

Curso de Difusão

# Introdução às *Graph Neural Networks* e suas aplicações em tarefas de aprendizado de máquina

Angelo Mendes

Marcos Gôlo

Ricardo Marcacini



# Apresentação

---



Angelo Mendes



Marcos Gôlo



Ricardo Marcacini

# Sumário

---

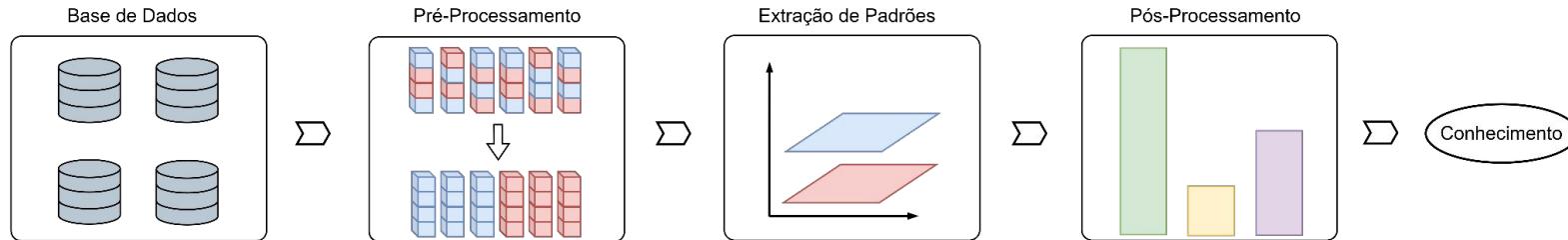
1. Introdução
2. Grafos
3. Modelagem de dados utilizando grafos
4. Graph Neural Networks
5. Contextos de exploração (Aplicações)
6. Prática 1: Reconhecimento de emoções em músicas utilizando Graph Neural Networks
7. Prática 2: Graph neural networks para detecção de notícias falsas por meio do aprendizado de uma classe

# Introdução

---

## Mineração de dados

- Construção e enriquecimento de bases de dados
- Análise dos dados, extração de características, criação de estrutura para os dados
- Aprendizado de padrões sobre os dados
- Análise e uso do conhecimento obtido



# Introdução

## Pré-processamento

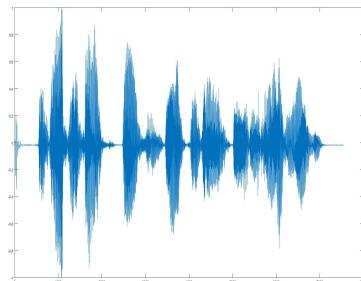
- Tipos de dados e extração de características

### Estruturado

| Nome  | Altura | Peso | IMC   |
|-------|--------|------|-------|
| José  | 1.80   | 75   | 23.15 |
| Maria | 1.65   | 69   | 25.34 |
| Paula | 1.71   | 68   | 23.26 |

Tabela com dados pessoais

### Não estruturado



Sinal de áudio

O curso foi ótimo.

Aprendi sobre GNN.

Ótimo curso sobre GNN.

Documento textual

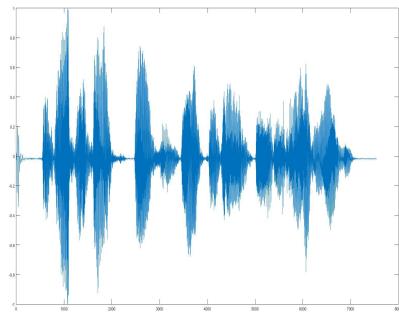
# Introdução

## Pré-processamento

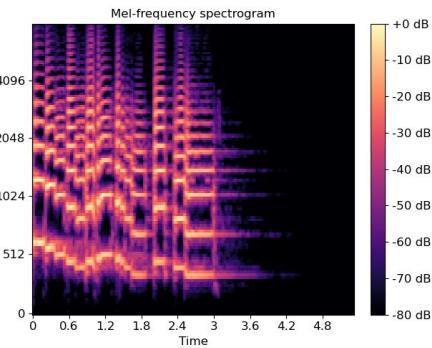
- Tipos de dados e extração de características

### Áudio

- características que englobam informações de tempo e frequência
- conseguimos enfatizar diferentes componentes do áudio



Sinal de áudio



Mel espectrograma

librosa. Disponível em: <https://librosa.org/doc>

# Introdução

---

## Pré-processamento

- Tipos de dados e extração de características

## Texto

- construção de vetores a partir de textos
- características que enfatizam informação de contexto
- características que possibilitam computar a similaridade entre textos

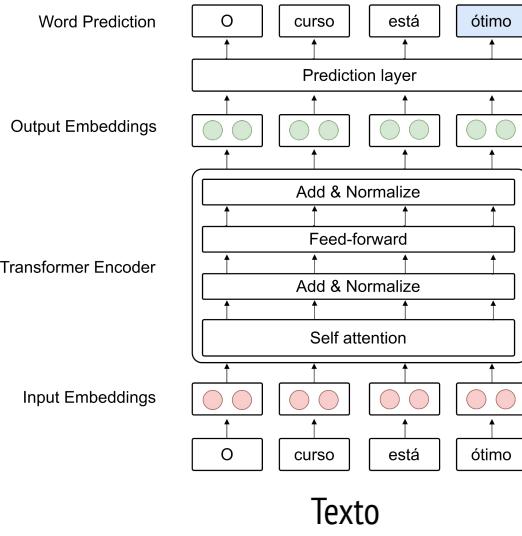
1. O curso foi ótimo
2. Aprendi sobre GNN
3. Ótimo curso sobre GNN

| 0 | curso | foi | ótimo | aprendi | sobre | GNN |
|---|-------|-----|-------|---------|-------|-----|
| 1 | 1     | 1   | 1     | 0       | 0     | 0   |
| 0 | 0     | 0   | 0     | 1       | 1     | 1   |
| 0 | 1     | 0   | 1     | 0       | 1     | 1   |

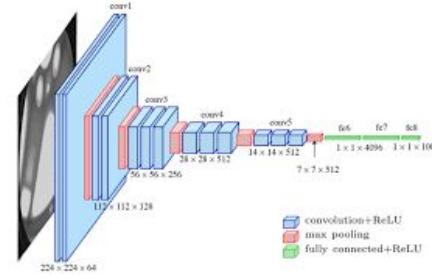
Caseli, H.M.; Nunes, M.G.V. (org.). 2023. Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português

# Introdução

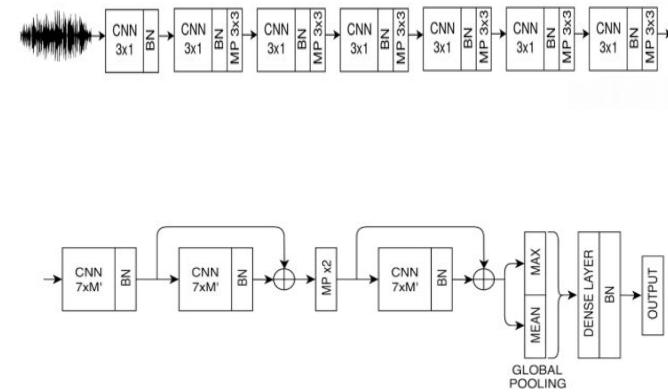
## Representação por *embeddings*



Texto



Imagen



Áudio

(bert) Jacob Devlin et al.. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.**

(vgg) Karen Simonyan, Andrew Zisserman. **Very Deep Convolutional Networks for Large-Scale Image Recognition.**

(musicnn) Jordi Pons et al.. **End-to-end learning for music audio tagging scale.**

# Grafos

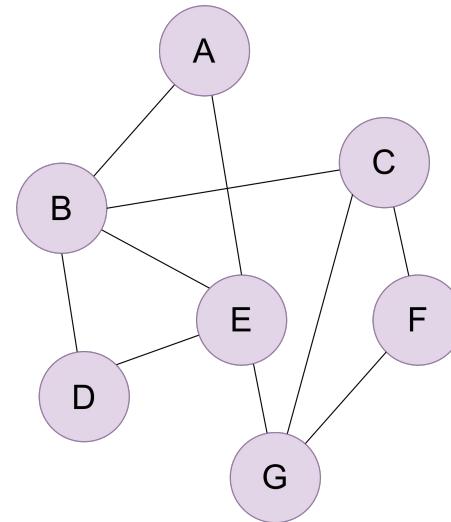
---

Definição de grafos  $G = (V, A, P)$

$$V = \{A, B, C, D, E, F, G\}$$

$$A = \{(A, B), (B, C), (B, D), \dots, (F, G)\}$$

$$P \subset \mathbb{R}$$



# Grafos

---

Definição de grafos  $G = (V, A, P)$

$V = \{A, B, C, D, E, F, G\}$

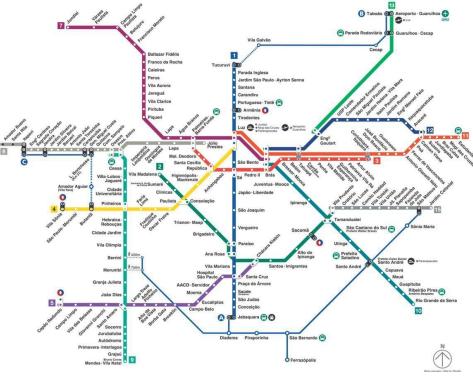
$A = \{(A, B), (B, C), (B, D), \dots, (F, G)\}$

$P \subset \mathbb{R}$

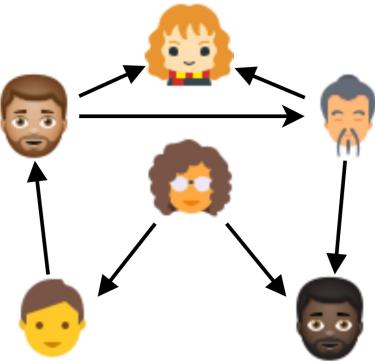
|          | <b>A</b> | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> | <b>G</b> |
|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>A</b> | 0        | <b>1</b> | 0        | 0        | <b>1</b> | 0        | 0        |
| <b>B</b> | <b>1</b> | 0        | <b>1</b> | <b>1</b> | <b>1</b> | 0        | 0        |
| <b>C</b> | 0        | <b>1</b> | 0        | 0        | 0        | <b>1</b> | <b>1</b> |
| <b>D</b> | 0        | <b>1</b> | 0        | 0        | <b>1</b> | 0        | 0        |
| <b>E</b> | <b>1</b> | <b>1</b> | 0        | <b>1</b> | 0        | 0        | <b>1</b> |
| <b>F</b> | 0        | 0        | <b>1</b> | 0        | 0        | 0        | <b>1</b> |
| <b>G</b> | 0        | 0        | <b>1</b> | 0        | <b>1</b> | <b>1</b> | 0        |

Matriz de adjacência

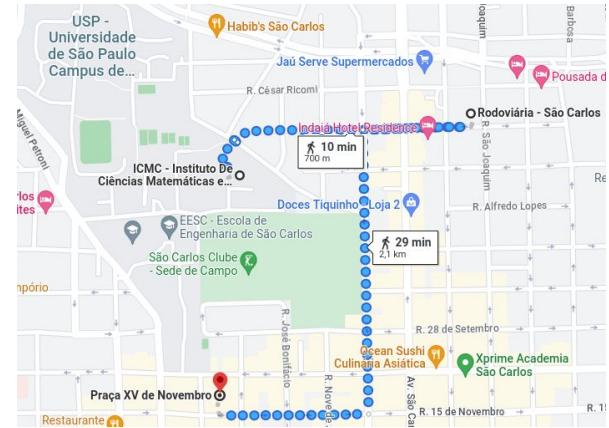
# Grafos



## Grafo não direcionado



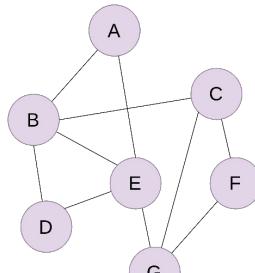
## Grafo direcionado



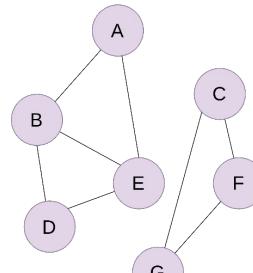
## Grafo ponderado

# Grafos

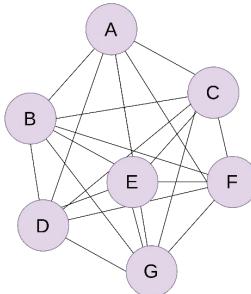
---



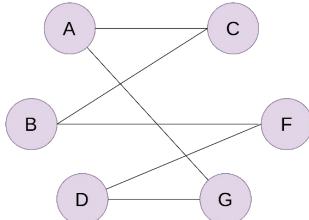
Conexo



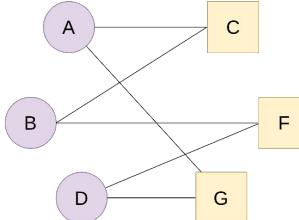
Desconexo



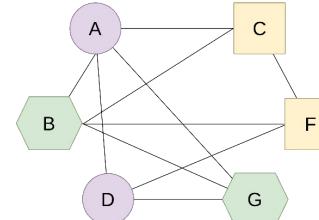
Completo



Homogêneo



Heterogêneo e bipartido



Heterogêneo

Alguns tipos de grafos

# Modelagem de dados em grafos

---

- Vamos adotar a definição de grafo como  $G = (V, A, P)$
- Usaremos **vértices**, porém podemos falar **objetos** ou **nós**
- Usaremos **arestas**, porém podemos falar **relações** ou **links**
- **Pesos** podem estar presentes, ou não
- Tentem imaginar um grafo com suas componentes definidas como classes e objetos

**class** Grafo:

vertices: list(Vertice)  
arestas: list(Aresta)

**class** Vertice:

id: str  
tipo: str  
caracteristica: list  
rotulo: str

**class** Aresta:

id: str  
origem: Vertice  
destino: Vertice  
peso: float  
tipo: str  
rotulo: str

# Modelagem de dados em grafos

---

| musica_id | artista   | letra        | audio                |
|-----------|-----------|--------------|----------------------|
| 1         | Artista 1 | [0, 0, 0, 1] | [0.1, 0.1, 0.1, 0.1] |
| 2         | Artista 2 | [0, 0, 1, 0] | [0.2, 0.2, 0.2, 0.2] |
| 3         | Artista 3 | [0, 1, 0, 0] | [0.3, 0.3, 0.3, 0.3] |
| 4         | Artista 4 | [1, 0, 0, 0] | [0.4, 0.4, 0.4, 0.4] |

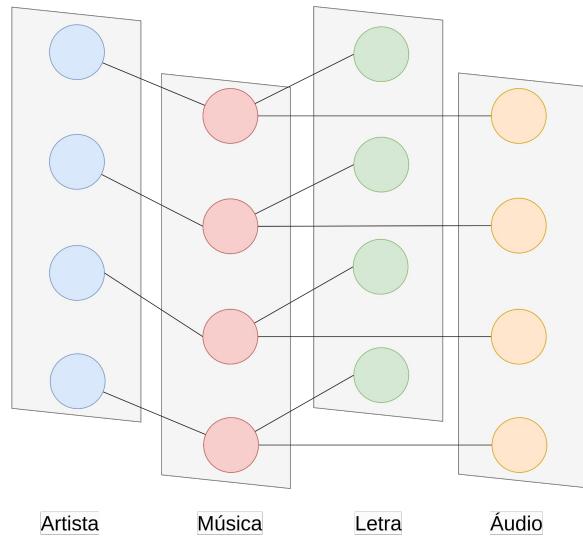
Base de dados musicais com 4 amostras

# Modelagem de dados em grafos

```
vertices = [  
    Vertice(id="musica_1", tipo="musica"),  
    Vertice(id="artista_1", tipo="artista"),  
    Vertice(id="letra_1", tipo="letra", caracteristica=[0, 0, 0, 1]),  
    Vertice(id="audio_1", tipo="audio", caracteristica=[0, 0, 0, 1]),  
    ...,  
    Vertice(id="audio_4", tipo="audio", caracteristica=[1, 0, 0, 0]),  
]
```

```
arestas = [  
    Aresta(id=1, origem=vertices[0], destino=vertices[1], tipo="musica_artista"),  
    Aresta(id=2, origem=vertices[0], destino=vertices[2], tipo="musica_letra"),  
    Aresta(id=3, origem=vertices[0], destino=vertices[3], tipo="musica_audio"),  
    ...,  
    Aresta(id=13, origem=vertices[12], destino=vertices[15], tipo="musica_audio")  
]
```

```
grafo = Grafo(vertices=vertices, arestas=arestas)
```



Grafo resultante, com vértices em camadas

# Modelagem de dados em grafos

|    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 0  | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 1  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 2  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 3  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 4  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 5  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 6  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 7  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 8  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1  | 1  | 0  | 0  | 0  | 0  |
| 9  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 1  | 1  | 1  | 1  |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 1  | 0  | 0  | 0  | 0  |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 1  | 0  | 0  | 0  | 0  |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 1  | 0  | 0  | 0  | 0  |

Tipos de vértice

música

letra

áudio

artista

Tipos de aresta

musica\_audio

musica\_letra

musica\_artista

artista\_musica

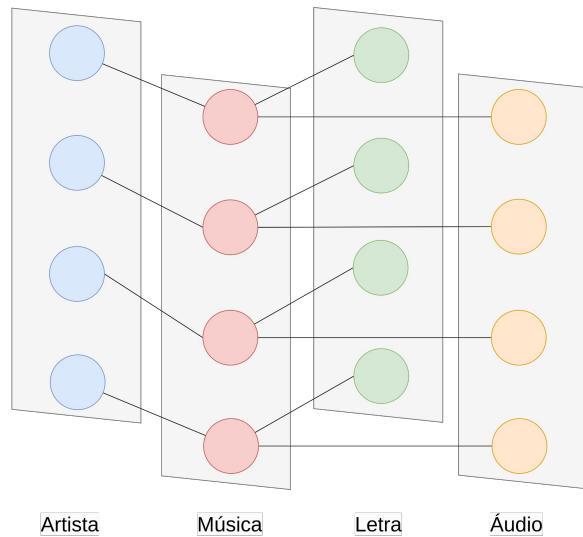
letra\_musica

audio\_musica

# Modelagem de dados em grafos

## Grafo desconexo

- Não há interação entre músicas distintas
- Não exploramos o potencial da modelagem dos dados em grafos
- Podemos enriquecer esse grafo!



**Grafo resultante, com vértices em camadas**

# Modelagem de dados em grafos

## Buscando informações de relacionamentos em fontes externas

### Músicas relacionadas

Para obter uma relação de músicas relacionadas com a música encontrada, utilize o valor `relmus` no parâmetro extra. Por exemplo, para encontrar músicas relacionadas a "One" do U2, usamos a seguinte url:

URL: [https://api.vagalume.com.br/search.php?art=U2&mus=One&extra=relmus&nolyrics=1&apikey=\[key\]](https://api.vagalume.com.br/search.php?art=U2&mus=One&extra=relmus&nolyrics=1&apikey=[key])

```
1 // Exemplo de retorno da requisição JSON
2 {
3   "type": "exact",
4   "art": {
5     "id": "3ade68b2g3b86eda3",
6     "name": "U2",
7     "url": "https://www.vagalume.com.br/u2/"
8   },
9   "mus": [
10    {
11      "id": "3ade68b3gdb86eda3",
12      "name": "One",
13      "url": "https://www.vagalume.com.br/u2/one.html",
14      "lang": 2,
15      "translate": [
16        {
17          "id": "3ade68b6g8048fda3",
18          "lang": 1,
19          "url": "https://www.vagalume.com.br/u2/one-traducao.html"
20        }
21      ],
22      "related": [
23        {
24          "art": {
25            "id": "3ade68b3g0f86eda3",
26            "name": "R.E.M.",
27            "url": "https://www.vagalume.com.br/r-e-m/"
28          },
29          "mus": [
30            {
31              "id": "3ade68b4gf596eda3",
32              "lang": 2,
33              "name": "Losing My Religion",
34              "url": "https://www.vagalume.com.br/r-e-m/losing-my-religion.html"
35            }
36          ]
37        }
38      ],
39      "art": { /* outro artista relacionado */ },
40      "mus": { /* outra música relacionada */ }
41    }
42  ]
43}
```

### Artistas relacionados

Para obter uma relação de artistas relacionados ao artista encontrado, utiliza-se o valor `relart` no parâmetro extra. Exemplo:  
URL: [https://api.vagalume.com.br/search.php?art=madonna&extra=relart&extra=relart&extra=relart&apikey=\[key\]](https://api.vagalume.com.br/search.php?art=madonna&extra=relart&extra=relart&extra=relart&apikey=[key])

```
1 // Exemplo de retorno da requisição JSON
2 {
3   "type": "song_notfound",
4   "art": {
5     "id": "3ade68b3g1f86eda3",
6     "name": "Madonna",
7     "url": "https://www.vagalume.com.br/madonna/",
8     "related": [
9       {
10         "id": "3ade68b2g4b86eda3",
11         "name": "Kid Abelha",
12         "url": "https://www.vagalume.com.br/kid-abelha/"
13       },
14       {
15         "id": "3ade68b5g67d7eda3",
16         "name": "Britney Spears",
17         "url": "https://www.vagalume.com.br/britney-spears/"
18       }
19     ]
20   }
21 }
```

Informações de músicas e artistas relacionados usando a API do Vagalume

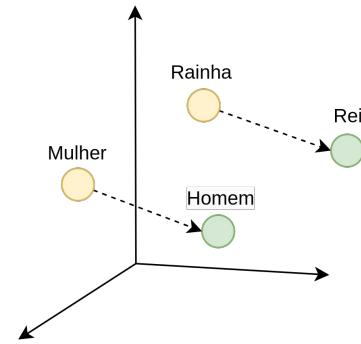
API do Vagalume. Disponível em: <https://api.vagalume.com.br/docs/>

A. C. M. da Silva and M. P. S. Gôlo and R. M. Marcacini. **Unsupervised Heterogeneous Graph Neural Network for Hit Song Prediction through One Class Learning**, 2022

# Modelagem de dados em grafos

## Construindo arestas baseando-se em medidas de similaridade

- Definir a característica que representa cada vértice
- Definir um critério e uma medida de distância/similaridade/dissimilaridade para relacionar os vértices
- Métodos baseados em vizinhança (como KNN) podem apoiar



Projeção de palavras em um espaço vetorial

# Modelagem de dados em grafos

Construindo arestas baseando-se em medidas de similaridade

| palavra | representação        |
|---------|----------------------|
| Rainha  | [0.2, 0.5, 0.6, 0.9] |
| Mulher  | [0.4, 0.8, 0.1, 0.2] |
| Rei     | [0.1, 0.7, 0.5, 0.5] |
| Homem   | [0.3, 0.9, 0.4, 0.3] |

Representação das palavras

|        | Rainha | Mulher | Rei  | Homem |
|--------|--------|--------|------|-------|
| Rainha | 1      | 0.65   | 0.93 | 0.79  |
| Mulher | 0.65   | 1      | 0.81 | 0.95  |
| Rei    | 0.93   | 0.81   | 1    | 0.94  |
| Homem  | 0.79   | 0.95   | 0.94 | 1     |

Matriz de dissimilaridade usando distância de cosseno

# Modelagem de dados em grafos

## Construindo arestas baseando-se em métodos de agrupamento

- Definir a característica que representa cada vértice
- Definir um algoritmo de agrupamento e seus parâmetros

### Pontos de atenção:

**1- Rede muito conexa: vértices com representações muito similares**

**2- Rede muito desconexa: baixa troca de informações entre os vértices**

| audio_id | representação        | cluster |
|----------|----------------------|---------|
| 1        | [0.2, 0.5, 0.6, 0.9] | 1       |
| 2        | [0.4, 0.8, 0.1, 0.2] | 1       |
| 3        | [0.1, 0.7, 0.5, 0.5] | 2       |
| 4        | [0.3, 0.9, 0.4, 0.3] | 2       |

Adicionando coluna com informação dos clusters de áudios

# Modelagem de dados em grafos

---

## Vantagens

- Suporta dados multimodais
- Complementaridade das informações
- Versatilidade para domínios complexos
- Propagação de informação
- Construir características apenas com a topologia do grafo

## Pontos de atenção

- grafo desconexo -> criar links
- nós com informações faltantes -> regularização (embeddings propagation)
- falta de label -> regularização (label propagation)
- over smoothing - embeddings muitos similares
- cenário indutivo

# Regularização de grafos

---

**Características ausentes em alguns vértices**

**Objetivo da regularização:**

- propagar informações entre os nós

**Premissas:**

- objetos que estão relacionados devem ter representações similares
- objetos que já possuem características, devem ter representação final similar à inicial

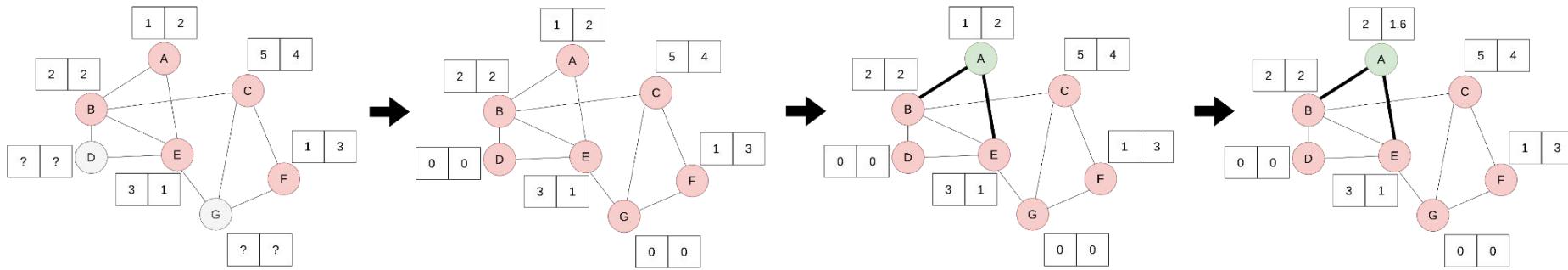
**Resultado:**

- todos os objetos possuem informações dos objetos de modalidades diferentes

Algoritmos de regularização: <https://github.com/BrucceNeves/GraphTLP/tree/main>

# Regularização de grafos

## Características ausentes em alguns vértices



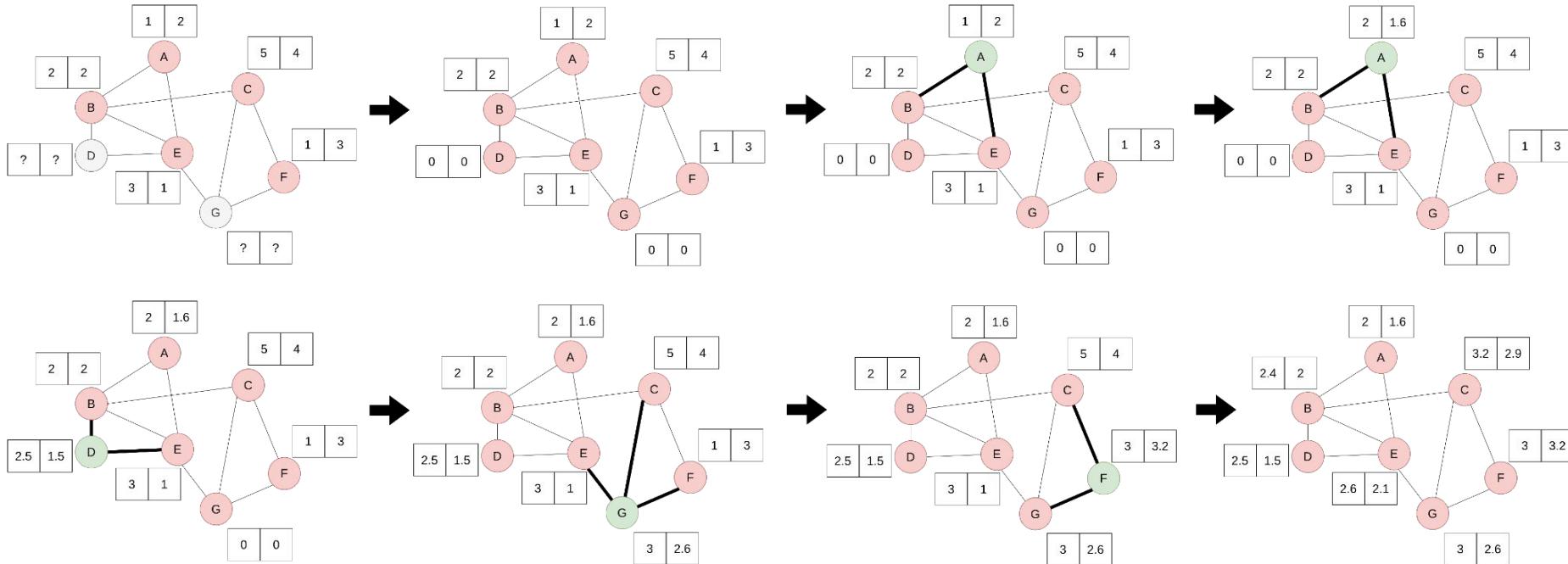
Estado inicial do grafo. Vértices D e G não tem características

Inicialização das características dos vértices D e G

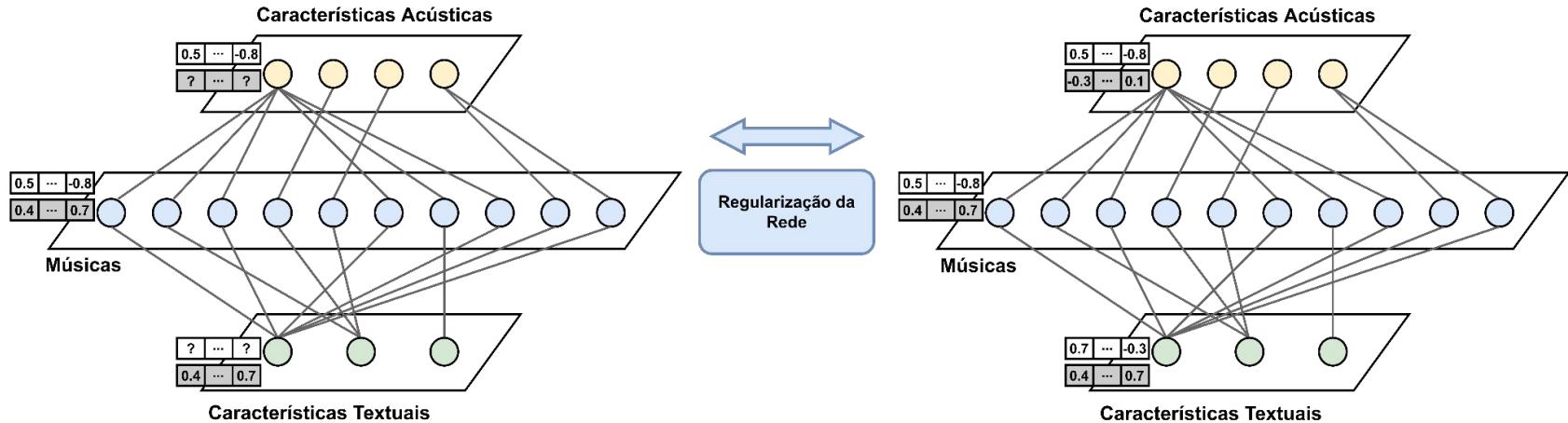
Caminhada aleatória e propagação das características entre vértices vizinhos

# Regularização de grafos

Características ausentes em alguns vértices



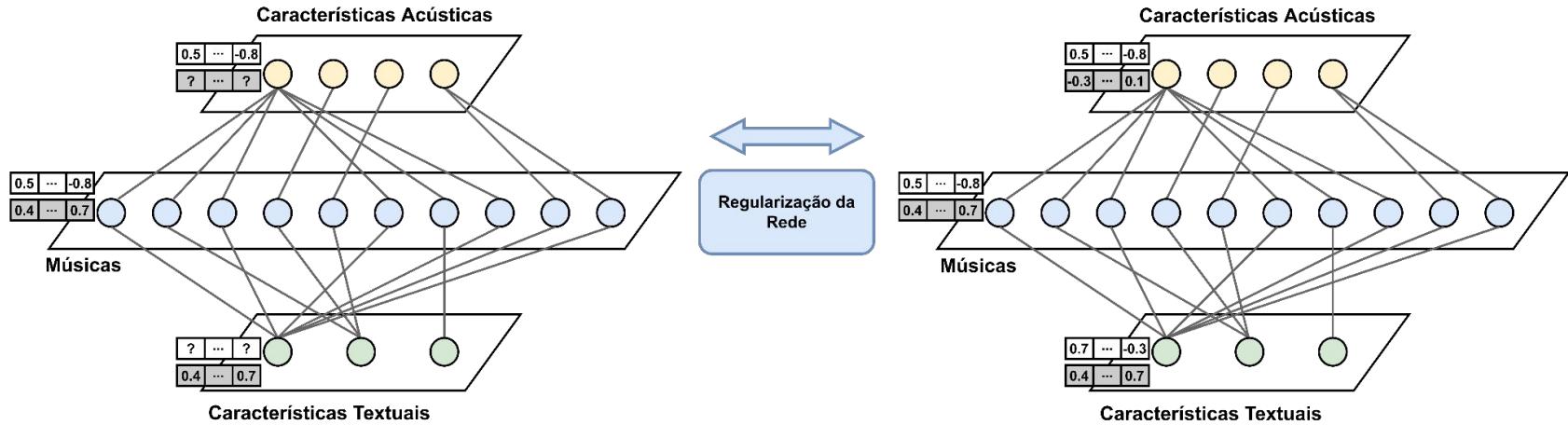
# Regularização de grafos



$$Q(\mathbf{Z}) = \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2 + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

ZHOU et. al. 2004. Learning with local and global consistency.

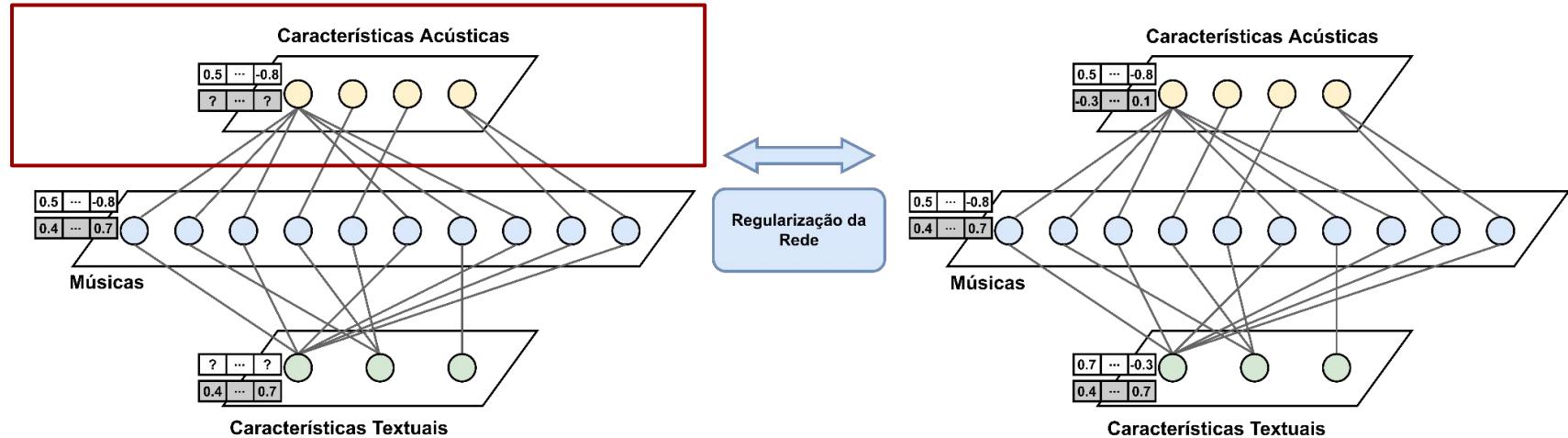
# Regularização de grafos



## Topologia da rede

$$Q(\mathbf{Z}) = \frac{1}{2} \underbrace{\sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2}_{\text{Topologia da rede}} + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

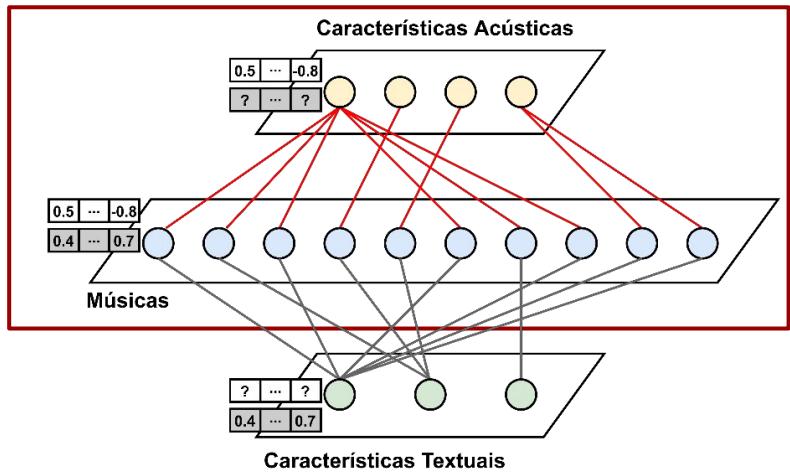
# Regularização de grafos



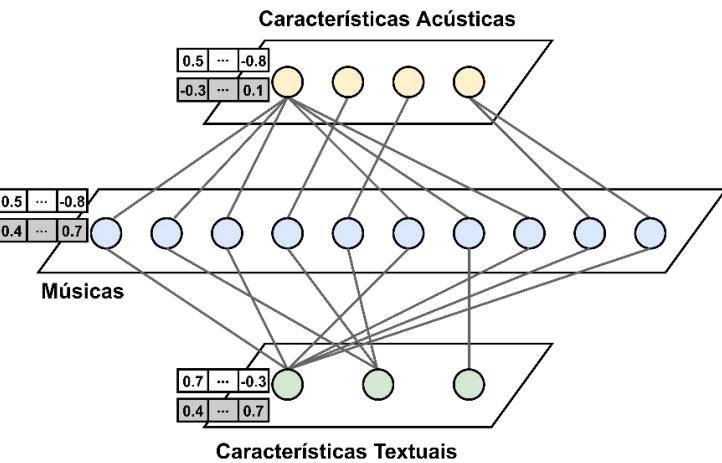
## Topologia da rede

$$Q(\mathbf{Z}) = \frac{1}{2} \underbrace{\sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2}_{\text{Term 1}} + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

# Regularização de grafos



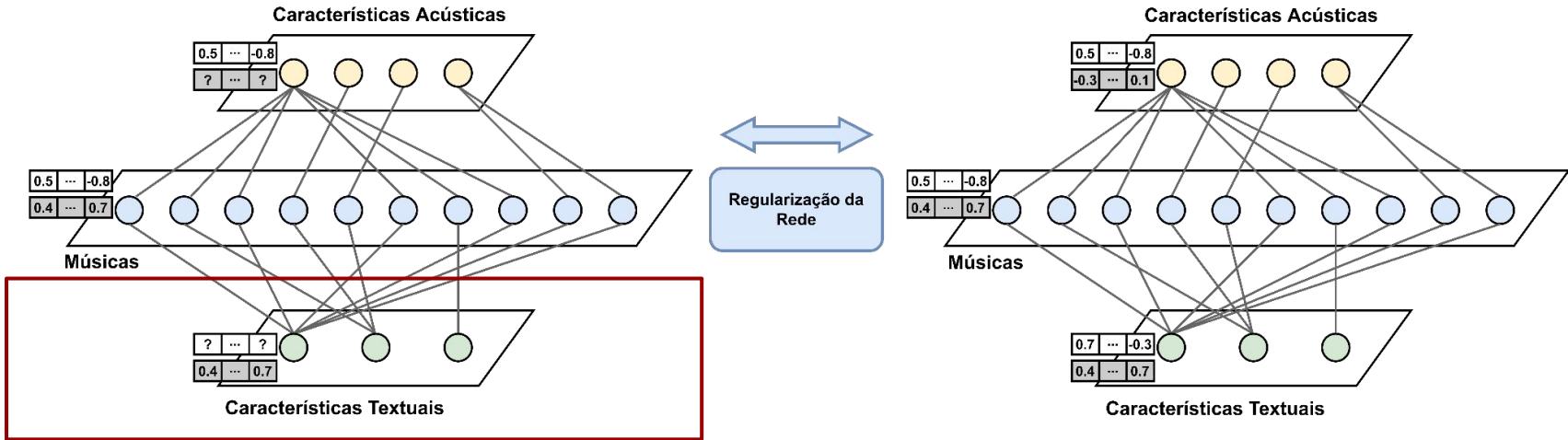
Regularização da  
Rede



## Topologia da rede

$$Q(\mathbf{Z}) = \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2 + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

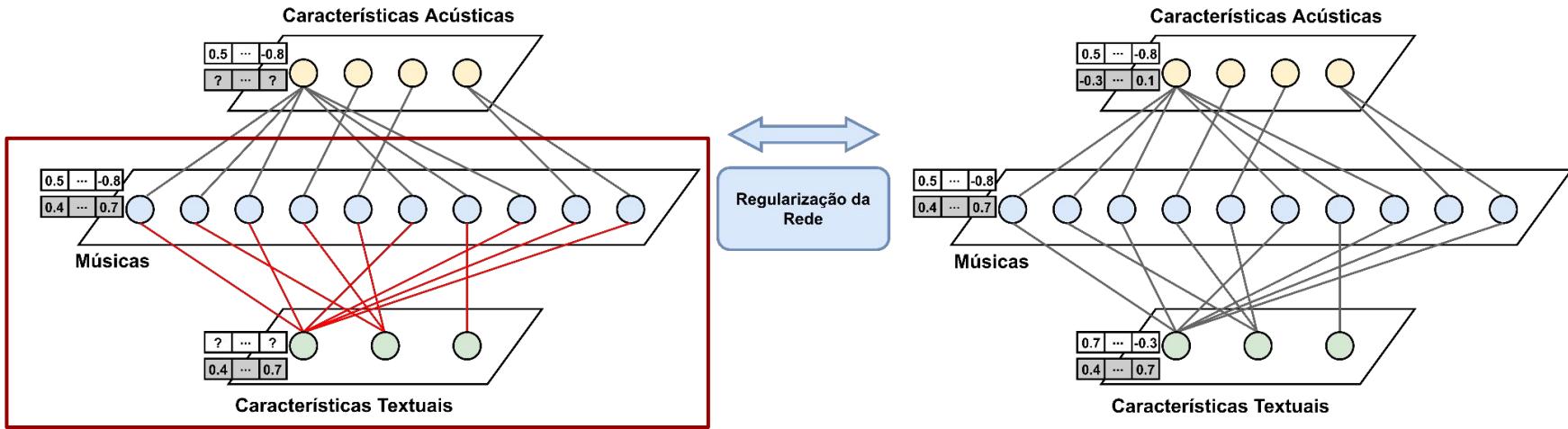
# Regularização de grafos



## Topologia da rede

$$Q(\mathbf{Z}) = \frac{1}{2} \underbrace{\sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2}_{\text{Topologia da rede}} + \frac{1}{2} \sum_{o_m \in O_M} \underbrace{\sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2}_{\text{Regularização da rede}} + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

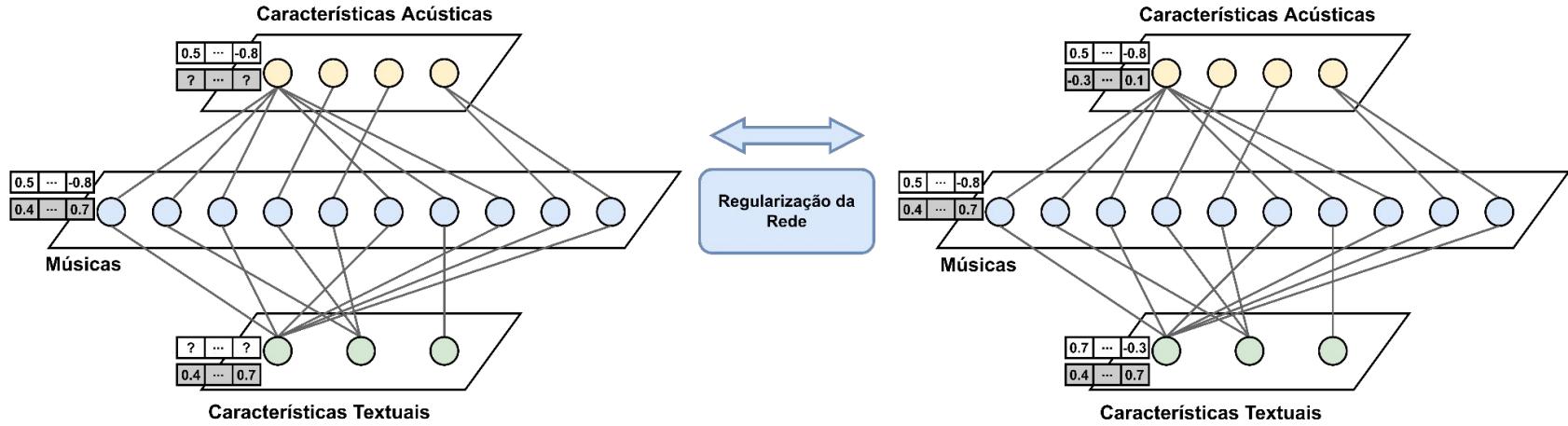
# Regularização de grafos



## Topologia da rede

$$Q(\mathbf{Z}) = \frac{1}{2} \underbrace{\sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2}_{\text{Term 1}} + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} \boxed{w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2} + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2$$

# Regularização de grafos



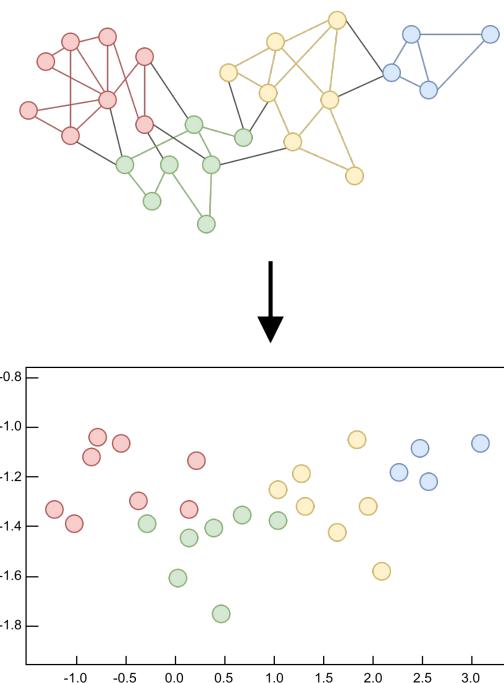
$$Q(\mathbf{Z}) = \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m, o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2 + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m, o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \underbrace{\sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2}_{\text{Regularization term}}$$

# Graph Embedding

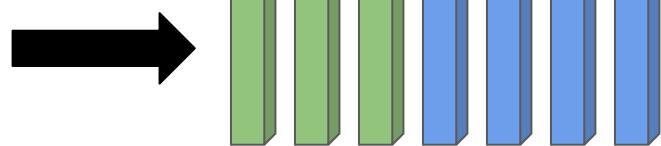
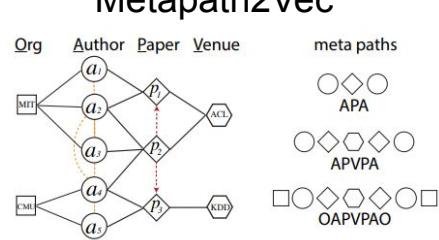
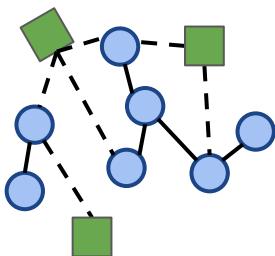
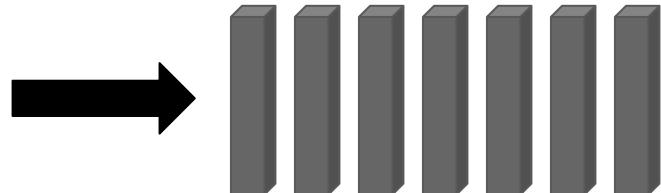
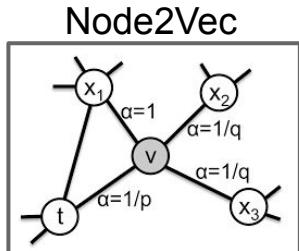
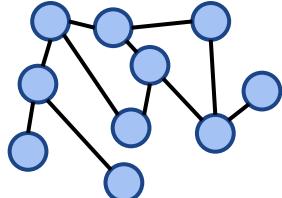
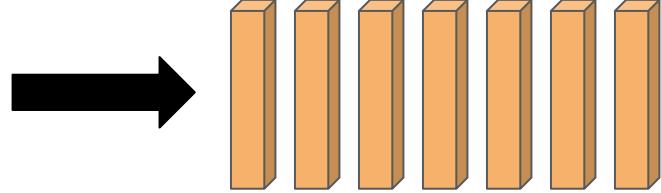
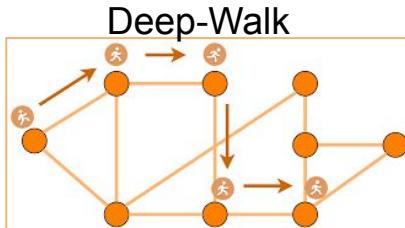
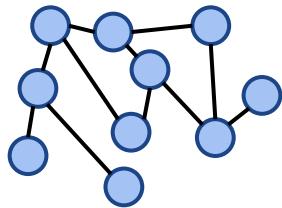
## Vértices sem características

Objetivo: Encontrar embeddings para os vértices em um espaço  $d$ -dimensional que preserve as noções de similaridade

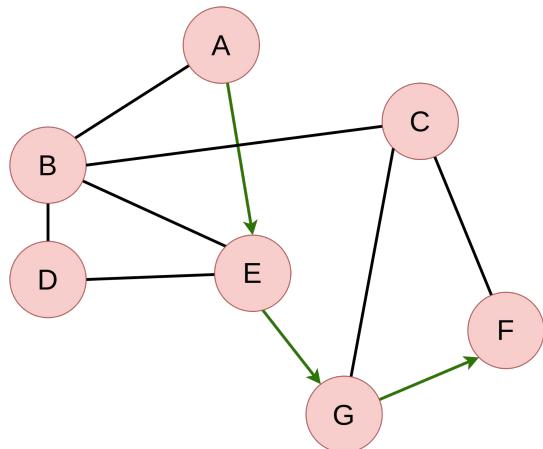
Ideia: Aprender uma representação que aproxime no espaço construído vértices que sejam vizinhos no grafo vizinhos



# Graph Embedding

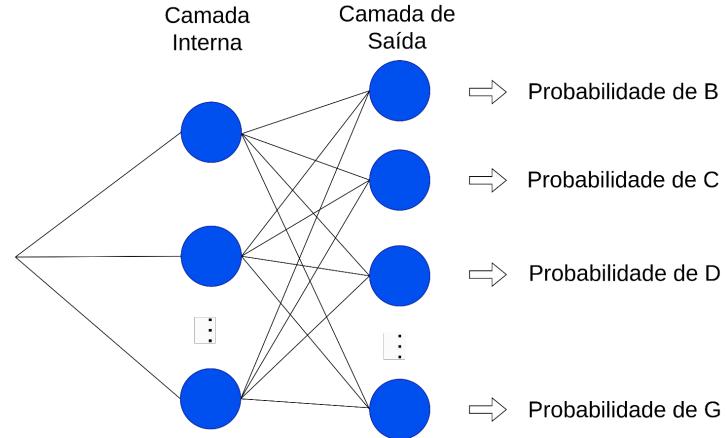


# Graph Embedding



Posição do vértice A

| Input |
|-------|
| 1     |
| 0     |
| 0     |
| 0     |
| 0     |
| 0     |
| 0     |



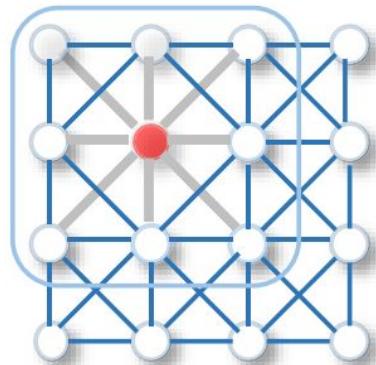
Construção de caminhadas entre os vértices

Utilização de um modelo de linguagem para aprender as representações

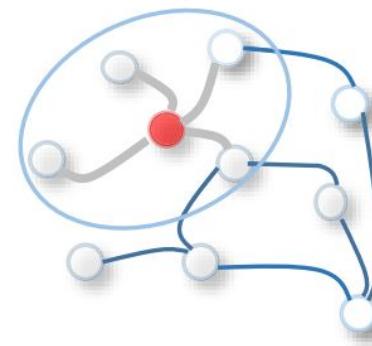
# Justificativa para GNNs

---

- Avanços das redes neurais em diferentes cenários de dados euclidianos
- E para dados não euclidianos, como os **grafos**?
- Propuseram as Redes Neurais para Grafos, ou as *Graph Neural Networks*



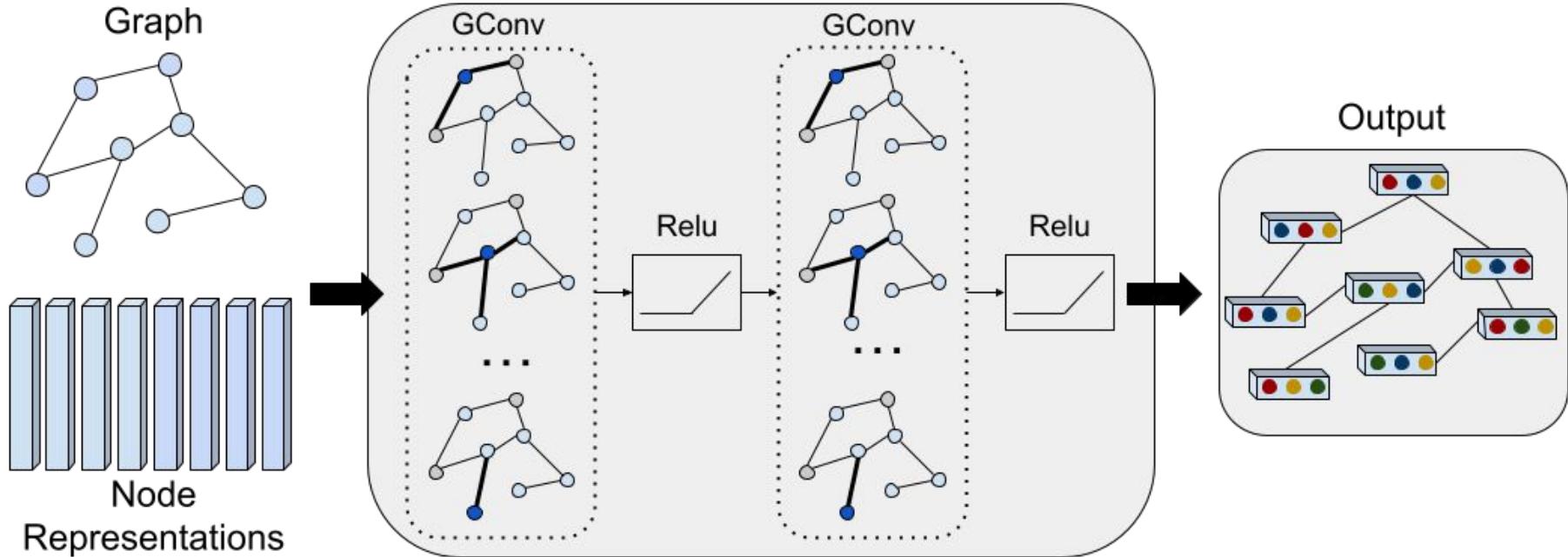
Rede Neural para imagem



Rede Neural para Grafo

Fonte: Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems 32.1 (2020):

# Graph Neural Networks



# Graph Neural Networks

---

- GNN:  $g(O, A; W)$
- $O$ : Representações dos objetos
- $A$ : Matriz de adjacência
- $W$ : Pesos da rede neural
- Uma GNN genérica pode ser definida com dois passos:
  - **Agregação:** agrupa informações dos vizinhos de um nó
  - **Combinação:** combina a representação aprendida pelo neurônio com a representação agregada

# Graph Neural Networks

---

$$\mathbf{H}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{A}; \mathbf{W}^{(l)})$$

$$\mathbf{a}_{o_i}^l = AGGREGATE^{(l)}(\{\mathbf{h}_{o_j}^{l-1} : o_j \in N_{o_i}\})$$

$$\mathbf{h}_{o_i}^l = COMBINE^{(l)}(\mathbf{h}_{o_i}^{l-1}, \mathbf{a}_{o_i}^l)$$

# Graph Neural Networks

$$\mathbf{H}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{A}; \mathbf{W}^{(l)})$$

$$\mathbf{H}^{(l+1)} = g(\mathbf{H}^{(l)}, \mathbf{A}; \mathbf{W}^{(l)})$$

representações de saída

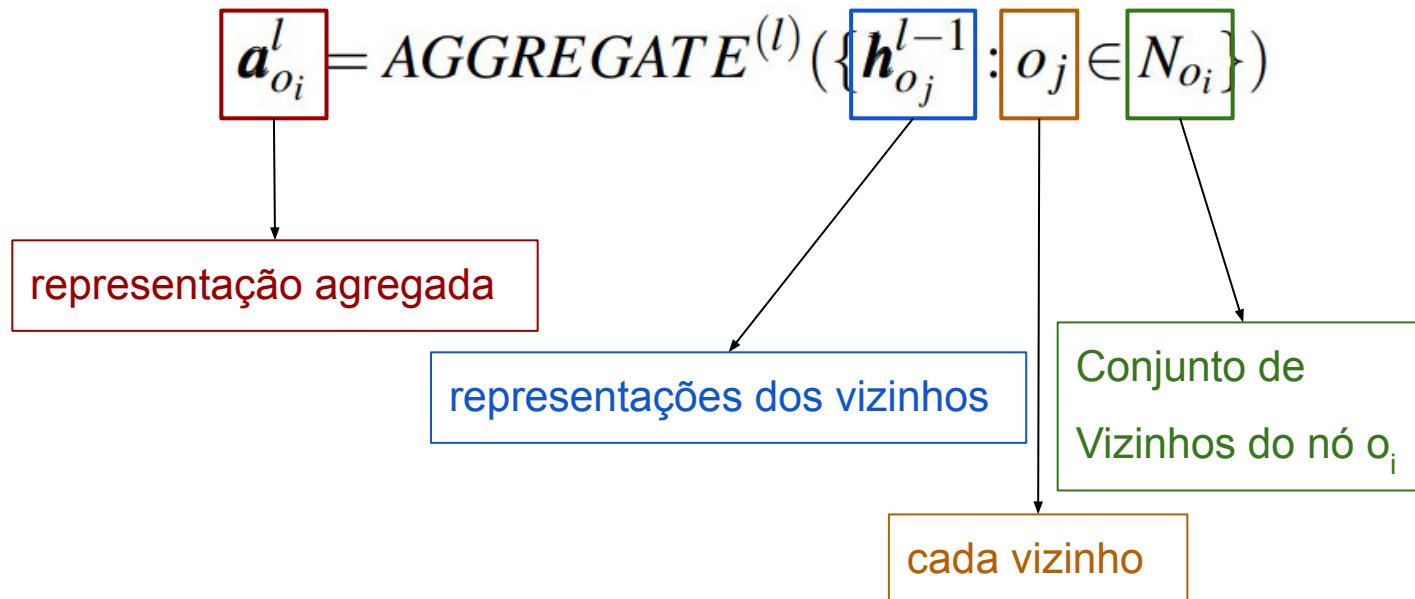
Pesos da rede neural

representações de entrada

matriz de adjacência

# Graph Neural Networks

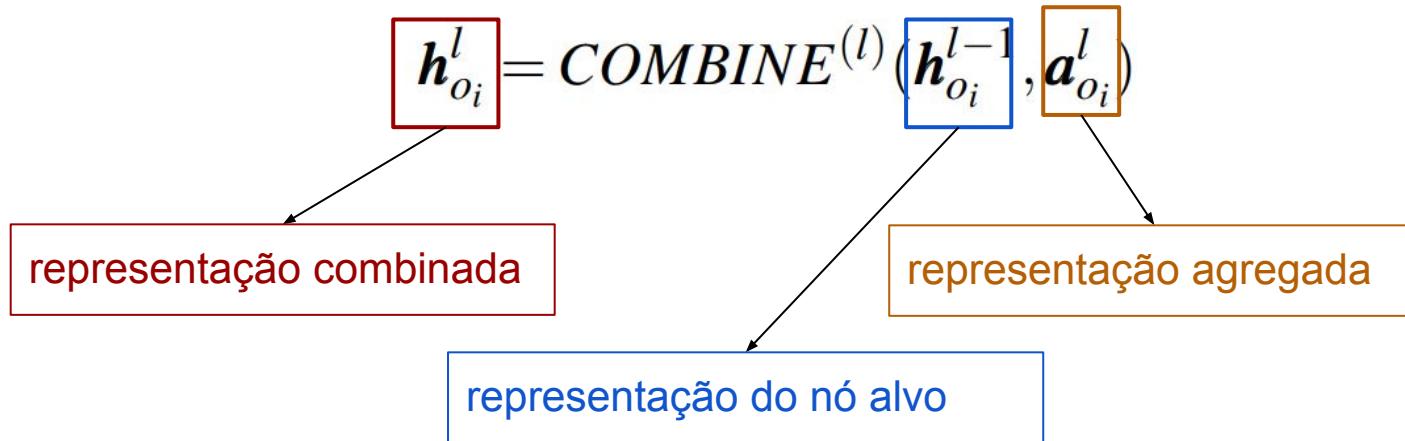
$$\mathbf{a}_{o_i}^l = AGGREGATE^{(l)}(\{\mathbf{h}_{o_j}^{l-1} : o_j \in N_{o_i}\})$$



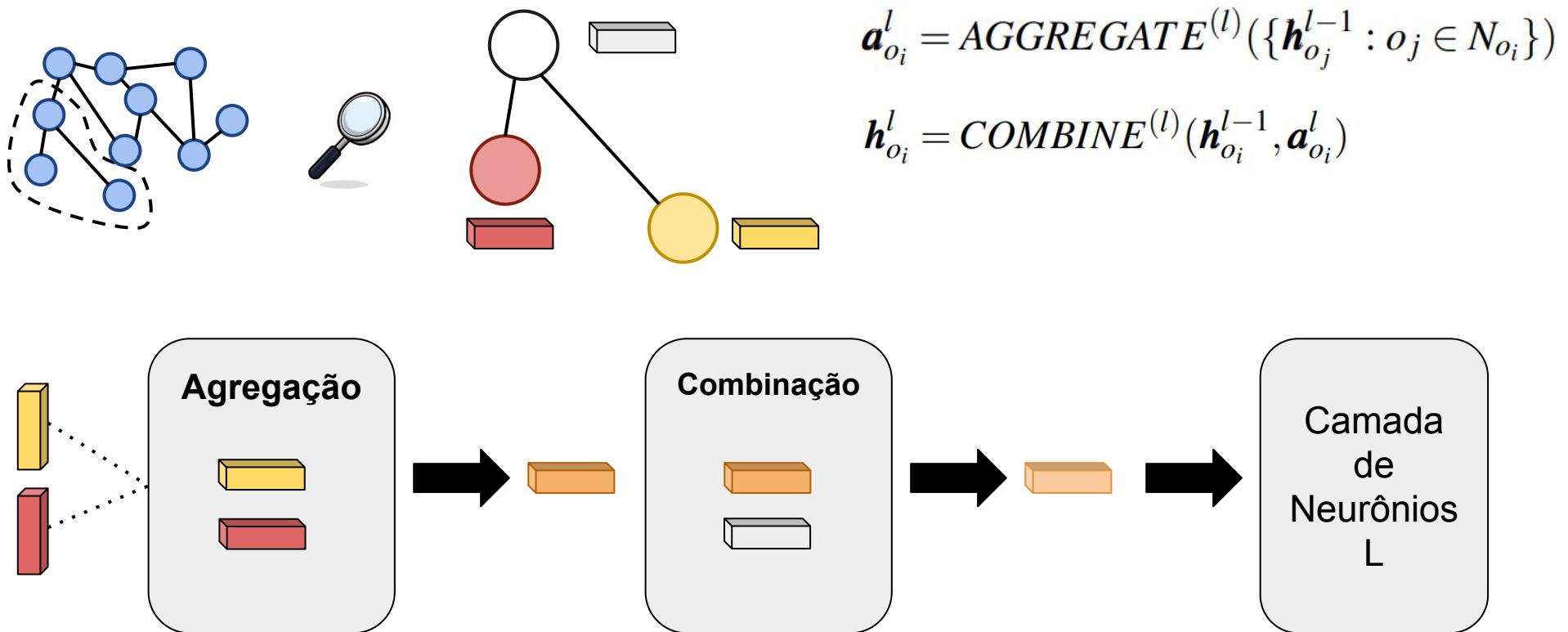
# Graph Neural Networks

---

$$\mathbf{h}_{o_i}^l = \text{COMBINE}^{(l)}(\mathbf{h}_{o_i}^{l-1}, \mathbf{a}_{o_i}^l)$$



# Graph Neural Networks



# Graph Convolutional Networks

---

$$\mathbf{h}_{o_i}^l = \sigma(\mathbf{W}^{(l)} \cdot MEAN\{\mathbf{h}_{o_j}^{l-1} : o_j \in \{N_{o_i} \cup o_i\}\})$$



**Agregação e  
Combinação**

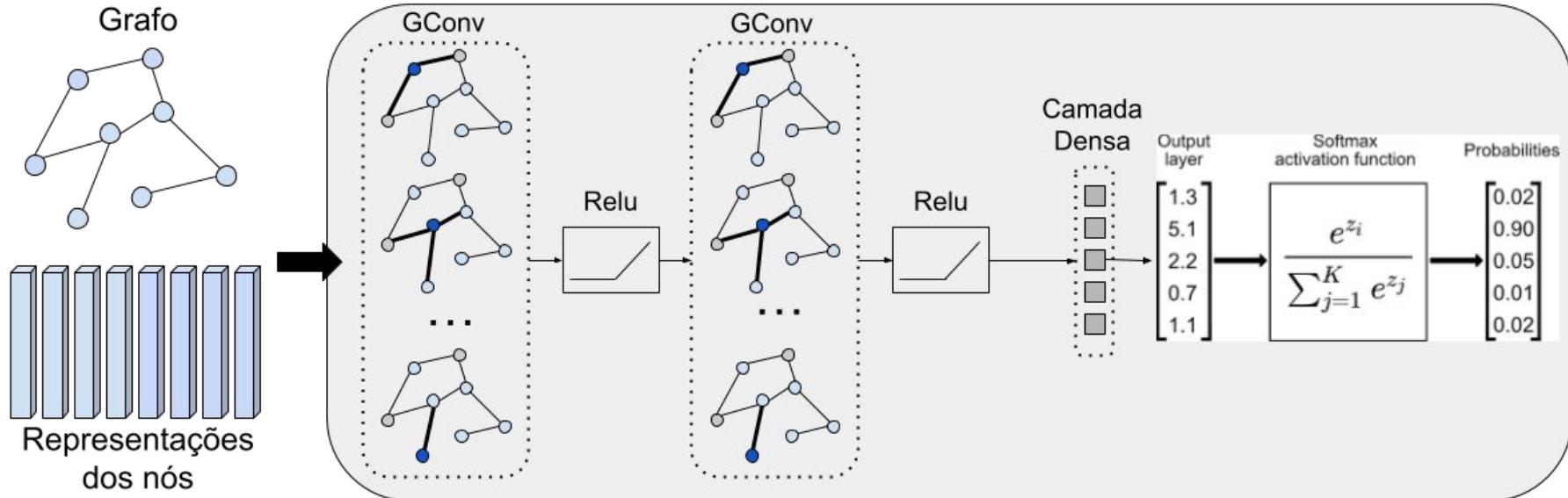
Adicionou o próprio nó na  
vizinhança (“se adicionou na média”)

# GAT e GraphSAGE

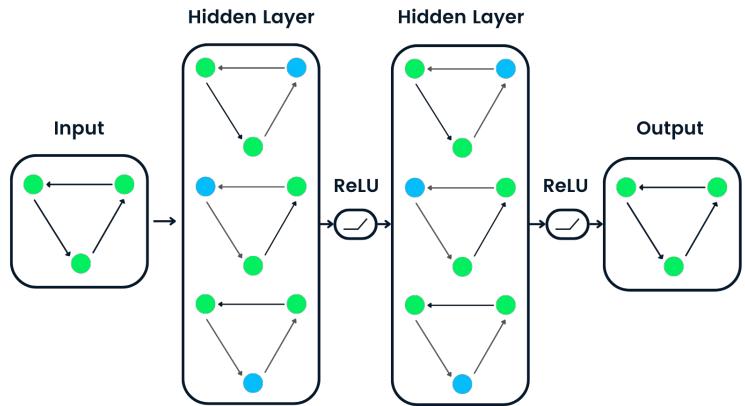
---

- **Graph Attention Network (GAT)**
  - Na GCN, a importância dos nós vizinhos é a mesma
  - Ruídos podem influenciar negativamente no processo de aprendizagem
  - A GAT foca as arestas mais importantes através do mecanismo de atenção
  - A GAT tem atenção às principais relações do grafo, melhorando a agregação das informações
- **GraphSAGE**
  - Novo método de combinação e diferentes agregadores.
  - GraphSAGE realiza uma amostragem nos vizinhos
  - Reduz o tempo e a complexidade da memória

# Graph Neural Networks multi-classe



# Graph Autoencoder (GNN não supervisionada)



GNN vista anteriormente

$$GAE = \begin{cases} \boxed{\text{Encoder : } \mathbf{H}^{(L)} = g(\mathbf{O}, \mathbf{A}; \mathbf{W})} \\ \boxed{\text{Decoder : } \hat{\mathbf{A}} = \sigma(\mathbf{H}^{(L)} \cdot \mathbf{H}^{(L)\top})} \end{cases}$$

Matriz de adjacência reconstruída       $\mathbf{H}^L$  transposto

Função de perda: Diferença entre  $\mathbf{A}$  e  $\hat{\mathbf{A}}$

# Vantagens e Desvantagens da GNN

---

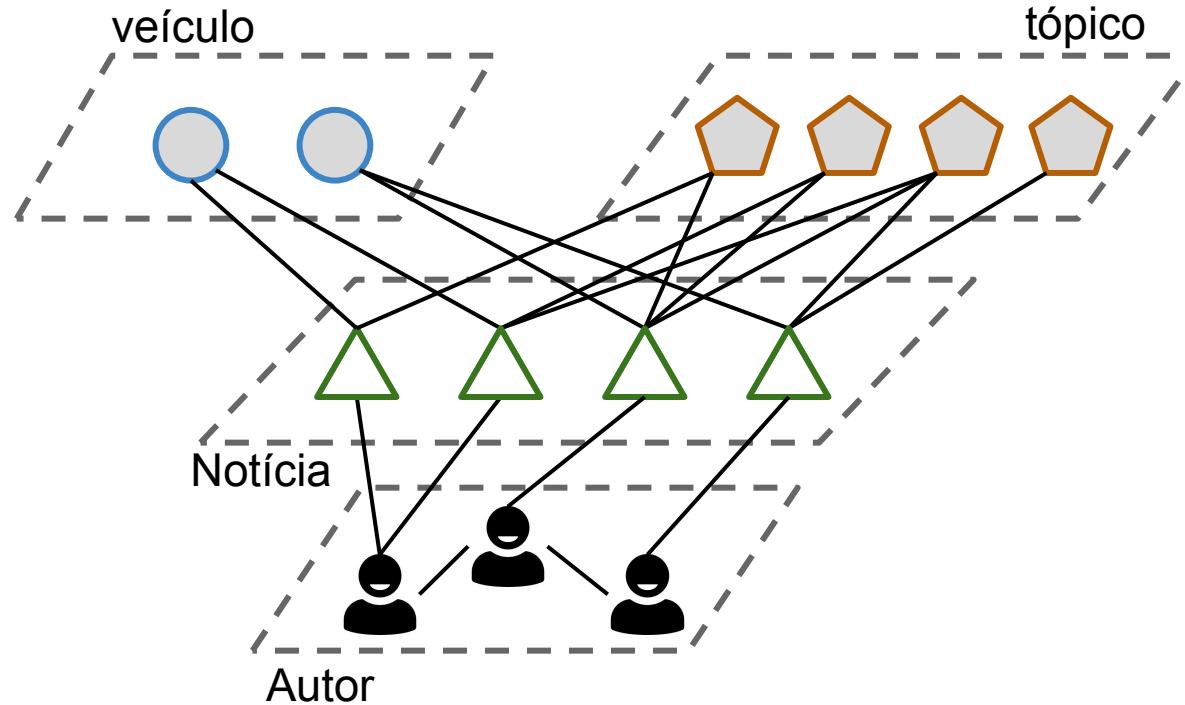
- **Vantagens**

- Resolve **diferentes tarefas** de forma **end-to-end**
  - classificação de nós
  - classificação de aresta
  - predição de aresta
  - classificação de grafos
- Adaptabilidade para diferentes tipos e modelagens de grafo
- Resultados estado-da-arte

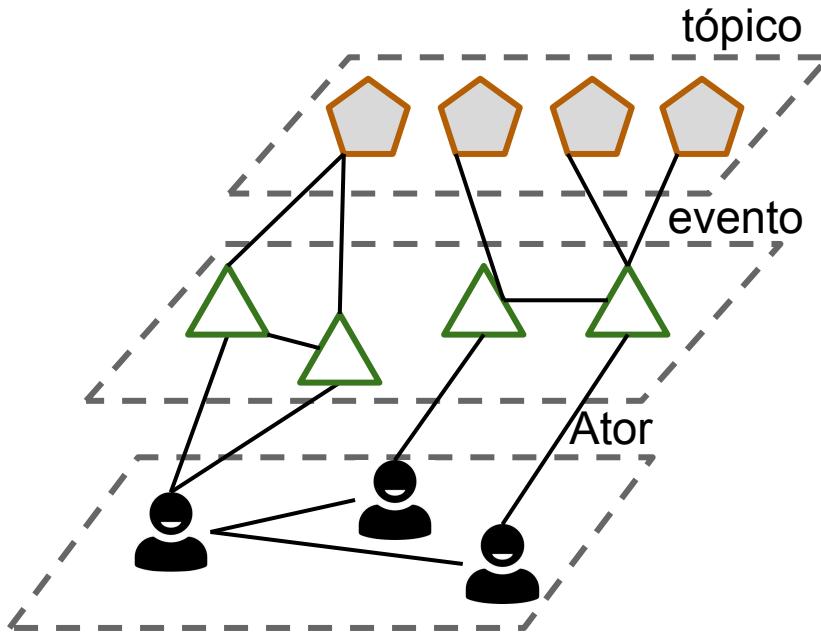
- **Desvantagens**

- Representações convergem para mesmo local com muitas camadas
- Black-box (interpretabilidade)

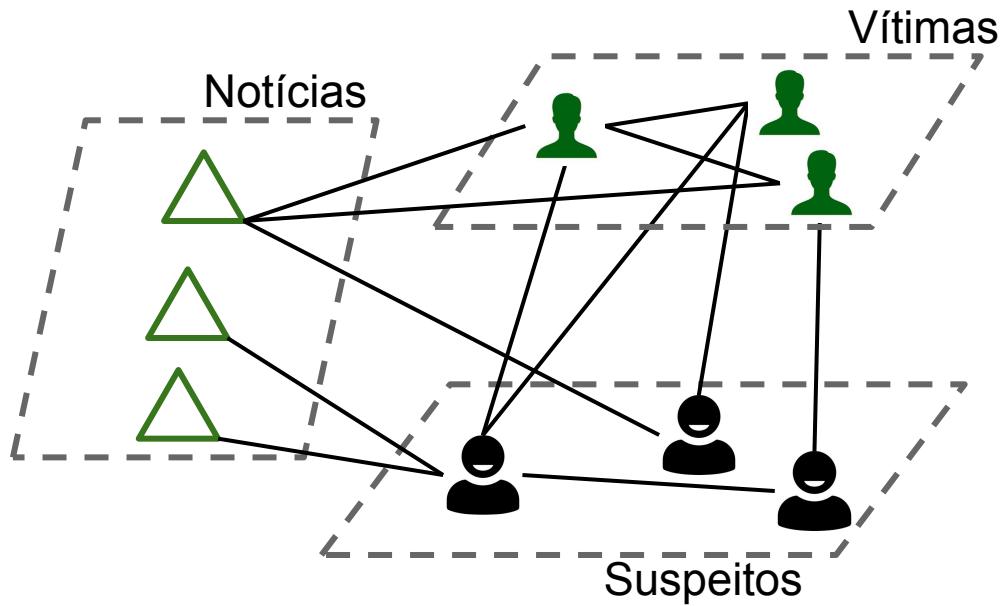
# Contextos de exploração - Detecção Notícias Falsas



# Contextos de exploração - Detecção de Eventos

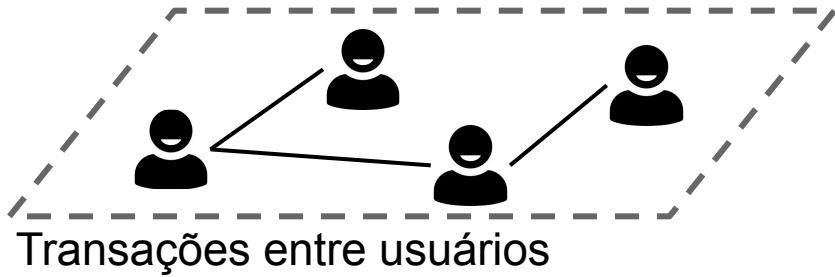


# Contextos de exploração - Detecção de Crimes



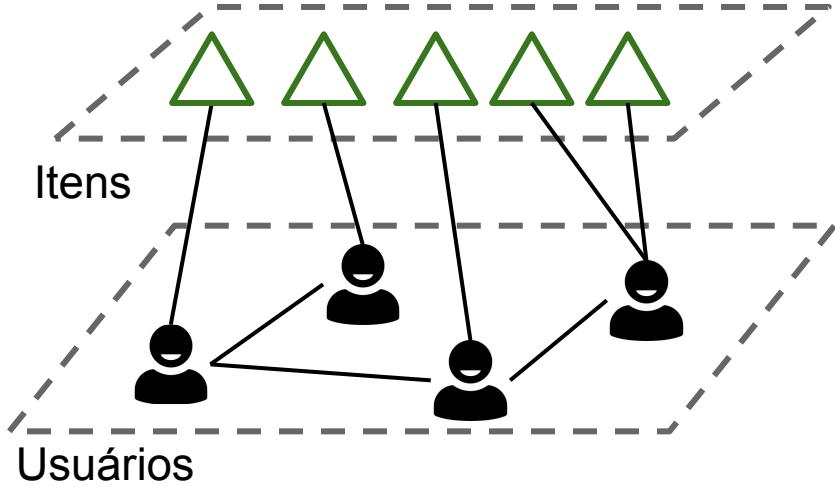
# Contextos de exploração - Detecção de Fraude

---

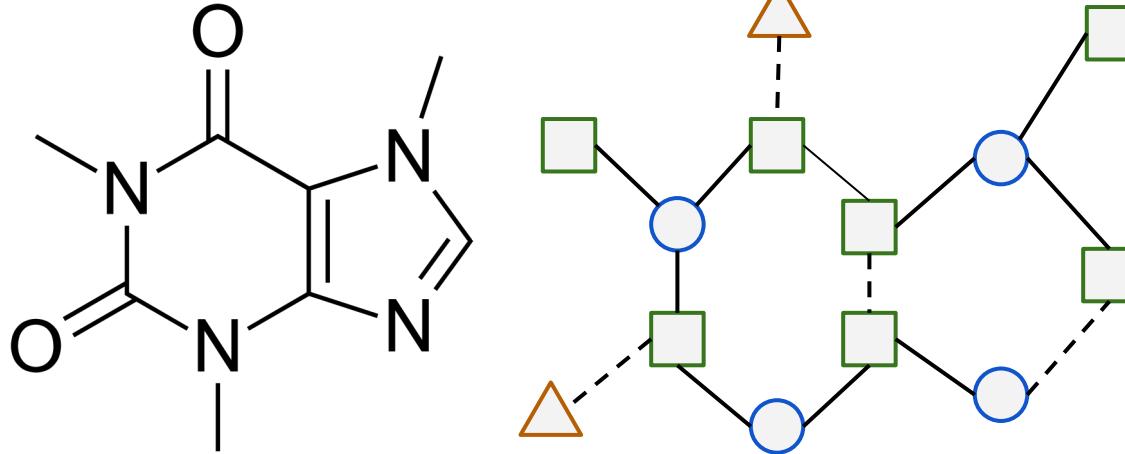


# Contextos de exploração - Recomendação

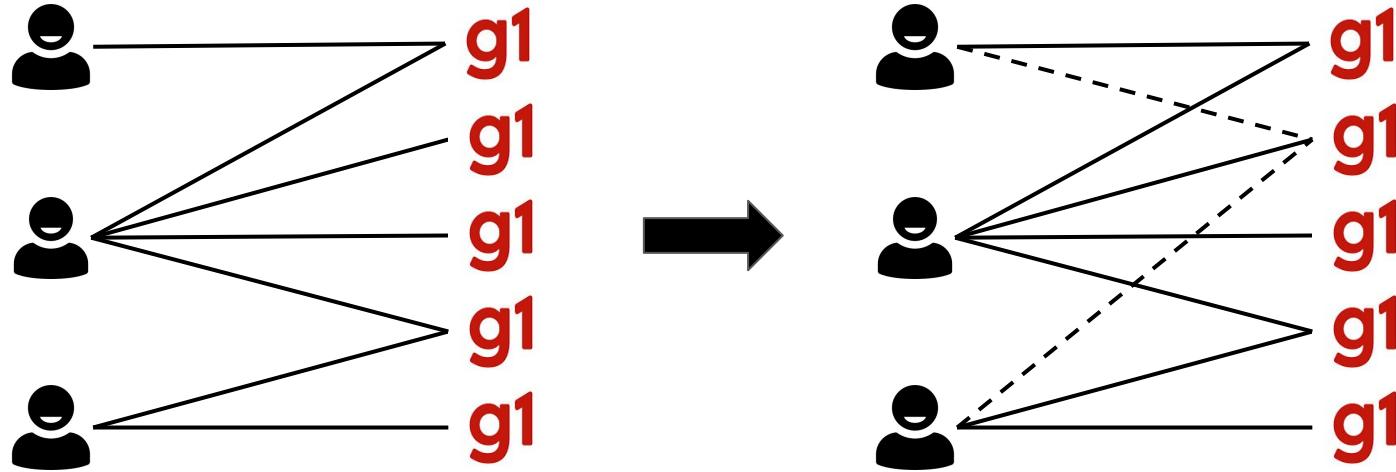
---



# Contextos de exploração - Classificação de Grafos

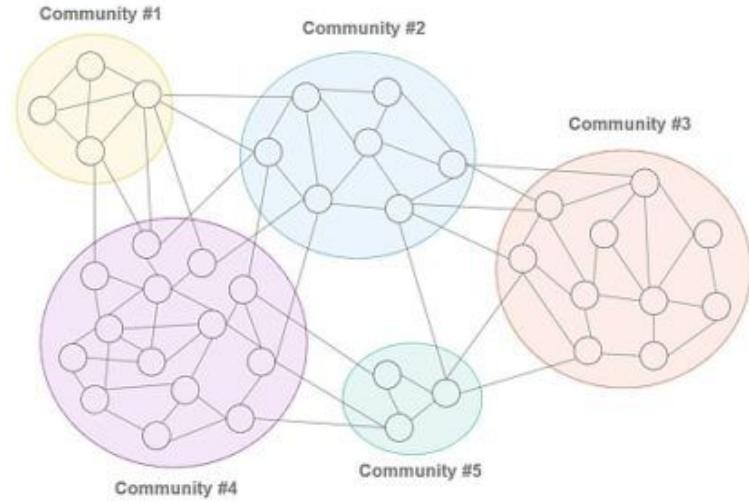
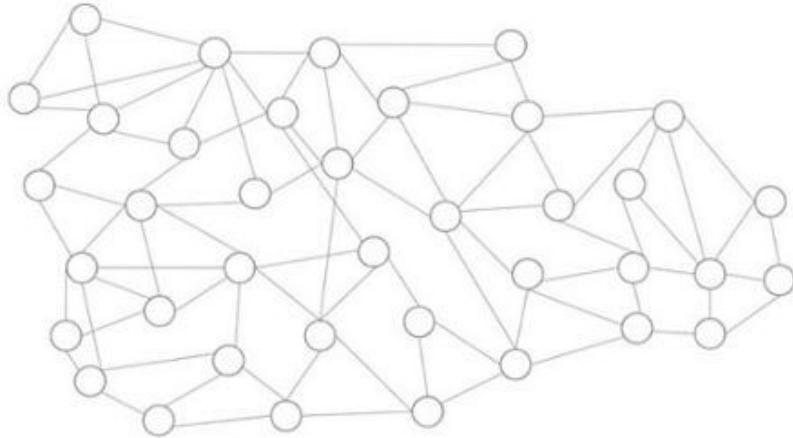


# Contextos de exploração - Recomendação de Notícias



# Contextos de exploração - Detecção de Comunidades

---



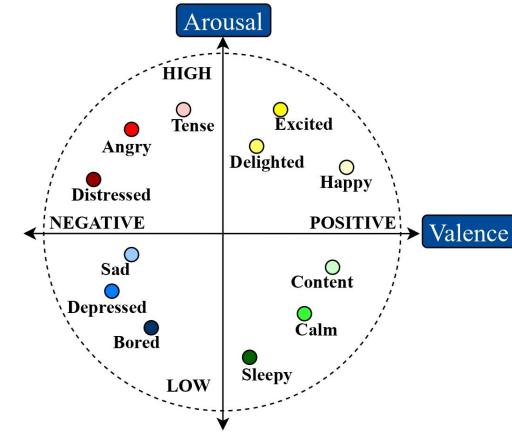
# Contextos de exploração

---

- Exemplos específicos de aplicação que serão explorados na parte prática:
  - Reconhecimento de Emoções em Músicas
  - Fake News Detection

# Reconhecimento de Emoções em Músicas

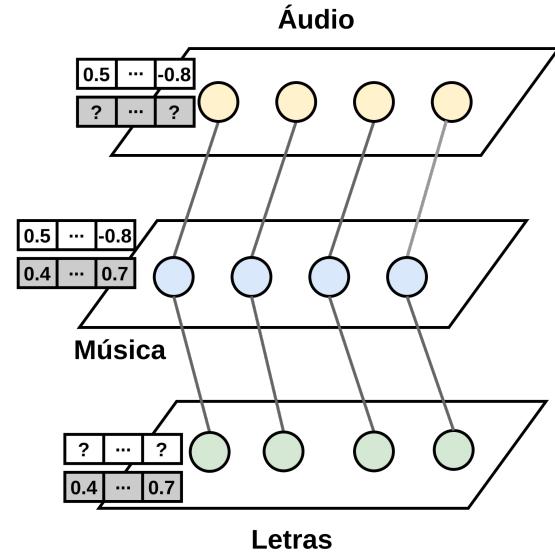
- Reconhecer emoções é uma tarefa importante na área de Recuperação de Informação Musical
- Músicas podem conter rótulos emoções
- Áudio e Letras podem expressar emoções distintas
- Utilizar GNNs é um recurso para combinar as modalidades de áudio e texto e aprender uma representação única mais relacionada à emoção



Mapeamento de emoções nos domínios de excitação e valência

# Reconhecimento de Emoções em Músicas

- Temos um grafo desconexo e vértices com modalidades incompletas
- Iremos construir arestas com agrupamento
- Iremos regularizar o grafo para propagar as embeddings
- Utilizaremos a GNN para aprender uma representação multimodal
- Iremos considerar emoções apenas como Positiva, Negativa e Neutra



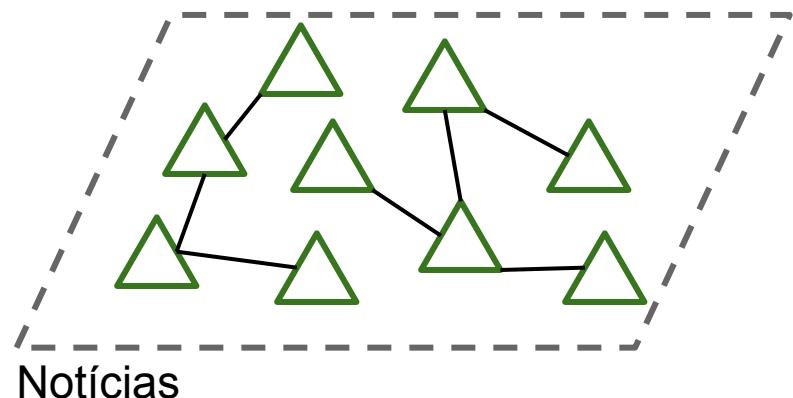
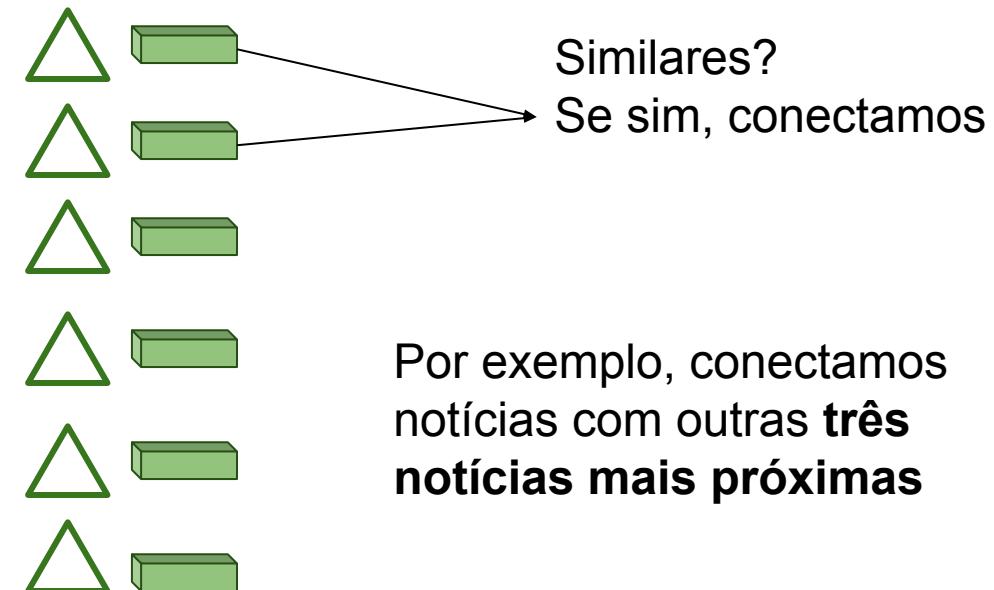
Grafo com vértices de 3 tipos

# Detecção de Notícia Falsa: Dataset

---

- Notícias Falsas sobre política
- Notícias Falsas de 2019
- Classe Real e Falsa
- Anotação por humanos
- Notícias: 2064
- Notícias Falsas: 1044
- Notícias Reais: 1020

# Detecção de Notícia Falsa: Grafo



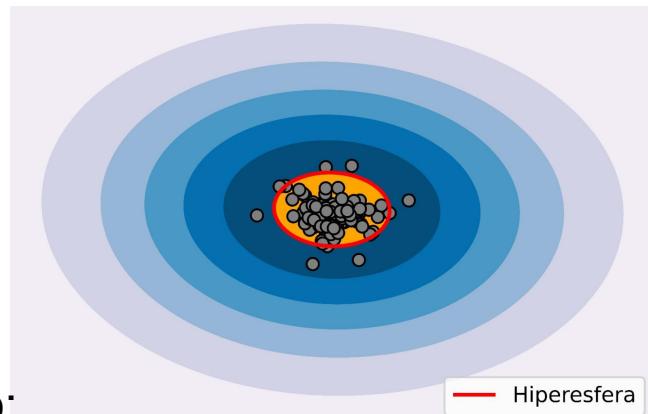
# Detecção de Notícia Falsa

---

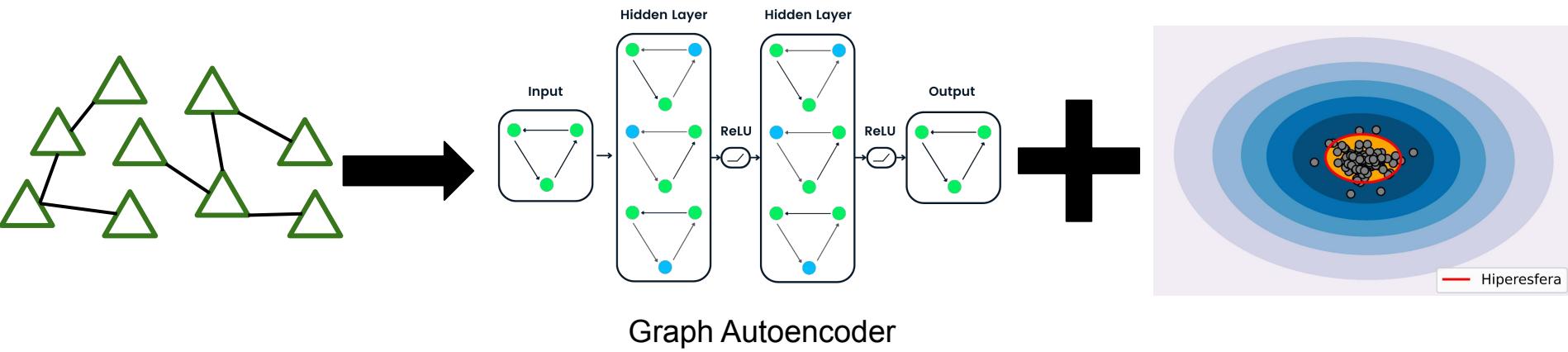
- GNN para duas classes?
- No mundo real, teríamos que **rotular tanto notícias falsas quanto reais.**
- Rotulação é **caro!!!**
- Será que podemos utilizar algum **aprendizado que só precisamos rotular as notícias falsas** que são nosso foco?
- **SIM!!!**

# *One-Class Learning (OCL)*

- No OCL o treinando é apenas em amostras de **uma classe** (classe de interesse). **Ausência de contra-exemplos**.
- OCL **reduz esforços** de rotulação e **não exige cobertura** abrangente da classe de não interesse
- Aplicações de **domínio aberto** ou quando há **interesse em uma única classe** do problema
- Após treinar, funciona com um classificador binário:
  - Classe de interesse
  - Não classe de interesse

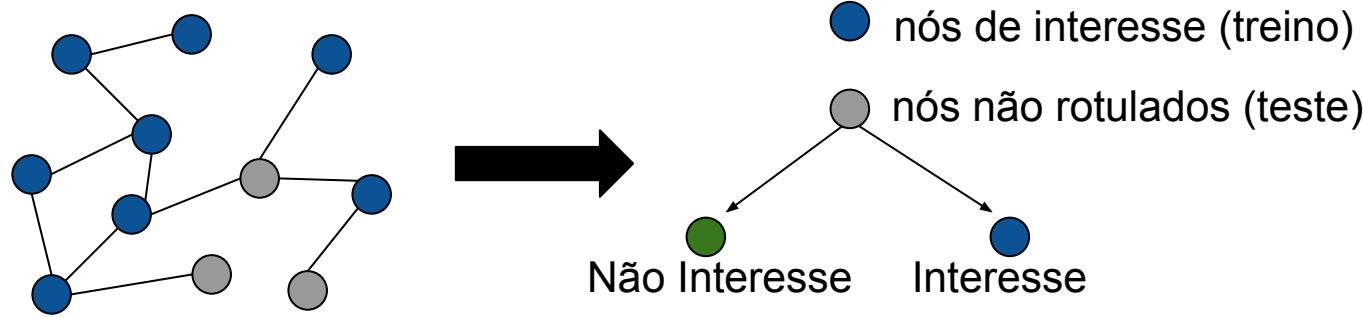


# Pipeline



# Tarefa

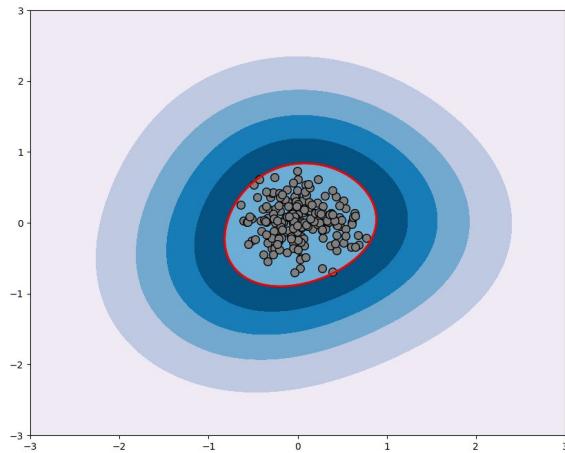
---



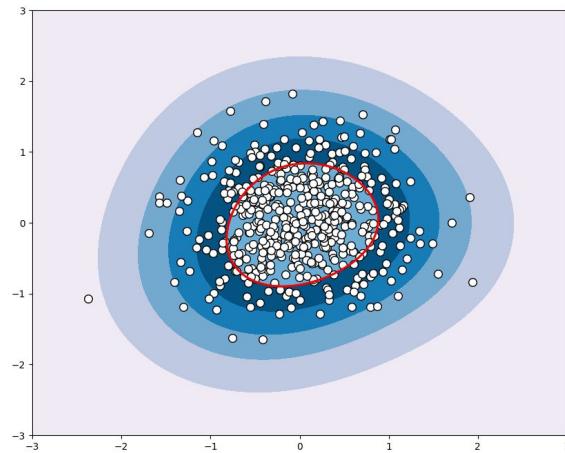
# One-Class Learning

---

Treino



Teste



- notícias falsas
- notícias não rotuladas

# Vamos praticar?

---

- [Código 1](#)
- [Código 2](#)

Obrigado!

## Introdução às *Graph Neural Networks* e suas aplicações em tarefas de aprendizado de máquina

Angelo Mendes

Marcos Gôlo

Ricardo Marcacini

