Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Guilherme Henrique Messias

Redes Neurais em Grafos para Aprendizado Positivo: Uma abordagem utilizando Reescrita

São Carlos
2024

Guilherme Henrique Messias

Redes Neurais em Grafos para Aprendizado Positivo: Uma abordagem utilizando Reescrita

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Alan Demétrius Baria Valejo
Coorientador: Sylvia Iasulaitis

São Carlos
2024

Resumo

Diante do crescente volume de dados gerado pela sociedade, a realização de tarefas de classificação tem se tornado ainda mais complexa, uma vez que alguns algoritmos de classificação supervisionada demandam uma quantidade substancial de dados rotulados a fim de alcançar um bom desempenho. Por outro lado, mesmo com a utilização de poucos exemplos rotulados da classe positiva, resultados relevantes podem ser obtidos por algoritmos de aprendizagem semissupervisionada baseados em aprendizado positivo e não rotulado (PUL). Esses algoritmos podem ser aplicados a uma variedade de tipos de dados, incluindo dados relacionais, que podem ser representados por grafos. Nesse contexto, algoritmos de aprendizado de máquina baseados em redes neurais apresentam resultados relevantes, principalmente as Redes Neurais em Grafos (GNNs). Ainda, a recente área de pesquisa reescrita em grafos (do inglês *Graph Rewiring*) tem mostrado um ganho de performance nas GNNs. Esse trabalho busca explorar a utilização de reescrita e redes neurais em grafos para tratar de problemas de PUL, sob a hipótese de que essas abordagens sejam capazes de aprender um espaço de representação latente enviesado pelos dados positivos. Os resultados em experimentos feitos com uma gama de conjuntos de dados mostram que a abordagem proposta é superior a algoritmos clássicos e atuais da literatura, tanto na extração de dados negativos quanto na tarefa de PUL.

Palavras-chave: Aprendizado Positivo, Grafos, Redes Neurais para Grafos.

Abstract

Given the increasing volume of data generated by society, the task of classification has become even more complex, as some supervised classification algorithms require a substantial amount of labeled data to achieve good performance. On the other hand, even with the use of a few labeled examples from the positive class, relevant results can be obtained by semi-supervised learning algorithms based on Positive and Unlabeled Learning (PUL). These algorithms can be applied to a variety of data types, including relational data, which can be represented by graphs. In this context, machine learning algorithms based on neural networks show relevant results, particularly Graph Neural Networks (GNNs). Additionally, the recent research area of graph rewiring has shown a performance gain in GNNs. This work aims to explore the use of graph rewiring and graph neural networks to address PUL problems, under the hypothesis that these approaches are capable of learning a latent representation space biased by the positive data. The results of experiments conducted on a range of datasets show that the proposed approach is superior to classical and current algorithms in the literature, both in negative data extraction and in the PUL task.

Keywords: Positive and Unlabeled Learning. Graph Neural Networks.

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

Com o desenvolvimento de plataformas em rede, os grafos se tornaram ferramentas fundamentais para a modelagem de dependência de dados (XIA et al., 2021). Assim, os grafos tem sido usados cada vez mais em aplicações de problemas reais para formular relações em redes sociais (NEWMAN; WATTS; STROGATZ, 2002), moléculas (GUO et al., 2021), tráfego de veículos (XIA et al., 2020), redes de citações (LIU et al., 2020), redes de comunicações (LIU et al., 2022), entre outros. Porém, sistemas modelados em grafos representam estruturas complexas que carregam características topológicas e funcionais de um conjunto de observações, dificultando a análise e absorção de informações, principalmente quando a quantidade de dados escala para centenas de milhões de exemplos.

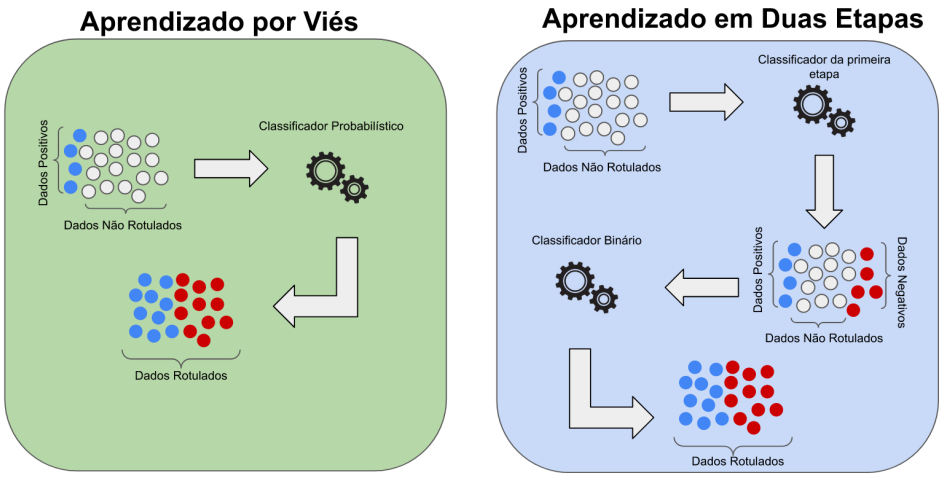
Dada a extensa aplicação de grafos em modelagem de problemas, modelos de aprendizado de máquina (AM) passaram a ser utilizados em grafos. Grover e Leskovec (2016) propuseram uma abordagem para transformar a estrutura do grafo em uma representação vetorial, tornando possível a aplicação de métodos tradicionais de AM em dados relacionais. Na mesma linha, Ahmed et al. (2013) apresentaram um *framework* para lidar com grafos em grande escala.

Com o avanço das pesquisas e com os expressivos resultados de redes neurais, alguns *frameworks* foram propostos para sua utilização em grafos, conhecidos pelo nome Redes Neurais em Grafos (do inglês *Graph Neural Networks*). Sperduti e Starita (1997) foram os pioneiros na utilização de redes neurais para trabalhar diretamente na estrutura de grafos, propondo um modelo baseado no uso de redes neurais recorrentes. Scarselli et al. (2008) definiram o conceito de *Graph Neural Network* (GNN) utilizando também redes neurais recorrentes. Diversos outros trabalhos desenvolveram modelos de redes neurais em grafos utilizando redes neurais profundas (KIPF; WELLING, 2016a; VELIČKOVIĆ et al., 2017) e *AutoEncoders* (KIPF; WELLING, 2016b; NG et al., 2019; CEN et al., 2020). O sucesso

das redes neurais em grafos ampliou a área de reconhecimento de padrões e análise de dados em sistemas com características relacionais (WU et al., 2021).

O crescimento da quantidade de dados produzidos pela sociedade também torna cada vez mais complexa tarefas de classificação. Existem diversos tipos de classificação, sendo uma delas a classificação binária, cujo objetivo é aprender um modelo que seja capaz de distinguir entre dados positivos e negativos. Uma variante do modelo de classificação binária é o aprendizado por exemplos positivos e não rotulados (*Positive and Unlabeled Learning* – PUL), no qual os dados de treino consistem apenas de exemplos rotulados da classe positiva e de exemplos não rotulados (BEKKER; DAVIS, 2020). Dentre as aplicações de PUL estão: classificação de notícias falsas (SOUZA et al., 2021), interações entre fármacos (ZHENG et al., 2019), identificação de genes (YANG et al., 2012; YANG et al., 2014) e classificação automática de textos (PENG; ZUO; HE, 2007). Um dos desafio do PUL é que existem relativamente poucos dados rotulados positivos e muitos dados não rotulados, tornando complexo o processo de aprendizado.

Existem duas técnicas principais de PUL: o aprendizado por viés (*biased learning*) e o aprendizado em duas etapas (*two-step technique*) (BEKKER; DAVIS, 2020). O aprendizado por viés trata exemplos não rotulados como negativos com ruído, por isso, grande parte dos algoritmos que utiliza essa técnica vem de modelos baseados em Máquinas de Vetor de Suporte (*Support Vector Machine* – SVM), pelo fato de que SVMs conseguem penalizar a classificação errada de um exemplo. O aprendizado em duas etapas consiste em: (1) Identificar exemplos negativos seguros que são mais dissimilares dos positivos e (2) classificar os dados utilizando um classificador binário. A Figura 1 ilustra o funcionamento das duas abordagens. Nesta dissertação o foco serão as técnicas de PUL em duas etapas.



(a) Aprendizado por Viés (b) Aprendizado em Duas Etapas

Figura 1 – Abordagens de Aprendizado Positivo. Na Figura (a) e (b) são ilustradas as abordagens de aprendizado por viés e em duas etapas, respectivamente.

Considerando o aprendizado em duas etapas e partindo do pressuposto de que exem-

plos próximos tendem a compartilhar os mesmos rótulos, os algoritmos presentes na literatura utilizam técnicas de agrupamento ou de similaridade para inferir os rótulos negativos seguros. Nesse contexto, algoritmos baseados em redes neurais para grafos são escassos e as pesquisas atuais não exploram completamente o desenvolvimento teórico desses modelos. Além disso, de acordo com o nosso conhecimento, nenhum trabalho explorou a utilização de GNN no processo de separação de dados positivos e negativos, ou seja, na primeira etapa.

1.1 Objetivos

O objetivo geral desse trabalho é explorar o potencial das GNNs em um algoritmo de PUL baseado em duas etapas. Deste objetivo geral foram derivados três objetivos específicos, descritos a seguir.

1. Apresentação de um modelo capaz de se beneficiar da conectividade entre vértices positivos;

2. Desenvolver algoritmos que explorem a topologia do grafo com a finalidade de facilitar a transferência de mensagem; e

3. Analisar algoritmos da literatura de PUL em duas etapas quanto à inferência de dados negativos seguros.

No primeiro objetivo específico, para explorar o potencial do uso de GNNs no problema de PUL, apresentamos um modelo que se beneficia da conectividade entre os vértices positivos. Esse modelo, inspirado no modelo *Relational Graph Convolutional Network* (RGCN) (SCHLICHTKRULL et al., 2018), considera que arestas entre vértices positivos e arestas incidentes em vértices não rotulados desempenham papéis distintos no processo de aprendizado, uma vez que o modelo prioriza a transferência de mensagem entre vértices positivos. De acordo com o nosso conhecimento, este modelo ainda não foi testado em um contexto de PUL.

Para alcançar o segundo objetivo específico iremos nos embasar na recente área de pesquisa conhecida como reescrita em grafos (do inglês *Graph Rewiring*), que consiste em modificar (criar, remover e/ou reponderar) as arestas de um grafo para que o fluxo de informações seja otimizado para uma tarefa específica, como classificação de vértices ou predição de arestas. Os resultados dessa área mostraram um ganho de performance nas GNNs visto que resulta em um facilitador da propagação de informação pelo grafo (BARBERO et al., 2024; AKANSHA, 2024).

Como as GNNs utilizam a vizinhança de um vértice para transferir informações a respeito de suas características, é plausível, considerando que o conjunto de vértices positivos é menor que o conjunto de vértices não rotulados, modelar uma abordagem de reescrita

capaz de adicionar novas conexões entre esses vértices. Além disso, algoritmos de reescrita em grafos são utilizados majoritariamente no contexto semissupervisionado e, de acordo com o nosso conhecimento, não encontramos um trabalho que avalie tais algoritmos no processo de inferência de dados negativos em tarefas de PUL.

Dessa forma, o modelo de GNN e de reescrita irão trabalhar em conjunto e o desenvolvimento desses algoritmos se apoia sobre a hipótese de que a arquitetura relacional e o modelo de reescrita serão capazes de gerar um espaço de representação latente (do inglês *embedding*) que distancia os dados positivos dos dados negativos, facilitando a aplicação de uma medida simples de similaridade para encontrar os exemplos negativos seguros.

Assim, a abordagem proposta se resume da seguinte forma: (1) arestas artificiais são utilizadas para facilitar a propagação de informação entre vértices positivos; (2) um modelo de GNN é utilizado para gerar um *embedding* para os vértices; (3) é gerado um ranking com os vértices não rotulados mais distantes dos vértices positivos e os top-k vértices mais distantes são rotulados como vértices negativos seguros e (4) por fim, um algoritmo de classificação binário é treinado com os vértices positivos e vértices negativos seguros e utilizado para classificar os vértices ainda não rotulados.

Finalmente, a fim de compreender o funcionamento tanto dos algoritmos da literatura quanto da abordagem proposta, o terceiro objetivo específico consiste em realizar uma análise empírica da primeira etapa (ou seja, na inferência de dados negativos seguros) de algoritmos da literatura para problemas de PUL em duas etapas.

Diversos algoritmos de PUL baseados em duas etapas foram propostos na literatura (CHAUDHARI; SHEVADE, 2012; MA; ZHANG, 2017; YU; LI, 2007; LIU; PENG, 2014; LI; LIU, 2003), porém, nenhum desses apresenta avaliações da primeira etapa de forma isolada. No geral, os métodos existentes são avaliados considerando a solução final do classificador, ou seja, após a segunda etapa. Avaliar a primeira etapa desses algoritmos é considerar quão bem o algoritmo separa exemplos positivos e negativos seguros, sem considerar sua classificação final após a segunda etapa.

É importante avaliar a primeira etapa, pois o classificador binário da segunda etapa pode depender de quão bem os dados negativos seguros são inferidos, melhorando o resultado final da classificação. Esse objetivo pretende, além de verificar a acurácia dos algoritmos nessa etapa, também verificar o quanto o desempenho na separação de dados positivos e negativos influencia na classificação do algoritmo. Dessa forma, será possível identificar limitações de algoritmos clássicos e também encontrar lacunas que possam ser exploradas em trabalhos futuros.

Para validar o modelo proposto foram realizados três experimentos. Primeiro, o algoritmo proposto foi testado com diferentes parâmetros (análise paramétrica). O objetivo é entender a influência dos parâmetros em diferentes conjunto de dados. Em seguida avaliou-se o algoritmo quanto à inferência de dados negativos seguros, ou seja, se o algoritmo é realmente capaz de inferir dados negativos a partir de exemplos positivos.

Finalmente, foi avaliada a classificação geral de todos os exemplos na tarefa de PUL. Nos dois últimos experimentos, nossa proposta foi confrontada com cinco algoritmos clássicos e atuais da literatura.

Os resultados obtidos pelos experimentos mostram que o modelo proposto é capaz de aprender um *embedding* enviesado pelos dados positivos, ou seja, exemplos positivos encontram-se distantes de exemplos negativos nesse espaço, facilitando a inferência de exemplos negativos seguros e, consequentemente, facilitando a classificação binária na última etapa. Além disso, observou-se que o modelo proposto apresenta resultado superior quando comparado à uma GCN tradicional, tanto na inferência de dados negativos quanto na classificação binária.

1.2 Contribuições

Este trabalho possui cinco principais contribuições, a saber:

1. Modelo de reescrita sequencial em grafos para problemas de PUL, denominado *Positive Sequential Rewiring via Breadth Search* (PSRB);

2. Modelo de GCN baseado no RGCN (SCHLICHTKRULL et al., 2018), denominado PSRB+RGCN, capaz de entender a relação entre os vértices positivos e não rotulados, bem como explorar a criação de novas arestas criadas pelo modelo de reescrita PSRB;

3. Análise paramétrica do modelo proposto;

4. Análise de desempenho, confrontando cinco algoritmos da literatura tanto na inferência de dados negativos quanto na tarefa de PUL;

5. Criação de um *framework* contemplando os principais algoritmos de PUL baseados em duas etapas;

6. Adaptação da GCN (KIPF; WELLING, 2016a) como um baseline para comparação com o modelo proposto na separação de dados positivos e negativos. De acordo com o nosso conhecimento, esse modelo ainda não foi utilizado para esse propósito. O trabalho mais próximo encontrado foi o de Wu et al. (2021a) que utiliza uma GCN como *embedding* para modelos de PUL baseados em aprendizado por viés.

7. Submissão de um artigo com o modelo proposto.

1.3 Organização do texto

O texto está organizado da seguinte forma:

- ❏ No Capítulo 2 são apresentados conceitos que fundamentam o trabalho, como aprendizado positivo, redes neurais, algoritmos para atualização de pesos da rede, etc;
- ❏ No Capítulo 3 são apresentados os materiais e os métodos, tais quais as abordagens propostas e as ferramentas para testar os algoritmos;
- ❏ No Capítulo 4 são apresentados resultados preliminares e testes paramétricos; e
- ❏ No Capítulo 5 são apresentadas as discussões e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Esse capítulo apresenta a fundamentação teórica para a compreensão do trabalho. Na seção 2.1 estão presentes os conceitos básicos de aprendizado de máquina, a seção 2.2 apresenta os conceitos de aprendizado semissupervisionado e não supervisionado em grafos. A seção 2.3 descreve arquiteturas de redes neurais tradicionais e em grafos e o algoritmo *back-propagation*, além dos modelos de GNN que serão utilizados nesse trabalho. Finalmente a seção 2.4 apresenta os modelos de PUL que serão utilizados como *baselines* para comparação.

2.1 Aprendizado de Máquina

A complexidade dos problemas computacionais e o crescente volume de dados produzidos pela sociedade moderna têm servido como motivação para o desenvolvimento de ferramentas computacionais autônomas, independentes da intervenção humana, destinadas à resolução de tarefas (FACELI et al., 2011). O Aprendizado de Máquina (AM) é uma subárea da Inteligência Artificial (IA) que busca modelos que possam aprender por meio da experiência (MITCHELL, 1997).

Segundo Mitchell (1997) AM consiste em um programa de computador que aprende a partir de uma experiência **E** com respeito a uma classe de tarefas **T** e uma medida de performance **P**, se a performance na tarefa **T** medida por **P** melhora com a experiência **E**.

A aprendizagem indutiva representa uma abordagem amplamente difundida no campo de AM, cujo objetivo é a inferência de regras ou padrões com base em conjuntos de dados previamente observados. O aprendizado indutivo pode ser dividido em quatro

principais tipos: supervisionado, semissupervisionado, não-supervisionado e por reforço (RANI; NAYAK; VYAS, 2015).

- Aprendizado supervisionado: Consiste na criação de um modelo capaz de aprender a partir de um conjunto de treinamento rotulado e generalizar para novos exemplos (KOTSIANTIS et al., 2007). Para esse tipo de aprendizado, tem-se a necessidade da figura de um "professor externo" que apresenta o conhecimento do ambiente por conjuntos de exemplos na forma de entrada e saída desejada (HAYKIN, 2009). Assim, um algoritmo de aprendizado de máquina é treinado com um conjunto de dados rotulados X e tem por objetivo mapear esse conjunto de dados para os rótulos corretos do conjunto Y (ALPAYDIN, 2014; FACELI et al., 2011).

- Aprendizado não supervisionado: Nesse tipo não é fornecida nenhuma informação adicional sobre os dados, ou seja, o algoritmo trabalha somente com as características dos dados procurando uma estrutura topológica intrínseca aos exemplos (ZHU; GOLDBERG, 2009). O algoritmo de AM aprende a representar (ou agrupar) as entradas submetidas conforme uma estrutura natural presente nos dados.

- Aprendizado por reforço: O modelo deve aprender uma série de reforços, que podem ser recompensas ou punições, não sendo necessário treinamento em lotes estáticos. Nesse paradigma, o modelo aprende a executar ações com base nas informações de entrada do ambiente e nas recompensas ou punições recebidas em reposta a essas ações (GARCIA; FERNÁNDEZ, 2015).

- Aprendizado semissupervisionado: Uma limitação natural do aprendizado supervisionado é a necessidade de um grande conjunto de instâncias rotuladas para alcançar um bom desempenho. No aprendizado semissupervisionado os dados de treinamento possuem poucos exemplos rotulados e um grande número de exemplos não rotulados. O objetivo do modelo é fazer uso efetivo dos dados disponíveis, não apenas dos rotulados. O modelo aprende tanto com os dados rotulados quanto com a estrutura topológica dos exemplos não rotulados (ZHU; GOLDBERG, 2009; SONG et al., 2022). O aprendizado semissupervisionado pode ser dividido principalmente em duas categorias:

  1. Dado um conjunto de treino $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, um algoritmo de aprendizado indutivo semissupervisionado aprende uma função $f : \mathcal{X} \to \mathcal{Y}$ tal que f seja um bom classificador para dados futuros.

  2. Dados dois conjuntos de treino $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, $\{\mathbf{x_j}\}_{j=l+1}^{l+u}$, um algoritmo de aprendizado transdutivo aprende uma função $f : \mathcal{X}^{l+u} \to \mathcal{Y}^{l+u}$ tal que f seja um bom classificador para os dados no conjunto $\{\mathbf{x_j}\}_{j=l+1}^{l+u}$

2.2 Aprendizado de Máquina em Grafos

Grafos são amplamente utilizados como a representação de uma estrutura de dados conectada, podendo ser encontrado no domínio de sistemas sociais, ecossistemas, redes biológicas e sistemas de informação. De forma geral, modelos de aprendizado em grafo procuram mapear as características de um grafo para vetores de mesma característica em um espaço latente. Grande parte dos algoritmos são baseados em técnicas de aprendizado profundo. Assim, a representação de um grafo pode ser usada em tarefas como classificação, predição de link ou diretamente como um processo de *embedding* (XIA et al., 2021).

Nesse capítulo serão apresentados conceitos fundamentais de aprendizado em grafos e paradigmas de aprendizado semissupervisionado e não supervisionado em grafos.

2.2.1 Conceitos Fundamentais

Um grafo pode ser representado por $G = (V, E, \mathcal{X}, \mathcal{Y})$, onde $V = \{v_i\}_{i=1}^{n}$ é o conjunto com n vértices. $E = \{e_{i,j} = (v_i, v_j)\}_{i=1}^{n}$ é o conjunto de arestas ligando o vértice v_i ao vértice v_j. $\mathcal{X} = \{x_i \in \mathbb{R}^m\}_{i=1}^{n}$ é o conjunto de m características para cada vértice v_i. Para tarefas de classificação, $\mathcal{Y} = \{y_i = 0, 1\}_{i=1}^{n}$ é o conjunto das classes para cada nó v_i (WU et al., 2021b).

Os vértices do conjunto E podem ser representados por uma matriz de adjacência A onde $A_{i,j} = 1$ se $(v_i, v_j) \in E$ e 0 caso contrário. Para fins de simplificação, os conjuntos $\mathcal{X}$ e $\mathcal{Y}$ serão representados por matrizes $X_{n\times m}$ e $Y_{n\times 1}$. Uma comunidade em um grafo se refere a um subconjunto $V' \in V$ tal que os vértices em V' são mais densamente conectados do que vértices que não pertencem a V'.

O conceito de *Representation Learning* se refere a encontrar uma boa representação das características dos dados, ou seja, é sobre aprender características dos dados que facilitem a extração de informação para construir classificadores ou outros modelos de predição (WU et al., 2022). No contexto de grafos, *Network Representation Learning* (NRL) busca aprender uma representação latente dos vértices em um grafo, enquanto preserva a estrutura topológica do conteúdo de vértices e outras informações intrínsecas. Assim que representações vetoriais são aprendidas, a análise de grafos pode ser feita efetivamente com algoritmos convencionais de AM (WU et al., 2022).

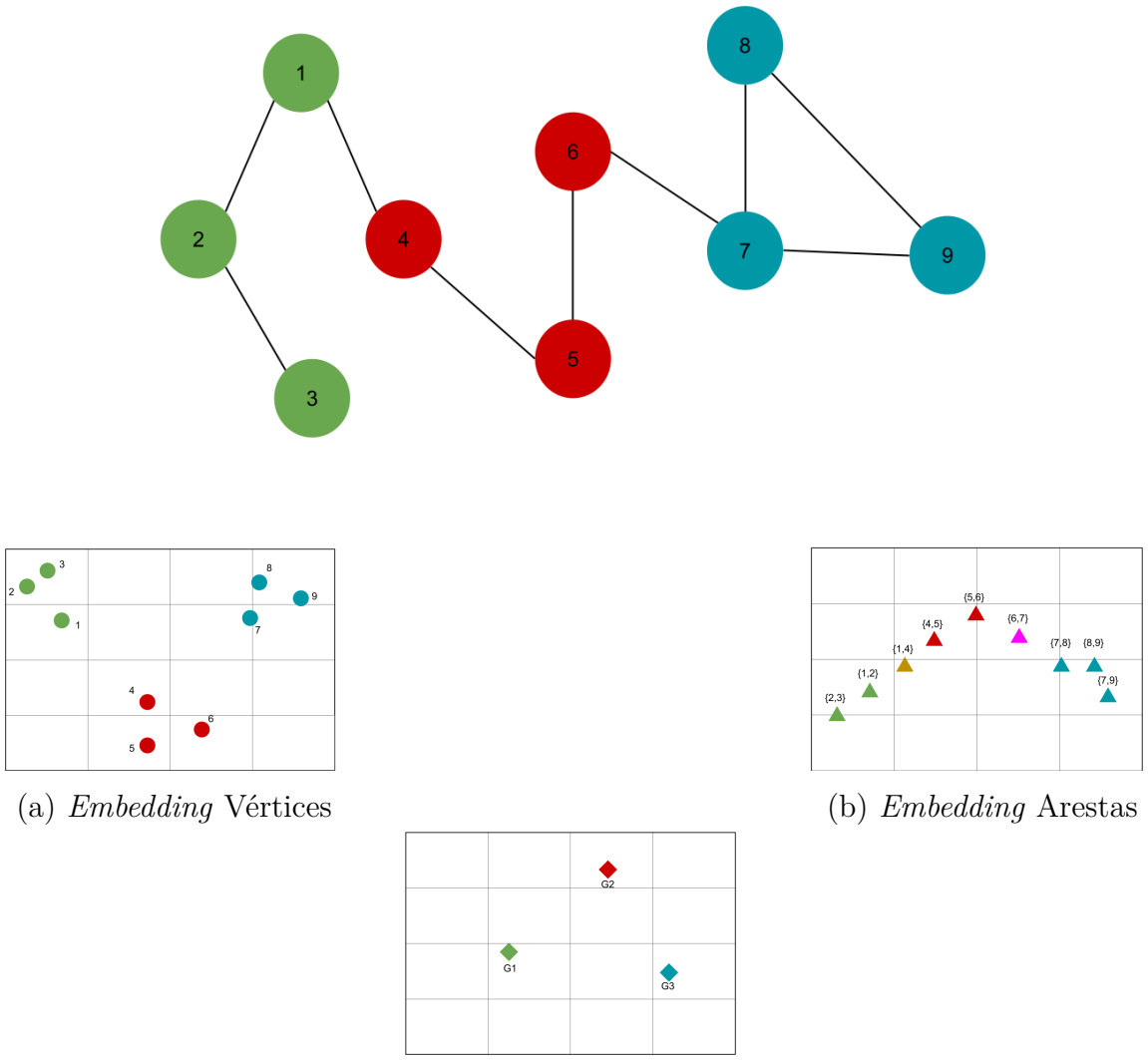
2.2.2 Aprendizado Não Supervisionado em Grafos

As tarefas de aprendizado não supervisionado se referem à identificação de informações relevantes nos dados sem a presença de um elemento externo para guiar o aprendizado (FACELI et al., 2011). O aprendizado consiste na identificação de propriedades intrínsecas

aos dados, de maneira a construir representações que possam servir a diversos propósitos (SOUTO et al., 2003).

No contexto de grafos, uma aplicação direta de aprendizado não supervisionado é o *embedding*. O *embedding* procura representar um grafo com vetores em um espaço vetorial de dimensão menor, preservando a estrutura do grafo. Existem diversas características de grafos que podem ser representadas como dados vetoriais, tais quais arestas, vértices, subgrafos e grafos completos (CAI; ZHENG; CHANG, 2018). A Figura 2 mostra o resultado do *embedding* de um grafo.

Figura 2 – Representação do *embedding* de um grafo



(a) *Embedding* Vértices

(b) *Embedding* Arestas

(c) *Embedding* Subgrafos

Fonte: Elaborado pelo autor

Além do *embedding*, a detecção de comunidades é outra aplicação de aprendizado não supervisionado em grafos. Detecção de comunidades se resume em encontrar subconjuntos de vértices densamente conexos entre si, e pouco conectados a vértices externos ao subconjunto.

2.2.3 Aprendizado semissupervisionado em Grafos

O aprendizado semissupervisionado (*self supervised learning* - SSL) utiliza os dados não rotulados para melhorar a hipótese do classificador. Segundo Engelen e Hoos (2020), o SSL, em um nível mais geral, pode ser dividido em indutivo e transdutivo.

No SSL indutivo o objetivo é induzir um classificador que consiga rotular novos exemplos fora do conjunto de dados não rotulados disponíveis para o treinamento. Ou seja, nesse caso, é gerado um modelo (generalização), com base nos dados de treinamento que, posteriormente, seja capaz de rotular dados desconhecidos (nunca vistos pelo modelo).

No SSL transdutivo, o algoritmo recebe simultaneamente os dados rotulados e não rotulados. Nesse caso, o algoritmo já aprende a classificar os dados sem rótulos, e não são apresentados novos dados. Não há a necessidade de generalizar o modelo para novos exemplos, pois os dados a serem rotulados são conhecidos previamente, de modo que o treinamento e a classificação ocorrem simultaneamente (GUTIÉRREZ, 2010).

Um exemplo das aplicações de SSL em grafos de maneira transdutiva é a utilização na rotulação de vértices previamente vistos na fase de treinamento. Já a aplicação transdutiva se refere à exemplos nunca antes vistos, como no caso de uma classificação de grafos.

Ou seja, enquanto que o SSL trandutivo conhece previamente todos os exemplos dos quais o rótulo precisa ser predito, o SSL indutivo não os conhece. Com base nessa premissa, dependendo das características do problema, deve-se selecionar a abordagem mais adequada, indutiva ou transdutiva.

Em grafos, a abordagem de aprendizado semissupervisionada mais comum é a transdutiva, em que o modelo recebe de maneira simultânea exemplos de ambos os tipos de dados, rotulados e não rotulados. Essa técnica é comumente utilizada na classificação de vértices, isto é, o objetivo é classificar vértices considerando uma determinada categoria. Por exemplo, suponha um grafo de documentos científicos, nos quais os vértices representam os documentos e as arestas representam a similaridade entre documentos (considerando a frequência e o número de palavras em comum entre os documentos). Suponha que alguns dos documentos foram classificados em diferentes áreas do saber, tais como física, química, matemática, história e geografia. O objetivo aqui é utilizar um algoritmo de aprendizado semissupervisionado para classificar os documentos (ou seja, os vértices) que não foram previamente rotulados em alguma das áreas do saber.

A Propagação de Rótulos ou *Label Propagation* (LP) é um algoritmo popular, simples e econômico em tempo de processamento, comumente usado na tarefa de aprendizado semissupervisionado transdutivo em grafos. Considerando a classificação de vértices, inicialmente vértices rotulados propagam seus rótulos para vértices vizinhos. A cada iteração, cada vértice não rotulado é atualizado com o rótulo mais frequente em sua vizinhança. Ao final, cada vértice é atribuído ao rótulo ao qual a maioria de seus vizinhos pertence. Intuitivamente, grupos de vértices densamente conectados convergirão para rótulos seme-

lhantes. Finalmente, os vértices atribuídos ao mesmo rótulo serão considerados em um único grupo ou comunidade e serão classificados com o mesmo rótulo. A Figura 3 ilustram a classificação dos vértices após a execução do algoritmo LP.

(a) Grafo parcialmente rotulado com duas classes

(b) Grafo totalmente rotulado com o algoritmo LP

Figura 3 – Propagação de Rótulos

2.3 Redes Neurais

Uma Rede Neural Artificial (RNA), ou simplesmente rede neural, é um modelo computacional inspirado no funcionamento do sistema nervoso humano (HAYKIN, 2009; FACELI et al., 2011). Nessa seção serão apresentados os modelos de redes neurais *perceptron*, *multilayer perceptron* e *AutoEncoder*.

2.3.1 Perceptron

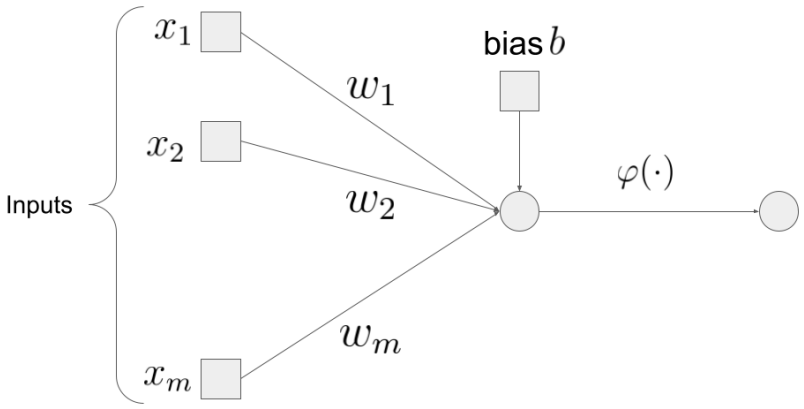
O modelo mais simples de rede neural, chamado *perceptron* (MCCULLOCH; PITTS, 1943), é formada por um algoritmo de aprendizado supervisionado utilizado para classificação binária. Essa unidade é conhecida como neurônio artificial e aplica transformações lineares e não lineares nos dados.

Seja $x = (x_1, x_2, \cdots, x_m), w = (w_1, w_2, \cdots, w_m) \in \mathbb{R}^m$ um vetor de entrada e um vetor de pesos, $b \in \mathbb{R}$ os pesos sinápticos da rede, $\varphi(\cdot)$ uma função de ativação não linear e y a saída da rede. Um perceptron pode ser definido como

$$y = \varphi\left(\sum_{i=1}^{m} x_i w_i + b\right),$$

a Figura 4 ilustra o funcionamento do perceptron.

Figura 4 – Estrutura do funcionamento de um perceptron



Fonte: Adaptado de Haykin (2009)

Duas das principais funções de ativação utilizadas são: a função limiar (*threshold*)

$$\varphi(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \tag{1}$$

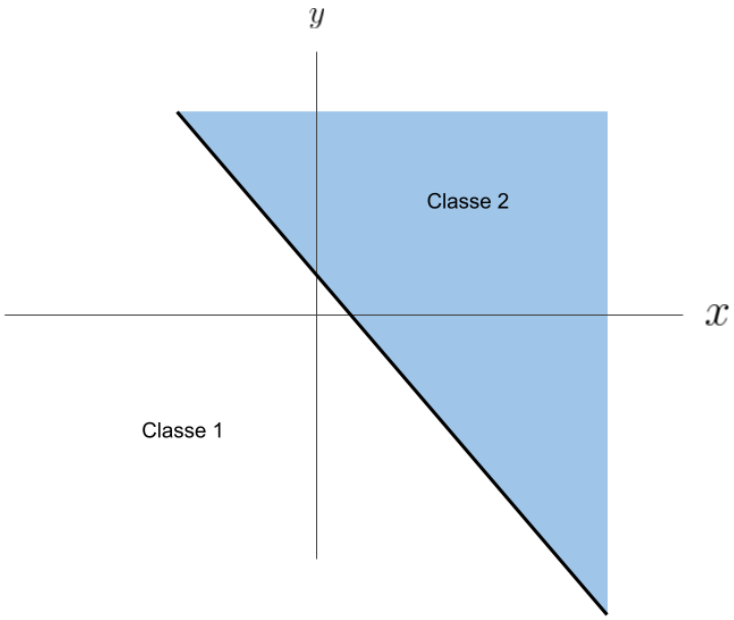
ou a função sigmoide

$$\varphi(x) = \frac{1}{1 + \exp(-ax)} \tag{2}$$

O modelo do perceptron consegue tratar somente problemas de classificação binária linearmente separáveis, ou seja, problemas cujos dados podem ser divididos por um hiperplano. Apesar dessa limitação, o perceptron foi a base para a criação de modelos de processamento mais complexos, capazes de lidar com uma variedade maior de problemas (HAYKIN, 2009).

A Figura 5 mostra um exemplo de fronteira de decisão para um problema de classificação binária linearmente separável.

Figura 5 – Fronteira de Decisão de um Problema Linearmente Separável



Fonte: Adaptado de Haykin (2009)

Ao classificar um exemplo incorretamente, o modelo faz a adaptação dos pesos utilizando o seguinte algoritmo (HAYKIN, 2009):

1. Se o n-ésimo vetor de entrada $\mathbf{x}(n)$ é corretamente classificado pelo vetor de pesos $\mathbf{w}(n)$, então nenhuma alteração é feita no vetor $\mathbf{w}(n)$. Ou seja, supondo C_1 como a classe que representa $y = 1$ e C_2 a classe que representa $y = 0$, temos

$$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ pois } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ para } \mathbf{x}(n) \text{ pertencente a classe } C_1$$

$$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ pois } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ para } \mathbf{x}(n) \text{ pertencente a classe } C_2,$$

2. Caso contrário, o vetor de pesos do perceptron é atualizado de acordo com a regra

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \text{ se } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ e } \mathbf{x}(n) \text{ pertence a classe } C_2$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \text{ se } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ e } \mathbf{x}(n) \text{ pertence a classe } C_1,$$

onde $\eta(n)$ representa uma taxa de aprendizado.

2.3.2 Redes Perceptron Multicamadas

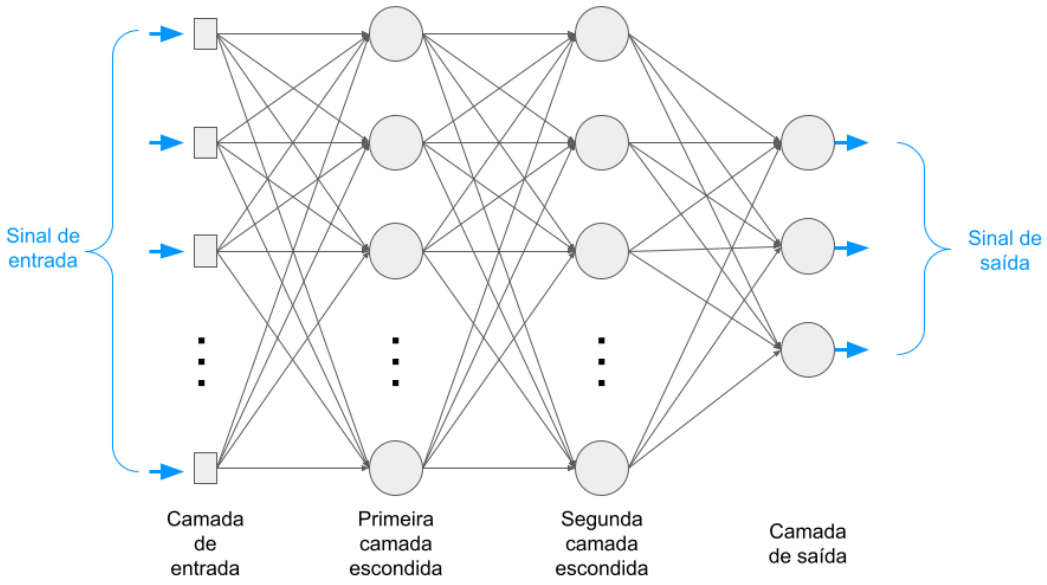
Um modelo de rede perceptron multicamada (*Multilayer Perceptron* - MLP) se diferencia de uma rede perceptron simples, pois pode conter camadas escondidas. O termo "escondido" se refere ao fato de que existem neurônios na rede que não se conectam diretamente nas entradas e nem são neurônios que representam a saída da rede.

As principais características de uma MLP são:

- ❏ O modelo presente em cada neurônio da rede inclui uma função de ativação não linear diferenciável;
- ❏ Contém uma ou mais camadas escondidas;
- ❏ Possui um alto grau de conectividade, denominado pelos pesos sinápticos; e
- ❏ É capaz de lidar com dados não linearmente separáveis.

A Figura 6 ilustra o funcionamento de uma MLP.

Figura 6 – Arquitetura de uma Rede Neural Multicamada



Fonte: Adaptado de Haykin (2009)

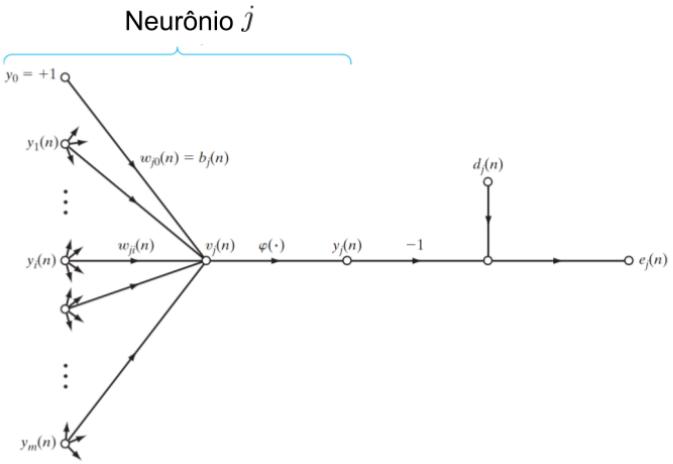
O objetivo da MLP é aproximar uma função f^*. Por exemplo, para um classificador, $y = f^*(\mathbf{x})$ mapeia uma entrada x para uma categoria y. Uma rede *feedforward* define uma função $\mathbf{y} = f(\mathbf{x}; \theta)$ e aprende os valores dos parâmetros θ que resultam na melhor aproximação (GOODFELLOW; BENGIO; COURVILLE, 2016).

De modo geral, a MLP consiste em uma rede *feedforward* que aceita uma entrada $\mathbf{x}$ e produz uma saída $\hat{y}$. Durante a fase de treino, a propagação produz uma função de custo $j(\theta)$, onde θ são os pesos da rede. O algoritmo *back-propagation* (RUMELHART; HINTON; WILLIAMS, 1986), que será discutido adiante, permite que a função de custo seja retro propagada pela rede para computar o gradiente. Para atualizar os pesos, o algoritmo mais usado é o gradiente descendente (ROBBINS; MONRO, 1951), que utiliza o gradiente da função de custo para atualizar os pesos da rede.

2.3.3 Algoritmo *Back Propagation*

Vamos considerar um neurônio j de uma rede neural como na Figura 7

Figura 7 – Neurônio MLP



Fonte: Adaptado de Haykin (2009)

O valor $v_j(n)$ produzido pela matriz de pesos W e pelos valores de entrada no neurônio j no n-ésimo exemplo de treino pode ser dada pela equação

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n), \tag{3}$$

onde m é o número de entradas (desconsiderando o bias) aplicadas diretamente no neurônio j. O peso sináptico w_{j0} representa o *bias* b_j aplicado ao neurônio j. Assim, o sinal de saída $y_j(n)$ pode ser dada da seguinte forma:

$$y_j(n) = \varphi_j(v_j(n)), \tag{4}$$

onde φ_j é a função de ativação do neurônio j.

O algoritmo *back-propagation* aplica uma correção $\Delta w_{ji}(n)$ no peso sináptico $w_{ji}(n)$, que é proporcional a derivada parcial $\partial\xi(n)/\partial w_{ji}(n)$, representando um fator de sensibilidade, onde ξ é a função de erro dados os pesos. A derivada procura a direção de busca no espaço de pesos w_{ji}. A derivada parcial pode ser calculada de acordo com a regra da cadeia:

$$\frac{\partial\xi(n)}{\partial w_{ji}(n)} = \frac{\partial\xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{5}$$

Podemos expressar, através do algoritmo *least mean square* (LMS), a seguinte derivada

$$\frac{\partial\xi(n)}{\partial e_j(n)} = \frac{1}{2} \sum_{j\in C} e_j^2(n) \tag{6}$$

Onde C representa o conjunto de todos os neurônios na camada de saída. O erro $e_j(n)$ da saída do j-ésimo neurônio pode ser calculado pela equação $e_j(n) = d_j(n) - y_j(n)$, onde $d_j(n)$ é a saída desejada do neurônio j. Assim,

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{7}$$

Agora, derivando ambos os lados da Equação 4 temos:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'_j(v_j(n)) \tag{8}$$

Finalmente, derivando Equação 3:

$$\frac{\partial v_j(n)}{\partial w_{ji}} = y_j(n) \tag{9}$$

Usando os valores da Equação 6 até a Equação 9 na Equação 5 temos

$$\frac{\partial \xi(n)}{\partial w_{ji}} = -e_j(n)\varphi_j(v_j(n))y_i(n) \tag{10}$$

A correção $\Delta w_{ji}(n)$ aplicada ao peso $w_{ji}(n)$ é definida pela regra delta, ou

$$\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)} \tag{11}$$

Onde η é a taxa de aprendizado do algoritmo *back-propagation*. O uso do sinal negativo vem do gradiente descendente no espaço vetorial dos pesos.

Assim, podemos definir o valor $\Delta w_{ji}(n)$ como

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{12}$$

onde

$$\begin{aligned} \delta_j(n) &= \frac{\partial \xi(n)}{\partial v_j(n)} \\ &= \frac{\partial \xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\varphi'_j(v_j(n)) \end{aligned} \tag{13}$$

Temos então que o gradiente local $\delta_j(n)$ de um neurônio j é igual ao produto do erro $e_j(n)$ para aquele neurônio e a derivada da função de ativação $\varphi'_j(v_j(n))$.

Quando o neurônio j é um neurônio de saída, podemos calcular seu erro utilizando a equação $e_j(n) = d_j(n) - y_j(n)$. Com o valor do erro, utilizamos Equação 13 para computar o gradiente local.

Se j é um neurônio localizado na camada escondida, não é possível estipular um valor desejado para o neurônio. Assim, com o sinal de erro do neurônio de saída, devemos calcular recursivamente o erro até o neurônio j a partir daqueles neurônios diretamente conectados a j.

De acordo com a Equação 13, podemos definir o gradiente $\delta_j(n)$ como

$$\begin{aligned}\delta_j(n) &= -\frac{\partial \xi(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial \xi(n)}{\partial y_j(n)}\varphi'_j(v_j(n))\end{aligned} \tag{14}$$

Onde a segunda linha da Equação 14 pode ser justificada pela Equação 8. Para calcular a derivada parcial $\partial\xi(n)/\partial y_j(n)$, considere

$$\xi(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n) \tag{15}$$

Derivando a equação 15 em relação ao sinal $y_j(n)$ temos

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{16}$$

Utilizando a regra da cadeia para calcular a derivada parcial $\partial e_k(n)/\partial y_j(n)$ e reescrevendo a Equação 16 temos:

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)} \tag{17}$$

Assim, como $e_k(n) = d_k(n) - y_k(n) = d_k(n) = \varphi_k(v_k(n))$:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi'_k(v_k(n)) \tag{18}$$

Para um neurônio k temos que:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{19}$$

onde m é a dimensão do vetor de entrada. Devirando a Equação 19:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{20}$$

Aplicando as Equações 18, 20 e 17 temos a derivada parcial desejada

$$\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi'_k(v_k(n))w_{kj}(n) \tag{21}$$

$$= -\sum_k \delta_k(n) w_{kj}(n) \tag{22}$$

Utilizando as Equações 21 e 14 temos a formula do algoritmo *back-propagation*

$$\delta_j(n) = \varphi'_j(v_j(n))\sum_k \delta_k(n) w_{kj}(n), \tag{23}$$

onde o neurônio j é um neurônio escondido. Assim, é possível verificar que a cada iteração, a correção nos pesos sinápticos $\Delta w_{ij}(n)$ é definida pela regra delta:

$$\Delta w_{ij}(n) = \eta \times \delta_j(n) \times y_i(n) \tag{24}$$

O gradiente local $\delta_j(n)$ depende do neurônio j ser um neurônio de saída ou um neurônio da camada escondida.

1. Se o neurônio j é um neurônio de saída, $\delta_j(n)$ é o produto da derivada $\varphi'_j(v_j(n))$ e do erro $e_j(n)$. Ambas associadas ao neurônio j.

2. Se o neurônio j é um neurônio escondido, $\delta_j(n)$ é igual ao produto da derivada $\varphi'_j(v_j(n))$ e da soma ponderada dos valores de δ computados nos neurônios da próxima camada escondida ou da camada de saída.

2.3.4 AutoEncoders

Um *AutoEncoder* (AE) é uma rede neural MLP que é treinada para copiar sua entrada na saída. O objetivo é aprender uma forma não supervisionada de representação dos dados. Formalmente, o problema consiste em aprender funções $g_1 : \mathbb{R}^n \to \mathbb{R}^p$ (*encoder*) e $g_2 : \mathbb{R}^p \to \mathbb{R}^n$ (*decoder*) que satisfaça

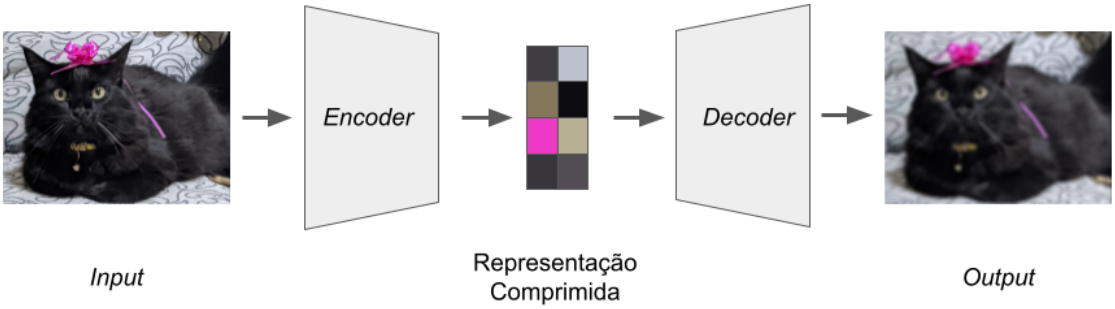
$$\arg\min_{g_1,g_2} E\left[\Delta(\mathbf{x}, g_2 \circ g_1(\mathbf{x}))\right], \tag{25}$$

onde E é o valor esperado da variável $\mathbf{x}$ e Δ é a função de reconstrução do erro, que mede a distância entre o *output* do *decoder* e o *input* (GOODFELLOW; BENGIO; COURVILLE, 2016; BANK; KOENIGSTEIN; GIRYES, 2021). Assim, definimos a saída $\tilde{\mathbf{x}}$ do *AutoEncoder* como

$$\tilde{\mathbf{x}} = f(\mathbf{x}) = g_2 \circ g_1(\mathbf{x}) \tag{26}$$

A Figura 8 mostra um modelo do *AutoEncoder*.

Figura 8 – AutoEncoder



Fonte: Adaptado de Bank, Koenigstein e Giryes (2021)

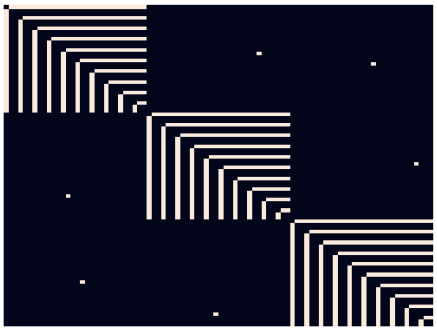
Na esquerda é possível verificar uma imagem que, ao passar pela rede, tem sua saída próxima à entrada. A sua versão comprimida é chamada de *bottleneck* e representa uma versão em menor dimensão da imagem. Para os propósitos deste trabalho, tanto a entrada quanto a saída serão de informações referentes aos nós de um grafo.

2.3.5 Redes Neurais em Grafos

O sucesso recente das redes neurais e de modelos de *deep learning* trouxeram um avanço significativo no reconhecimento de padrões e na mineração de dados. Enquanto esses algoritmos são efetivos em capturar padrões em dados euclidianos, existe um crescente número de aplicações modeladas por grafos que não se beneficiam dessas estruturas (WU et al., 2021).

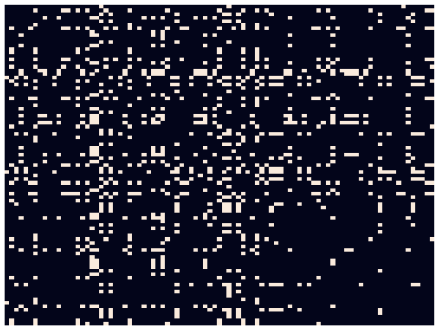
Um exemplo dessa limitação é o fato da matriz de adjacência de um grafo não ter representação única. As Figuras 9 e 10 mostram a representação de duas matrizes de adjacência que representam o mesmo grafo. Nesse exemplo podemos verificar que operações euclidianas elementares, como somas, produtos, cálculo de distâncias, convoluções, entre outros, são inviáveis pois são operações que respeitam a ordem dos elementos.

Figura 9 – Matriz de adjacência ordenada em comunidades



Fonte: Produzido pelo autor

Figura 10 – Matriz de adjacência ordenada de forma aleatória



Fonte: Produzido pelo autor

Para preencher essa lacuna, diversos métodos baseados em redes neurais foram propostos para trabalhar com grafos. Esses modelos podem ser divididos em quatro categorias, sendo elas: Redes Neurais Recorrentes em Grafos (*Recurrent Graph Neural Networks* – RecGNN), Redes Neurais Espaciais-Temporais em Grafos (*Spatial-temporal Graph Neural Networks* – STGNNs), *AutoEncoders* em Grafos (*Graph AutoEncoders* – GAE) e Redes Neurais Convolucionais em Grafos (*Graph Convolutional Networks* - GCN) (WU et al., 2021).

Recurrent Graph Neural Networks são os modelos pioneiros na utilização de redes neurais em grafos. O objetivo é aprender uma representação dos vértices com a

utilização de redes neurais recorrentes. Partindo da hipótese que um vértice em um grafo troca informações com seus vizinhos até que um estado de equilíbrio é alcançado. Os modelos de rede recorrente se dividem principalmente em modelos de aprendizado supervisionado e semissupervisionado. Alguns deles são descritos a seguir:

Spatial Temporal Graph Neural Networks são arquiteturas com o objetivo de encontrar padrões em grafos espaciais-temporais. Grafos espaciais-temporais são grafos onde os atributos dos nós mudam dinamicamente ao longo do tempo. A ideia principal de uma STGNN é considerar uma dependência espacial e temporal simultaneamente.

Graph AutoEncoders são modelos de aprendizado não supervisionado que fazem o *encode* dos vértices ou das arestas em um espaço euclidiano. Uma propriedade importante do GAE é a capacidade de entender a distribuição dos vértices em um grafo (BANK; KOENIGSTEIN; GIRYES, 2021), podendo ser utilizado para geração de grafos, vértices e arestas e também o *embedding* da rede.

Graph Convolutional Networks são modelos que generalizam o conceito de convolução de dados em grade para grafos. A ideia principal é gerar a representação de um nó v a partir do vetor de características x_v e dos vetores $x_u : u \in \mathcal{N}(v)$. Diferente das redes recorrentes, as redes convolucionais fazem uma pilha de camadas convolucionais. A ideia de convolução para grafos foi apresentada em Kipf e Welling (2016a).

Todas as quatro categorias de GNN sucedem da ideia de que as arestas de um grafo carregam informações importantes sobre os dados. Um conceito fundamental que idealiza essa hipótese é o conceito de Transferência de Mensagem (*Message Passing* – MP).

2.3.5.1 Transferência de Mensagem

O MP é o *framework* mais utilizado para construção de GNNs (GILMER et al., 2017).

Dado um grafo $G = (V, E, X)$, o *framework* faz o *encoding* de cada vértice $v \in V$ com um vetor de representação $\mathbf{h}_v$ e continua atualizando aquela representação iterativamente coletando representações de seus vizinhos e aplicando uma camada de rede neural para performar uma transformação não linear dessas representações.

O modelo consiste basicamente em dois passos (WU et al., 2022):

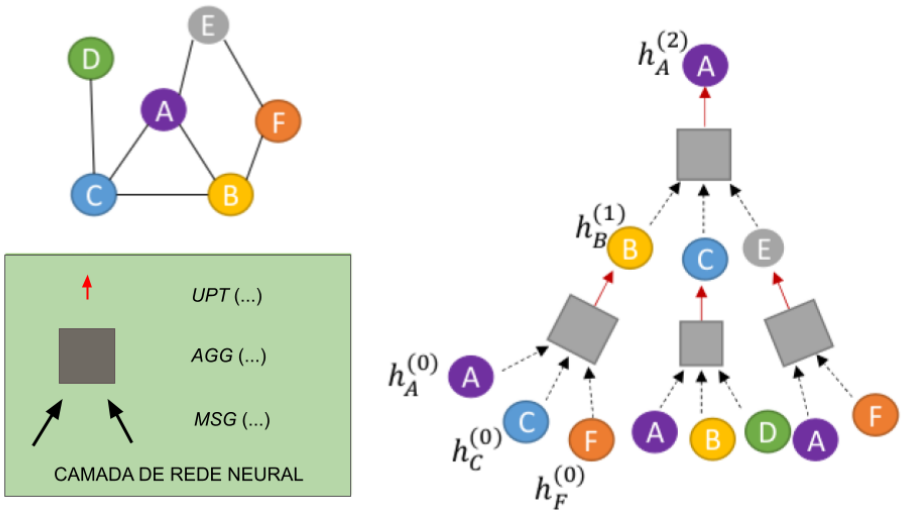
1. Inicializa a representação de um vértice a partir do vetor $\mathbf{h}_v^{(0)} = \mathbf{x}_v, \forall v \in V$

2. Atualiza cada vetor de representação baseado na mensagem transmitida pela estrutura do grafo. Na l-ésima camada, $l = 1, 2, \cdots, L$, performa os seguintes passos:

$$\begin{aligned} \text{Message:} \quad & \mathbf{m}_{uv}^{(l)} = \text{MSG}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_u^{(l-1)}\right), \forall (u,v) \in E \\ \text{Aggregation:} \quad & \mathbf{a}_v^{(l)} = \text{AGG}\left(\{\mathbf{m}_{uv}^{(l)} | u \in N_v\}\right), \forall v \in V \\ \text{Update:} \quad & \mathbf{h}_v^{(l)} = \text{UPT}(\mathbf{h}_v^{(l-1)}, \mathbf{a}_v^{(l)}), \forall v \in V \end{aligned} \tag{27}$$

As operações MSG, AGG e UPT são tipicamente implementadas por uma rede neural $feedforward$, sendo que a operação AGG deve ser uma operação comutativa (WU et al., 2022). A Figura 11 exemplifica o funcionamento do MP para um vértice específico do grafo.

Figura 11 – Rede Neural em Grafo



Fonte: (WU et al., 2022)

Considerando que as operações descritas pela Equação 27 podem ser feitas por redes neurais convencionais, o algoritmo de retropropagação pode ser utilizado nesse caso. Além disso, as funções de erros mais utilizadas nas redes neurais, tal qual a MSELoss$(\cdot,\cdot)$, podem ser utilizadas para ajustar os parâmetros da rede.

O modelo de transferência de mensagem é a base do funcionamento de diversos *frameworks* de GNN.

2.3.5.2 Frameworks

Diversos *frameworks* são propostos na literatura, os principais para esse trabalho são: Rede Neural Convolucional em Grafos (GCN) (KIPF; WELLING, 2016a), *Attributed Network AutoEncoder* (ANAE) (CEN et al., 2020), *Relational Graph Convolutional Networks* (RGCN) (SCHLICHTKRULL et al., 2018).

A topologia de *Graph Convolutional Network* (GCN) foi proposta em Kipf e Welling (2016a) e é uma das mais populares e efetivas *baselines* de redes neurais em grafos (SONG et al., 2021).

Uma rede neural convolucional em grafos pode ser construída considerando a matriz laplaciana e a operação de convolução. Suponha $L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ a matriz laplaciana normalizada de um grafo G onde D é a matriz diagonal dos vértices, $D_{i,i} = \sum_j A_{i,j}$. Essa matriz pode ser fatorada em $L = U\Lambda U^T$, onde $U = [u_0, u_1, \cdots u_{n-1}] \in \mathbb{R}^{n\times n}$ é a matriz dos autovetores ordenada pelos autovalores da matriz diagonal de autovalores Λ. Dado $X \in \mathbb{R}^{n\times m}$ uma matriz de características dos vértices do grafo, uma transformada de Fourier em grafos pode ser definida como $\mathcal{F}(x) = U^T x$, sendo sua inversa $\mathcal{F}^-1(x) = Ux$.

A transformada de Fourier em grafos projeta o grafo de entrada em um espaço vetorial ortonormal onde a base é formada pelos autovetores da matriz laplaciana normalizada. Agora, o sinal X de entrada de um grafo pode ser convolucionada em relação a um filtro $g \in \mathcal{R}^n$ através da equação

$$x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) = U(U^T x \odot U^T g), \tag{28}$$

onde $\odot$ representa o produto direto de dois vetores. A diferença principal entre as redes convolucionais encontra-se na escolha do filtro g.

A GCN (KIPF; WELLING, 2016a) utiliza a convolução da seguinte forma:

$$x *_G g_\theta = \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x \tag{29}$$

Para habilitar múltiplas entradas de *inputs* e *outputs*, a GCN modifica a Equação 29 da seguinte forma:

$$H = X *_G G_\theta = f(\overline{A}X\theta) \tag{30}$$

onde $\overline{A} = I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ e $f(\cdot)$ é uma função de ativação. Utilizando esse valor de $\overline{A}$ causa instabilidade a GCN. Para lidar com esse problema, é feito uma troca do valor de $\overline{A}$ para $\overline{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ com $\tilde{A} = A + I_n$ e $\tilde{D} = \sum_j \tilde{A}_{ij}$

Dessa forma, a GCN considera um modelo de rede neural $f(X, A)$ com a seguinte regra de propagação:

$$H^{(l+1)} = \varphi\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{31}$$

onde $\tilde{A} = A + I_N$ é a matriz de adjacência de um grafo não direcionado G com arestas de laço e I_N é a matriz identidade. $\tilde{D}_{i,i} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ é uma matriz de pesos treináveis de uma camada l, $\varphi(\cdot)$ se refere a uma função de ativação, como RELu$(\cdot)$ por exemplo. $H^{(l)} \in \mathbb{R}^{N\times D}$ é a matriz de ativação da l-ésima camada e $H^0 = X$, onde X é a matriz de características dos vértices no grafo.

Para lidar com grafos que representam diferentes relações, Schlichtkrull et al. (2018) apresentaram o modelo de *Ralational Graph Convolutional Network* (RGCN). Ao contrário dos GCNs padrão, que aplicam a mesma transformação a todas as arestas, os R-GCNs aplicam diferentes transformações com base no tipo de relação entre os nós. Isso significa

que a mensagem passada de um vértice para outro é modulada pelo tipo específico de aresta (ou relação) que os conecta. A Equação 32 define a agregação em cada camada de uma RGCN

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}\right) \tag{32}$$

onde $\mathcal{N}_i^r$ denota o conjunto de índices de vizinhos do vértice i sob a relação $r \in \mathcal{R}$. $c_{i,r}$ é uma constante de normalização específica do problema que pode ser aprendida ou escolhida antecipadamente (como $c_{i,r} = |\mathcal{N}_i^r|$).

Esse modelo foi desenvolvido para lidar com grafos relacionais. Nesse trabalho, esse modelo será utilizado considerando os diferentes grafos gerados pelo modelo de reescrita.

O modelo do *Graph AutoEncoder* (GAE) (KIPF; WELLING, 2016b) é um modelo que aprende a representação dos dados. A arquitetura busca aprender representações compactas da estrutura de um grafo. Portanto, o GAE não aprende representações dos vértices individuais, apenas da estrutura do grafo.

Para lidar com essa lacuna, Cen et al. (2020) propôs a arquitetura *Attributed Network AutoEncoder* (ANAE), que consiste em um *framework* criado para lidar com problemas sobre grafos com características nos vértices. O objetivo da ANAE é aprender o conteúdo de um nó para um grafo com atributos, levando em consideração também a estrutura do grafo.

Similar ao modelo ANAE, Ng et al. (2019) propõe o uso da regra de propagação da Equação 31 na saída do *encoder* de um *AutoEncoder*, aplicando o algoritmo MP na representação latente. Nesse trabalho, consideraremos uma arquitetura *Graph AutoEncoder* (GAE) como o modelo apresentado em Ng et al. (2019).

Considerando X como a matriz de características formada pelos elementos $\mathbf{x}$, a Equação 26 pode ser escrita como

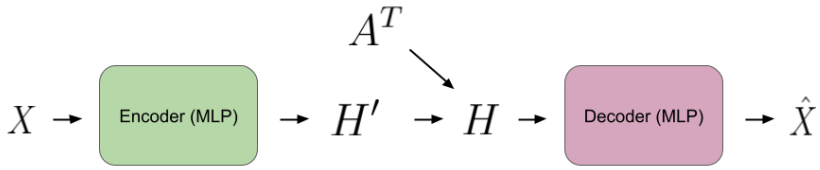
$$\tilde{X} = f(X, A) = g_2\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}g_1(X)\right) \tag{33}$$

Assim, o algoritmo consiste em aprender funções g_1 e g_2 que minimizem a expectativa do erro Δ, conforme a Equação 34:

$$\arg\min_{g_1,g_2} E\left[\Delta\left(X, g_2\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}g_1(X)\right)\right)\right] \tag{34}$$

A Figura 12 apresenta o esquema de funcionamento do *Graph AutoEncoder*

Figura 12 – Esquematização de um GAE



Fonte: Adaptado de Ng et al. (2019)

Dessa forma, a rede é capaz de aprender a representação dos nós considerando a estrutura topológica do grafo, ou seja, dos vizinhos diretos de um nó alvo.

2.3.5.3 Aprendizado semissupervisionado com Redes Neurais em Grafos

A propagação no grafo feita pela arquitetura GCN pode ser usada para um problema de classificação semissupervisionado, pois é possível diminuir o valor da saída da última camada para representar a classe do objeto e, além disso, são utilizadas todas as características dos vértices. Os resultados de Kipf e Welling (2016a) sugerem que a classificação é robusta quando a matriz de adjacência A contém informações que não estão presentes na matriz de características X.

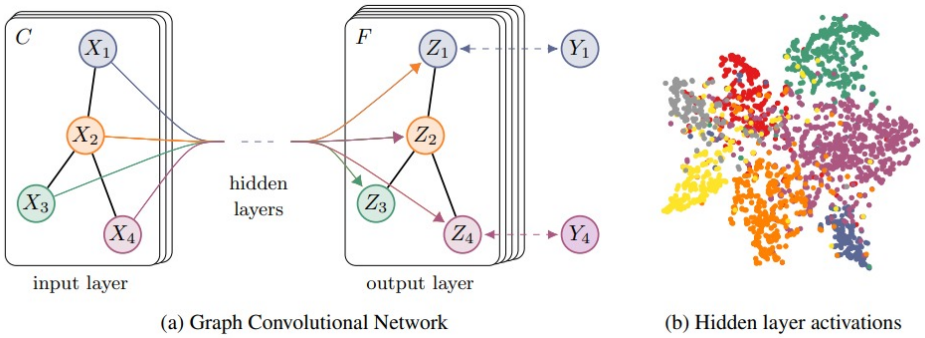
O conjunto de dados CORA (SEN et al., 2008) é um conjunto amplamente utilizado para validação de modelos de aprendizado de máquina (KIPF; WELLING, 2016a; WU et al., 2021b; YOO et al., 2021). Considere uma aplicação da GCN como na Equação 35:

$$Z = f(X, A) = \text{softmax}\left(\tilde{A}\ \text{RELu}\left(\tilde{A}XW^{(0)}\right)\ W^{(1)}\right) \tag{35}$$

A Figura 13 mostra uma visualização da representação vetorial do conjunto de dados CORA, através do algoritmo t-SNE (MAATEN; HINTON, 2008) no modelo GCN.

Na esquerda, a Figura 13 mostra uma representação esquemática de uma GCN com múltiplas camadas para aprendizagem semissupervisionada, com C neurônios de entrada e F neurônios na camada de saída. A estrutura do grafo (arestas representadas por linhas pretas) é compartilhada entre as camadas e seus rótulos são denotados por Y_i. Na direita, a figura mostra uma visualização pelo algoritmo t-SNE (MAATEN; HINTON, 2008) de uma GCN treinada com duas camadas de ativação no conjunto de dados CORA

Figura 13 – Classificação de nós



Fonte: Kipf e Welling (2016a)

(SEN et al., 2008), usando 5% dos dados rotulados. As cores demonstram as classes dos documentos.

A arquitetura da GCN usa um modelo de propagação de características de um determinado grafo. Experimentos em um número considerável de conjuntos de dados mostraram que a proposta do modelo da GCN é capaz de utilizar tanto as características dos nós, quanto a estrutura do grafo. Dadas essas condições, o modelo performou significativamente melhor que outros métodos propostos (KIPF; WELLING, 2016a).

2.3.5.4 Reescrita em Grafos

Considerando que cada camada de uma GNN baseada em MP consegue capturar informações de uma vizinhança, seriam necessárias diversas camadas para capturar informações à longo alcance. Mesmo que utilizadas, essas camadas teriam que lidar com uma quantidade muito grande de informação que seria sumarizada em um vetor de tamanho unitário, acarretando em perda de informações (KIPF; WELLING, 2016a; WU et al., 2021b). Esse problema é conhecido na literatura como *over-squashing*.

Uma possível solução para agregar informações de vizinhanças distintas seria a criação de arestas artificiais, essa solução é dada na literatura como reescrita em grafos. Reescrita são técnicas utilizadas em GNNs para modificar a estrutura do grafo original de forma a melhorar a propagação de informação, adicionando ou reestruturando arestas para facilitar a fase de agregação dessas redes neurais.

Assim, a ideia principal do uso de reescrita em GNNs é decompor o grafo original em grafos que serão utilizados para computação da representação latente. Assim, estratégias de reescrita consistem em aplicar um operador R em $G = (V, E, X, Y)$ produzindo um novo grafo $R(G) = (V, R(E), X, Y)$ com os mesmos vértices porém com conectividade diferente. Agora, um modelo de MPNN pode agregar informações de uma vizinhança tanto em G quanto em $R(G)$. Considerando uma MPNN sobre uma perspectiva de reescrita, definimos a atualização em cada camada t da forma:

$$x_v^{(t)} = \mathrm{UP}^{(t)}(x_v^{(t-1)}, \mathrm{AG}_G^{(t)}\left(\{x_u^{(t-1)} : (u,v) \in E\}\right), \mathrm{AG}_{R(G)}^{(t)}(\{x_u^{(t-1)} : (u,v) \in R(E)\})\Big). \quad (36)$$

onde AG e UP são funções de agregação e atualização respectivamente. Diversos modelos da literatura consideram a função $\mathrm{AG}_G = 0$, atualizando as representações apenas informações do grafo após a operação de reescrita. Os métodos da literatura utilizam duas abordagens para estratégias de reescrita, os modelos espectrais e os modelos espaciais.

Modelos de reescrita espaciais consideram operações localizadas em cada vértice, agregando informações diretamente na MPNN. Brüel-Gabrielsson et al. (2023) (BRüEL-GABRIELSSON; YUROCHKIN; SOLOMON, 2023) utilizam ligações diretas de vizinhos de uma r-vizinhança. Abboud et al. (2023) (ABBOUD; DIMITROV; CEYLAN, 2023) utilizam caminhos mínimos para propagar a informação em uma MPNN. As arquiteturas *Graph Transformers* também são amplamente utilizadas para soluções baseadas em reescrita (CAI et al., 2023; KREUZER et al., 2021; YING et al., 2021). Esses modelos reduzem o *over-squashing* criando novas arestas no grafo original, diminuindo o diâmetro e aumentando a propagação de informação ao longo da rede.

Por outro lado, métodos espectrais consideram operações globais no grafo. (BARBERO et al., 2023) aponta que duas abordagens são exploradas para métodos espectrais, sendo elas a partir do *spectral gap* (ARNAIZ-RODRIGUEZ et al., 2022) e de *effective resistance* (BLACK et al., 2023). Esses métodos adicionam apenas algumas arestas melhorando alguma métrica de conectividade.

Grande parte dos algoritmos de reescrita (principalmente espectrais) adicionam arestas de forma instantânea, ou seja, o resultado consiste em somente um grafo com mais arestas. O algoritmo LASER (*Locality-Aware SEquential Rewiring*) (BARBERO et al., 2023) utiliza uma abordagem sequencial para a criação do grafo final, a partir de uma mescla de abordagens espectrais e espaciais. Dessa forma a topologia do grafo é alterada a cada passo do algoritmo, gerando grafos distintos.

A abordagem LASER consiste em, dado um nó v em um estágio l da aplicação do algoritmo, considerar a ligação direta de vértices à uma distancia $l+1$ de v. Um parâmetro ρ é utilizado para adicionar arestas com menor *score* de conectividade μ. A equação 37 exemplifica a criação sequencial dos grafos.

$$G = G_0 \xrightarrow{R_1} G_1 \xrightarrow{R_2} G_2 \cdots \xrightarrow{R_L} G_L. \quad (37)$$

Dessa forma, com a criação sequencial de arestas permite uma evolução mais suave a partir do grafo G_0, sendo mais robusta ao *over-squashing*, preservando informações da topologia do grafo (BARBERO et al., 2023).

As informações que originalmente estavam presentes apenas no grafo G_0 agora estão divididas entre os novos grafos criados sequencialmente. Para lidar com esses novos grafos,

um modelo de GNN é proposto para levar em consideração todos esses grafos gerados a cada camada.

2.4 Classificação Binária

2.4.1 Classificação em Única Classe

A técnica de classificação de uma única classe (*One Class Classification* – OCC) procura identificar a pertinência de um objeto a uma classe específica, previamente observada durante o treinamento. A classificação em única classe é utilizada em problemas de classificação binários, e a classe observada durante o treinamento é chamada de classe positiva. Ao contrário do aprendizado multirrótulo, os parâmetros de aprendizagem e as fronteiras de decisão tornam-se mais complexos em problemas de OCC (PERERA; OZA; PATEL, 2021). Exemplos de aplicação de OCC são previsão de incêndios, predição de crimes e também classificação de textos (SHIN; KIM, 2019; MANEVITZ; YOUSEF, 2007).

Depois de gerar um classificador f, um valor $f(\mathbf{x_i})$ é dado para uma nova observação $\mathbf{x_i}$. Se esse valor ultrapassar um limiar, então ele será classificado como pertencente à classe positiva. Formalmente:

$$\text{classe} = \begin{cases} f(\mathbf{x_i}) & \geq \text{limiar} \rightarrow \text{positiva} \\ f(\mathbf{x_i}) & < \text{limiar} \rightarrow \text{negativa} \end{cases} \tag{38}$$

Dado o contexto, existem diversos trabalhos que avaliam métricas de similaridade para esses limiares, entre eles estão o uso do algoritmo *k-Nearest Neighbor Density-Based* (k-NN) (MARUTHI, 2020), *k-means* (WU; IANAKIEV; GOVINDARAJU, 2002), Redes neurais (*dense autoencoder*) (MANEVITZ; YOUSEF, 2007).

No limiar baseado no algoritmo k-NN, um objeto é classificado de acordo com o voto da maioria de seus vizinhos. Para determinar os vizinhos, é usada a distância euclidiana como medida de similaridade. Para $k = 1$ em um espaço N-dimensional, a Equação 39 descreve o valor da distância d entre os exemplos x e y:

$$f(\mathbf{x_i}) = \sum_{\mathbf{y} \in N(\mathbf{x_i}, k)} \sqrt{\sum_{k=1}^{N} (x_{ik} - y_k)^2} / |\mathcal{N}(\mathbf{x_i}, k)| \tag{39}$$

Onde $\mathcal{N}(\mathbf{x_i}, \|)$ representa os k vizinhos rotulados como classe positiva de $\mathbf{x_i}$. O algoritmo k-NN sofre com a maldição da dimensionalidade (FACELI et al., 2011), ou seja, pode não performar bem quando uma observação possui muitas características. Para lidar com esse problema, é necessário utilizar técnicas de seleção de atributos (MARUTHI, 2020).

Existem outras métricas de similaridade para o algoritmo k-NN, como por exemplo a similaridade por cosseno

$$f(\mathbf{x_i}) = \sum_{\mathbf{y} \in N(\mathbf{x_i}, k)} \cos(\mathbf{x_i}, \mathbf{y}) / |\mathcal{N}(\mathbf{x_i}, k)| \tag{40}$$

No algoritmo *k-Means*, os exemplos da classe positiva são separados em k grupos. Sejam $D^+ = G_1 \cup G_2 \cup \cdots \cup G_k$ esses grupos, onde $G_i \subset D^+$ são representados pelos centroides $\mathbf{g_i}$. Cada centróide $\mathbf{g_i}$ é a média dos exemplos de um conjunto G_i. Assim que todos os centroides são computados, cada novo exemplo é avaliado de acordo com a similaridade de seu centroide mais próximo. Essa similaridade pode ser calculada pela distância euclidiana ou pela similaridade por cosseno:

$$f(\mathbf{x_i}) = \max_{G_j \subset D^+} \cos(\mathbf{x_i}, \mathbf{g_j}) \tag{41}$$

$$f(\mathbf{x_i}) = \max_{G_j \subset D^+} \sqrt{\sum_{k=1}^{N} (x_{ik} - g_{jk})^2} \tag{42}$$

O uso de redes neurais para OCC foi proposto em (MANEVITZ; YOUSEF, 2007). O *autoencoder* é um modelo de arquitetura de rede neural treinado para copiar sua entrada na saída (GOODFELLOW; BENGIO; COURVILLE, 2016). O objetivo é aprender uma forma não supervisionada de representação dos dados. Formalmente, o problema consiste em aprender funções $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^p$ (*encoder*) e $\mathcal{B} : \mathbb{R}^p \to \mathbb{R}^n$ (*decoder*) que satisfaça

$$\arg\min_{\mathcal{A},\mathcal{B}} E[\Delta(\mathbf{x}, \mathcal{B} \circ \mathcal{A}(\mathbf{x})] \tag{43}$$

Onde E é o valor esperado da variável $\mathbf{x}$, Δ é a função de reconstrução do erro, que mede a distância entre o *output* do *decoder* e o *input*.

2.4.2 Aprendizado por exemplos positivos e não rotulados

O aprendizado por exemplos positivos e não rotulados (*Positive and Unlabeled Learning* – PUL) é um caso especial da classificação binária (BEKKER; DAVIS, 2020).

O problema pode ser definido da seguinte forma: Dado um conjunto de exemplos $\mathcal{V} = \{v_1, \cdots, v_l, v_{l+1}, \cdots, v_l + u\} \subset \mathbb{R}^m$ e uma classe de rótulos $\mathcal{C} = \{-1, 1\}$, 1 representando a classe positiva e -1 representando a classe negativa. Os primeiros l elementos de $\mathcal{V}$ são rotulados da classe positiva 1, formando o conjunto de exemplos rotulados $\mathcal{P}$. Os u elementos restantes de $\mathcal{V}$ são exemplos não rotulados, formando o conjunto de exemplos não rotulados $\mathcal{U}$, ainda, $u >> l$ e $|\mathcal{V}| = |\mathcal{P} \cup \mathcal{U}|$. Um algoritmo é dito de PUL quando aprende com o conjunto $\mathcal{P}$ e rotula os exemplos presentes em $\mathcal{U}$ (MA; ZHANG, 2017).

Os métodos de aprendizagem positiva podem ser divididas principalmente em três categorias: *two-step technique*, *biased learning* e *class prior incorporation*.

O método *biased learning* considera que os dados não rotulados contém ruído e são da classe negativa. Como o ruído para os exemplos negativos é uma constante, os dados são considerados pela hipótese de serem selecionados completamente ao acaso.

O método *class prior incorporation* contém três subcategorias: pós-processamento, pré-processamento e modificação de método. A categoria de pós-processamento treina um classificador probabilístico não tradicional considerando os dados não rotulados como negativos e modificando as probabilidades de saída. O método de pré-processamento modifica o conjunto de dados usando a probabilidade de cada classe. A modificação de método modifica o classificador para incorporar a distribuição dos dados (BEKKER; DAVIS, 2020; YU; LI, 2007).

O método *two-step technique* consiste em duas etapas: (1) identificar os exemplos negativos e (2) treinar um classificador baseado nos exemplos positivos, negativos e não rotulados. A primeira etapa pode ser definida como: Dados $\mathcal{X} = \{x_1, x_2, \cdots, x_m\}$, $\mathcal{Y} = \{y_i | y_i = \pm 1, i = 1, \cdots, m\}$ e $\mathcal{P} \subset \mathcal{X}$, $\mathcal{P} = \{x_k | y_k = 1\}$ queremos encontrar o conjunto de dados negativos $\mathcal{RN} \in \mathcal{X}$, $\mathcal{RN} = \{x_k | y_k = -1\}$.

No contexto de grafos, a primeira etapa de um algoritmo *two-step technique* pode ser definido como: Dado um grafo G com m nós, um conjunto de vetores $\mathcal{X} = \{x_1, \cdots, x_l, x_{l+1}, \cdots, x_{l+n}\} \in \mathbb{R}^m$ chamado de vetor de características dos nós e um conjunto de rótulos $\mathcal{C} = \{-1, 1\}$, os primeiros l elementos de $\mathcal{X}$ são rotulado como $y_i = 1, i = 1, \cdots, l$. Os restantes n elementos são não rotulados, ou seja, $y_i \in \mathcal{C}, i = l + 1, \cdots, n$. Portanto essa etapa visa encontrar o conjunto $\mathcal{RN} \subset \mathcal{X}$ tal que $y_i = -1, \forall x_i \in \mathcal{RN}, y_i \in \mathcal{C}$

Os algoritmos baseados em *two-step technique* assumem duas propriedades fundamentais dos dados (BEKKER; DAVIS, 2020).

1. Separabilidade dos dados: Por definição, os dados são separáveis se existe uma função f no espaço de hipóteses que mapeia todos os exemplos positivos para um valor maior ou igual a um limiar τ e todos os exemplos negativos para um valor menor que τ. Assim:

$$f(x_i) \geq \tau, \quad y = 1$$
$$f(x_i) < \tau, \quad y = 0$$

2. Suavidade dos dados: Se duas instâncias x_1 e x_2 são similares, então as probabilidades $\Pr(y = 1|x_1)$ e $\Pr(y = 1|x_2)$ serão similares.

Diversos algoritmos foram propostos na literatura para inferir sobre os exemplos negativos. Entre eles os mais comuns são algoritmos baseados em agrupamento e algoritmos baseados em Máquinas de Vetor de Suporte (SVM).

2.4.2.1 Algoritmos de Inferência de Exemplos Negativos em Problemas de Aprendizado Positivo

Diversos métodos foram propostos na literatura para inferir sobre os dados negativos. A técnica *Maximum Margin Clustering with Least Square* (MCLS) (CHAUDHARI; SHEVADE, 2012) propôs uma abordagem de agrupamento de dados utilizando o algoritmo

kMeans. O Algoritmo 1 mostra a execução proposta pelo MCLS. Um grupo é classificado como positivo ou negativo de acordo com a quantidade de elementos que pertencem a P. Depois disso, uma métrica de similaridade é utilizada para encontrar os elementos presentes nos grupos negativos que são mais dissimilares dos centroides dos grupos positivos. Os elementos com maior dissimilaridade são classificados como pertencentes ao conjunto $\mathcal{RN}$.

Algoritmo 1 *Maximum Margin Clustering with Least Square* – MCLS

Require: Conjunto de dados positivos $\mathcal{P}$, conjunto de dados não rotulados $\mathcal{U}$, quantidade de grupos k, quantidade de elementos num_neg.
Ensure: Conjunto de dados negativos $\mathcal{RN}$
  Agrupar utilizando o algoritmo kMeans todos os elementos de $\mathcal{P} \cup \mathcal{U}$
  for grupo $k_i, i = 1, \cdots, K$ **do**
    if k_i contém mais elementos de P do que de U **then**
      k_i é classificado como grupo positivo
    else
      k_i é classificado como grupo negativo
    end if
  end for
  for grupo k_j negativo **do**
    for $v_i \in k_j$ **do**
      calcula a distância entre v_i e os centróides dos grupos positivos
    end for
  end for
  Fazer um ranking com as distâncias
  Selecionar os num_neg elementos de $\mathcal{U}$ com os maiores distâncias para compor $\mathcal{RN}$

O algoritmo *Clustering-based Method for Collect Reliable Negative Examples* (C-CRNE) (LIU; PENG, 2014) utiliza *Agglomerative Clustering* para encontrar os dados negativos. O conjunto de dados positivos é dividido em grupos e o centroide de cada grupo é calculado. Depois disso, todos os exemplos não rotulados são considerados potenciais negativos. Enfim, são escolhidos aqueles com a maior distância entre os centroides dos grupos que contém elementos positivos para formar o conjunto $\mathcal{RN}$. O Algoritmo 2 exemplifica o funcionamento do CCRNE.

Algoritmo 2 *Clustering-based Method for Collect Reliable Negative Examples* (C-CRNE)

Require: Conjunto de dados positivos $\mathcal{P}$, conjunto de dados não rotulados $\mathcal{U}$
Ensure: Conjunto de dados negativos $\mathcal{RN}$
considere $X = \{x_i, i = 1, \cdots, m\} \in \mathcal{P}$
$O_p \leftarrow \frac{\sum_{i=1}^{m} x_i}{m}$, $r \leftarrow \max_{x_k \in P} d(x_k, O_p)$, $r_p \leftarrow \frac{r \times \varphi(m)}{\varphi(m)+1}$
$C_1 \leftarrow \{x_1\}$, $O_1 \leftarrow x_1$, numCluster $\leftarrow 1$, $Z \leftarrow \{x_2, \cdots, x_m\}$, $n_j \leftarrow 0$
while $Z \neq \emptyset$ **do**
  $O_j \leftarrow \arg_j \min_{k=1}^{\text{numCluster}} d(x_k, O_k)$
  if $d(x_i, O_j) < r_p$ **then**
    Adiciona X_i na classe c_j e ajusta o centro da classe j
    $O_j \leftarrow \frac{n_j \cdot O_j + x_i}{n_j + 1}$, $n_j \leftarrow n_j + 1$
  else
    Cria um novo grupo $C_{\text{numCluster}} \leftarrow \{x_i\}$
    numCluster $\leftarrow$ numCluster $+ 1$, $O_{\text{numCluster}} \leftarrow x_i$
  end if
end while
$\mathcal{RN} \leftarrow \mathcal{U}$
for $i \leftarrow 1$ até numCluster **do**
  for $x_i \in \mathcal{RN}$ **do**
    if $d(x_i, O_j) < r$ **then**
      $\mathcal{RN} \leftarrow \mathcal{RN} - \{x_i\}$
    end if
  end for
end for

No Algoritmo 2, $\varphi(m) = \log(m) + 1$, numCluster representa o número de classes geradas na iteração atual, m é o total de elementos para separação em grupos, $C_1, C_2, \cdots, C_{\text{numCluster}}$ são os grupos finais, O_j é o centróide de C_j e n_j é a quantidade de elementos do grupo C_j.

Li e Liu (2003) propuseram uma técnica que envolve a similaridade de dois vetores, criados de acordo com os exemplos presentes nos dados positivos e nos dados não rotulados. Assim, um novo exemplo é rotulado de acordo com a similaridade entre ambos os vetores. Esse método trata o conjunto dos dados não rotulados $\mathcal{U}$ como negativo e usa o conjunto $\mathcal{P}$ e $\mathcal{U}$ como dados de treino para criar um classificador *Rocchio*. O Algoritmo 3 exemplifica a inferência de dados negativos.

Algoritmo 3 *Rocchio*

Require: Conjunto de dados positivos $\mathcal{P}$, conjunto de dados não rotulados $\mathcal{U}$
Ensure: Conjunto de dados negativos $\mathcal{RN}$

$\vec{c}_+ \leftarrow \alpha \frac{1}{|\mathcal{P}|} \sum_{d \in \mathcal{P}} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|\mathcal{U}|} \sum_{d \in \mathcal{U}} \frac{\vec{d}}{||\vec{d}||}$
$\vec{c}_- \leftarrow \alpha \frac{1}{|\mathcal{U}|} \sum_{d \in \mathcal{U}} \frac{\vec{d}}{||\vec{d}||} - \beta \frac{1}{|\mathcal{P}|} \sum_{d \in \mathcal{P}} \frac{\vec{d}}{||\vec{d}||}$
for $d' \in \mathcal{U}$ **do**
  if $\text{sim}\left(\vec{c}_+, \vec{d'}\right) \leq \text{sim}\left(\vec{c}_-, \vec{d'}\right)$ **then**
    $\mathcal{RN} \leftarrow \mathcal{RN} \cup \{d'\}$
  end if
end for

Alguns métodos são propostos na literatura para encontrar os dados negativos em um problema de aprendizado positivo em grafos. O método *Label Propagation for Positive and Unlabeled Learning* (LP-PUL) (CARNEVALI et al., 2021) utiliza o algoritmo de Dijkstra (DIJKSTRA, 1959) para encontrar a média do menor caminho entre um vértice não rotulado do grafo e os vértices positivos. Então, os vértices não rotulados que tem a maior distância até os vértices positivos são escolhidos para compor o conjunto $\mathcal{RN}$. O Algoritmo 4 mostra a extração de exemplos negativos do algoritmo LP-PUL

Algoritmo 4 *Label Propagation for Positive and Unlabeled Learning* (LP-PUL)

Require: Conjunto de dados positivos $\mathcal{P}$, conjunto de dados não rotulados $\mathcal{U}$, quantidade de elementos negativos a serem inferidos *num_neg*, grafo G
for $p \in \mathcal{P}$ **do**
  $d \leftarrow Dijkstra(G, p)$
  for $u \in \mathcal{U}$ **do**
    $\mathbf{a}_u \leftarrow \mathbf{a}_u + d_u$
  end for
end for
for $u \in \mathcal{U}$ **do**
  $\mathbf{a}_u \leftarrow \mathbf{a}_u / |\mathcal{U}|$
end for
Retorna os *num_neg* elementos com maior distância em $\mathbf{a}$

Ma e Zhang (2017) propuseram o algoritmo *Positive and Unlabeled Learning by Label Propagation* (PU-LP) que utiliza o índice de Katz para encontrar a centralidade de cada par de vértices no grafo. Assim, encontra-se a menor dissimilaridade média entre os nós rotulados como positivos e os não rotulados para aumentar a cardinalidade do conjunto $\mathcal{P}$. Depois disso, utiliza-se a mesma métrica para encontrar a menor dissimilaridade entre o novo conjunto $\mathcal{P}$ e os dados não rotulados, formando o conjunto $\mathcal{RN}$ de mesma cardinalidade de $\mathcal{P}$. O Algoritmo 5 mostra a extração de exemplos negativos.

Algoritmo 5 *Positive and Unlabeled Learning by Label Propagation* (PU-LP)

Require: Conjunto de dados positivos $\mathcal{P}$, conjunto de dados não rotulados $\mathcal{U}$, número de iterações m, o parâmetro $\lambda \in (0,1)$ que controla o tamanho do conjunto $\mathcal{RP}$, a matriz de similaridade W calculada utilizando o índice de Katz.

$\mathcal{RP} \leftarrow \emptyset, \mathcal{P}' \leftarrow \mathcal{P}, \mathcal{U}' \leftarrow \mathcal{U}$

for $k = 1 : m$ **do**

Baseado em W, calcular $\overline{S_{v_i}}, v_i \in \mathcal{U}', \overline{S_{v_i}} = \dfrac{\sum_{j=1}^{|\mathcal{P}'|} w_{ij}}{|\mathcal{P}'|}$

Ordenar cada exemplo v_i de acordo com S_{v_i}

$\mathcal{RP}' \leftarrow$ os maiores $\dfrac{\lambda}{m} \times |\mathcal{P}|$ exemplos ordenados em $\mathcal{U}'$

$\mathcal{P}' \leftarrow \mathcal{P}' \cup \mathcal{RP}', \mathcal{U}' \leftarrow \mathcal{U}' - \mathcal{RP}', \mathcal{RP} \leftarrow \mathcal{RP} \cup \mathcal{RP}'$

end for

Com base em W, computa $\overline{S_{v_i}}, v_i \in \mathcal{U}' - \mathcal{RP}, \overline{S_{v_i}} = \dfrac{\sum_{j=1}^{|\mathcal{P}' \cup \mathcal{RP}|} w_{ij}}{|\mathcal{P}' \cup \mathcal{RP}|}$

Ordenar cada exemplo v_i de acordo com $S_{v_i} \in \mathcal{U} - \mathcal{RP}$

$RN \leftarrow$ os menores $|\mathcal{P} \cup \mathcal{RP}|$ exemplos em $\mathcal{U} - \mathcal{RP}$

A Tabela 1 mostra um resumo dos algoritmos selecionados.

Algoritmo	Estrutura	Parâmetros
MCLS	Baseado em Vetores	$k, \mathcal{U}, \mathcal{P}, num\_neg$
LP_PUL	Baseado em Grafos	$\mathcal{U}, \mathcal{P}, num\_neg, G$
CCRNE	Baseado em Vetores	$\mathcal{U}, \mathcal{P}$
PU_LP	Baseado em Grafos	$\mathcal{U}, \mathcal{P}, m, \lambda, G, W$
RCSVM	Baseado em Vetores	$\mathcal{U}, \mathcal{P}$

Tabela 1 – Resumo da estrutura utilizada nos algoritmos selecionados

Os algoritmos de aprendizado positivo possuem diversas aplicações em problemas de várias áreas do conhecimento. Algumas dessas aplicações são exploradas na próxima seção.

2.4.2.2 Aplicações de Aprendizado Positivo

Uma quantidade muito grande de problemas de classificação reais se encaixam no cenário de aprendizado positivo (JASKIE; SPANIAS, 2019). Essa seção apresenta uma breve discussão a respeito de aplicações do paradigma de aprendizado positivo.

O algoritmo LP-PUL foi utilizado no trabalho de Souza et al. (2021) em um problema de classificação sobre *fake news*. Os resultados obtidos foram satisfatórios principalmente quando utilizado Doc2Vec para realizar o *embedding* dos documentos.

No contexto de identificação de interações entre fármacos, Zheng et al. (2019) propuseram um método *DDI-PULearn* que gera exemplos negativos a partir de *One-Class Support Vector Machines* (OCSVM), utilizando tanto os exemplos negativos quanto exemplos positivos para treino.

Algoritmos de PUL podem ser utilizados também para identificar genes relacionados à doenças. Yang et al. (2012) desenvolveram um algoritmo de PUL para a identificação de genes utilizando *Support Vector Machines* (SVM). Para lidar com o mesmo problema de pesquisa, Yang et al. (2014) aplicaram o algoritmo de Propagação de Rótulos (*Label Propagation*) para classificar exemplos não rotulados.

Para classificação automática de textos, Peng, Zuo e He (2007) apresentaram uma abordagem utilizando PUL e SVM iterativamente para criar um conjunto confiável de classificadores. Para classificação de páginas web, Yu, Han e Chang (2004) apresentaram um *framework* que elimina a necessidade de classificação manual de páginas negativas utilizando PUL juntamente com SVMs.

Sistemas de recomendação são grande campo de aplicação de PUL. Para recomendação de páginas web, uma página de interesse pode ser rotulada com uma classe positiva, enquanto outras páginas pertencem ao conjunto de dados não rotulados (JASKIE; SPANIAS, 2019). A abordagem de Zhang e Lee (2005) utiliza probabilidade para tratar de problemas de aprendizado positivo. Amostras de dados são selecionadas aleatoriamente para compor o conjunto de dados positivos e, sobre essa hipótese, se todos os exemplos restantes forem rotulados como negativos, não haveria erro em um exemplo negativo, apenas exemplos positivos seriam classificados incorretamente com uma probabilidade conhecida.

2.4.2.3 Métricas de Validação

A validação de algoritmos de PUL através de métricas para conjuntos de dados desbalanceados é algo interessante a se considerar, porém, as métricas amplamente utilizadas na literatura são as tradicionais, como acurácia e F1 *score*.

A acurácia é um conceito fundamental na análise e avaliação de modelos de aprendizado de máquina. Ela representa a medida da precisão de um modelo em relação aos resultados obtidos na tarefa de classificação. Para calcular a acurácia, utiliza-se a equação

$$\text{acurácia} = \frac{\text{Número de previsões corretas}}{\text{Total de previsões}} \tag{44}$$

Que verifica a proporção de exemplos que produziram uma saída correta (GOODFELLOW; BENGIO; COURVILLE, 2016).

Além da acurácia, existem outras técnicas de validação de desempenho. As mais utilizadas são a precisão (*precision*), a revocação (*recall*) e o F1 *score*. Para entender essas métricas é necessário dividir as predições em quatro valores diferentes:

- Verdadeiro Positivo (*True Positive* – TP): Representa o número de instâncias que foram classificadas como positivas e que realmente pertencem à classe positiva.
- Falso Negativo (*False Negative* – FN): Representa o número de instâncias que foram classificadas como negativas mas na verdade pertencem à classe positiva.

- Verdadeiro Negativo (*True Negative* – TN): Representa o número de instâncias que foram classificadas como negativas e que realmente pertencem à classe negativa.
- Falso Positivo (*False Positive* – FP): Representa o número de instâncias que foram classificadas como positivas mas na verdade pertencem à classe negativa.

Segundo Faceli et al. (2011):

- A precisão é a proporção de exemplos positivos classificados corretamente entre todos aqueles preditos como positivos, podendo ser calculada através da equação

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{45}$$

- A revocação corresponde à taxa de acerto na classe positiva, e pode ser definida como

$$\text{Revocação} = \frac{VP}{VP + FN} \tag{46}$$

- O F1 *score* Micro é uma métrica que combina a precisão e a revocação para avaliar um modelo de classificação. O valor do F1 *score* pode ser obtido multiplicando os valores da precisão e da revocação e dividindo pela soma deles conforme a equação

$$\text{F1\_Score} = \frac{2 \cdot \text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{47}$$

O F1 *score* Micro será utilizado para a validação dos experimentos apresentados no Capítulo 4. Além disso, a classe positiva será substituída pela classe negativa, a fim de verificar a eficiência dos algoritmos na inferência de dados negativos em um problema de aprendizado positivo.

Capítulo 3

Reescrita em GNNs para Aprendizado Positivo

Esse capítulo apresenta os métodos propostos para inferência de dados negativos em um problema de PUL e para classificação binária. Além disso, apresenta as ferramentas utilizadas tais quais os conjuntos de dados e também as avaliações experimentais e testes paramétricos.

Os algoritmos de PUL baseados na estratégia de duas etapas são formados por: (1) inferir os dados negativos através de dados positivos; (2) criar um classificador binário capaz de inferir os rótulos de todos os exemplos.

A abordagem proposta consiste primeiramente na criação de múltiplos grafos através de uma estratégia de reescrita; em seguida, esses grafos são utilizados por um GAE cujo *encoder* é feito por uma RGCN. A representação latente aprendida por essa rede é capaz de separar exemplos positivos e negativos, facilitando a extração de dados negativos.

Assim que esses dados são extraídos, qualquer classificador binário pode ser utilizado para classificar esses exemplos.

As Seções 3.1, 3.2, 3.3 e 3.4 detalham o funcionamento do algoritmo proposto.

3.1 Reescrita Positiva Sequencial via Busca em Largura

O empilhamento de camadas de GCN com o objetivo de agregar informações em um raio maior se mostra ineficiente na maioria dos conjuntos de dados (KIPF; WELLING, 2016a). Assim, estratégias de reescrita mostram-se consistentes para mitigar esse pro-

blema (RUSCH; BRONSTEIN; MISHRA, 2023). A arquitetura LASER (*Locality Aware Sequential Rewiring*) proposta por Barbero et al. (2023) consiste em conectar uma parcela ρ de vértices que melhore uma medida de conectividade μ do grafo. Essa medida de conectividade pode ser diferente para cada problema, dificultando na formulação de uma medida ideal.

Para adequar uma métrica de conectividade ao problema de PUL, este trabalho propõe uma nova estratégia. Considerando uma abordagem sequencial de L iterações e supondo que a troca de informações entre vértices positivos facilita a convergência para a solução do problema, o modelo proposto de Reescrita Positiva Sequencial via Busca em Largura (***P**ositive **S**equential **R**ewiring via **B**readth Search* – PSRB) consiste na aplicação de uma função de probabilidade geométrica ϕ para a criação de novas arestas entre vértices positivos, ou seja, a estratégia de reescrita está restrita ao conjunto $\overline{E} \subset E$ onde $\overline{E} = \{(u, v) : u, v \in \mathcal{P}\}$. A cada aplicação de ϕ um novo grafo é gerado. Ao final de L iterações, são gerados L grafos.

A função ϕ é definida como:

$$\phi(u, v|P, L) = \frac{1}{d(u, v)^{\alpha} \cdot |P|^{\beta} \cdot L^{\gamma}}, \tag{48}$$

onde $|P|$ representa a quantidade de elementos positivos rotulados e $\alpha, \beta, \gamma \in \mathbb{R}$. A distância d calculada pelo algoritmo de busca em largura é tal que $d \in [1, r]$ onde r é o raio do grafo G

A escolha dessa função densidade de probabilidade se dá ao fato de que:

1. A distância deve ser considerada para preservar a localidade. Além disso, deve ser possível ligar vértices cuja distância é infinito (vértices de componentes conexas distintas) escolhendo $\alpha = 0$.

2. A quantidade de elementos positivos deve ser levada em consideração para a escolha da adição de arestas. Porém, deve-se manter o grafo esparso, pois conjuntos de dados muito grandes podem resultar em uma adição muito grande de arestas.

3. A quantidade de grafos gerados deve ser considerada para a escolha de adição de novas arestas, com o objetivo de que haja um limitante superior para a quantidade de elementos em E.

A Figura 14 exemplifica a aplicação da estratégia de *rewiring* proposta pelo Algoritmo 6.

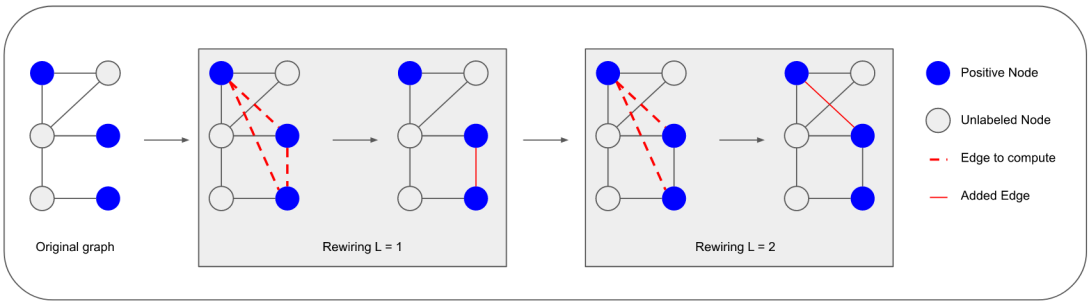


Figura 14 – Ilustração da abordagem de *rewiring* proposta. A aplicação de *rewiring* é feita para cada valor de L, resultando em um grafo diferente que é utilizado pelo modelo para agregar informação de vizinhanças distintas.

Algoritmo 6 Reescrita Positiva Sequencial via Busca em Largura

Input: Conjunto de vértices positivos P, grafo G
Output: Grafos Reescritos *rewiring_graphs*
alpha, beta, gamma $\leftarrow$ parâmetros de função de probabilidade
rewiring_graphs $\leftarrow$ []
for $l \in [1, L]$ **do**
  for all $(u, v) \in P$ **do**
    Calcula distância BFS $d(u, v)$ entre u e v
    Calcula a probabilidade $\phi(u, v|P, L) = \frac{1}{d(u,v)^{\alpha} \cdot |P|^{\beta} \cdot L^{\gamma}}$
    Gera um número aleatório $r \sim \text{UNIFORM}(0, 1)$
    if $r < \phi(u, v|P, L)$ **then**
      Adiciona a aresta (u, v) em G'
    end if
  end for
  adiciona G' em *rewiring_graphs*
end for
return *rewiring_graphs*

Essa abordagem permite a criação de múltiplos grafos no conjunto *rewiring_graphs*. Para que seja possível extrair informações desses múltiplos grafos o modelo de rede neural em grafos deve ser capaz de operar transferências de mensagem sobre múltiplos vértices.

3.2 RGCN em múltiplos grafos

Para lidar com L grafos gerados a partir do modelo de reescrita aplicado, considera-se uma GNN similar ao modelo RGCN proposto por Schlichtkrull et al. (2018). Porém, cada grafo distinto gerado pelo modelo de reescrita será considerado como uma relação, ou seja, a transferência de mensagem será feita de forma diferente para cada grafo e, como a reescrita é aplicada a vértices positivos, os mesmos se beneficiarão dessa arquitetura.

Considerando uma GCN como na Equação 31 a partir da perspectiva de reescrita da Equação 37, o modelo proposto consiste em empilhar T camadas da seguinte forma:

$$x_v^{(t)} = \varphi\left(\sum_{u \in N(v) \cup v} \frac{1}{\sqrt{d_u d_v}} W_0^{(t)} x_u^{(t-1)} + \sum_{l=1}^{L} \sum_{u \in N_\phi^l(v) \cup \{v\}} \frac{1}{\sqrt{d_{u,l} d_{v,l}}} W_l^{(t)} x_u^{(t-1)}\right) \tag{49}$$

onde d_u representa o grau do vértice u, $d_{u,l}$ representa o grau de u no grafo l e $N_\phi^l(v)$ o conjunto dos vizinhos de v da l-ésima iteração do algoritmo a partir da função densidade de probabilidade ϕ, φ é uma função de ativação não linear.

O modelo proposto na Equação 49 servirá como *encoder* de um modelo GAE da forma como proposto por Kipf e Welling (2016b). O *decoder* do modelo será feito através de do produto do resultado do *encoder* por sua transposta. A matriz $\hat{A}$, que consiste na matriz de adjacência reconstruída então é definida como

$$\hat{A} = \sigma(ZZ^T), \ \textbf{com} \ Z = \mathrm{RGCN}(X, A) \tag{50}$$

Dessa forma, o resultado obtido pelo *embedding* após o treinamento leva em consideração a densidade de conexões geradas pelo *rewiring* e a matriz de características do grafo.

Esse espaço de representação latente (ou seja, *embedding*) gerado por meio da RGCN tende a agrupar os vértices positivos e, consequentemente, mantendo os vértices negativos mais distantes, facilitando inferir vértices negativos seguros. Isso permite que modelos de AM tenham um ganho significativo quando usados nesse espaço.

Para a inferência dos dados negativos seguros, utiliza-se uma medida de similaridade no espaço de representação latente gerado pelo *embedding*, classificando os top *num_neg* exemplos mais distantes como negativos seguros. Para esse trabalho, a medida de similaridade será a distância euclidiana. O Algoritmo 7 exemplifica o funcionamento da inferência dos dados negativos seguros.

Algoritmo 7 Extração de Negativos Seguros

```
model ← Train(G)
lat_rep ← GAE_Encode(model, G)
avg_distances ← {}
for u in U do
    dist ← 0
    for v in P do
        dist ← dist + Distance(lat_rep[u], lat_rep[v])
    end for
    avg_distances[u] ← dist / |P|
end for
sorted_vertices ← Sort_By_Avg_Dist(avg_distances)
for i from 1 to num_neg do
    u ← sorted_vertices[i]
    Label(u) ← negative
end for
return Labeled_Graph(G, P, U)
```

3.3 GCN como baseline para inferência de dados negativos

Embora o modelo GCN já tenha sido usado em trabalhos de PUL (WU et al., 2021a) usando uma abordagem baseada em aprendizado por viés, não foram encontrados trabalhos que utilizam a GCN para a inferência de dados negativos seguros para problemas de PUL considerando técnicas baseadas em duas etapas. Essa abordagem, embora não seja novidade, servirá como *baseline* para o modelo proposto. Portanto, para fins de reprodutibilidade, será apresentado como esse modelo foi utilizado como baseline.

Para comparar o modelo proposto com abordagens já apresentadas na literatura, será feita uma adaptação no modelo GCN (KIPF; WELLING, 2016a). Para isso, será utilizado um modelo GAE como proposto por Kipf e Welling (2016b), cujo *encoder* é um modelo GCN e o *decoder* é o produto do *encoder* pela sua transposta. Assim, a matriz $\hat{A}$, que consiste da matriz de adjacência reconstruída pelo GAE é definida pela Equação 51.

$$\hat{A} = \sigma(ZZ^T), \ \textbf{com} \ Z = \mathrm{GCN}(X, A) \tag{51}$$

Similar ao modelo PSRB+RGCN, os top *num_neg* exemplos mais distantes serão classificados como negativos seguros e a medida de similaridade usada será a distância euclidiana.

3.4 Classificador Binário

A segunda etapa do algoritmo proposto consiste em utilizar um classificador binário para classificar o restante dos exemplos a partir dos dados positivos e dos dados negativos encontrados na primeira etapa. Qualquer algoritmo pode ser utilizado, tais quais SVMs, redes neurais, algoritmos de propagação de rótulos. Para esse trabalho foi utilizada uma rede neural MLP acoplada ao final do GAE da etapa anterior, sendo feito o treinamento apenas da MLP, congelando os outros parâmetros da rede. Além disso, os pesos utilizados na função de erro são proporcionais à quantidade de dados positivos presentes em cada um dos conjuntos de dados.

A Figura 15 exemplifica o funcionamento do modelo proposto.

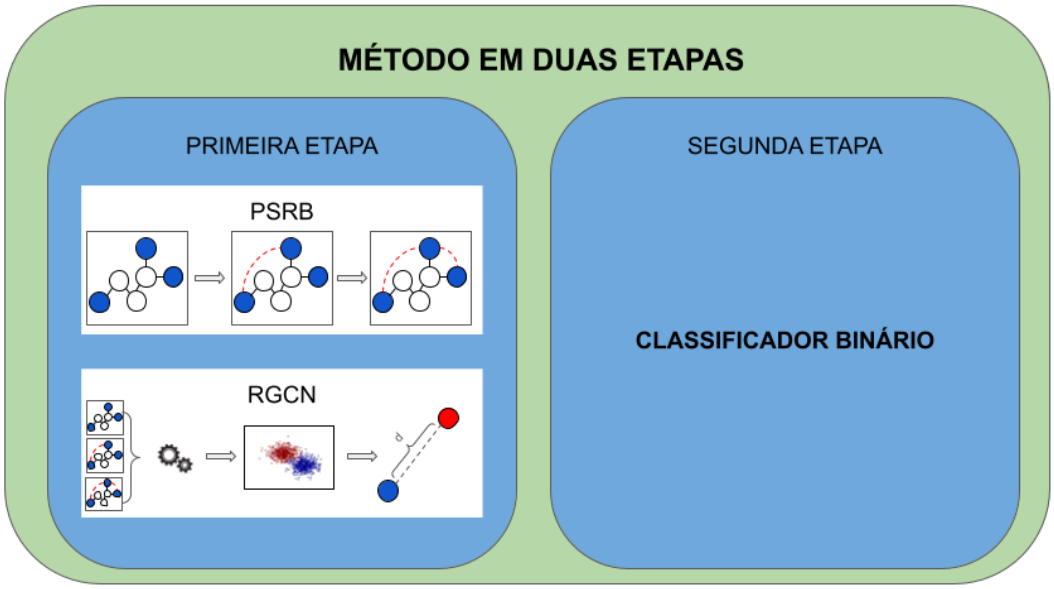


Figura 15 – Esquematização do Algoritmo Proposto

Capítulo 4

Análise Experimental

Ao que consta, não foram encontrados trabalhos que façam a análise da Primeira Etapa dos algoritmos de PUL de forma isolada. Os artigos da literatura avaliam esses algoritmos considerando a resposta obtida no final das duas fases (identificação de dados negativos e classificação binária).

Dessa forma, limitações encontradas na primeira etapa são mascaradas pelo resultado final, dificultando que tais limitações sejam endereçadas. Assim, como primeiro resultado desse mestrado, vamos avaliar a Primeira Etapa de algoritmos de PUL de forma isolada.

Esse capítulo apresenta as ferramentas, os conjuntos de dados utilizados, os testes paramétricos e os resultados dos testes realizados com o modelo proposto e *benchmarks* de algoritmos da literatura.

4.1 Ferramentas

Durante a execução dos experimentos foi utilizada a linguagem de programação *Python*, com as bibliotecas *Numpy* (HARRIS et al., 2020), *PyTorch Geometric* (FEY; LENSSEN, 2019a), Networkx (HAGBERG; SCHULT; SWART, 2008) e *Pandas* (MCKINNEY, 2010). A Tabela 2 mostra as ferramentas utilizadas em cada etapa da pesquisa.

Etapa da pesquisa	Ferramenta Utilizada	Bibliotecas
Elaboração dos Algoritmos	*Python* 3.10.12	*Numpy, Pytorch Geometric, NetworkX, Networkit*
Testes paramétricos	*Python* 3.10.12	*Numpy, Pytorch Geometric, NetworkX, Pandas, Matplotlib, Networkit*
Avaliação dos resultados	*Python* 3.10.12	*Numpy, Pytorch Geometric, NetworkX, Pandas, Matplotlib, Networkit*

Tabela 2 – Ferramentas

4.2 Conjuntos de dados e *Baselines*

Os experimentos foram realizados de três formas distintas. Primeiro o algoritmo foi testado com diferentes parâmetros. O objetivo é entender a influência dos algoritmos para cada conjunto de dados. Em seguida avaliou-se o algoritmo quanto à inferência de dados negativos, ou seja, se o algoritmo é realmente capaz de inferir dados negativos a partir de exemplos positivos. Finalmente, foi avaliada a classificação geral de todos os exemplos. Para inferir os rótulos de todos os exemplos, os algoritmos PU_LP e LP_PUL utilizaram o algoritmo *Learning with Local and Global Consistency* (LLGC) (ZHOU et al., 2003). O algoritmo GAE utilizou a arquitetura GCN. Os demais algoritmos utilizaram uma SVM tradicional com o uso da biblioteca Scikit-Learn (PEDREGOSA et al., 2011).

Para todos os três experimentos foram utilizados os conjuntos de dados Cora, CiteSeer, PubMed, Amazon Photo e DBLP (YANG; COHEN; SALAKHUTDINOV, 2016; BOJCHEVSKI; GüNNEMANN, 2018; SHCHUR et al., 2019), com as classes positivas sendo 3, 2, 2, 6 e 0 respectivamente. As escolhas das classes representam os elementos com maior frequência nos conjuntos de dados, transformando o problema de classificação multiclasse em um problema binário, assim como feito em (WU et al., 2021a).

A Tabela 3 mostra as estatísticas dos conjuntos de dados selecionados. A densidade é a razão do número de arestas pelo quadrado dos vértices.

Tabela 3 – Estatísticas dos Conjuntos de Dados

Dataset	Vértices	Arestas	Densidade	Características	Positivos (%)	Positivos (abs)
Cora	2708	5278	0.0007	1433	30.21%	818
CiteSeer	3327	4552	0.0004	3703	20.08%	668
PubMed	19717	44302	0.0001	500	39.09%	7875
Photo	7650	119081	0.0002	745	25.37%	1941
DBLP	17716	52867	0.0003	1639	44.71%	7920

Para fazer uma comparação justa do modelo proposto com os outros algoritmos, selecionamos as seguintes *baselines* com as adaptações necessárias, comparando o nosso

modelo com métodos clássicos de PUL focados em duas etapas. Para todos os exemplos, a quantidade de elementos rotulados utilizada foi igual à quantidade de positivos.

- **MCLS**: Para o agrupamento k-Means foi utilizado um valor de $k = 7$. Para impedir que o conjunto de clusters positivos fosse vazio, caso um cluster obtivesse mais da média de positivos, o mesmo era classificado como positivo.

- **CCRNE**: A fim de evitar que a execução do algoritmo não fosse capaz de retornar elementos negativos foi considerado um ratio $r = 0.3$ para diminuir essa distância dos centróides positivos, com o intuito de evitar que o conjunto de negativos fosse vazio.

- **RCSVM**: Para a aplicação desse algoritmo foi utilizada a distância euclidiana e os valores de $\alpha = 0.7, \beta = 0.3$.

- **PU_LP**: Os parâmetros utilizados para a execução desse algoritmo foram $\alpha = 0.1, \lambda = 1, m = 3$.

- **LP_PUL**: O algoritmo LP_PUL utiliza a distância através do algoritmo de Dijkstra para encontrar os elementos mais distantes, não sendo necessário parâmetros adicionais.

- **GCN**: Os parâmetros utilizados foram $\text{N\_LAYERS} = [\text{NUM\_FEATURES}, 64, 16]$ com otimizador Adam, por 100 épocas de treinamento do GAE e 200 épocas do treinamento da GCN com taxa de aprendizagem 0.001, com uso da biblioteca PyTorch Geometric (FEY; LENSSEN, 2019b). Para classificar os dados negativos, foi acoplada uma nova camada de GCN treinada com os dados positivos e com dados negativos obtidos na primeira etapa.

- **PSRB**: Para a abordagem proposta foram utilizados valores de $L = 2$, $\text{N\_LAYERS} = [\text{NUM\_FEATURES}, 64, 16]$ com otimizador Adam, por 100 épocas de treinamento do GAE com taxa de aprendizagem 0.001, com uso da biblioteca PyTorch Geometric. Para rotular o restante dos exemplos, foi utilizada uma camada de MLP no final do embedding do GAE, acoplada à ultima camada de RGCN. As camadas já treinadas foram deixadas congeladas, somente a camada MLP foi treinada por 200 épocas e taxa de aprendizagem 0.001.

4.3 Avaliação Paramétrica

Para avaliar os parâmetros do modelo proposto foram realizados dois experimentos distintos. O primeiro tem como objetivo entender quais são as influências dos parâmetros de probabilidade α, β e γ no *rewiring*. O segundo busca encontrar qual o melhor valor de L

para cada conjunto de dados, mostrando como o algoritmo se comporta com informações de um número maior de grafos.

4.3.1 Variáveis Probabilísticas

Para entender a influência dos parâmetros de probabilidade α, β e γ no *rewiring* foram feitos testes com os valores de $\alpha = [0, 0.4]$, $\beta = [0, 1]$ com intervalo 0.1 e $\gamma = [0, 4]$ com intervalo 1. Para esses testes foram utilizados os parâmetros $L = 2$ e a *rate* de dados positivos foi fixa em 0.1. A Tabela 4 ilustra a média dos 10 maiores e menores valores obtidos de F1 *score* para diferentes parâmetros testados.

	Highest Scores				**Lowest Scores**			
Dataset	α	β	γ	**F1 *score***	α	β	γ	**F1 *score***
Amazon Photo	0.15	0.11	1.6	0.6438	0.09	0.63	2.3	0.4103
DBLP	0.2	0.71	1.7	0.8613	0.14	0.04	2.8	0.4301
CiteSeer	0	0.56	2	0.5743	0.2	0.27	3.4	0.2989
Cora	0.23	0.46	1.6	0.7447	0	0.15	1.6	0.3468
PubMed	0.17	0.36	2.2	0.8049	0.29	0.93	2.8	0.5628

Tabela 4 – Parâmetros probabilísticos de reescrita

Os valores de α consideram a distância entre dois vértices para a adição de uma nova aresta (Equação 48). Para conjuntos com grande número de componentes conexas, o valor de $\alpha = 0$ implica que haverá conexão entre componentes conexas.

O parâmetro β consiste na variação da probabilidade de adição de novas arestas a partir da quantidade de dados positivos. Pelos resultados obtidos, esse valor aparenta ser o mais importante da rede. Para conjuntos de dados com grande número de vértices positivos e alta densidade, considera-se um valor de β maior do que conjuntos de dados com grande número de vértices positivos e baixa densidade. Para conjuntos de dados com número menor de vértices positivos e densidade média, considera-se valores de β próximos de 0.5.

O parâmetro γ funciona como uma forma de conter a adição excessiva de arestas à medida que a quantidade de grafos gerados pelo *rewiring* aumenta.

4.3.2 Número de Grafos Gerados

Para avaliar a eficiência do algoritmo a partir do parâmetro L vamos considerar 5 amostras para valores de 0.01, 0.05, 0.10, 0.15, 0.20 e 0.25 de dados rotulados para os conjuntos de dados Cora e PubMed. Para o cálculo da probabilidade de *rewiring* foram utilizados valores de $\alpha = 0.23, \beta = 0.46, \gamma = 2$ para o conjunto de dados Cora e $\alpha = 0.17, \beta = 0.36, \gamma = 2$ para o conjunto de dados PubMed.

As Figuras 16 e 17 ilustram os resultados obtidos para os experimentos.

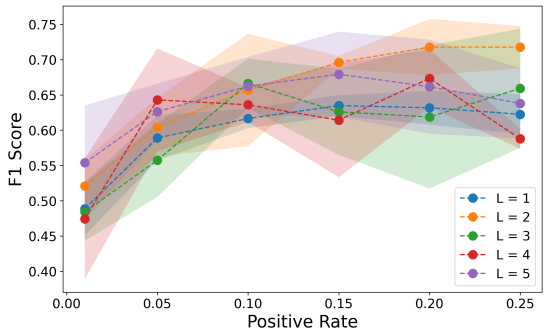


Figura 16 – F1 score for Cora dataset for diferent values of L

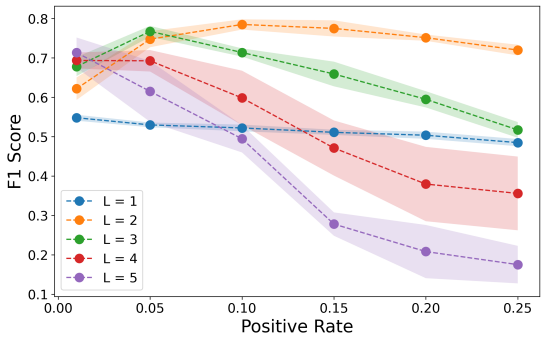


Figura 17 – F1 score for PubMed dataset for diferent values of L

O algoritmo possui diferentes desempenhos para diversos valores de L. À medida que os valores de L aumentam, a pontuação do algoritmo diminui. Esse fenômeno pode ser justificado pelo *over-smoothing*, visto que muitos valores de L fazem com que a representação latente de um vértice tenha que agregar muita informação em um vetor de tamanho fixo. Esse acontecimento é similar aos experimentos feitos por Kipf e Welling (2016a) e por Wu et al. (2021a), onde muitas camadas de convolução ou a agregação de vizinhanças muito distintas resultavam em pontuações baixas. Em ambos os experimentos o valor de $L = 2$ obteve os resultados mais relevantes.

4.3.3 Crescimento da Quantidade de Arestas

O parâmetro γ da Equação 48 representa a importância do iterador na adição de novas arestas. Inicialmente, esse parâmetro não foi considerado, resultando em, além de uma quantidade muito grande de arestas adicionadas, um aumento na adição em cada iteração do algoritmo, gerando *over-squashing*.

As Figuras 18 - 21 mostram a diferença entre considerar o valor γ sendo diferente de zero para os conjuntos de dados PubMed e Cora.

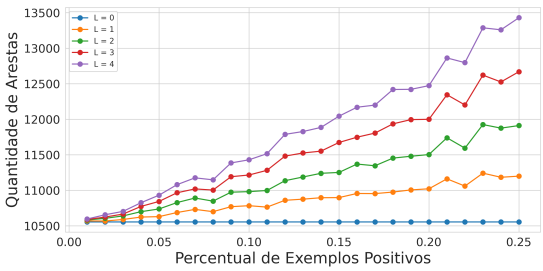


Figura 18 – Quantidade de arestas por rate no conjunto de dados Cora para $\alpha = 0.2, \beta = 0.8, \gamma = 0$.

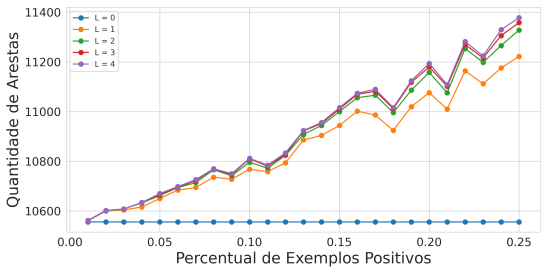


Figura 19 – Quantidade de arestas por rate no conjunto de dados Cora para $\alpha = 0.2, \beta = 0.8, \gamma = 3$.

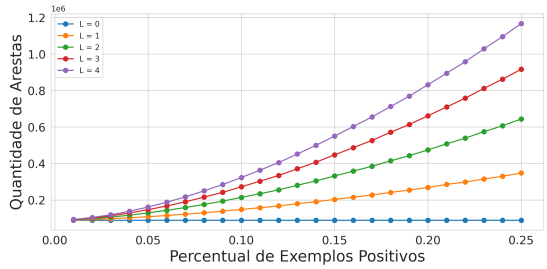


Figura 20 – Quantidade de arestas por rate no conjunto de dados PubMed para $\alpha = 0.2, \beta = 0.4, \gamma = 0$.

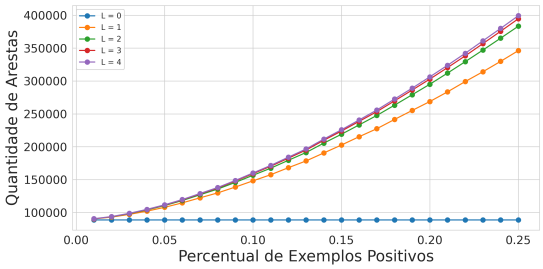


Figura 21 – Quantidade de arestas por rate no conjunto de dados PubMed para $\alpha = 0.2, \beta = 0.4, \gamma = 3$.

O parâmetro γ não foi considerado em uma versão inicial da função densidade de probabilidade ϕ. Depois de observar o crescimento na quantidade de arestas, concluímos que a consideração da quantidade de vértices tende a conservar o grafo esparso.

Os valores obtidos no conjunto de dados Cora e PubMed mostram a importância do parâmetro γ em preservar a estrutura topológica do grafo. Desconsiderar esse parâmetro pode gerar um grafo com até dez vezes mais arestas. A escolha desse parâmetro é um desafio do algoritmo. Essa discussão se apresenta no Capítulo 5

4.3.4 Avaliação de *Embedding*

Modelos *AutoEncoder* tem como função principal a representação latente das variáveis de cada exemplo, ou seja, são utilizados para representar os dados em espaços de dimensão menores. O modelo PSRB utiliza estratégias de reescrita para conectar densamente exemplos positivos em um problema de PUL. Isso afeta diretamente as representações latentes obtidas pela arquitetura proposta.

Para gerar o *embedding* a ser avaliado, foram utilizados dois modelos, um GAE com uma GCN como *embedding* (KIPF; WELLING, 2016b) e um GAE com o modelo de rewiring PSRB. Ambos os modelos treinaram por 200 épocas com otimizador Adam e taxa de aprendizagem 0.001. As camadas utilizadas foram $[\text{num\_features}, 64, 2]$ para gerar uma representação visível em um gráfico de duas dimensões.

As Figura 22 - 25 mostram a representação latente obtida pelos modelos GAE + GCN e PSRB + RGCN. Em azul estão os vértices positivos, em verde os negativos, em vermelho os vértices positivos e rotulados e em branco os vértices atribuídos como negativo pelo modelo baseado em distância. Além disso, os elementos classificados incorretamente estão em preto.

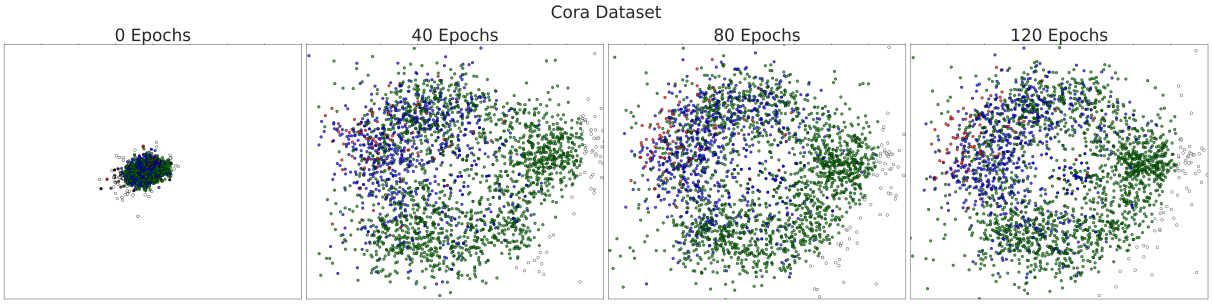


Figura 22 – *Embedding Graph Autoencoder* PSRB + RGCN no conjunto de dados Cora.

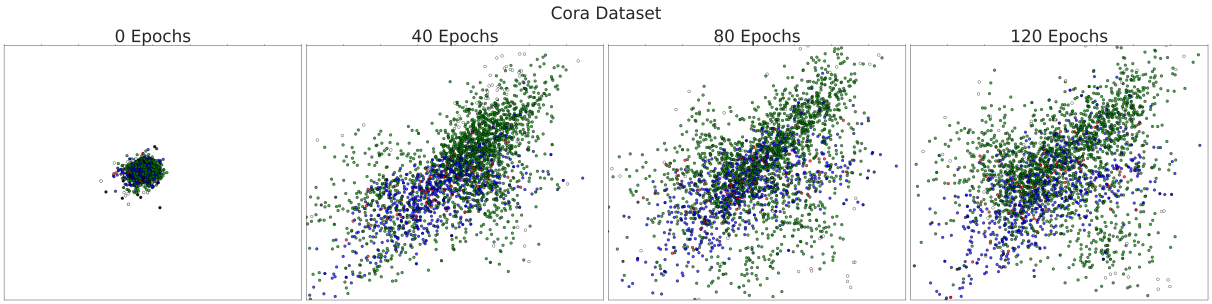


Figura 23 – *Embedding Graph Autoencoder* GCN no conjunto de dados Cora.

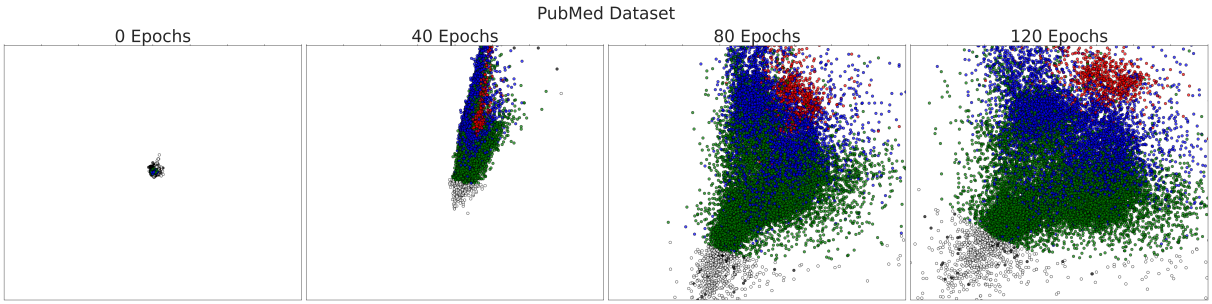


Figura 24 – *Embedding Graph Autoencoder* PSRB + RGCN no conjunto de dados PubMed.

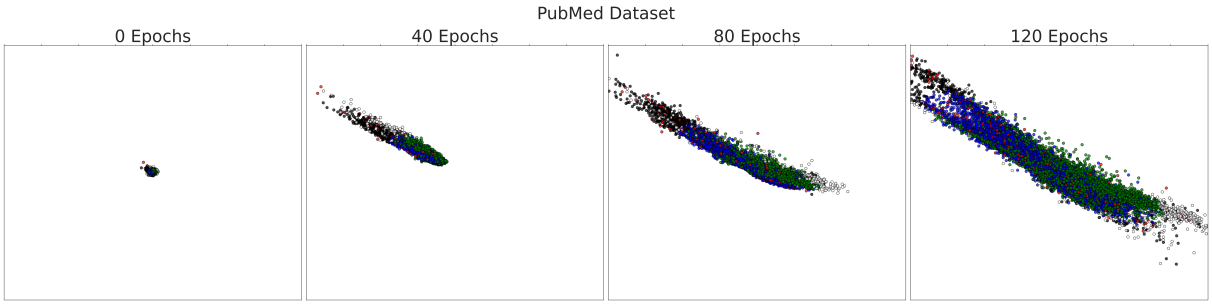


Figura 25 – *Embedding Graph Autoencoder* GCN no conjunto de dados PubMed.

O *embedding* gerado pelo modelo PSRB + RGCN tende a diminuir as distâncias entre os vértices positivos, influenciando outros vértices positivos a se distanciarem, tornando o espaço de representação latente mais simples para a classificação proposta.

4.4 Extração de Exemplos Negativos

Para avaliar a qualidade da extração de exemplos negativos, foram considerados valores de dados positivos rotulados num intervalo unitário entre 1 e 25%. Os valores negativos a serem inferidos são os mesmos dos valores positivos. Cada algoritmo treinou 5 vezes em cada conjunto de dados e os resultados obtidos representam as médias desse treinamento.

As figuras 26 - 30 ilustram os resultados obtidos nos conjuntos de dados testados.

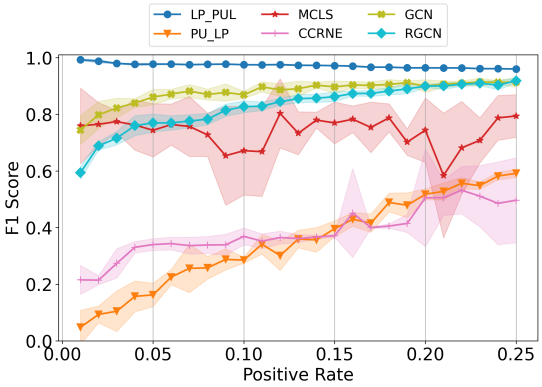


Figura 26 – F1 score para o conjunto de dados DBLP com os valores $\alpha = 0.2, \beta = 0.8, \gamma = 0$.

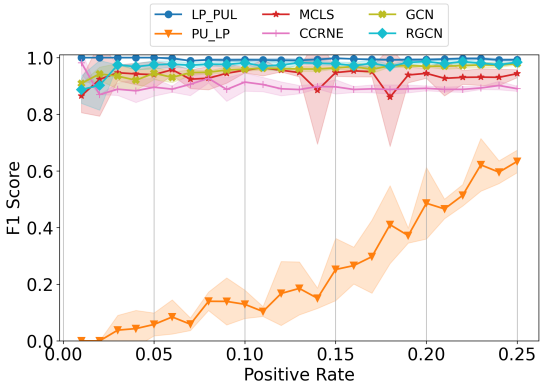


Figura 27 – F1 score para o conjunto de dados CiteSeer com os valores $\alpha = 0, \beta = 0.5, \gamma = 0$.

Para o conjunto de dados DBLP e CiteSeer, o modelo PSRB apresenta-se competitivo com os algoritmos do estado da arte. Obtendo os melhores *scores* à medida que a rate de dados positivos aumenta. No conjunto de dados DBLP a abordagem proposta é superior ao algoritmo GAE, mostrando a eficiência da utilização de *rewiring* sequencial. Para nenhum dos dois conjuntos de dados foi encontrado um valor de α e β para que o algoritmo RCSVM fosse capaz de inferir dados negativos.

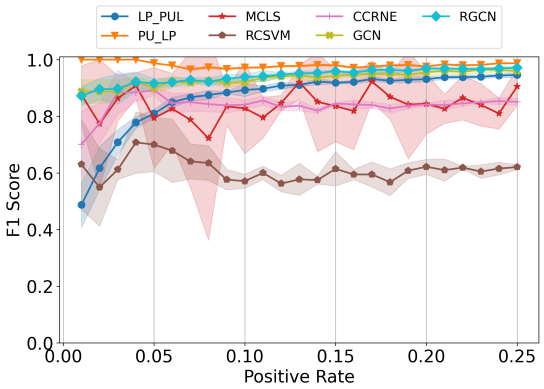


Figura 28 – F1 Score para o conjunto de dados Cora com os valores $\alpha = 0, \beta = 0.8, \gamma = 3$.

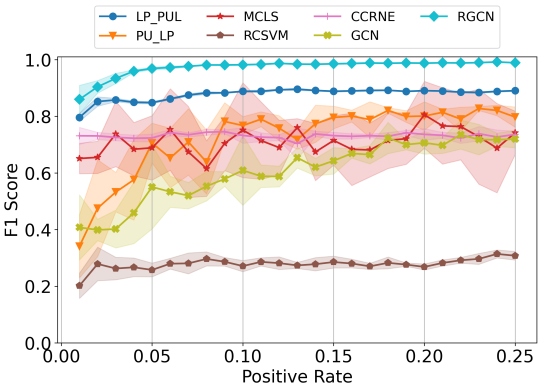


Figura 29 – F1 Score para o conjunto de dados PubMed com os valores $\alpha = 0, \beta = 0.4, \gamma = 3$.

No conjunto de dados Cora, o algoritmo MCLS apresenta alto desvio padrão, dada a dificuldade de inferir uma quantidade proporcional de dados negativos, visto que o al-

goritmo infere automaticamente a quantidade desses dados. Para o conjunto de dados PubMed, o algoritmo PSRB obteve um *score* superior aos algoritmos da literatura, principalmente quando comparado à abordagem GAE, mostrando a eficiência de *rewiring* em grafos menos densos.

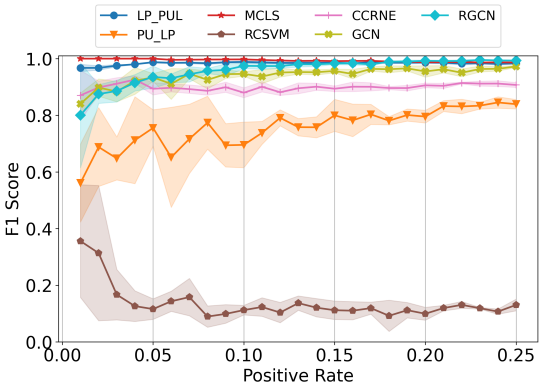


Figura 30 – F1 Score para o conjunto de dados Amazon Photo com os valores $\alpha = 0, \beta = 0.1, \gamma = 3$.

Para o conjunto de dados Amazon Photo a abordagem PSRB obteve os melhores *scores* junto aos algoritmos LP_PUL e MCLS a partir da *rate* 0.1. No geral, a abordagem proposta oferece uma flexibilidade de resultados expressivos em todos os conjuntos de dados, mostrando uma maior robustez à mudança dos dados. Além disso, no contexto da extração de dados negativos, o algoritmo PSRB demonstra facilidade na seleção dos parâmetros de probabilidade.

4.5 Avaliação de Desempenho na tarefa de PUL

Para avaliar a classificação dos exemplos, foram utilizados os mesmos conjuntos de dados. Os algoritmos foram testados para inferir uma quantidade igual de elementos negativos quanto positivos.

As Tabelas 5, 6, 7, 8, 9 ilustram os resultados obtidos para os conjuntos de dados Photo, DBLP, CiteSeer, Cora e PubMed respectivamente.

Tabela 5 – F1 Score médio para o conjunto de dados Amazon Photo

Model/Rate	**0.01**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**
CCRNE	0.4628 ± 0.01	0.5031 ± 0.03	0.4688 ± 0.02	0.4613 ± 0.04	0.4688 ± 0.05	0.4454 ± 0.02
LP_PUL	0.4538 ± 0.00	0.5106 ± 0.00	0.5054 ± 0.00	0.5162 ± 0.01	0.5263 ± 0.00	0.5195 ± 0.00
MCLS	0.4253 ± 0.00	0.4137 ± 0.00	0.401 ± 0.00	0.3896 ± 0.00	0.3767 ± 0.00	0.3651 ± 0.00
PU_LP	0.3655 ± 0.03	0.3732 ± 0.02	0.3805 ± 0.05	0.5092 ± 0.19	**0.7258 ± 0.11**	**0.7888 ± 0.03**
RCSVM	0.355 ± 0.02	0.3137 ± 0.01	0.2711 ± 0.03	0.2827 ± 0.02	0.2744 ± 0.02	0.2737 ± 0.01
GCN	0.4630 ± 0.03	0.4916 ± 0.03	0.4918 ± 0.02	0.4813 ± 0.01	0.4934 ± 0.00	0.4597 ± 0.02
PSRB + RGCN	**0.4923 ± 0.04**	**0.5623 ± 0.02**	**0.5749 ± 0.09**	**0.5849 ± 0.04**	0.6280 ± 0.05	0.6450 ± 0.08

Analisando os resultados obtidos para o conjunto de dados Amazon Photo (Tabela 5), observa-se que os melhores resultados estão divididos entre os modelos. O algoritmo

RCSVM obteve o menor *score* quando comparado à outras abordagens. O algoritmo PU_LP obteve ganho significativo com o aumento da rate quando comparado à menores valores. A abordagem proposta obteve os melhores resultados para as rates 0.05 0.10 e 0.15, crescendo de forma constante.

Tabela 6 – F1 Score médio para o conjunto de dados DBLP

Model/Rate	**0.01**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**
CCRNE	0.5591 ± 0.02	0.5854 ± 0.01	0.5841 ± 0.00	0.6042 ± 0.02	0.6076 ± 0.04	0.6609 ± 0.04
LP_PUL	**0.7111 ± 0.00**	**0.763 ± 0.02**	0.7702 ± 0.02	0.7786 ± 0.01	0.7786 ± 0.00	0.7875 ± 0.01
MCLS	0.6059 ± 0.01	0.5787 ± 0.03	0.5623 ± 0.03	0.543 ± 0.03	0.5179 ± 0.02	0.5088 ± 0.02
PU_LP	0.1547 ± 0.03	0.2673 ± 0.03	0.3278 ± 0.04	0.423 ± 0.02	0.6056 ± 0.03	0.6719 ± 0.02
GCN	0.6677 ± 0.01	0.7468 ± 0.02	0.7758 ± 0.02	0.7886 ± 0.02	0.7903 ± 0.01	0.7928 ± 0.00
PSRB + RGCN	0.6077 ± 0.02	0.7455 ± 0.05	**0.8521 ± 0.00**	**0.8406 ± 0.03**	**0.8617 ± 0.01**	**0.8661 ± 0.01**

Para o conjunto de dados DBLP (Tabela 6) o modelo proposto obteve os melhores *scores* para *rates* a partir de 0.10. Quando comparados os resultados obtidos pela GCN e pelo LP_PUL observa-se um ganho consideravel, mostrando a eficiência na utilização tanto da matriz de características quanto da matriz de adjacência. Para o algoritmo RCSVM executou-se um teste paramétrico e não foram encontrados parâmetros que obtivessem resultados válidos.

Tabela 7 – F1 Score médio para o conjunto de dados CiteSeer

Model/Rate	**0.01**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**
CCRNE	0.3454 ± 0.03	**0.4655 ± 0.04**	**0.5797 ± 0.03**	**0.5911 ± 0.02**	**0.593 ± 0.02**	**0.5718 ± 0.06**
LP_PUL	0.3293 ± 0.00	0.4076 ± 0.01	0.4214 ± 0.00	0.418 ± 0.00	0.4157 ± 0.00	0.4157 ± 0.00
MCLS	0.3075 ± 0.09	0.3666 ± 0.04	0.3694 ± 0.04	0.4323 ± 0.06	0.456 ± 0.03	0.4021 ± 0.0
PU_LP	0.0910 ± 0.03	0.1496 ± 0.02	0.2060 ± 0.01	0.2016 ± 0.01	0.2137 ± 0.03	0.2338 ± 0.02
GCN	**0.3825 ± 0.03**	0.3824 ± 0.00	0.3914 ± 0.01	0.3952 ± 0.01	0.3686 ± 0.00	0.3593 ± 0.01
PSRB + RGCN	0.3824 ± 0.01	0.4525 ± 0.06	0.5149 ± 0.07	0.5247 ± 0.05	0.5611 ± 0.04	0.5385 ± 0.05

Para o conjunto de dados CiteSeer (Tabela 7) o algoritmo CCRNE obteve melhores resultados. A abordagem proposta obteve os resultados mais próximos ao algoritmo CCRNE. Comparando os resultados obtidos com o modelo GCN, observa-se um ganho considerável, resultado da abordagem de *rewiring*. Em contrapartida, o algorimo PU_LP obteve *scores* menores, justificados pela quantidade de componentes conexas do conjunto de dados.

Tabela 8 – F1 Score médio para o conjunto de dados Cora

Model/Rate	**0.01**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**
CCRNE	0.3962 ± 0.06	0.5627 ± 0.02	0.5761 ± 0.03	0.5845 ± 0.02	0.6163 ± 0.03	0.6198 ± 0.03
LP_PUL	0.4688 ± 0.01	0.5036 ± 0.01	0.5422 ± 0.02	0.5829 ± 0.03	0.5997 ± 0.02	0.6103 ± 0.02
MCLS	0.4536 ± 0.01	0.446 ± 0.01	0.4404 ± 0.02	0.4082 ± 0.01	0.4061 ± 0.03	0.39 ± 0.03
PU_LP	**0.5739 ± 0.1**	**0.6801 ± 0.02**	0.6716 ± 0.02	0.6795 ± 0.02	0.6677 ± 0.01	0.639 ± 0.01
RCSVM	0.3723 ± 0.12	0.4034 ± 0.08	0.3689 ± 0.04	0.3167 ± 0.03	0.3344 ± 0.03	0.3126 ± 0.01
GCN	0.5045 ± 0.05	0.5886 ± 0.01	0.6104 ± 0.02	0.6298 ± 0.03	0.6247 ± 0.03	0.6341 ± 0.01
PSRB + RGCN	0.5627 ± 0.05	0.6609 ± 0.06	**0.7279 ± 0.03**	**0.7429 ± 0.03**	**0.7278 ± 0.04**	**0.7407 ± 0.01**

Para o conjunto de dados Cora (Tabela 8), o algoritmo PU_LP obteve melhores resultados para quantidade de dados rotulados até 0.05, enquanto o algoritmo proposto

obteve melhor desempenho para as *rates* restantes. O algoritmo proposto obteve crescimento constante à medida que a quantidade de dados rotulados aumentou.

Tabela 9 – F1 Score médio para o conjunto de dados PubMed

Model/Rate	**0.01**	**0.05**	**0.10**	**0.15**	**0.20**	**0.25**
CCRNE	**0.6255 ± 0.03**	0.7073 ± 0.01	0.7167 ± 0.01	0.7108 ± 0.03	0.6716 ± 0.07	0.6952 ± 0.02
LP_PUL	0.5778 ± 0.0	0.5821 ± 0.01	0.5867 ± 0.0	0.5849 ± 0.0	0.5845 ± 0.0	0.5808 ± 0.0
MCLS	0.5542 ± 0.01	0.5778 ± 0.06	0.5508 ± 0.06	0.5723 ± 0.03	0.5527 ± 0.05	0.5044 ± 0.03
PU_LP	0.3693 ± 0.02	0.4059 ± 0.03	0.5189 ± 0.07	0.6866 ± 0.04	0.7184 ± 0.0	0.7132 ± 0.03
RCSVM	0.4245 ± 0.01	0.3794 ± 0.01	0.3617 ± 0.01	0.3346 ± 0.01	0.3129 ± 0.0	0.2917 ± 0.01
GCN	0.5435 ± 0.01	0.534 ± 0.02	0.5422 ± 0.02	0.5184 ± 0.02	0.5313 ± 0.02	0.5109 ± 0.02
PSRB + RGCN	0.5978 ± 0.01	**0.7355 ± 0.01**	**0.7840 ± 0.01**	**0.7729 ± 0.01**	**0.7443 ± 0.01**	**0.7189 ± 0.01**

Para o conjunto de dados PubMed (Tabela 9) o algoritmo proposto obteve melhores resultados quando comparado ao modelo GCN. Esse resultado mostra a eficiência do algoritmo em conjuntos de dados de baixa densidade e com uma porcentagem maior de dados positivos (Tabela 3). O algoritmo CCRNE obteve resultados relevantes para *rates* 0.01, ao contrário do LP_PUL que obteve menores resultados em rates menores, mostrando a dificuldade em obter informações apenas da matriz de adjacência.

Capítulo 5

Conclusão

Nessa dissertação foi apresentado o modelo *Positive Sequential Rewiring via Brendth Search* (PSRB). O modelo, baseado em *two-step technique* para PUL, consiste na aplicação de uma estratégia de reescrita em grafos considerando somente ligações entre vértices positivos, aumentando a densidade entre esses vértices e facilitando a propagação dessas informações. Para lidar com os grafos criados pela estratégia de reescrita, desenvolvemos um modelo GNN que considera a informação de múltiplos grafos. A inferência dos dados negativos é feita por meio da distância euclidiana entre os exemplos mais distantes dos positivos no espaço de representação latente.

Para classificar o restante dos exemplos utilizamos um modelo MLP acoplado ao final do *encoder* pelo modelo PSRB. Apesar de obter os melhores resultados, o modelo não se limita a essa escolha de um classificador binário. Resultados obtidos em uma quantidade considerável de exemplos da literatura mostram a eficiência da abordagem proposta, confirmando a hipótese levantada.

Com os experimentos conclui-se que o algoritmo é capaz de aprender um *embedding* que separa os dados positivos dos negativos, como ilustrado no experimento da subseção 4.3.4.

5.1 Contribuição

O *framework* proposto é a contribuição principal desse trabalho. Essa contribuição é composta pelo PSRB e pela adaptação feita ao modelo RGCN. A abordagem PSRB contribui para a utilização de estratégias de rewiring em PUL e consequentemente com o avanço da utilização de reescrita em problemas de classificação.

Os experimentos realizados explicitam as principais dificuldades e vantagens das abordagens da literatura, além de mostrar as comparações do *framework* proposto com os al-

goritmos da literatura, tanto na inferência de dados negativos seguros quanto na tarefa de PUL. Além disso, a análise paramétrica auxilia na escolha dos parâmetros para conjuntos de dados distintos.

5.2 Limitação

As limitações do algoritmo proposto se resumem principalmente no ajuste dos parâmetros e no espaço de aplicação.

Sobre o tempo de execução, o algoritmo de reescrita apresenta complexidade linear $O(L * (V + E))$, facilitando a reescrita em grafos com muitos vértices e arestas, porém, limita o algoritmo ao espaço dos grafos não ponderados, visto que a busca em profundidade é aplicável apenas a esse caso.

O ajuste dos parâmetros também é uma limitação do algoritmo. Inicialmente foi feita a suposição de que conjuntos de dados com uma quantidade grande de exemplos necessitariam de parâmetros menores, principalmente sobre o parâmetro β. Porém, os testes paramétricos detectaram diferentes comportamentos para conjuntos de dados com números de arestas próximos, como o caso do conjunto PubMed e DBLP. Além disso, esses parâmetros são diretamente relacionados à quantidade de dados positivos disponíveis para treinamento.

O parâmetro α representa a importância da distância entre vértices positivos para adição da aresta. Esse parâmetro se mostra importante em grafos com muitas componentes conexas, como o caso do conjunto de dados CiteSeer. Qualquer valor diferente de zero resulta em uma importância dada a essa distância, não permitindo que vértices de diferentes componentes se conectem. Isso se dá pelo fato de que a distância entre dois vértices de componentes desconexas é infinita. A escolha desse parâmetro como zero em grafos desconexos resulta na não transferência de mensagem entre suas componentes. Esse fato mostra outra limitação da abordagem proposta.

Além da dificuldade de parametrização, o modelo não lida bem com conjuntos de dados com várias componentes conexas. Por outro lado, grafos com únicas ou poucas componentes conexas se beneficiam da estratégia de reescrita, facilitando inclusive na escolha do parâmetro α para próxima de zero.

5.3 Trabalhos Futuros

Apesar da dificuldade de lidar com esses parâmetros, é possível identificar que existe alguma relação entre esses valores e o conjunto de dados utilizado. Como trabalho futuro pretende-se encontrar alguma forma de fixar esses parâmetros, seja relacionando a quantidade de vértices e arestas do grafo, seja considerando a densidade dos vértices rotulados, vizinhança em comum, entre outras hipóteses.

A aplicação de reescrita dada pelo parâmetro L leva o modelo facilmente a um estado de *over-smoothing*, essa limitação pode ser contornada através de mecanismos de regularização. Originalmente o modelo de RGCN proposto apresenta uma forma de regularização para lidar com esse problema. Pretende-se, como trabalho futuro, aplicar regularização na abordagem proposta.

O modelo se mostra eficiente na separabilidade dos exemplos positivos e negativos, principalmente no conjunto de dados PubMed. Durante a realização dos testes paramétricos, foi considerada a hipótese de refazer a reescrita, agora considerando os vértices negativos. Pretende-se aprofundar esse modelo de *embedding* semi-supervisionado, considerando problemas multiclasse.

Considerando pesquisas realizadas pelo grupo, pretende-se aplicar o modelo desenvolvido nesse trabalho para a detecção de *fake news*, seja pela aplicação direta ou por utilização de grafos heterogêneos k-partidos. Outra possível adaptação da técnica proposta consiste na utilização de um comitê de classificação na segunda etapa. Esses trabalhos futuros mostram a importância do desenvolvimento dessa pesquisa para os interesses do grupo Interfaces e também do MIDAS.

Referências

ABBOUD, R.; DIMITROV, R.; CEYLAN İsmail İlkan. **Shortest Path Networks for Graph Property Prediction**. 2023.

AHMED, A. et al. Distributed large-scale natural graph factorization. In: **Proceedings of the 22nd international conference on World Wide Web**. [S.l.: s.n.], 2013. p. 37–48.

AKANSHA, S. **Over-Squashing in Graph Neural Networks: A Comprehensive survey**. 2024.

ALPAYDIN, E. **Introduction to Machine Learning**. 3. ed. Cambridge, MA: MIT Press, 2014. (Adaptive Computation and Machine Learning). ISBN 978-0-262-02818-9.

ARNAIZ-RODRIGUEZ, A. et al. **DiffWire: Inductive Graph Rewiring via the Lovász Bound**. 2022.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. **Autoencoders**. 2021.

BARBERO, F. et al. **Locality-Aware Graph-Rewiring in GNNs**. 2023.

_____. **Locality-Aware Graph-Rewiring in GNNs**. 2024.

BEKKER, J.; DAVIS, J. Learning from positive and unlabeled data: a survey. **Machine Learning**, Springer Science and Business Media LLC, v. 109, n. 4, p. 719–760, abr. 2020. Disponível em: <https://doi.org/10.1007/s10994-020-05877-5>.

BLACK, M. et al. **Understanding Oversquashing in GNNs through the Lens of Effective Resistance**. 2023.

BOJCHEVSKI, A.; GüNNEMANN, S. **Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking**. 2018.

BRüEL-GABRIELSSON, R.; YUROCHKIN, M.; SOLOMON, J. **Rewiring with Positional Encodings for Graph Neural Networks**. 2023.

CAI, C. et al. **On the Connection Between MPNN and Graph Transformer**. 2023.

CAI, H.; ZHENG, V. W.; CHANG, K. C.-C. **A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications**. 2018.

CARNEVALI, J. C. et al. A graph-based approach for positive and unlabeled learning. **Information Sciences**, v. 580, p. 655–672, 2021. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025521009130>.

CEN, K. et al. **ANAE: Learning Node Context Representation for Attributed Network Embedding**. 2020.

CHAUDHARI, S.; SHEVADE, S. Learning from positive and unlabelled examples using maximum margin clustering. In: HUANG, T. et al. (Ed.). **Neural Information Processing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 465–473. ISBN 978-3-642-34487-9.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische mathematik**, Springer, v. 1, n. 1, p. 269–271, 1959.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. **Machine learning**, Springer, v. 109, n. 2, p. 373–440, 2020.

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FEY, M.; LENSSEN, J. E. Fast graph representation learning with PyTorch Geometric. In: **ICLR Workshop on Representation Learning on Graphs and Manifolds**. [S.l.: s.n.], 2019.

_____. Fast graph representation learning with PyTorch Geometric. In: **ICLR Workshop on Representation Learning on Graphs and Manifolds**. [S.l.: s.n.], 2019.

GARCIA, J.; FERNÁNDEZ, F. A comprehensive survey on safe reinforcement learning. **Journal of Machine Learning Research**, v. 16, n. 1, p. 1437–1480, 2015.

GILMER, J. et al. **Neural Message Passing for Quantum Chemistry**. 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GROVER, A.; LESKOVEC, J. **node2vec: Scalable Feature Learning for Networks**. 2016.

GUO, Z. et al. Few-shot graph learning for molecular property prediction. In: **Proceedings of the Web Conference 2021**. New York, NY, USA: Association for Computing Machinery, 2021. (WWW '21), p. 2559–2567. ISBN 9781450383127. Disponível em: <https://doi.org/10.1145/3442381.3450112>.

GUTIÉRREZ, V. A. L. **Classificação semi-supervisionada baseada em desacordo por similaridade**. Tese (Doutorado) — Universidade de São Paulo, 2010.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). **Proceedings of the 7th Python in Science Conference**. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

JASKIE, K.; SPANIAS, A. Positive and unlabeled learning algorithms and applications: A survey. In: **2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)**. [S.l.: s.n.], 2019. p. 1–8.

KIPF, T. N.; WELLING, M. **Semi-Supervised Classification with Graph Convolutional Networks**. arXiv, 2016. Disponível em: <https://arxiv.org/abs/1609.02907>.

_____. Variational graph auto-encoders. **arXiv preprint arXiv:1611.07308**, 2016.

KOTSIANTIS, S. B. et al. Supervised machine learning: A review of classification techniques. **Emerging artificial intelligence applications in computer engineering**, Amsterdam, v. 160, n. 1, p. 3–24, 2007.

KREUZER, D. et al. **Rethinking Graph Transformers with Spectral Attention**. 2021.

LI, X.; LIU, B. Learning to classify texts using positive and unlabeled data. In: **Proceedings of the 18th International Joint Conference on Artificial Intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. (IJCAI'03), p. 587–592.

LIU, J. et al. **Web of Scholars: A Scholar Knowledge Graph**. 2022.

_____. **Shifu2: A Network Representation Learning Based Model for Advisor-advisee Relationship Mining**. 2020.

LIU, L.; PENG, T. Clustering-based method for positive and unlabeled text categorization enhanced by improved tfidf. **J. Inf. Sci. Eng.**, Citeseer, v. 30, n. 5, p. 1463–1481, 2014.

MA, S.; ZHANG, R. Pu-lp: A novel approach for positive and unlabeled learning by label propagation. In: **2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)**. [S.l.: s.n.], 2017. p. 537–542.

MAATEN, L. van der; HINTON, G. Visualizing data using t-SNE. **Journal of Machine Learning Research**, v. 9, p. 2579–2605, 2008. Disponível em: <http://www.jmlr.org/papers/v9/vandermaaten08a.html>.

MANEVITZ, L.; YOUSEF, M. One-class document classification via neural networks. **Neurocomputing**, Elsevier BV, v. 70, n. 7-9, p. 1466–1481, mar. 2007. Disponível em: <https://doi.org/10.1016/j.neucom.2006.05.013>.

MARUTHI, R. A. Classification of imbalanced datasets using one-class svm, k-nearest neighbors and cart algorithm. **International Journal of Advanced Computer Science and Applications**, v. 11, n. 11, 2020. Disponível em: <https://www.proquest.com/scholarly-journals/classification-imbalanced-datasets-using-one/docview/2655124572/se-2>.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 127–147, 1943.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1.

NEWMAN, M. E.; WATTS, D. J.; STROGATZ, S. H. Random graph models of social networks. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 99, n. suppl_1, p. 2566–2572, 2002.

NG, I. et al. **A Graph Autoencoder Approach to Causal Structure Learning**. 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PENG, T.; ZUO, W.; HE, F. SVM based adaptive learning method for text classification from positive and unlabeled documents. **Knowledge and Information Systems**, Springer Science and Business Media LLC, v. 16, n. 3, p. 281–301, set. 2007. Disponível em: <https://doi.org/10.1007/s10115-007-0107-1>.

PERERA, P.; OZA, P.; PATEL, V. M. **One-Class Classification: A Survey**. 2021.

RANI, M.; NAYAK, R.; VYAS, O. An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. **Knowledge-Based Systems**, v. 90, p. 33–48, 2015. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705115003779>.

ROBBINS, H.; MONRO, S. A stochastic approximation method. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 22, n. 3, p. 400–407, 1951. ISSN 00034851. Disponível em: <http://www.jstor.org/stable/2236626>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Springer Science and Business Media LLC, v. 323, n. 6088, p. 533–536, out. 1986. Disponível em: <https://doi.org/10.1038/323533a0>.

RUSCH, T. K.; BRONSTEIN, M. M.; MISHRA, S. A survey on oversmoothing in graph neural networks. **arXiv preprint arXiv:2303.10993**, 2023.

SCARSELLI, F. et al. The graph neural network model. **IEEE transactions on neural networks**, IEEE, v. 20, n. 1, p. 61–80, 2008.

SCHLICHTKRULL, M. et al. Modeling relational data with graph convolutional networks. In: SPRINGER. **The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15**. [S.l.], 2018. p. 593–607.

SEN, P. et al. Collective classification in network data. **AI Magazine**, Wiley, v. 29, n. 3, p. 93–106, set. 2008. Disponível em: <https://doi.org/10.1609/aimag.v29i3.2157>.

SHCHUR, O. et al. **Pitfalls of Graph Neural Network Evaluation**. 2019.

SHIN, S. Y.; KIM, H.-j. Autoencoder-based one-class classification technique for event prediction. In: **Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things**. New York, NY, USA: Association for Computing Machinery, 2019. (CCIOT '19), p. 54–58. ISBN 9781450372411. Disponível em: <https://doi.org/10.1145/3361821.3361831>.

SONG, Z. et al. **Graph-based Semi-supervised Learning: A Comprehensive Review**. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2102.13303>.

_____. Graph-based semi-supervised learning: A comprehensive review. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–21, 2022.

SOUTO, M. C. P. et al. Técnicas de aprendizado de máquina para problemas de biologia molecular. In: **Congresso da Sociedade Brasileira de Computação**. [S.l.]: SBC, 2003.

SOUZA, M. C. de et al. A network-based positive and unlabeled learning approach for fake news detection. **Machine Learning**, Springer Science and Business Media LLC, v. 111, n. 10, p. 3549–3592, nov. 2021. Disponível em: <https://doi.org/10.1007/s10994-021-06111-6>.

SPERDUTI, A.; STARITA, A. Supervised neural networks for the classification of structures. **IEEE Transactions on Neural Networks**, v. 8, n. 3, p. 714–735, 1997.

VELIČKOVIĆ, P. et al. Graph attention networks. **arXiv preprint arXiv:1710.10903**, 2017.

WU, L. et al. **Graph Neural Networks: Foundations, Frontiers, and Applications**. Singapore: Springer Singapore, 2022. 725 p.

WU, M. et al. Learning graph neural networks with positive and unlabeled nodes. **ACM Transactions on Knowledge Discovery from Data (TKDD)**, ACM New York, NY, v. 15, n. 6, p. 1–25, 2021.

_____. Learning graph neural networks with positive and unlabeled nodes. **ACM Trans. Knowl. Discov. Data**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 6, jun 2021. ISSN 1556-4681. Disponível em: <https://doi.org/10.1145/3450316>.

WU, Y.; IANAKIEV, K.; GOVINDARAJU, V. Improved k-nearest neighbor classification. **Pattern recognition**, Elsevier, v. 35, n. 10, p. 2311–2318, 2002.

WU, Z. et al. A comprehensive survey on graph neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 32, n. 1, p. 4–24, jan 2021. Disponível em: <https://doi.org/10.1109%2Ftnnls.2020.2978386>.

XIA, F. et al. Graph learning: A survey. **IEEE Transactions on Artificial Intelligence**, Institute of Electrical and Electronics Engineers (IEEE), v. 2, n. 2, p. 109–127, apr 2021. Disponível em: <https://doi.org/10.1109%2Ftai.2021.3076021>.

_____. Ranking station importance with human mobility patterns using subway network datasets. **IEEE Transactions on Intelligent Transportation Systems**, v. 21, n. 7, p. 2840–2852, 2020.

YANG, P. et al. Ensemble positive unlabeled learning for disease gene identification. **PLOS ONE**, Public Library of Science, v. 9, n. 5, p. 1–11, 05 2014. Disponível em: <https://doi.org/10.1371/journal.pone.0097079>.

_____. Positive-unlabeled learning for disease gene identification. **Bioinformatics**, v. 28, n. 20, p. 2640–2647, 08 2012. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/bts504>.

YANG, Z.; COHEN, W. W.; SALAKHUTDINOV, R. **Revisiting Semi-Supervised Learning with Graph Embeddings**. 2016.

YING, C. et al. **Do Transformers Really Perform Bad for Graph Representation?** 2021.

YOO, J. et al. Accurate graph-based pu learning without class prior. In: **2021 IEEE International Conference on Data Mining (ICDM)**. [S.l.: s.n.], 2021. p. 827–836.

YU, H.; HAN, J.; CHANG, K.-C. Pebl: Web page classification without negative examples. **IEEE Transactions on Knowledge and Data Engineering**, v. 16, n. 1, p. 70–81, 2004.

YU, S.; LI, C. PE-PUC: A graph based PU-learning approach for text classification. In: **Machine Learning and Data Mining in Pattern Recognition**. Springer Berlin Heidelberg, 2007. p. 574–584. Disponível em: <https://doi.org/10.1007/978-3-540-73499-4_43>.

ZHANG, D.; LEE, W. S. A simple probabilistic approach to learning from positive and unlabeled examples. In: **Proceedings of the 5th annual UK workshop on computational intelligence (UKCI)**. [S.l.: s.n.], 2005. p. 83–87.

ZHENG, Y. et al. DDI-PULearn: a positive-unlabeled learning method for large-scale prediction of drug-drug interactions. **BMC Bioinformatics**, Springer Science and Business Media LLC, v. 20, n. S19, dez. 2019. Disponível em: <https://doi.org/10.1186/s12859-019-3214-6>.

ZHOU, D. et al. Learning with local and global consistency. **Advances in neural information processing systems**, v. 16, 2003.

ZHU, X.; GOLDBERG, A. B. **Introduction to Semi-Supervised Learning**. Morgan and Claypool Publishers, 2009. (Synthesis Lectures on Artificial Intelligence and Machine Learning). Disponível em: <http://dx.doi.org/10.2200/S00196ED1V01Y200906AIM006>.

FUNDAÇÃO UNIVERSIDADE FEDERAL DE SÃO CARLOS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO - PPGCC/CCET

Rod. Washington Luís km 235 - SP-310, s/n - Bairro Monjolinho, São Carlos/SP, CEP 13565-905

Telefone: (16) 33518233 - http://www.ufscar.br

DECLARAÇÃO

Declaro, para os devidos fins, que a banca de defesa de dissertação de mestrado, conforme abaixo, foi realizada com a presença de todos os membros.

17 de julho de 2024	09h00min	
Candidato	Guilherme Henrique Messias	
Presidente	Alan Demétrius Baria Valejo	UFSCar
Titular	Murilo Coelho Naldi	UFSCar
Titular	Ricardo Marcacini	USP

O(a) aluno(a) está apto(a) agora a receber o título de “Mestre em Ciência da Computação”. A homologação do título e o respectivo processo de expedição e registro de diploma aguardam a finalização da versão final do texto da dissertação e de seu depósito no Repositório Institucional da UFSCar para se iniciarem.

Ivan Rogério da Silva

Secretaria do PPGCC"

Documento assinado eletronicamente por **Ivan Rogerio da Silva**, **Assistente em Administração**, em 23/07/2024, às 11:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufscar.br/autenticacao, informando o código verificador **1503938** e o código CRC **B564AF36**.

Referência: Caso responda a este documento, indicar expressamente o Processo nº 23112.020129/2024-58

SEI nº 1503938

Modelo de Documento: Declaração, versão de 02/Agosto/2019