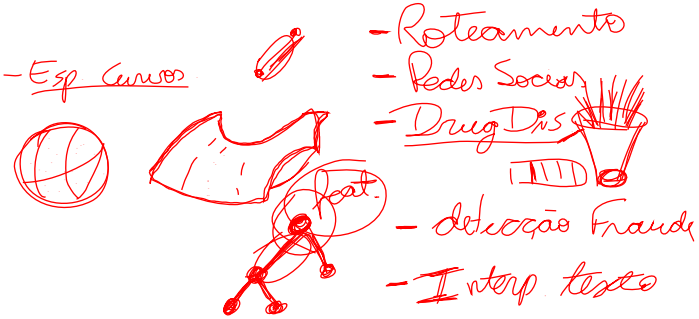


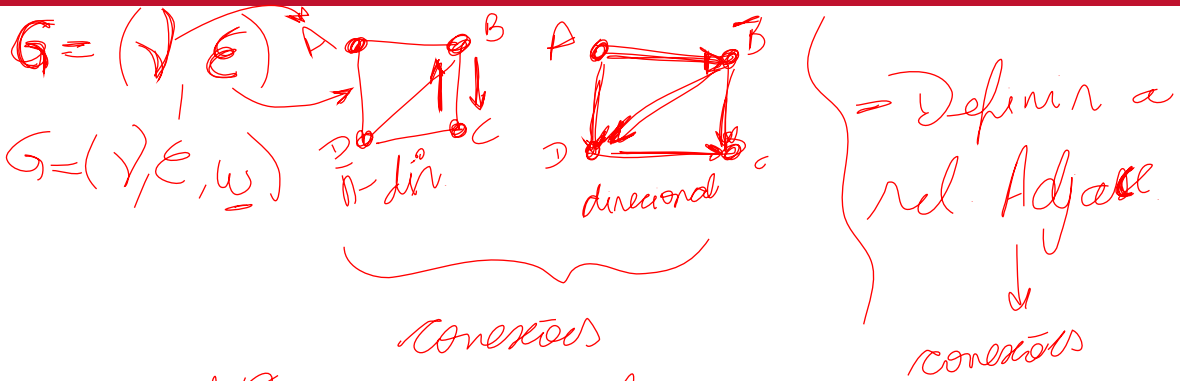
Advanced Institute for Artificial Intelligence

Graph Neural Networks

Advanced Institute for Artificial Intelligence – AI2

https://advancedinstitute.ai

Note


G = (V, E)
G = (V, E, W)
A
B
C
D
não-dir
direcional
conexões
= Defina a
rel Adjac
conexões
cíclico
A-cíc

Introdução

Sabemos que um grafo pode ser definido como $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, em que $\mathcal{V}$ e $\mathcal{E}$ correspondem ao conjunto de **vértices** (ou nós) e **arestas**, respectivamente. Grafos podem ser utilizados para modelar diversas entidades, tais como imagens, textos, interações entre usuários em redes sociais e moléculas, por exemplo.

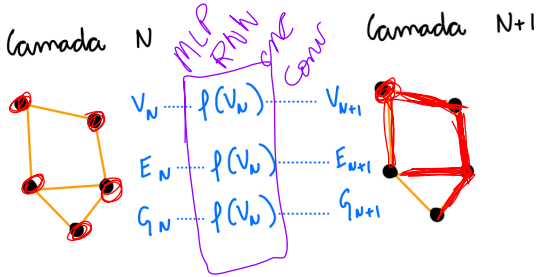
Tarefas de predição em grafos podem ser efetuadas em três níveis:

- Grafo: a tarefa é a de predizer as propriedades do grafo propriamente dito (ex: qual o tipo de molécula que estamos analisando?).
- Vértice: a tarefa é a predizer o papel de cada nó no grafo (ex: classificação de um determinado usuário em uma rede social).
- Arestas: a tarefa é a de predizer a ação representada por uma determinada aresta (ex: classificação de ações em imagens).

Graph Neural Networks (GNNs) podem ser entendidas como uma transformação otimizável em todos os atributos do grafo (nós, arestas e o grafo propriamente dito) que **preserva as suas simetrias** (invariâncias à permutações). De maneira geral, podemos usar diferentes informações para cada nível de atuação de uma GNN:

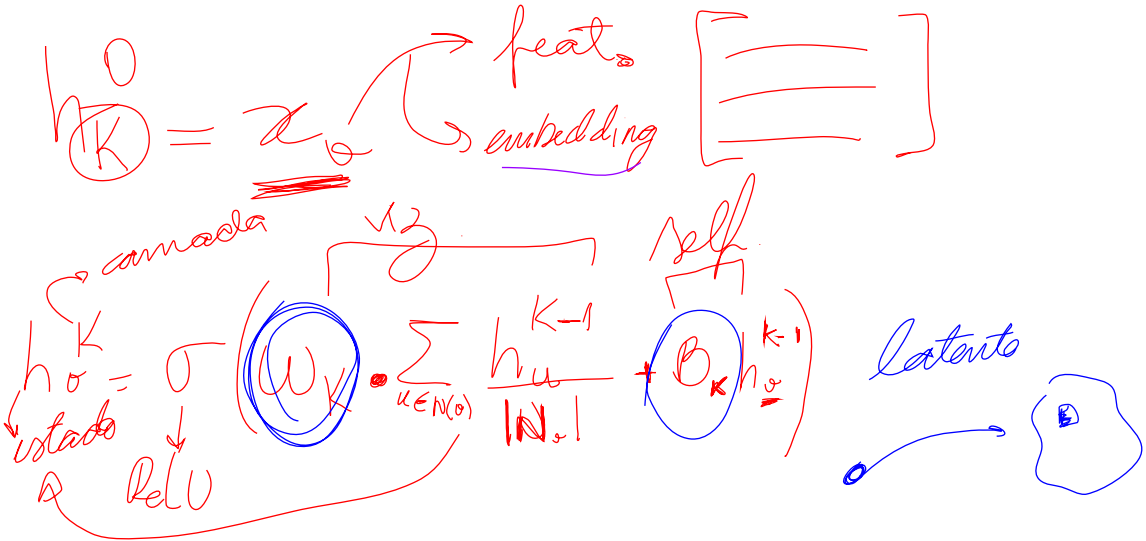
- Grafo: número de nós, caminho mais longo, valor calculado na MST (*Minimum Spanning Tree*), etc.
- Vértice: número de vizinhos, peso do nó, etc.
- Arestas: peso da aresta, etc.

Usualmente, uma GNN emprega uma rede MLP (*Multilayer Perceptron*) em cada componente do grafo, dando origem ao que chamamos de **camada GNN**.

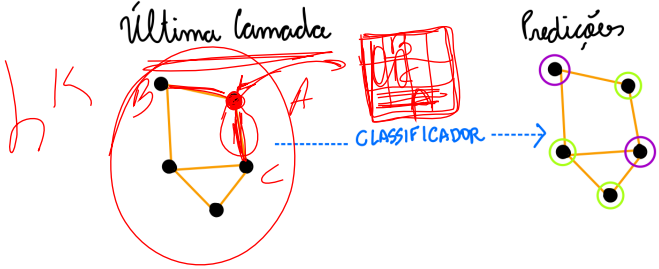


Assim, temos que $f(\cdot)$ é a função (MLP) que recebe como entrada uma entidade (grafo, vértice ou aresta) e retorna algum tipo de processamento sobre ele. Note, portanto, que uma GNN não atualiza a conectividade do grafo, representada pela sua lista de adjacências.

Note



Dado que temos uma GNN, como podemos utilizá-la para fazer predições? Vamos considerar um caso de **classificação binária**. Caso a tarefa seja a de realizar predições nos nós e eles já possuem alguma informação (*node embedding*), basta aplicarmos um classificador em cada um deles para obter a predição.



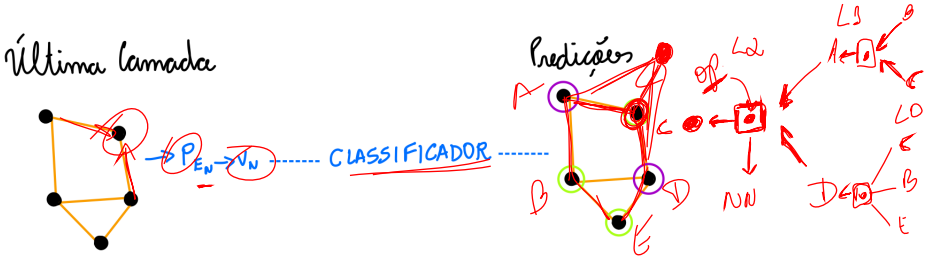
No entanto, podemos enfrentar situações mais complicadas como, por exemplo, ter informações nas arestas mas ainda querer realizar predições nos nós. Assim, precisamos de uma maneira para coletar informações das arestas e passá-las aos nós para fins de predição.

Podemos realizar tal etapa por meio de uma operação de *pooling*. Podemos realizar isto em duas etapas:

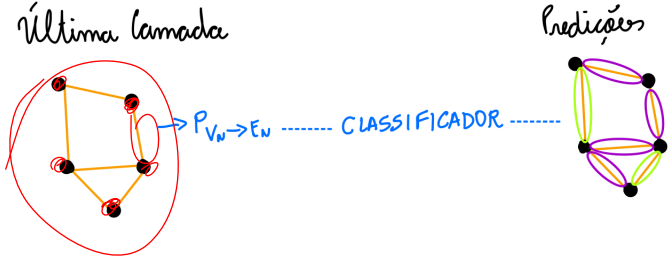
- Para cada entidade (aresta) que iremos aplicar o *pooling*, podemos criar uma matriz concatenando suas informações (embeddings). Ex: para um nó com duas arestas adjacentes, suas respectivas informações serão agrupadas.
- Agregar as informações (usualmente uma operação de soma é aplicada à todas elas).

Podemos representar a operação de pooling pela letra ρ e denotar que estamos juntando (passando) informações de arestas para os nós da seguinte maneira: $\rho_{\mathcal{E}_N \to \mathcal{V}_N}$, em que N representa a camada em questão que a operação está sendo realizada.

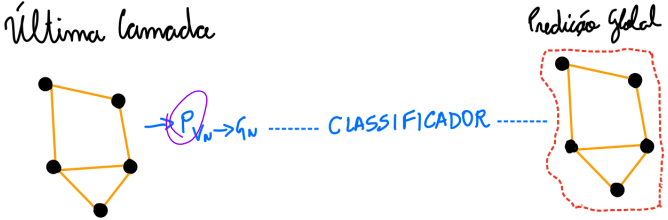
Assim sendo, caso tenhamos apenas informações sobre arestas e queiramos realizar predições nos nós, podemos utilizar *pooling* para passar (rotear) informações para serem utilizadas posteriormente.



Por outro lado, caso tenhamos apenas informações sobre os nós e queiramos realizar predições nas arestas, podemos realizar algo semelhante.



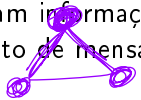
Ainda, caso tenhamos informações sobre os nós/arestas e queiramos inferir algo a respeito do grafo como um todo, podemos adotar algo similar, ou seja, rotear informações dos nós/arestas e tomar a decisão. Isso seria equivalente à operação de *global average pooling* de uma CNN.



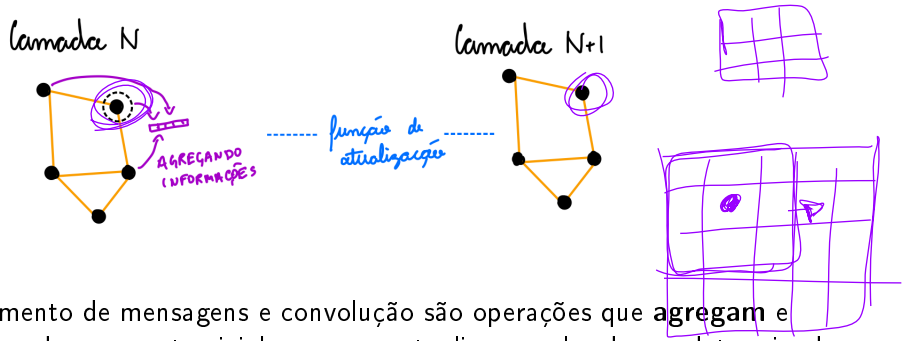
Mostramos, então, como usar uma GNN para realizar predições simples em um problema de classificação binário roteando (passando) informações entre **diferentes partes do grafo**. Caso tenhamos informações adicionais acerca das entidades do grafo, basta apenas que definamos uma maneira de rotear essas informações.

Roteando Mensagens

Podemos realizar predições mais sofisticadas por meio de operações de *pooling* em camadas GNN de tal forma a levar em consideração informações de conectividade do grafo. A ideia é fazer uso da chamada **roteamento de mensagens**, em que nós/arestas vizinhas trocam informações e, assim, influenciam a atualização dos seus respectivos *embeddings*. Roteamento de mensagens, de maneira geral, funciona da seguinte maneira:



1. Para cada nó do grafo, junte todas as informações ou mensagens dos nós vizinhos.
2. Realize uma agregação dessas informações utilizando alguma **função de agregação**.
3. Para todas as mensagens que passaram por alguma operação de agregação, encaminhe-as para alguma **função de atualização** (ex: rede neural).

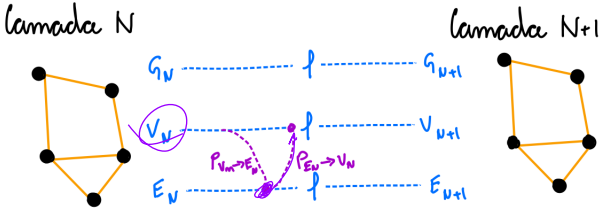


Essencialmente, roteamento de mensagens e convolução são operações que **agregam** e **processam** informações de uma certa vizinhança para atualizar o valor de um determinado elemento. Em GNNs, cada elemento é um vértice; já em uma CNN cada elemento é um *pixel*. Empilhando essas camadas GNN de roteamento de mensagens é uma maneira de fazer com que os nós recebam informações do grafo como um todo.

Aprendendo Representações de Arestas

Nossa base de dados nem sempre irá conter todo o tipo de informação (nós, arestas e contexto global do grafo). Já mostramos como fazer uso de uma operação de *pooling* para rotear mensagens de arestas para nós (ou vice-versa), mas somente na última camada antes da predição final. No entanto, podemos fazer o mesmo entre camadas internas de uma GNN.

No entanto, representações de nós e arestas não possuem, necessariamente, o mesmo tipo e tamanho. Assim, ainda não está claro como combinar esses tipos de informação. Uma alternativa seria aprender um **mapeamento linear** do espaço dos nós para o espaço de arestas e vice-versa, ou ainda **concatená-los**.


Camada N
Camada N+1
G_N
G_{N+1}
V_N
V_{N+1}
E_N
E_{N+1}
$P_{V_m \to E_n}$
$P_{E_n \to V_N}$

Conclusões

GNNs são modelos poderosos em duas principais situações:

- Dados estão, naturalmente, representados em uma estrutura de grafo.
- Dados podem ser representados em uma estrutura de grafo com interferência mínima em sua estrutura.