne[v]}, \boldsymbol{l}_{ne[v]})$ é um vetor de características de v que depende de se seus vizinhos, $ne[v]$.

$\boldsymbol{o}_v = g_{\boldsymbol{w}}(\boldsymbol{l}_v, \boldsymbol{x}_v)$ é a saída de v.

$f_{\boldsymbol{w}}$ e $g_{\boldsymbol{w}}$ são funções parametrizadas por pesos $\boldsymbol{w}$. Na prática, f e g são redes neurais, MLPs ou RNNs parametrizadas por conjuntos de pesos diferentes ($\boldsymbol{w}_f$ e $\boldsymbol{w}_g$).

GNN: EMBEDDING NODES



REDES NEURAIS DE GRAFOS: GNN

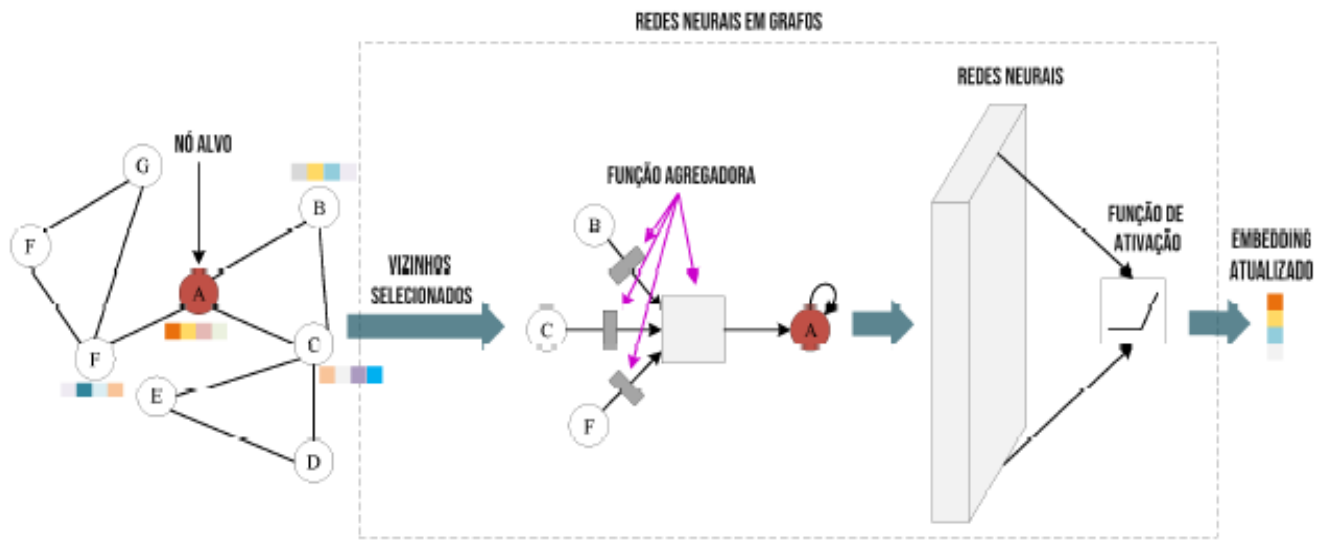


Figura 2. Processamento de uma GNN. Fonte: [Zeng and Tang 2021].

- Graph Neural Networks (GNNs), modelo proposto por [Scarselli et al. 2009]
- Objetivo do modelo
- O processo de aprendizagem
- O funcionamento de uma GNN. [Zeng and Tang 2021]

Encontrar $\boldsymbol{w}$ de forma a aproximar $\boldsymbol{o}_v$ de uma saída esperada.

A base de exemplos é composta por p triplas (G, v, t), onde G é um grafo, v é um vértice de G e t é a saída esperada para uma função $\phi_{\boldsymbol{w}}(G_i, v_i) = \boldsymbol{o}_v$.

O valor de $\boldsymbol{w}$ é encontrado minimizando uma função de erro, como o erro quadrático:

$$e_{\boldsymbol{w}} = \sum_{i=1}^{p} (t_i - \phi_{\boldsymbol{w}}(G_i, v_i))^2$$

OBJETIVO DA GNN

GNN

ALGORITMO DE APRENDIZADO

Passo 1: Estabilização de x_v

- x_v depende de $x_{ne[v]}$.
- Necessário atualizar iterativamente $x_v(t) = f_{\boldsymbol{w}}(l_v, x_{ne[v]}(t-1), l_{ne[v]})$ até alcançar um ponto fixo estável em $t = T$.

Passo 2: cálculo dos pesos

- Calcular o gradiente $\frac{\partial e_{\boldsymbol{w}}(T)}{\partial \boldsymbol{w}}$ e atualizar $\boldsymbol{w}$ usando descida de gradiente.

INTERPRETAÇÃO

- nós representam objetos ou conceitos, descritos por um vetor de características.
- arestas representam relações (estradas, ligações moleculares).
- a GNN otimiza seu desempenho em uma tarefa (*e.g.* classificação, regressão) modelando as interações entre os objetos em seus pesos $\boldsymbol{w}$.

PROPRIEDADES

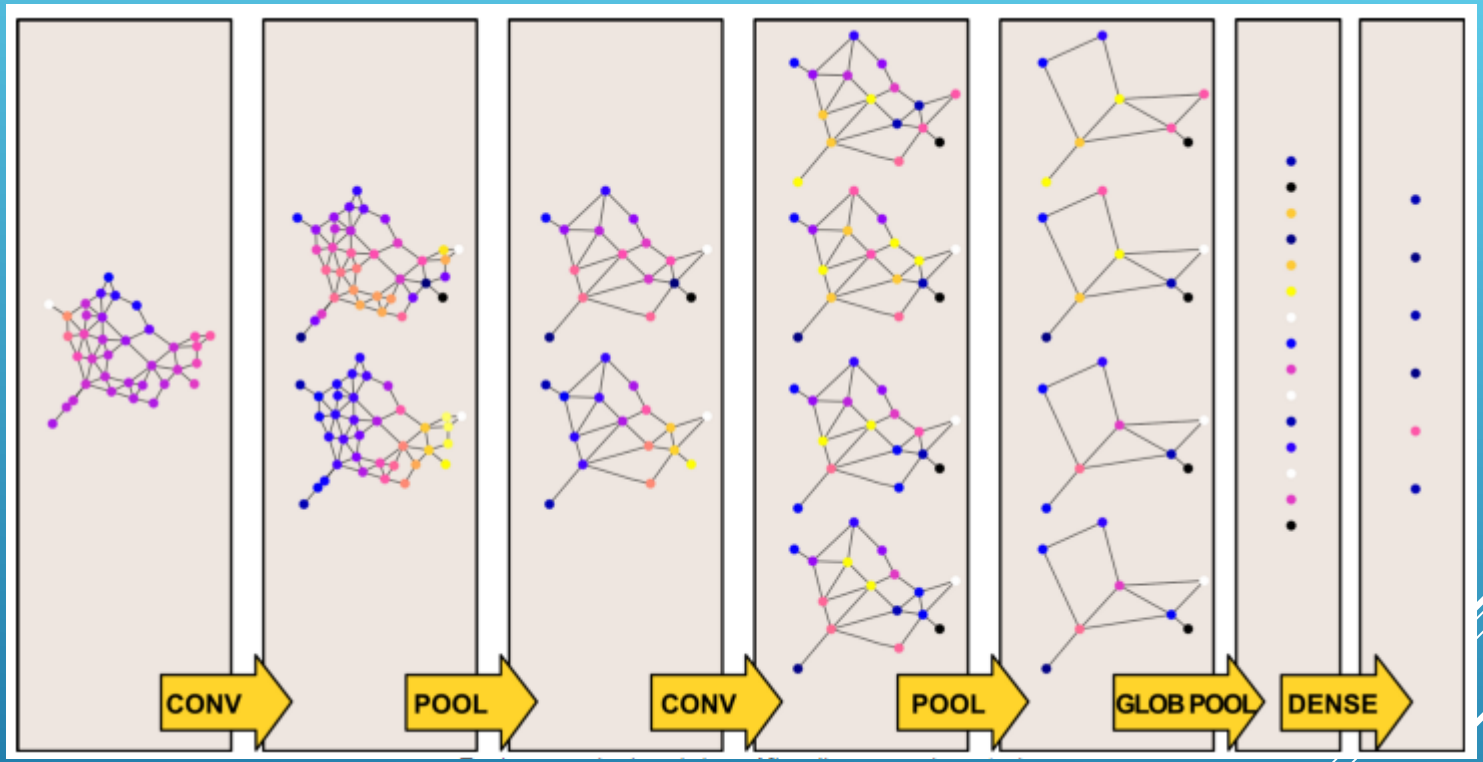
Compartilhamento de parâmetros
Ambas as funções $f_{\boldsymbol{w}}$ e $g_{\boldsymbol{w}}$ são aplicadas a todos os nós dos grafos, o que significa que $\boldsymbol{w}$ é utilizada em mais de um local na entrada dos dados.

Grafos de tamanho arbitrário
A ordem na qual $ne[v]$ é utilizado em $f_{\boldsymbol{w}}$ importa? Se não, $f_{\boldsymbol{w}}$ pode ser uma *função de agregação* (e.g. máximo, média), ignorando a ordem e quantidade de vértices.

Permite processar grafos de tamanhos arbitrários!

GCN

- Uma rede convolucional de grafos (GCN), implementa a convolução em um **grafo**, ao invés de em uma imagem composta de pixels.



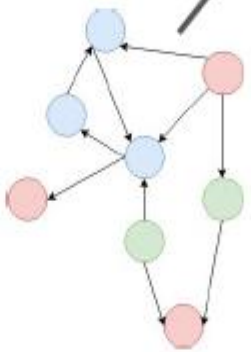
GCN-GRAPH CONVOLUTION NETWORK

GCN

Graph Convolution Layer

- Idêntica à convolução tradicional, exceto por:
  - Vizinhança não é necessariamente em grid
  - Não temos mais um peso específico para cada vizinho
  - Ao invés disso: aplicamos uma transformação a cada feature

$$X_i' \leftarrow \sum_{j \in \mathcal{N}(i) \cup \{i\}} \Theta \times X_j$$

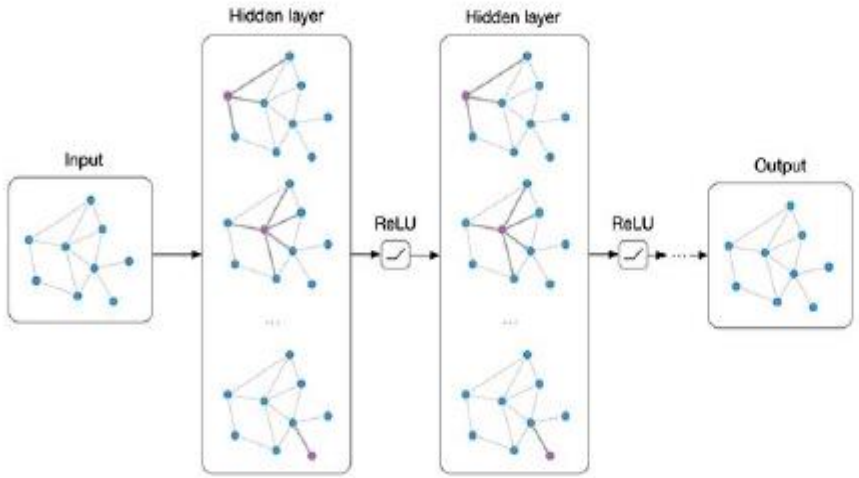


```python
import torch

# Nº of nodes
N = 8

# Adjacency matrix (NxN)
A = torch.tensor([
    [0,1,0,1,0,0,0,0],
    [0,0,1,0,0,0,0,0],
    [0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,1],
    [0,0,0,1,0,0,0,0],
    [0,0,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0],
]).float()

# Add self-edges
A += torch.eye(N)

# Dimensionality of feature vectors
d = 16
# Initial feature vectors
x = torch.randn(N, d)

# Parameters
Θ = torch.nn.Linear(d, d, bias=False)

# GC layer
x = torch.mm(A,Θ(x))
```

GCN

Graph Convolution Layer

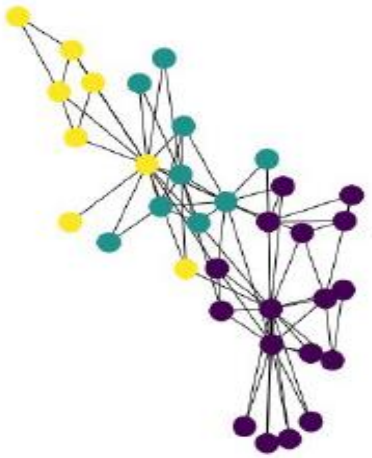
- Podemos empilhar múltiplas camadas de GC



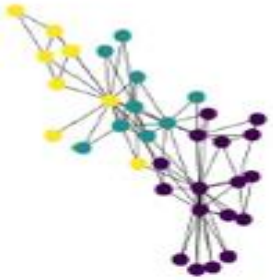

```python
import torch

# Nº of nodes
N = 8

# Adjacency matrix (NxN)
A = torch.tensor([
    [0,1,0,1,0,0,0,0],
    [0,0,1,0,0,0,0,0],
    [0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,1],
    [0,0,0,1,0,0,0,0],
    [0,0,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0],
]).float()

# Add self-edges
A += torch.eye(N)

# Dimensionality of feature vectors
d = 16
# Initial feature vectors
x = torch.randn(N, d)

# Parameters
θ1 = torch.nn.Linear(d, d, bias=False)
θ2 = torch.nn.Linear(d, d, bias=False)

# GC layers
x = torch.relu(torch.mm(A,θ1(x)))
x = torch.relu(torch.mm(A,θ2(x)))
```

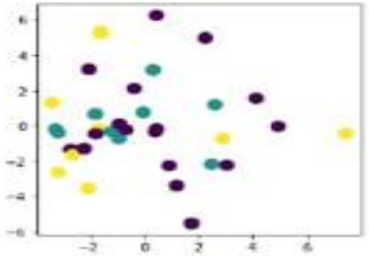
GCN

Graph Convolution Layer



```python
import torch
import numpy as np
import networkx as nx
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from networkx.algorithms.community import greedy_modularity_communities

# Get Zachary's Karate Club graph
G = nx.karate_club_graph()
A = torch.tensor(nx.to_numpy_matrix(G)).float()

# Split graph into communities
communities = greedy_modularity_communities(G)
labels = np.zeros(len(G))
for i in range(len(communities)):
    labels[list(communities[i])] = i+1

# Draw graph
nx.draw(G, node_color=labels)
plt.show()
```

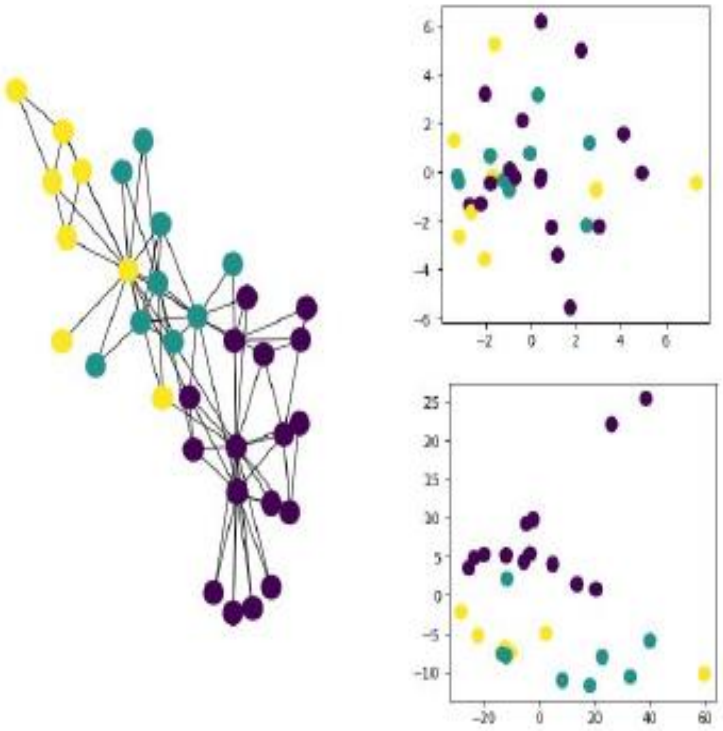





```python
# Util to project and plot feature vectors
def plot_feature_vectors(x):
    pca = PCA(n_components=2)
    x_proj = pca.fit_transform(x.detach().numpy())
    plt.figure(figsize=(4, 4))
    plt.scatter(x_proj[:,0], x_proj[:,1], c=labels, s=100)
    plt.show()

# Dimensionality of feature vectors
d = 100
# Initial feature vectors
x = torch.randn(len(G), d)

# Parameters
θ = [torch.nn.Linear(d, d, bias=False) for i in range(3)]

# Plot before GCs
plot_feature_vectors(x)
```

GCN

Graph Convolution Layer



```
# Plot before GCs
plot_feature_vectors(x)

# Update feature vectors with repeated GCs
for i in range(3):
    x = torch.relu(torch.mm(A,θ[i](x)))

# Plot after GCs
plot_feature_vectors(x)
```

GCN: Message Passing

obtendo a mensagem dos nós vizinhos

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

Atualização do estado do nó usando o estado oculto anterior e uma nova mensagem

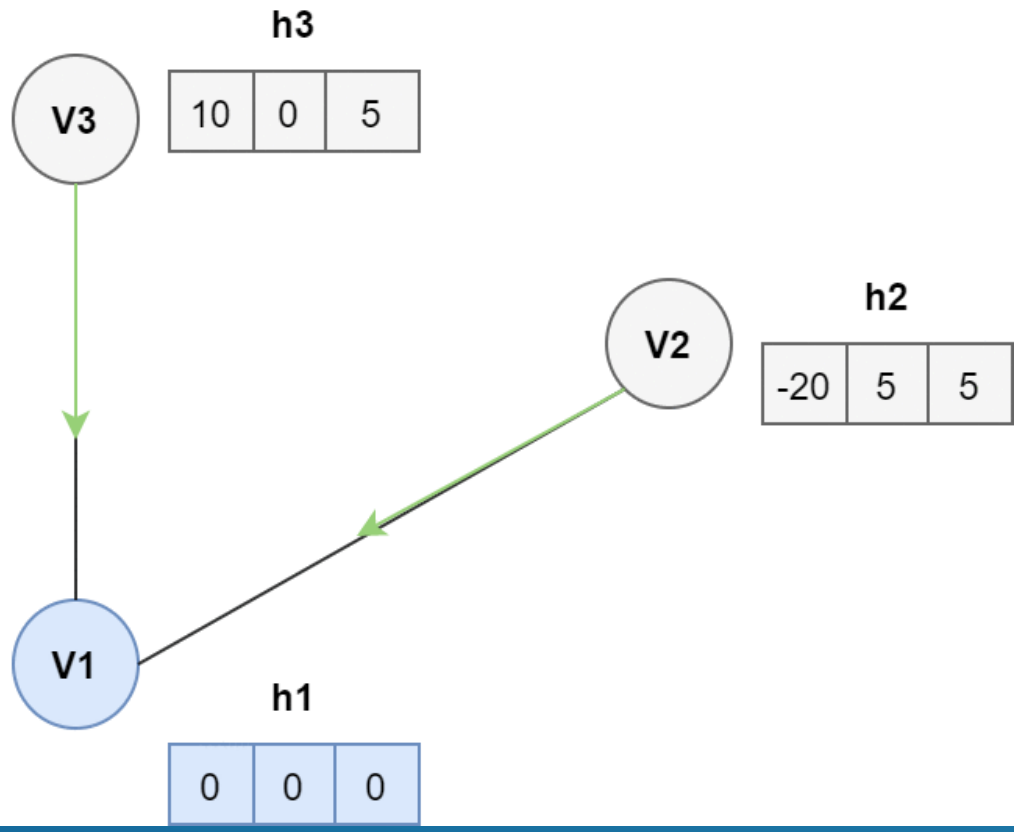
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

$$m_v^{t+1} = \sum_{w \in N(v)} h_w^t$$

$$h_v^{t+1} = average(h_v, m_v^{t+1})$$

ht - hidden state for each node

Message Passing for Node V1
for t = 1



ESTADO DA ARTE – APLICAÇÕES DE GCN

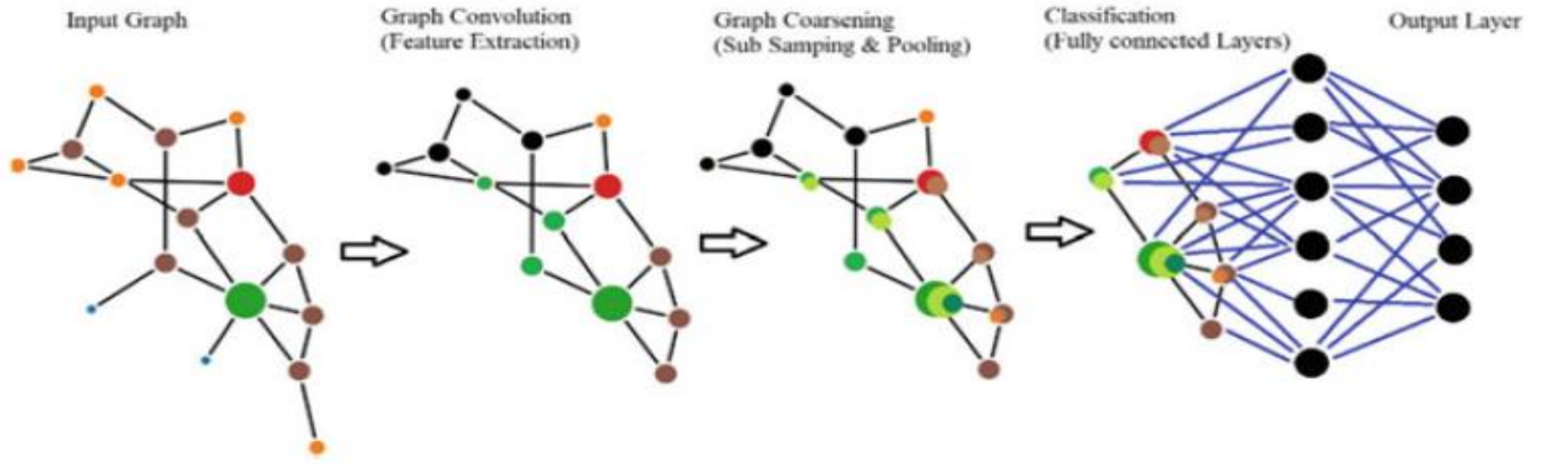


Fig. 1 Architecture of GCN

ESTADO DA ARTE – ARCHITECTURE OF GCN

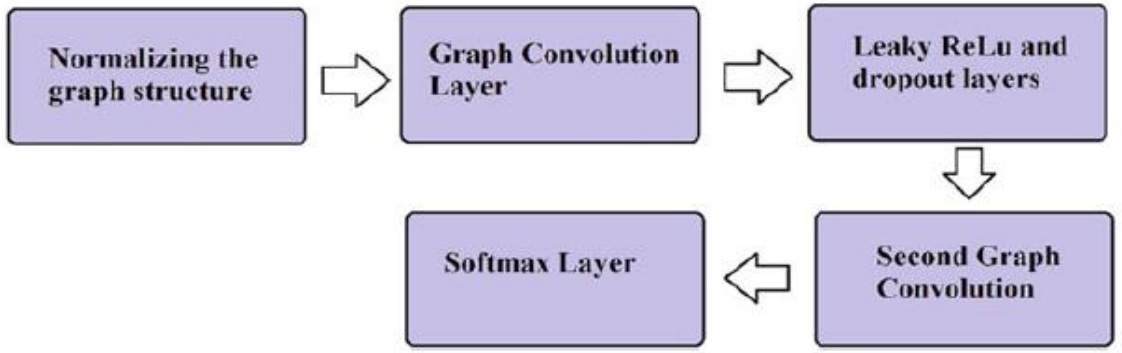


Fig. 2 Stages of a simple GCN architecture

ESTADO DA ARTE : SEMI-SUPERVISED LEARNING WITH GCN

- Classificação de nós de um grafo
- **Poucos labels** são conhecidos
- Como **embeddings** acumulam informação local, é possível treinar com menos labels

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Semi-Supervised Classification with Graph Convolutional Networks
Thomas N. Kipf, Max Welling

https://arxiv.org/abs/1609.02907

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

• Estado da arte → Related Work

Tabela 1 – Summary of Results of work based on GCN

S. No.	Research Paper	Application	Methodology	Fig.Number	Results
1	Benamira et al. [3]	Fake News detection	Embbending of articles + Graph Construction + Classification(GCN+AGNN)	Fig. 3	84.94% +- 2.30%
2	Bian et al. [4]	Rumour detection	Construct propagation and dispersion graphs+calculate the high level node representations + root feature enhacement+representation of propagation and dispersion for rumour classification	Fig. 4	96.1%
3	Dong et al. [5]	Multiple rumour source detection	GCNSI+LPSI+NetSleuth	Fig. 5	0.63 (error distance)
4	Li and GoldWasser [6]	Political perspective detection	GCN+SkipThought/GCN+HLSTM	Fig. 6	91.74%
5	Wu et al. [7]	Social spammer detection	GCN + markov random field(MRF)	-	83.9%
6	AlJohany et al. [8]	Bot prediction on social networks	Bot Detection using SNA(Community detection, degree and triangle, clustering coeficiente) + GCN	Fig. 7	71%
7	Ying et al. [9]	Web-scale recommender systems	PinSage(random-Walk graph convolutional network(GCN))	-	67%(hit-rate)
8	Yao et al. [10]	Text-classification	Text graph convolutional networks (text GCN)	-	86.34%
9	Marcheggiani and Titov [11]	Semantic role labelling	Word-embbending + BiLSTM Encoder+GCN encoder+Classifier	_	88.0%(F1-Score)

ESTADO DA ARTE: **SEMI-SUPERVISED FAKE NEWS DETECTION USING GCN**

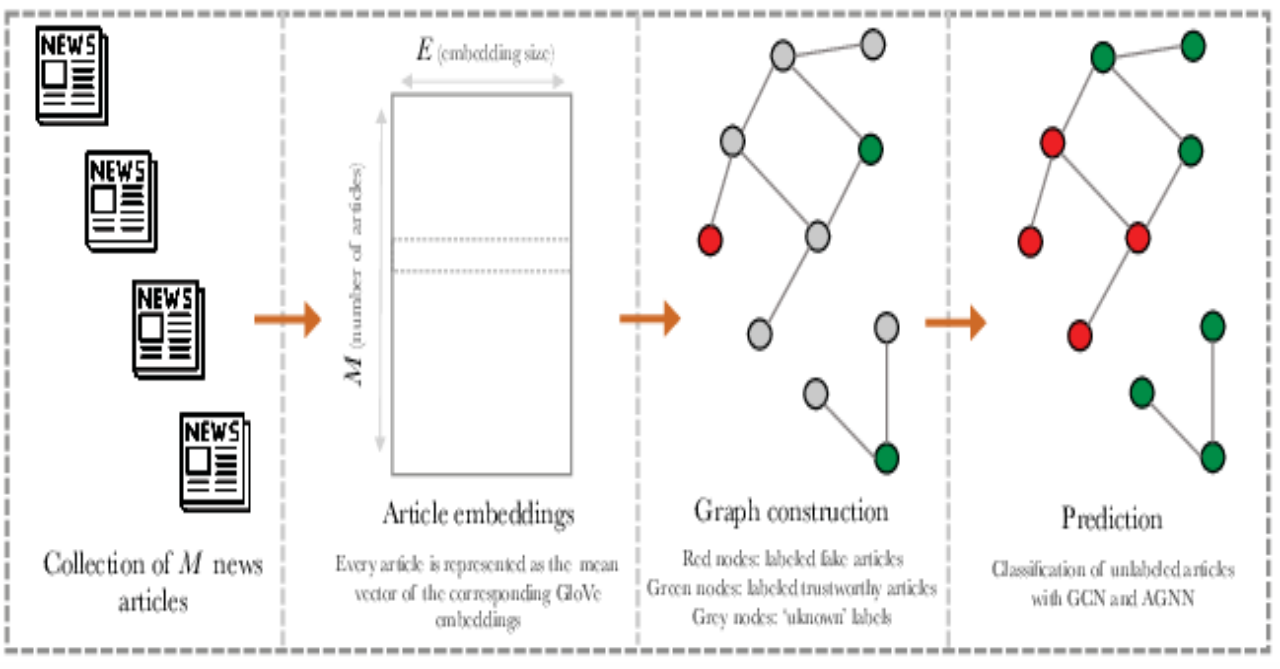


Fig. 1. Illustration of the proposed approach: M denotes the number of articles (real and fake) and E is the dimension of our GloVe embeddings (in our case, M = 150, E = 100). Finally, we use k = 4 nearest neighbours to build the graph

[Published in 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) 2019]

- Estado da arte → bi-direcional GCN for Rumour Detection

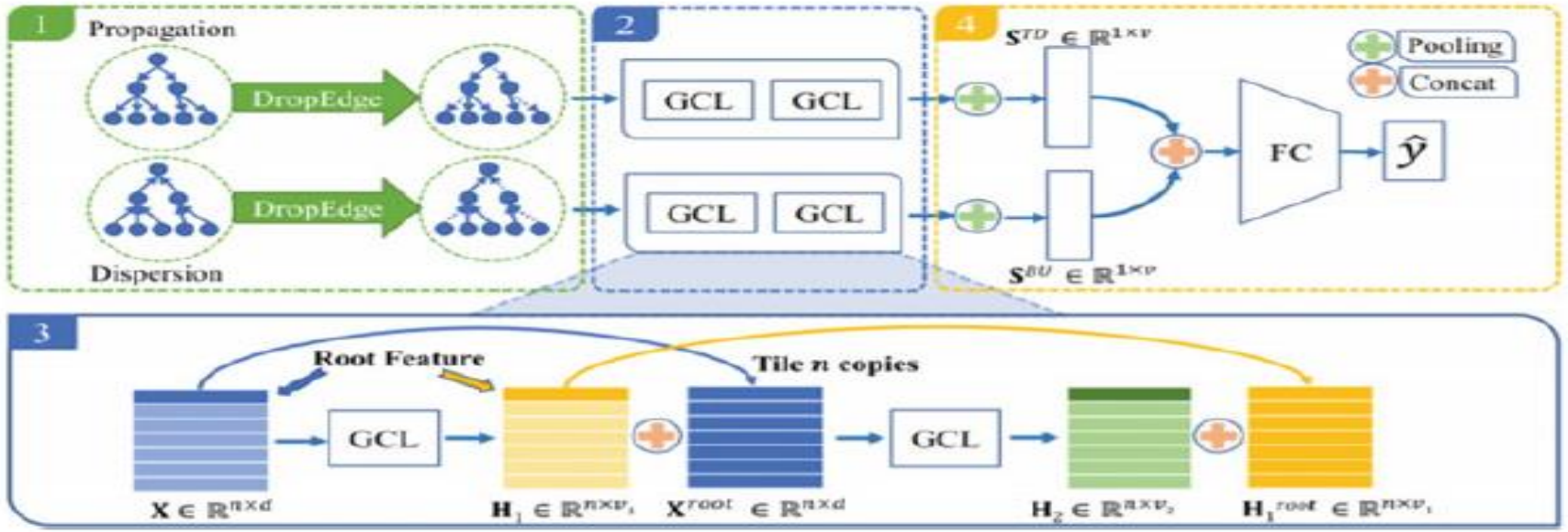


Fig. 4 Architecture of bi-directional GCN for rumour detection [4]

ESTADO DA ARTE → GCNSI FOR RUMOUR DETECTION

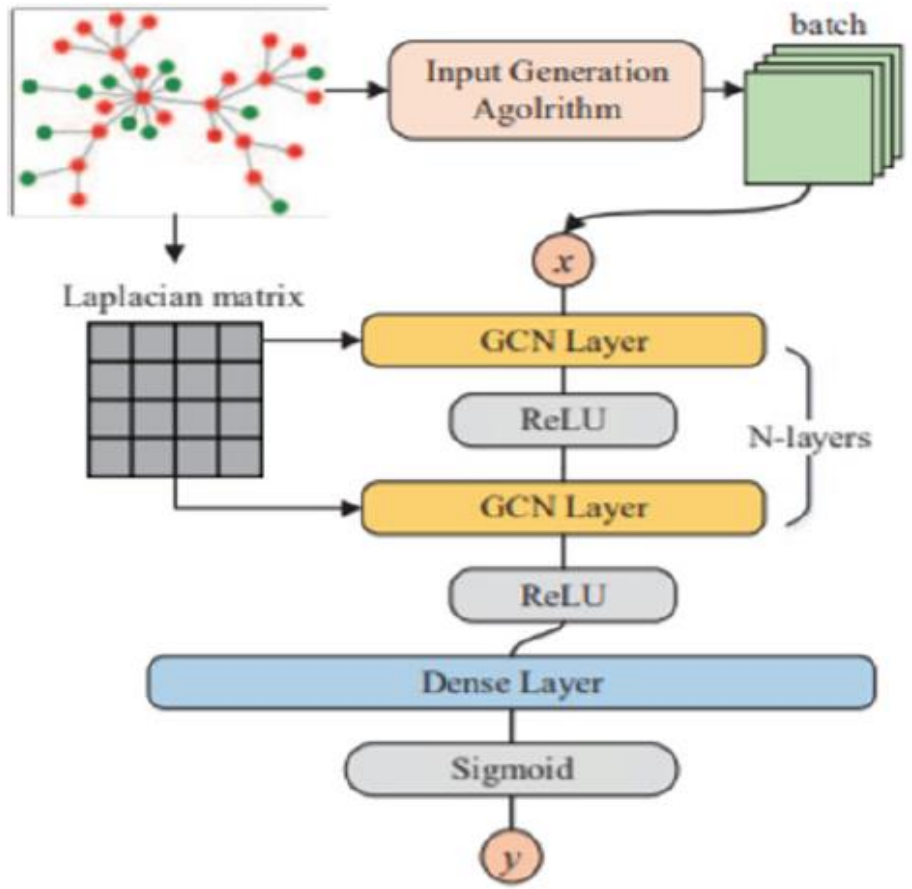


Fig. 5 Architecture of GCNSI for rumour detection [5]

ESTADO DA ARTE → BOT DETECTION IN POLITICAL PERSPECTIVE

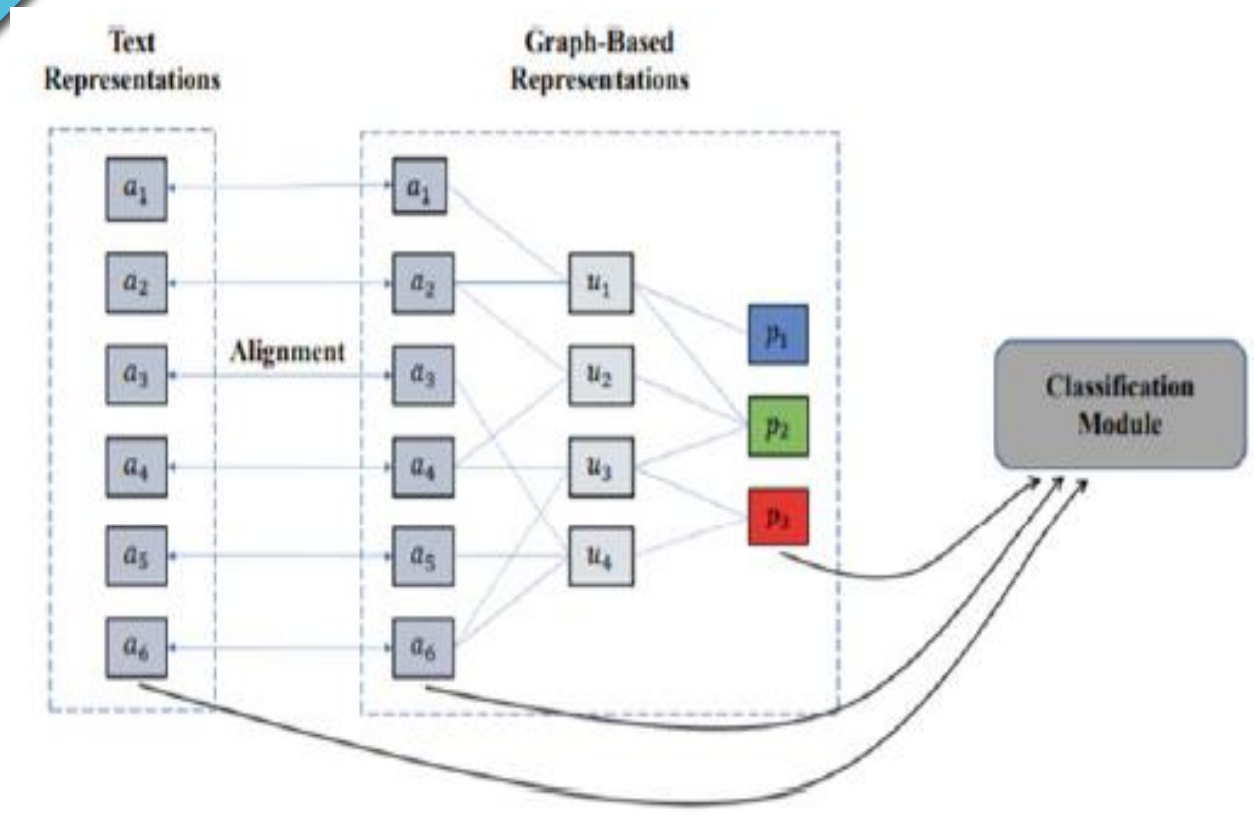


Fig. 6 Architecture of SNA + GCN for bot detection [6]

ESTADO DA ARTE → BOT DETECTION ON SOCIAL NETWORKS

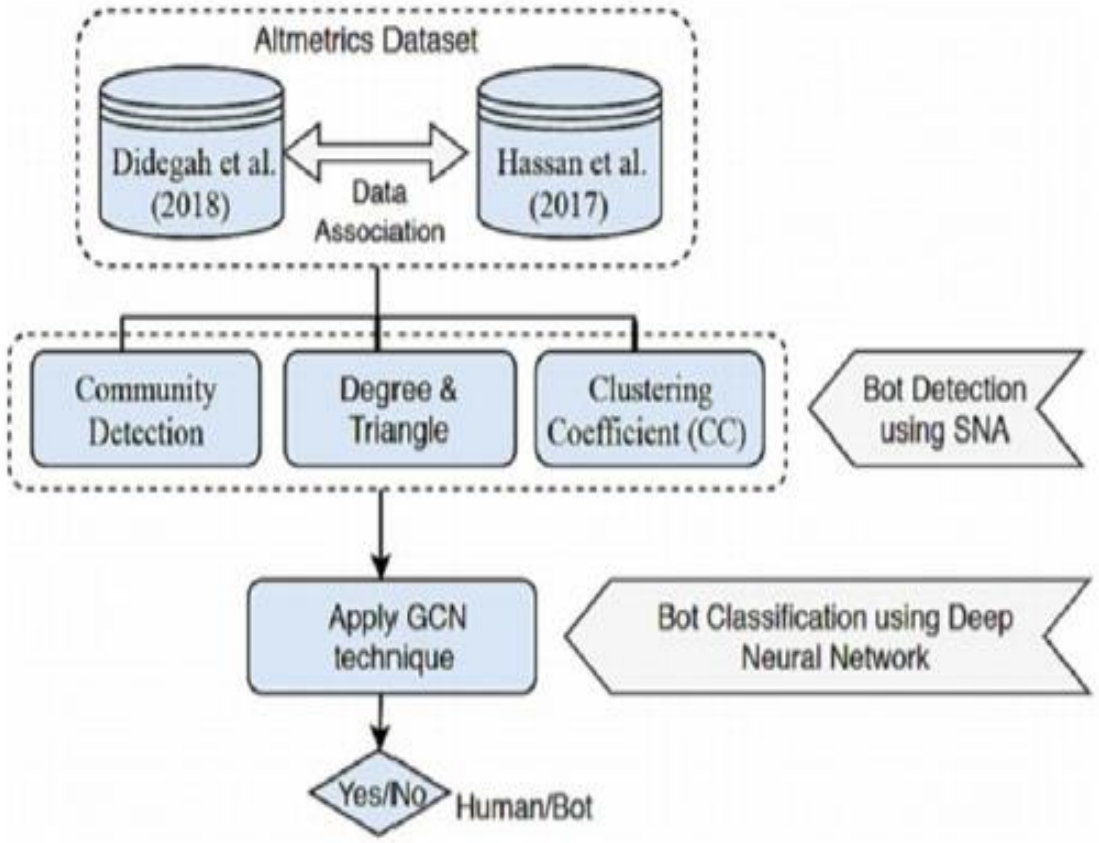
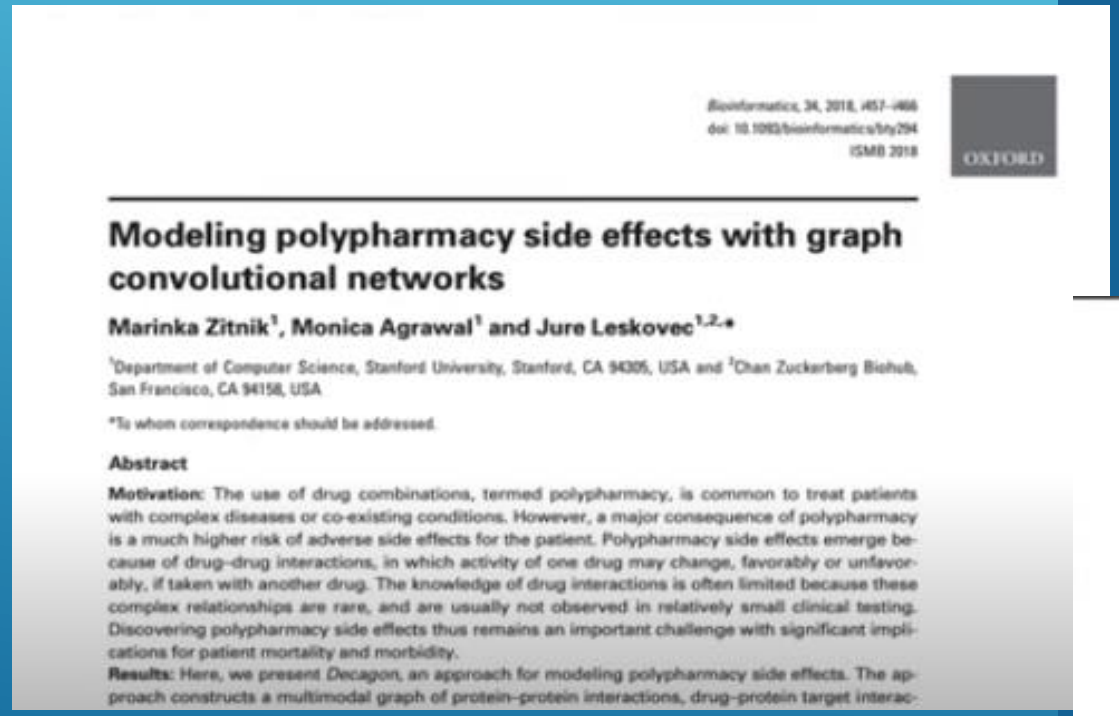


Fig. 7 Model architecture [8]

NOTAS FINAIS

- Melhoria: Permitir input de grafos dinâmicos.
- Deep Learning em grafos abre possibilidades em:
  - Redes sociais
  - Bio-tecnologia/Moléculas
  - Expressões simbólicas
  - Raciocínio relacional
  - Código
  - Otimização combinatória
  - Semi-supervised learning
  - Física
  - Química/Farmácia/BioMédica
  - ...



Bioinformatics, 34, 2018, i457–i466
doi: 10.1093/bioinformatics/bty294
ISMB 2018

OXFORD

Modeling polypharmacy side effects with graph convolutional networks

Marinka Zitnik[1], Monica Agrawal[1] and Jure Leskovec[1,2,*]

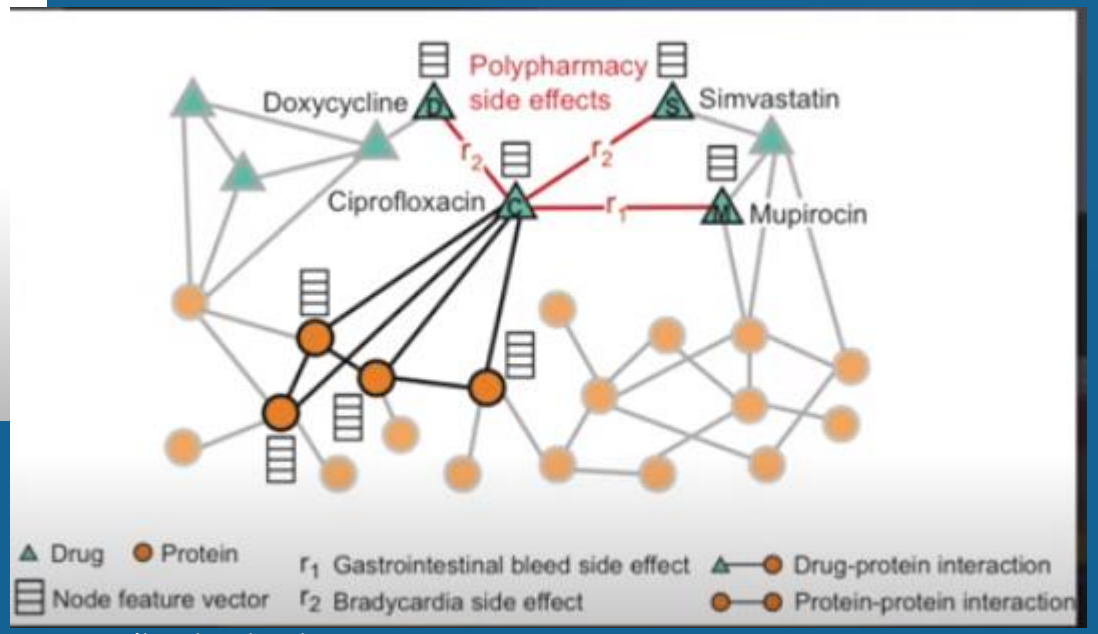
[1]Department of Computer Science, Stanford University, Stanford, CA 94305, USA and [2]Chan Zuckerberg Biohub, San Francisco, CA 94158, USA

*To whom correspondence should be addressed.

Abstract

Motivation: The use of drug combinations, termed polypharmacy, is common to treat patients with complex diseases or co-existing conditions. However, a major consequence of polypharmacy is a much higher risk of adverse side effects for the patient. Polypharmacy side effects emerge because of drug–drug interactions, in which activity of one drug may change, favorably or unfavorably, if taken with another drug. The knowledge of drug interactions is often limited because these complex relationships are rare, and are usually not observed in relatively small clinical testing. Discovering polypharmacy side effects thus remains an important challenge with significant implications for patient mortality and morbidity.

Results: Here, we present *Decagon*, an approach for modeling polypharmacy side effects. The approach constructs a multimodal graph of protein–protein interactions, drug–protein target interac-



Modelagem p/ predicão de efeitos colaterais em uso concomitante de drogas.

APLICAÇÃO GCN EM PYTHON+TENSORFLOW

VAMOS PARA O GOOGLE COLAB

- Referências

1. Huang K-H (2019) A gentle introduction to graph neural networks (basics, DeepWalk, and GraphSage), 10 Feb 2019. [Online]. Available: https://towardsdatascience.com/a-gentle-introduction-to-graph-neural-network-basics-deepwalk-and-graphsage-db5d540d50b3

2. Jepsen TS (2018) How to do deep learning on graphs with graph convolutional networks, 18 Sept 2018. [Online]. Available: https://towardsdatascience.com/how-to-do-deep-learning-ongraphs-with-graph-convolutional-networks-7d2250723780

3. Benamira A, Devillers B, Lesot E, Ray AK, Saadi M, Malliaros FD (2019) Semi-supervised learning and graph neural networks for fake news detection. In: IEEE/ACM International conference on advances in social networks analysis and mining, Chicago

4. Bian T, Xiao X, Xu T, Zhao P, Huang W, Rong Y, Huang J (2020) Rumor detection on social media with bi-directional graph convolutional networks. arXiv preprint arXiv:2001.06362, Chicago

5. Dong M, Zheng B, Hung NQV, Su H, Li G (2019) Multiple rumor source detection with graph convolutional networks. In: 28th ACM International conference on information and knowledge management, Harvard

6. Li C, Goldwasser D (2019) Encoding social information with graph convolutional networks for political perspective detection in news media. In: 57th Annual meeting of the association for computational linguistics, Harvard

- # Referências

7. Wu Y, Lian D, Xu Y,Wu L, Chen E (2020) Graph convolutional networks with markov random field reasoning for social spammer detection

8. Aljohani N, Fayoumi A, Hassan S (2020) Bot prediction on social networks of Twitter in altmetrics using deep graph convolutional networks. Soft Comput

9. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale. In: 24th ACMSIGKDD international conference on knowledge discovery & data mining, Chicago, pp 974–983

10. Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: AAAI Conference on artificial intelligence, Vancouver

11. Marcheggiani D, Titov I (2017) Encoding sentences with graph convolutional networks for semantic role labeling. arXiv preprint arXiv:1703.04826, Harvard

12. Zhang S, Tong H, Xu J, Maciejewski R (2019) Graph convolutional networks: a comprehensive review. 10 Nov 2019. [Online]. Available: https://link.springer.com/article/10.1186/s40649- 019-0069-y

13. Yang Z, Han S, Zhao J (2020) Poisson Kernel avoiding self-smoothing in graph convolutional networks. arXiv preprint arXiv:2002.02589, 7 Feb 2020, Vancouver

14. J. A. Bondy e U. S. R. Murty. Graph Theory. Springer London, 2008. DOI:10.1007/978-1-84628-970-5.

[GMS05] M. Gori, G. Monfardini e F. Scarselli. "A new model for learning in graph domains". English. Em: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. Vol. 2. cited By 70. IEEE, jul. de 2005, pp. 729–734. DOI: 10.1109/ijcnn.2005.1555942.

[Sca+09] F. Scarselli et al. "The Graph Neural Network Model". English. Em: IEEE Transactions on Neural Networks 20.1 (jan. de 2009). cited By 269, pp. 61–80. ISSN: 1045-9227. DOI: 10.1109/tnn.2008.2005605.

SLIDE EXTRA: PIADAS INFAMES: HOMENAGEM AO NOSSO PROF. DR. RENATO ROCHA

Existem 10 tipos de pessoas no mundo: as que entendem números binários e as que não entendem.

– Você sabe que é um vértice?
– Não.
– Então vamos ali no cantinho que eu te mostro.

92% dos brasileiros são ruins em matemática, os outros 16% são péssimos!

– Me sinto tão insignificante... – disse o número 1.
– Pelo menos você é mais do que nada. – respondeu o número zero.

Por que o 3 e o 7 não podem se casar?
Porque são primos. O que é uma pena, pois eles formariam um casal 10.

– Gata, seus pais são matemáticos?
– Não. Por quê?
– Porque você é um produto notável.