

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Tamara Arruda Pereira

***Distill n' Explain:* explicando redes neurais para grafos usando
substituto simples**

Fortaleza – CE

15 de dezembro de 2022

Tamara Arruda Pereira

***Distill n' Explain: explicando redes neurais para grafos
usando substituto simples***

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Prof. Dr. Amauri H. de Souza Jr

Coorientador: Prof. Dr. Diego P. P. Mesquita

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

15 de dezembro de 2022

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- P436d Pereira, Tamara Arruda.
 Distill n' Explain: explicando redes neurais para grafos usando substituto simples / Tamara Arruda
 Pereira. - 2022.
 71 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
 Fortaleza, 2022.
 Orientação: Prof. Dr. Amauri Holanda de Souza Jr.
 Coorientação: Prof. Dr. Diego Parente Paiva Mesquita.
1. Explicabilidade. 2. Redes neurais para grafos. 3. Destilação de conhecimento. I. Título.
- CDD 004
-



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Tamara Arruda Pereira

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

Orientador: Prof. Dr. Amauri H. de Souza Jr
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Coorientador: Prof. Dr. Diego P. P. Mesquita
Fundação Getúlio Vargas (FGV)

Prof. Dr. Pedro Olmo Stancioli Vaz De Melo
Universidade Federal de Minas Gerais (UFMG)

Prof. Dr. Guilherme de Alencar Barreto
Universidade Federal do Ceará (UFC)

Prof. Dr. Ajalmar Rêgo da Rocha Neto
Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Fortaleza – CE
15 de dezembro de 2022

*Dedico este trabalho a todos que amo,
em especial, minha mãe,
um exemplo de dedicação.*

Agradecimentos

Durante todo o período de mestrado tive a oportunidade de adquirir conhecimento em novas áreas e tecnologias, além de aprender lições para a vida profissional e pessoal. Nesse contexto, tais conquistas e experiências não seriam concretizadas sem apoio das pessoas envolvidas nessa caminhada.

Gostaria de agradecer primeiramente a Deus por minha vida, saúde e sabedoria. A minha família, em especial a minha mãe Maria de Fátima, que me ensinou que a educação e o conhecimento transformam realidades, aproveito esse momento também para fazer um agradecimento especial a minha irmã, Mayara, que desde muito cedo sempre me incentivou nos estudos. Devo satisfação também a todos os meus amigos, em especial Amanda, Beth, Camila, Hannah, Juliele, Júnior, Luan, Samara, Savio e muitos outros que me ajudaram de alguma forma no meu desenvolvimento pessoal e entenderam minha ausência nesse período de estudo.

Iniciar no mundo da pesquisa é uma tarefa desafiadora que só é possível com a orientação adequada. Portanto, dedico meu agradecimento especial ao professor e orientador Dr. Amauri H. Souza Jr., por me aceitar como sua aluna de mestrado. O professor Amauri me apresentou a uma área de pesquisa extremamente interessante, à qual me dediquei durante todo o meu mestrado e espero continuar me dedicando durante muito tempo. Além disso, gostaria de agradecê-lo por todo tempo e paciência dedicado a mim nesse período de orientação, além de todos os conselhos que levarei para a vida.

Agradeço também ao meu coorientador Dr. Diego Mesquita, que junto ao Amauri usou do seu tempo para me orientar no desenvolvimento da pesquisa. Agradeço aos meus colegas de pesquisa Erik e Gabriel pela contribuição e ao tempo compartilhado. Além disso, agradeço a todo o corpo docente do Programa de Pós-Graduação em Ciência da Computação (PPGCC) do IFCE que contribuíram para a minha formação acadêmica nesse período. A FUNCAP pelo apoio financeiro. E por fim, e não menos importante, gostaria de agradecer também a todos os meus companheiros de turma do mestrado, em especial ao Fagner, no qual tive o privilégio de compartilhar experiências acadêmicas e pessoais.

“Se você tem conhecimento, deixe os outros acenderem suas velas nele.”
(Margaret Fuller)

Resumo

Redes neurais para grafos (*graph neural networks*, GNNs) consistem em generalizações de arquiteturas neurais clássicas (e.g., redes convolucionais) para dados estruturados na forma de grafos. Essas redes têm se tornado a abordagem mais promissora para aprendizagem de representações para grafos, levando a resultados notáveis em diversas aplicações, incluindo geração de novas drogas, sistema de recomendação, tradução automática, e geolocalização de usuário. Apesar desse sucesso, GNNs possuem baixa interpretabilidade, principalmente devido a sua estrutura complexa com altos números de parâmetros. Modelos geradores de explicações buscam minimizar o problema da falta de interpretabilidade através da identificação de subestruturas da entrada (i.e., grafo) mais relevantes para as predições do modelo a ser explicado.

Encontrar essas subestruturas geralmente implica em retropropagação através da GNN, vinculando a complexidade (por exemplo, número de camadas) da GNN ao custo de explicá-la. Isso naturalmente levanta a questão: É possível quebrar esse vínculo explicando uma GNN substituta mais simples? Para responder a pergunta, este trabalho apresenta o *Distill n' Explain* (DnX) um modelo explicador que extrai explicações através de uma GNN substituta. Primeiro, o DnX aprende um substituto da GNN via destilação de conhecimento. Em seguida, o DnX extrai explicações de nível de nó ou aresta resolvendo um problema convexo simples. Também propomos o FastDnX, uma versão mais rápida do DnX que aproveita a estrutura simples do modelo substituto e extrai explicações usando decomposição linear. Experimentos mostram que o DnX e o FastDnX geralmente superam os explicadores de GNN apesar de sua simplicidade. Além disso, são apresentados resultados teóricos que sustentam a hipótese de que boas explicações de um substituto também caracterizam boas explicações para a GNN que queremos explicar.

Palavras-chave: Explicabilidade; redes neurais para grafos; destilação de conhecimento.

Abstract

Graph neural networks (GNN) consist of generalizations of classical neural architectures (e.g., convolutional networks) for data structured in the form of graphs. These networks have become the most promising approach for learning representations for graphs, leading to remarkable results in several applications, including drug design, recommendation systems, machine translation, and user geolocation. Despite this success, GNNs have low interpretability, mainly due to their complex structure with high numbers of parameters. Models that generate explanations seek to minimize the problem of lack of interpretability through the identification of substructures of the input (i.e., graph) most relevant to the predictions of the model to be explained.

Finding these substructures often implies backpropagation through the GNN, tying the complexity (e.g., number of layers) of the GNN to the cost of explaining it. This naturally raises the question: Can we break this bond by explaining a simpler surrogate GNN? To answer the question, this work presents *Distill n' Explain* (DnX), an explanatory model that extracts explanations through a surrogate GNN. First, DnX learns a surrogate from GNN via knowledge distillation. Then DnX extracts node or edge level explanations by solving a simple convex problem. We also propose FastDnX, a faster version of DnX that takes advantage of the simple structure of the surrogate model and extracts explanations using linear decomposition. Experiments show that DnX and FastDnX often outperform state-of-the-art explainers despite their simplicity. Furthermore, we show theoretical results that support the hypothesis that good explanations of a surrogate also characterize good explanations for the GNN we want to explain.

Keywords: Explainability; graph neural networks; knowledge distillation.

Lista de ilustrações

Figura 1	– Exemplos de como os grafos podem representar diferentes estruturas como imagens e textos que possuem um formato regular, além de representar estruturas complexas (com diferentes números de vizinhos) como uma rede social.	29
Figura 2	– Ilustração de um grafo e suas respectivas matriz de atributos de nós e matriz de adjacência.	30
Figura 3	– Ilustração de uma GNN de duas camadas aplicada a um problema de classificação de nós.	31
Figura 4	– Ilustração do funcionamento do modelo de passagem de mensagem com 2 camadas.	32
Figura 5	– Diagrama ilustrando a tarefa de geração de explicações a nível de instância para a tarefa de classificação de nós.	36
Figura 6	– Ilustração da geração de explicação $\mathcal{G}_s$ do modelo GNNExplainer dado um grafo de entrada $\mathcal{G}$	37
Figura 7	– Arquitetura do modelo explicador PGM-Explainer.	39
Figura 8	– Matriz de confusão do processo de destilação do modelo GCN para o conjunto de dados BA-Community. Classe 1 corresponde aos nós base. O destilador Ψ classifica incorretamente muitos nós do <i>motif</i> , mas ele é capaz de prever corretamente quase todos os nós base.	50
Figura 9	– Ilustração mostra como cada conjunto de dados sintéticos são montados com seus <i>motifs</i> e nós base. Os rótulos dos nós são representados por cores. Além disso, para avaliar como cada explicador se comporta, são mostrados exemplos de explicações extraídas do GNNExplainer, PGMExplainer, PGExplainer, DnX e FastDnX para a mesma predição do modelo GCN em cada conjunto de dados.	52
Figura 10	– Comparação de tempo. Os gráficos de barras mostram o tempo médio que cada método leva para explicar uma predição do GCN. FastDnX é consistentemente o método mais rápido, muitas vezes por uma grande margem. Para os conjuntos de dados com maior grau médio (conjuntos de dados Bitcoin), o FastDnX é 4 ordens de magnitude mais rápido que o PGMExplainer e 2 ordens mais rápido que os outros métodos.	54
Figura 11	– Distribuição de graus dos nós do <i>motif</i> e base. Embora seja possível distinguir em geral os nós do <i>motif</i> e base das informações de grau em conjuntos de dados baseados em BA, há uma sobreposição significativa em Tree-Cycles and Tree-Grids.	57

Lista de tabelas

Tabela 1	– Detalhes dos conjuntos dados sintéticos usados nos experimentos. . . .	48
Tabela 2	– Performance (média e desvio padrão da acurácia) para tarefa de classificação de nós dos modelos a serem explicados.	49
Tabela 3	– Acurácia de destilação e tempo decorrido do destilador Ψ para os modelos GCN, ARMA, GATED e GIN. Para todos os casos, a acurácia de Ψ apresentou resultados maiores que 86% e o destilador consegue aprender em menos de um minuto.	50
Tabela 4	– Performance (acurácia) de explicação a nível de nó dos métodos explicadores (i.e., as explicações são dadas por um conjunto de nós). Os números Azuis e Verdes denotam o melhor e o segundo melhor método, respectivamente. A média e o desvio padrão são calculados após 10 rodadas independentes do processo de geração de explicação, a destilação não inclusa. Como as explicações do FastDnX são determinísticas o desvio padrão foi marcado com não aplicável (NA). Na maioria dos casos, FastDnX alcança o melhor desempenho.	51
Tabela 5	– Performance (média da precisão) para explicações a nível de nó dos conjuntos de dados do mundo real para os primeiros 500 nós. O DnX é o método com melhor desempenho e sua variante FastNnX é a segunda melhor em todos os experimentos.	53
Tabela 6	– Performance (média da precisão) de explicação a nível de nó para os conjunto de dados reais, buscando os nós apenas na vizinhança de 1-hop, mesmo protocolo utilizado por Vu e Thai (2020). Os explicadores DnX e FastDnX conseguem melhores resultados que todos os outros explicadores.	54
Tabela 7	– Performance (media e desvio padrão da AUC) para explicações a nível de arestas dos modelos explicadores para os conjuntos de dados sintéticos. Os números Azuis e Verdes denotam o melhor e o segundo melhor resultados respectivamente. O FastDnX é o melhor ou o segundo melhor explicador para todos os modelos e conjuntos de dados. Para GCN e GATED, o PGExplainer produz os melhores resultados. No geral, tanto o DnX quanto o FastDnX superam o GNNExplainer e o PGMEExplainer.	55
Tabela 8	– Performance (AUC) para explicações a nível de aresta dos conjuntos de dados reais para os primeiros 500 nós. O FastNnX é o método com melhor desempenho e o DnX é a segunda melhor em todos os experimentos.	56

Lista de abreviaturas e siglas

ML	<i>Machine learning.</i>
GDL	<i>Geometric deep learning.</i>
GNN	<i>Graph neural network.</i>
CNN	<i>Convolutional neural network.</i>
GCN	<i>Graph convolutional network.</i>
GIN	<i>Graph isomorphism network.</i>
WL	<i>Weisfeiler-Lehman.</i>
MLP	<i>Multilayer perceptron.</i>
SGC	<i>Simple graph convolution.</i>
MSE	<i>Mean squared error.</i>
ReLU	<i>Rectified linear unit.</i>
KL	<i>Kullback-Leibler.</i>
RHS	<i>Right hand side.</i>
LHS	<i>Left hand side.</i>
BA	<i>Barabási-Albert.</i>
GATED	<i>Gated graph sequence neural network.</i>
ARMA	<i>Graph neural network with convolutional ARMA filters.</i>
AP	<i>Average precision.</i>
AUC	<i>Area under the ROC curve.</i>
ROC	<i>Receiver operating characteristic.</i>

Lista de símbolos

$\in$	Pertence.
$\cup$	União.
$\mathbb{R}$	Conjunto dos número reais.
$\subseteq$	Contido em; é subconjunto de.
$\mathcal{G}$	Grafo.
V	Vértices do grafo.
A	Matriz de adjacência.
$\tilde{A}$	Matriz de adjacência normalizada.
X	Matriz de atributos.
D	Matriz de graus.
$\mathcal{N}(u)$	Conjunto dos vizinhos de u .
Φ	Redes neurais para grafos a ser explicado.
Ψ	Modelo destilador.
σ	Função de ativação.
Θ	Parâmetros aprendíveis.
$\mathcal{E}$	Vetor de pesos de importância.
$\mathcal{D}$	Conjunto de dados amostrados.
$\hat{Y}$	Matriz de predições de classes para todos os nós do grafo.
L	Número de camadas.
$\mathcal{K}$	Conjunto de perturbações.
$\mathcal{G}'$	Versão perturbada de $\mathcal{G}$.
γ	Constante.
γ_1	Constante.

γ_2 Constante.

δ Constante.

ϵ Ruído.

Sumário

1	INTRODUÇÃO	25
1.1	Motivação e visão geral	25
1.2	Objetivos gerais e específicos	27
1.3	Produção científica	27
1.3.1	Trabalhos publicados	27
1.3.2	Trabalhos submetidos	28
1.4	Estrutura da dissertação	28
2	REDES NEURAIS PARA GRAFOS	29
2.1	Passagem de mensagem	31
2.1.1	Redes convolucionais para grafos	32
2.1.2	Rede de isomorfismo de grafos	32
2.1.3	Convoluções simples para grafos	33
2.2	Conclusão	34
3	EXPLICADORES PARA REDES NEURAIS PARA GRAFOS	35
3.1	Abordagens para geração de explicações	35
3.1.1	GNNExplainer	36
3.1.2	PGExplainer	37
3.1.3	PGMExplainer	38
3.2	Conclusão	39
4	<i>DISTILL N' EXPLAIN</i>: EXPLICANDO REDES NEURAIS PARA GRAFOS USANDO SUBSTITUTO SIMPLES	41
4.1	Destilação de conhecimento	41
4.2	Obtendo explicações	42
4.2.1	Encontrando $\mathcal{E}$ via otimização	42
4.2.2	Encontrando $\mathcal{E}$ via decomposição linear	43
4.3	Análise teórica	43
4.4	Conclusão	46
5	EXPERIMENTOS	47
5.1	Conjuntos de dados	47
5.2	Baselines	48
5.3	Configurações dos experimentos	48
5.3.1	Explicações a nível de nós e arestas.	49

5.4	Resultados	49
5.5	Discussões	55
5.5.1	Os <i>benchmarks</i> são simples demais?	55
5.5.2	Limitações	57
5.6	Conclusão	57
6	CONCLUSÃO	59
	 REFERÊNCIAS	 61
	 APÊNDICES	 65
	APÊNDICE A – PROVAS MATEMÁTICAS	67

1 Introdução

1.1 Motivação e visão geral

Aprendizagem de máquina (*machine learning*, ML) é uma subárea da inteligência artificial que compreende técnicas estatísticas ou bioinspiradas para construção de modelos matemáticos capazes de lidar com incertezas a partir de dados (BISHOP, 2006). Aprendizagem de máquina, especialmente aprendizado profundo (*deep learning*), tem se tornado a principal abordagem para resolver problemas na área da inteligência artificial. Por exemplo, nos últimos anos, houve avanços significativos em aplicações como reconhecimento facial (SCHROFF; KALENICHENKO; PHILBIN, 2015; CAO et al., 2018), carros autônomos (CHEN et al., 2017; SALLAB et al., 2017) ou mesmo robôs que operam na compra e venda de ativos na bolsa de valores (XU; KONSTANTINIDIS; MANDIC, 2020).

No entanto, a maioria dos métodos de aprendizado de máquina podem sofrer problemas, tais como baixa interpretabilidade devido sua estrutura complexa com alto número de parâmetros. Além disso, modelos de ML são frequentemente associados a problemas de alto viés, ou seja, quando os algoritmos produzem resultados tendenciosos. Problemas como esses geram desconfiança, principalmente, em modelos que atuam em áreas críticas, como saúde, justiça e mercado financeiro. Com o intuito de suavizar essas desconfianças associadas aos modelos de aprendizado de máquina, tem-se notado nos últimos anos um interesse crescente em tópicos como métodos de seleção de características (STRUMBELJ; KONONENKO; ROBNIK-SIKONJA, 2009; HENNI; MEZGHANI; GOUIN-VALLERAND, 2018) e explicadores (FONG; VEDALDI, 2017).

Modelos tradicionais de aprendizado profundo normalmente são aplicados a problemas em que os dados assumem valores em domínios regulares, tais como imagens (*grid* 2D) ou texto (*grid* 1D). Estender esses modelos para dados irregulares ou não-euclidianos, como grafos, nem sempre é uma tarefa trivial, e esse objetivo deu origem a uma nova área chamada de *geometric deep learning* (GDL) (BRONSTEIN et al., 2017). Nesse contexto, as redes neurais para grafos (*graph neural networks*, GNNs) (SCARSELLI et al., 2009) surgiram e se destacam como modelos capazes de aprender representações e fornecer predições para dados estruturados na forma de grafos.

As GNNs típicas recorrem a técnica de passagens de mensagens (*message passing*) em grafos para extrair representações significativas de nós/grafos para a tarefa em questão. Apesar do sucesso das GNNs em muitos domínios, por terem surgido na era de modelos profundos e como uma tentativa de generalizar tais modelos (e.g., redes convolucionais) para grafos seu projeto arquitetônico geralmente resulta em modelos com interpretabilidade

limitada. Isso naturalmente dificulta o diagnóstico de cenários em que as GNNs são enganadas por variáveis confusas ou não se alinham com o conhecimento de especialistas. Para mitigar essa falta de interpretabilidade, uma estratégia popular é usar métodos explicadores *post-hoc*. A ideia é aumentar a transparência do modelo destacando elementos de entrada/modelo que são particularmente importantes para as predições, ajudando os usuários a entender o que está acontecendo nos bastidores.

Recentemente, houve um aumento de modelos propostos para explicar GNNs (YUAN et al., 2022). Embora as explicações para GNN possam vir em diferentes formas (YING et al., 2019; YUAN et al., 2021; WANG et al., 2021; LUCIC et al., 2022; YUAN et al., 2020a), elas geralmente assumem a forma de subestruturas (mínimas) dos grafos de entrada que são altamente influentes para a predição que queremos explicar. O modelo explicador GNNExplainer (YING et al., 2019) propõe aprender uma máscara *soft* para ponderar as arestas dos grafos. Para encontrar máscaras significativas, o GNNExplainer maximiza a informação mútua entre as predições da GNN dado o grafo original e o mascarado. Para aliviar o custo de otimizar novamente sempre que for explicar um nó diferente, o PGExplainer (LUO et al., 2020) propõe o uso de *embeddings* de nós para parametrizar as máscaras.

No entanto, o GNNExplainer e o PGExplainer impõem fortes suposições sobre o acesso a GNN que será explicada. O primeiro assume ser capaz de retropropagar através da GNN. O último assume ainda acesso as ativações ocultas da GNN. O PGMEExplainer (VU; THAI, 2020) suaviza essas suposições aproximando o comportamento local da GNN com um modelo gráfico probabilístico (PGM), que pode ser usado para classificar a relevância de nós e arestas. Por outro lado, obter explicações do PGMEExplainer envolve aprender a estrutura de um PGM que pode não ser bem escalável.

Este trabalho propõe *Distill n' Explain* (DnX). DnX usa a destilação de conhecimento para aprender uma simples GNN Ψ , por exemplo convolução de grafos simples (SGC) (WU et al., 2019), que imita o comportamento da GNN Φ que será explicada. Em seguida, o DnX resolve um problema convexo simples para encontrar uma máscara que pondere a influência de cada nó na saída de Ψ . Neste trabalho, foi adotado a mesma configuração caixa preta do PGMEExplainer, mas foi reduzido severamente o custo computacional extraindo explicações de um modelo substituto global. Além do DnX, é proposta a variante FastDnX que aproveita a natureza linear do modelo substituto para acelerar o procedimento de explicação. Notavelmente, Φ só é utilizado para aprender o substituto Ψ e, depois que Ψ for treinado, pode ser usado para explicar qualquer predição em nível de nó. Para apoiar a intuição de que explicar um substituto em vez da GNN original é uma ideia sensata, são fornecidos resultados teóricos ligando a qualidade da destilação à fidelidade das explicações geradas.

Experimentos em oito conjuntos de dados populares de classificação de nós mostram

que o DnX e o FastDnX geralmente superam GNN-, PG- e PGM-Explainers. Também é demonstrado que tanto o DnX quanto o FastDnX são muito mais rápidos que os concorrentes. Notavelmente, o FastDnX apresenta uma aceleração de até $62K\times$ em relação ao PGMExplainer.

1.2 Objetivos gerais e específicos

Esta dissertação tem como objetivo geral investigar alternativas que simplifiquem a geração de explicações para redes neurais de grafos. O uso de abordagens simplificadas podem levar a modelos computacionalmente mais eficientes e interpretáveis.

Os objetivos específicos deste trabalho são:

- Desenvolver um modelo mais simples de geração de explicações para GNNs que tratam as GNNs como modelos caixa-preta e que dependam de modelos substitutos simples obtidos através de destilação do conhecimento.
- Fornecer análises teóricas sobre a qualidade das explicações baseadas nesse substituto, vinculando o erro no procedimento de destilação à fidelidade da explicação.
- Avaliar o desempenho do modelo proposto através da comparação com outros explicadores para GNNs disponíveis na literatura em termos de qualidade de explicações e eficiência computacional.
- Verificar a relevância e adequação dos conjuntos de dados existentes para validação de explicadores para GNNs.

1.3 Produção científica

1.3.1 Trabalhos publicados

- **PEREIRA, T. A.**; NASCIMENTO, E. J. F.; MESQUITA, D.; SOUZA, A. H.. ConveXplainer for graph neural networks. IN: 11th Brazilian Conference on Intelligent Systems (BRACIS), 2022.
- DUARTE, G. J. S.; **PEREIRA, T. A.**; NASCIMENTO, E. J. F.; MESQUITA, D.; SOUZA, A. H.. How do loss functions impact the performance of graph neural networks?. IN: 15th Congresso Brasileiro de Inteligência Computacional (SBIC), 2021.

1.3.2 Trabalhos submetidos

- **PEREIRA, T. A.**; NASCIMENTO, E. J. F.; RESCK L.; MESQUITA, D.; SOUZA, A. H.; Distill n' Explain: explaining graph neural networks using simple surrogates. IN: 26th International Conference on Artificial Intelligence and Statistics (AISTATS), 2023.

1.4 Estrutura da dissertação

O restante do trabalho está organizado da seguinte forma: no Capítulo 2 é apresentada a fundamentação teórica das redes neurais para grafos (GNN). Além disso, são descritas as principais GNNs da literatura. Em seguida, no Capítulo 3, são discutidas algumas técnicas de geração de explicações para grafos, além de descrever os principais modelos explicadores da literatura: GNNExplainer, PGExplainer e PGMExplainer.

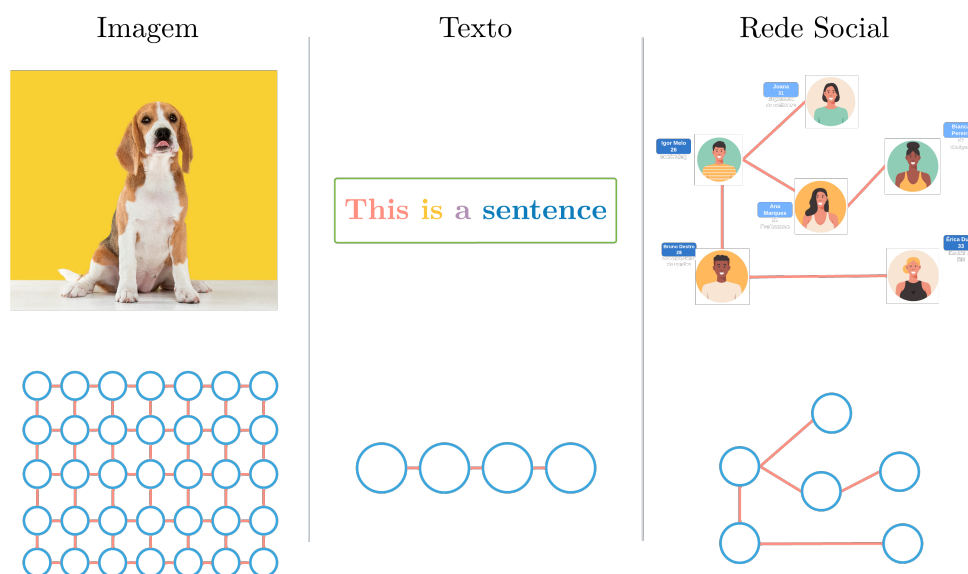
O modelo proposto DnX e sua variante FastDnX são apresentados no Capítulo 4. Também é fornecida a análise teórica sobre a qualidade das explicações. No Capítulo 5 são descritas as especificações das bases de dados e as configurações utilizadas para os experimentos, além disso, os resultados alcançados são apresentados e comparados com os explicadores do estado da arte (*state-of-the-art*, SOTA). Finalmente, as conclusões obtidas são descritas no Capítulo 6.

2 Redes neurais para grafos

Os grafos possuem uma estrutura natural capaz de representar um conjunto de objetos e suas relações. Na visão mais geral, um grafo é simplesmente uma coleção de objetos (nós), juntamente com um conjunto de interações (arestas) (HAMILTON, 2020). Os grafos possuem a capacidade descrever sistemas complexos, por exemplo, podem representar redes sociais, proteínas e suas interações biológicas ou iterações entre átomos e moléculas.

Dados em grafos são irregulares pois não possuem representação intrínseca no espaço euclidiano, ou seja, normalmente não podem ser representados através de sequências não esparsas de valores em uma ou duas dimensões, como imagens ou texto. Dados regulares como imagens possuem um mesmo tamanho e um padrão de conectividade regular, onde cada *pixel* é conectado com os seus oito *pixels* vizinhos, que podemos pensar como um grafo em grade de tamanho fixo. Já o texto ou fala são dados sequenciais, então podemos pensar neles como grafos em linha. No entanto, a maioria dos grafos não possuem uma estrutura fixa, além de ter um número variável de nós não ordenados, onde os nós podem ter diferentes quantidades de vizinhos. A Figura 1 mostra grafos representando estruturas mais simples (com número de vizinhos fixos) e mais complexas como redes sociais (com número variado de vizinhos).

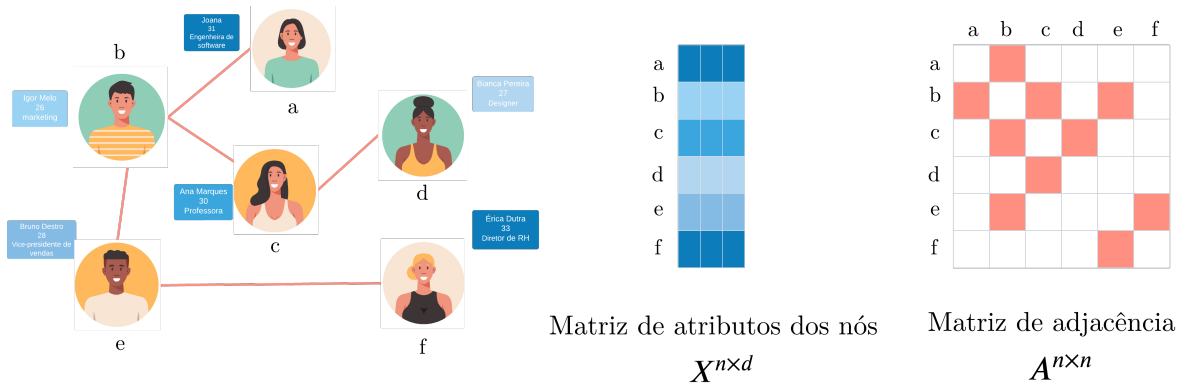
Figura 1 – Exemplos de como os grafos podem representar diferentes estruturas como imagens e textos que possuem um formato regular, além de representar estruturas complexas (com diferentes números de vizinhos) como uma rede social.



Fonte: Próprio autor.

Notação. Para uma descrição um pouco mais formal, o grafo $\mathcal{G} = (V, E)$ é definido por um conjunto de nós $V = \{1, \dots, n\}$ e um conjunto de arestas $E \subseteq V \times V$, esse relacionamento entre nós pode ser representado por uma matriz de adjacência $A \in \mathbb{R}^{n \times n}$, ou seja, A_{ij} é 1 se $(i, j) \in E$ e 0 caso contrário. Em muitos casos, também temos informações de atributos associados ao grafo, por exemplo, um nome associado a um usuário em uma rede social. Na maioria das vezes, esses são atributos de nível de nó que são representados usando uma matriz $X \in \mathbb{R}^{n \times d}$, como mostrado na Figura 2. Ao longo deste trabalho o grafo $\mathcal{G}$ também é representado por (A, X) . Além disso, define-se a matriz diagonal de graus D de $\mathcal{G}$ tal que $D_{ii} := \sum_j A_{ij}$. Finalmente, a matriz de adjacência normalizada com *self-loops* é denotada por $\tilde{A} = (D + I)^{-1/2}(A + I)(D + I)^{-1/2}$.

Figura 2 – Ilustração de um grafo e suas respectivas matriz de atributos de nós e matriz de adjacência.



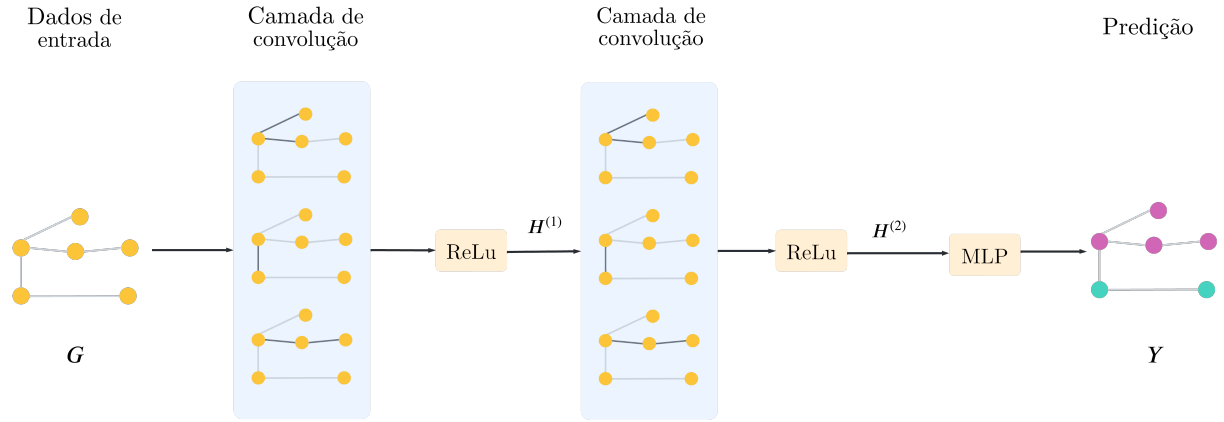
Fonte: Próprio autor.

Devido a estrutura irregular dos grafos, modelos de redes neurais tradicionais como redes neurais convolucionais (*convolutional neural network*, CNN) não são capazes de aprender representações para grafos, pois operações de convolução no domínio de grafos não são bem definidas. Mas como aprender representações de grafos de forma eficiente?

As redes neurais para grafos (*graph neural network*, GNN) têm atraído atenção ao longo dos últimos anos. As GNNs buscam preservar a estrutura do grafo e aprender representações dos nós, grafos e arestas através do processo de passagem de mensagens. Da mesma forma que as CNNs, uma GNN combina sequencialmente filtragem local, não linearidade e possivelmente operações de *pooling* para obter representações refinadas do grafo em cada camada.

Cada camada da GNN pode ser escrita como uma função, de modo que $H^{(\ell)} = f(H^{(\ell-1)}, A)$, com $H^{(0)} = X$ e $H^{(L)}$ é a saída após todas as L camadas. Os modelos específicos diferem apenas em como a função $f(\cdot)$ é definida. Após as L camadas, em um problema de predição de nó, um classificador é aplicado as representações aprendidas, desta forma $Y = \text{softmax}(MLP(H^{(L)}))$ onde MLP é uma rede neural multicamadas e $Y = y_{ic} \in \mathbb{R}^{n \times C}$ abrange predições de classe, de modo que y_{ic} denota a probabilidade do nó i pertencer à classe c , como ilustrado na Figura 3.

Figura 3 – Ilustração de uma GNN de duas camadas aplicada a um problema de classificação de nós.



Fonte: Próprio autor.

As GNNs podem ser aplicadas a uma grande diversidade de tarefas de modelagem como classificação de nós (HAMILTON; YING; LESKOVEC, 2017; VELIČKOVIĆ et al., 2018; KIPF; WELING, 2017), classificação de grafo (MA et al., 2020), previsão de arestas (ZHANG; CHEN, 2018) ou geração de grafos (KIPF; WELING, 2016; LIAO et al., 2019). Neste capítulo são descritas três principais GNNs: redes convolucionais para grafos (*graph convolutional networks*, GCNs) (KIPF; WELING, 2017), convoluções simples para grafos (*simple graph convolutions*, SGCs) (WU et al., 2019) e redes de isomorfismo em grafos (*graph isomorphism network*, GINs) (XU et al., 2019).

2.1 Passagem de mensagem

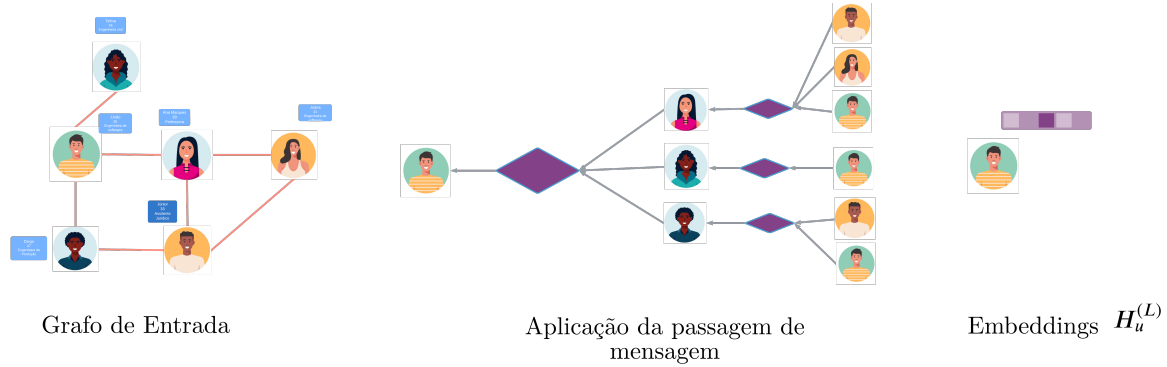
Uma definição simples para uma rede neural para grafos é que ela usa uma forma de passagem de mensagem na qual as mensagens vetoriais são trocadas entre os nós e atualizadas como redes neurais (HAMILTON, 2020). Em cada camada ℓ da passagem de mensagens em uma GNN, a matriz de representações (*embedding*) $H^{(\ell)} \in \mathbb{R}^{n \times d^{(\ell)}}$ é calculada, onde cada linha da matriz corresponde a $h_u^{(\ell)} \in \mathbb{R}^{1 \times d^{(\ell)}}$ que são as informações agregadas dos vizinhos de $u \in \mathcal{N}(u)$ e $d^{(\ell)}$ denota a dimensionalidade das características na ℓ -ésima camada. A função de atualização é definida na Equação 2.2.

$$H^{(\ell)} = \text{UPDATE}^{(\ell-1)}(H^{(\ell-1)}, \text{AGGREGATE}^{(\ell-1)}(A, H^{(\ell-1)})) \quad (2.1)$$

$$= \text{UPDATE}^{(\ell-1)}(H^{(\ell-1)}, M^{(\ell-1)}). \quad (2.2)$$

As funções UPDATE e AGGREGATE são funções diferenciáveis, $M^{(\ell)} = m_{\mathcal{N}(u)}^{(\ell)} \forall u \in V$ são as mensagens geradas a cada camada ℓ .

Figura 4 – Ilustração do funcionamento do modelo de passagem de mensagem com 2 camadas.



Fonte: Próprio autor.

A Figura 4 ilustra como a passagem de mensagem aplicada a um único nó funciona. A cada camada ℓ o modelo agrega mensagens dos seus vizinhos que, por sua vez, as mensagens provenientes desses vizinhos são baseadas em informações agregadas de suas respectivas vizinhanças e assim por diante. Para esse exemplo foi usado um modelo de passagem de mensagem de 2 camadas ($L = 2$).

2.1.1 Redes convolucionais para grafos

A rede convolucional para grafos (*graph convolutional network*, GCN) proposta por Kipf e Welling (2017) combina atributos de nós e a conectividade do grafo para aprender representações para grafos. Os autores desenvolveram um modelo que é a aproximação polinomial de primeira ordem da convolução espectral para grafo na forma de uma rede de passagem de mensagens, onde a informação é propagada ao longo dos nós vizinhos. A GCN usa apenas uma única matriz de parâmetros por camada, além disso, utiliza a matriz normalizada do grafo $\tilde{A}$, assim a função de atualização das representações $H^{(\ell)}$ é calculada recursivamente da seguinte forma:

$$H^{(\ell)} = \sigma \left(\tilde{A} H^{(\ell-1)} \Theta^{(\ell)} \right), \quad (2.3)$$

onde $H^{(0)} = X$ (isto é, a matriz de características de entrada) e $\sigma(\cdot)$ é uma função de ativação não linear, por exemplo, uma Unidade Linear Retificada (*Rectified Linear Unit*, ReLU).

2.1.2 Rede de isomorfismo de grafos

A rede de isomorfismo de grafos (*graph isomorphism network*, GIN) (XU et al., 2019) foi projetada para ter o mesmo poder de expressividade que o teste de isomorfismo de Weisfeiler-Lehman (WL) (WEISFEILER; LEHMAN, 1968).

O teste Weisfeiler-Lehman (WL) é um algoritmo clássico baseado em refinamento de cor. Cada nó mantém um estado (ou cor) que é atualizado em cada iteração agregando informações dos estados de seus vizinhos. Este processo não é garantido convergir, mas quando converge e as representações diferem entre dois grafos em qualquer estágio, pode-se concluir que os dois grafos são não isomórficos. A capacidade de WL de diferenciar grafos a partir de suas representações o torna mais poderoso do que muitas arquiteturas, incluindo GCNs. Pensando nisso surgiu o modelo GIN um novo agregador que produz diferentes representações (*embeddings*) ao lidar com grafos não isomórficos, para isso o modelo ajusta o peso do nó central por um parâmetro ϵ que pode ser aprendido ou fixado. A função de atualização é calculada da seguinte forma:

$$H^{(\ell)} = MLP \left((1 + \epsilon^{(\ell)}) \cdot H^{(\ell-1)} + AH^{(\ell-1)} \right), \quad (2.4)$$

onde $MLP(\cdot)$ representa um perceptron multicamada, $H^{(0)} = X$ (isto é, a matriz de atributos de entrada).

2.1.3 Convoluções simples para grafos

O modelo denominado convoluções simples para grafos (*simple graph convolution*, SGC) (WU et al., 2019) é uma simplificação da rede GCN. Conforme visto, a etapa de propagação em uma GCN na camada ℓ é calculada recursivamente da seguinte forma:

$$H^{(\ell)} = \sigma \left(\tilde{A}H^{(\ell-1)}\Theta^{(\ell)} \right), \quad (2.5)$$

$$= \sigma \left(\tilde{A} \sigma \left(\dots \sigma \tilde{A} \left(\tilde{A}X\Theta^{(1)} \right) \Theta^{(2)} \dots \right) \Theta^{(\ell)} \right), \quad (2.6)$$

onde a matriz $H^{(\ell)} \in \mathbb{R}^{n \times d^{(\ell)}}$ são as representações calculadas na ℓ -ésima camada. $\Theta^{(\ell)} \in \mathbb{R}^{d^{(\ell-1)} \times d^{(\ell)}}$ denota os parâmetros do modelo, $\sigma(\cdot)$ é uma função de ativação não linear e $\tilde{A}$ é a matriz normalizada definida anteriormente. Para reduzir o excesso de complexidade, o SGC remove as não linearidades entre as camadas da GCN, resultando em

$$H^{(\ell)} = \tilde{A} \dots \tilde{A} \tilde{A} X \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(\ell)}. \quad (2.7)$$

Para simplificar a notação as multiplicações repetidas da matriz de adjacência normalizada $\tilde{A}$ são substituídas por uma única matriz, elevando $\tilde{A}$ a ℓ -ésima potência $\tilde{A}^L$, além disso, os pesos são transformados em uma única matriz $\Theta = \Theta^{(1)}\Theta^{(2)} \dots \Theta^{(L)}$. Desta forma a função de atualização para calcular a matriz de representações do SGC é definida como:

$$H = \tilde{A}^L X \Theta. \quad (2.8)$$

O modelo SGC mostrou desempenho igual ou superior aos GCN para a tarefa de classificação de nós, sendo computacionalmente mais eficiente e ajustando menos parâmetros.

2.2 Conclusão

Neste capítulo foram apresentados os conceitos básicos de redes neurais para grafos. Dentre suas principais vantagens, as GNNs são capazes de aprender representações para grafos levando em consideração a sua estrutura, característica crucial para aprender representações de dados em grafos.

Foi discutida também a abordagem de passagens de mensagens que funciona na vizinhança local dos nós e captura as propriedades de um nó com base em seus vizinhos. Além disso, foram descritas três principais GNNs: as redes convolucionais para grafos (GCN) que sem dúvida é a GNN mais popular na literatura, convoluções simples para grafos (SGC), que consiste em uma GNN linear, e a rede de isomorfismo de grafos (GIN), que é baseada no teste de isomorfismo de Weisfeiler-Lehman.

3 Explicadores para redes neurais para grafos

Apesar do sucesso, a maioria das GNNs são modelos caixa-preta com um alto número de parâmetros, e naturalmente possuem baixa interpretabilidade. Uma solução para mitigar a baixa interpretabilidade de GNNs consiste em criar um novo modelo (explicador) capaz de identificar quais elementos da entrada são mais relevantes para a predição do modelo a ser explicado (GNN). Em muitos casos, essas explicações podem permitir que especialistas validem o modelo. Essa abordagem apresenta similaridades com técnicas clássicas de seleção de atributos em redes neurais (STRUMBELJ; KONONENKO; ROBNIK-SIKONJA, 2009) e com técnicas para identificação de mapas de saliência em visão computacional (ZEILER; FERGUS, 2013; VEDALDI, 2016). No contexto de grafos, além dos atributos de entrada, muitas vezes o interesse está em identificar subestruturas do grafo original que são mais relevantes para uma predição específica. Um exemplo disso é que normalmente membros de uma classe compartilham estruturas comuns. Na química, compostos mutagênicos costumam ter sequências químicas como NO_2 (um par de átomos de oxigênio junto com um átomo de nitrogênio).

Este capítulo fornece inicialmente uma visão geral das abordagens existentes para explicação de redes neurais para grafos. Em seguida, são discutidos métodos que compõem o estado da arte para explicação de GNNs, tais como GNNExplainer (YING et al., 2019), PGExplainer (LUO et al., 2020) e PGMEExplainer (VU; THAI, 2020).

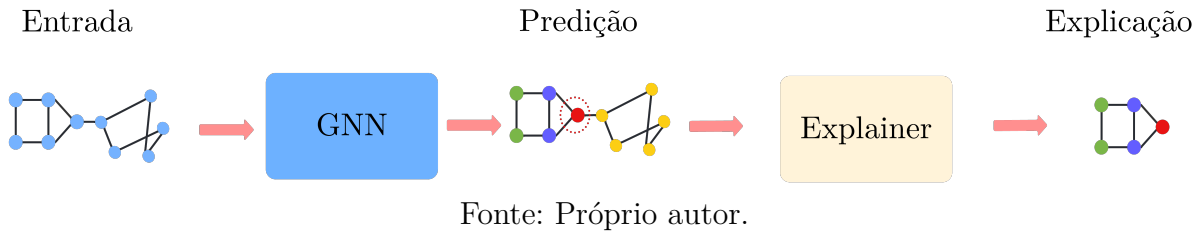
3.1 Abordagens para geração de explicações

Modelos explicadores podem gerar explicações a nível de instância (geram explicações para uma predição, ou seja, retorna as informações relevantes para uma determinada predição da GNN) (LUO et al., 2020; VU; THAI, 2020; YING et al., 2019) ou a nível de modelo (buscam padrões nos grafos que podem levar a um determinado comportamento da GNN) (YUAN et al., 2020b). Neste trabalho, nos concentramos em modelos explicadores a nível de instância.

Os modelos explicadores a nível de instância podem retornar aresta, nós, atributos ou subgrafos que são relevantes para a predição da GNN. Os explicadores podem tratar as GNNs como caixa-preta, desta forma só acessam as entradas e saídas do modelo a serem explicados (VU; THAI, 2020). Por outro lado, existem abordagens que precisam acessar os parâmetros ou representações ocultas para gerar explicações (LUO et al., 2020). Os modelos explicadores podem fornecer explicações para predições das tarefas de classificação de nó ou classificação de grafos.

A Figura 5 mostra o funcionamento geral para o processo de geração de explicações no nível de instância para a tarefa de classificação de nó. O grafo de entrada é passado para um modelo GNN treinado e então esse modelo retorna sua predição, ou seja, a classe que cada nó pertence (representado por diferentes cores na imagem). Para gerar a explicação para um determinado nó, o modelo explicador recebe o grafo de entrada e a predição a ser explicada, então o modelo explicador retorna apenas as informações relevantes para aquela predição.

Figura 5 – Diagrama ilustrando a tarefa de geração de explicações a nível de instância para a tarefa de classificação de nós.



Alguns exemplos de modelos que geram explicações em nível de instância são descritos a seguir.

3.1.1 GNNExplainer

O modelo GNNExplainer (YING et al., 2019) foi um dos primeiros explicadores de GNNs a ganhar significativa atenção. As explicações do GNNExplainer são encontradas através de um procedimento de otimização/busca aplicado independentemente a cada instância. O método é baseado em perturbações e gera explicações por meio do aprendizado de máscaras para arestas e atributos de nós que permitem remover informações que não impactam as predições da GNN.

Mais especificamente, a ideia combinar as máscaras de atributos e arestas com o grafo original $\mathcal{G}$, para encontrar um subgrafo $\mathcal{G}_s$ e o subconjunto de atributos dos nós X_s , onde o subgrafo $(\mathcal{G}_s, X_s)$ contém apenas informações importantes para a predição. Para isso, o GNNExplainer aprende as máscaras maximizando a informação mútua entre o possível subgrafo explicador $(\mathcal{G}_s, X_s)$ e a predição y da GNN, ou seja:

$$\max_{\mathcal{G}_s} \text{MI}(y, (\mathcal{G}_s, X_s)) = H(y) - H(y|\mathcal{G} = \mathcal{G}_s, X = X_s). \quad (3.1)$$

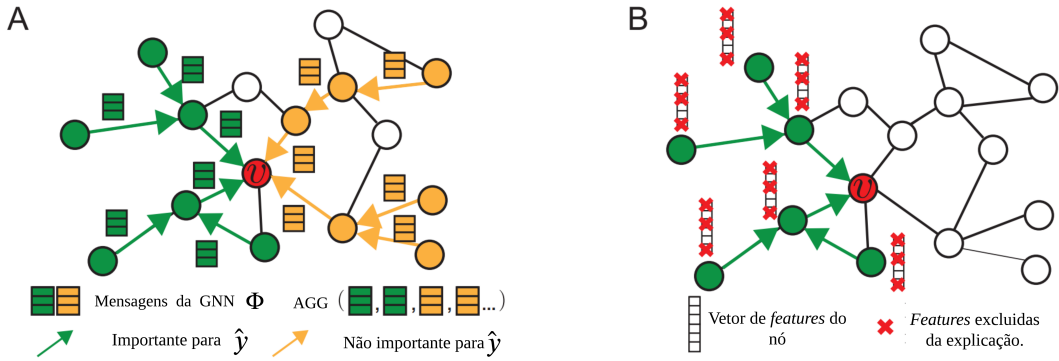
O termo de entropia $H(y)$ é constante, pois o modelo a ser explicado Φ é fixado após o treinamento. Assim, maximizar a informação mútua entre a distribuição de rótulos preditos y e a explicação $(\mathcal{G}_s, X_s)$ é equivalente a minimizar a entropia condicional $H(y|\mathcal{G} = \mathcal{G}_s, X = X_s)$, que pode ser expresso da seguinte forma:

$$H(y|\mathcal{G} = \mathcal{G}_s, X = X_s) = -\mathbb{E}_{y|\mathcal{G}_s, X_s}[\log P_\Phi(y|\mathcal{G} = \mathcal{G}_s, X = X_s)]. \quad (3.2)$$

Na prática a Equação 3.2 é um problema de otimização combinatória intratável. Para contornar essa limitação, Ying et al. (2019) relaxa o problema aprendendo uma máscara *soft* $M \in [0, 1]^{n \times n}$ que pondera as arestas do grafo de entrada $\mathcal{G}$.

A Figura 6 (A) mostra o grafo de entrada $\mathcal{G}$ (verde e amarelo) usado para fazer a predição do nó v . As arestas verdes em $\mathcal{G}_s$ são vias importantes de passagem de mensagens, que permitem que informações úteis do nó sejam propagadas através de $\mathcal{G}$ e agregadas em v para a predição, e as arestas amarelas são irrelevantes. O objetivo do GNNExplainer é identificar um pequeno conjunto de atributos e caminhos importantes (verde) que são cruciais para a predição. Já a Figura 6 (B) mostra o subgrafo $\mathcal{G}_s$ (verde), e os atributos de nós que são ou não relevantes para a predição da GNN que foram encontrados com o auxílio da máscara de atributos. Para encontrar as melhores máscaras o modelo maximiza a informação mútua entre a predição y do grafo original e a predição obtida com subgrafo $(\mathcal{G}_s, X_s)$.

Figura 6 – Ilustração da geração de explicação $\mathcal{G}_s$ do modelo GNNExplainer dado um grafo de entrada $\mathcal{G}$.



Fonte: Adaptada de Ying et al. (2019).

3.1.2 PGExplainer

O modelo PGExplainer (*parameterized explainer for graph neural network*) (LUO et al., 2020) adota uma rede neural profunda para parametrizar o processo de geração de explicações. Diferentemente do GNNExplainer que aprende máscaras tanto para arestas quanto para atributos de nó, o PGExplainer se concentra apenas na estrutura do grafo, para isso aprende máscaras discretas para arestas. O modelo recebe um grafo inicial $\mathcal{G}$ e busca um subgrafo $\mathcal{G}_s$ de modo que $\mathcal{G} = \mathcal{G}_s + \Delta\mathcal{G}$ onde $\mathcal{G}_s$ é o subgrafo que contém informações relevantes para a predição da GNN e $\Delta\mathcal{G}$ consiste nas arestas irrelevantes para a predição. Para a geração de explicações, o PGExplainer, assim como o GNNExplainer,

maximiza a informação mútua entre a predição y da GNN com o grafo de entrada $\mathcal{G}$ e a predição $\hat{y}(\mathcal{G}_s)$ da GNN com a entrada do subgrafo $\mathcal{G}_s$.

Devido à natureza discreta de $\mathcal{G}_s$, o PGExplainer relaxa os pesos das arestas de variáveis binárias para variáveis contínuas no intervalo $(0, 1)$ e adotam um truque de reparametrização para otimizar a função objetivo com métodos baseados em gradientes. Diferente do GNNExplainer que repete todo o processo de otimização para cada nó que quer explicar, o PGExplainer propõe explicar coletivamente as predições feitas por uma GNN, para isso, faz uso dos *embeddings* da GNN (é a matriz de representações de nós que codifica os atributos e a estrutura do grafo de entrada) para parametrizar as máscaras de arestas. Desta forma, o PGExplainer acessa representações ocultas da GNN para gerar explicações.

3.1.3 PGMEExplainer

O PGMEExplainer (VU; THAI, 2020) é um explicador baseado em modelos gráficos probabilísticos (*probabilistic graphical models*, PGMs). Mais especificamente, ele utiliza redes bayesianas para chegar a explicação de uma predição.

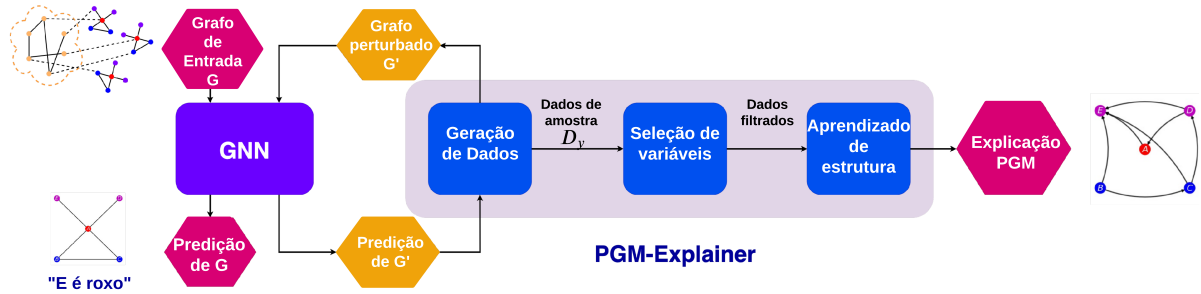
O PGM-Explainer consiste em três etapas principais: geração de dados, seleção de variáveis e aprendizado da estrutura. Dado um modelo GNN Φ e uma predição a ser explicada y , o objetivo da etapa de geração de dados no PGMEExplainer é gerar um conjunto de dados amostrados $\mathcal{D}_y$ a partir da função alvo Φ . Esse conjunto de dados amostrados $\mathcal{D}_y$ será utilizado nas etapas seguintes para aprender um PGM a partir dessas amostras. Como o explicador visa capturar os comportamentos de Φ , especialmente nas vizinhanças do grafo de entrada $\mathcal{G}$, o PGMEExplainer primeiro perturba algumas *features* de $\mathcal{G}$ para obter um conjunto de amostras perturbadas.

A etapa de aprendizado de estrutura nada mais é que a tarefa de aprender um PGM a partir de $\mathcal{D}_y$. Na prática, encontrar um PGM ideal a partir dos dados amostrados é intratável, visto que, para uma GNN típica, o conjunto de nós possíveis (L -hop vizinhos de y em $\mathcal{G}$) pode conter milhares de nós e a busca por uma rede bayesiana ótima é muito cara. Desta forma, na etapa de seleção de variáveis o PGMEExplainer corta ainda mais o conjunto de amostras. Para reduzir o número de variáveis em $\mathcal{D}_y$, o PGMEExplainer precisa identificar quais variáveis são importantes para evitar eliminá-las. Para isso, ele utiliza o algoritmo Grow-Shrink (GS) (MARGARITIS; THRUN, 1999). A etapa final do PGMEExplainer é aprender a explicação a partir de uma rede bayesiana de $\mathcal{D}_y$. O PGMEExplainer pode fornecer explicações tanto para tarefa de classificação de nós quanto para a de classificação de grafos.

A Figura 7 mostra as etapas de geração de explicação do PGMEExplainer, dado um grafo de entrada $\mathcal{G}$ e uma predição a ser explicada, o PGMEExplainer gera perturbações

nos atributos de $\mathcal{G}$ e registra as predições da GNN após essas perturbações. Ao repetir esses procedimentos várias vezes, um conjunto de amostras D_y é gerado e passado para etapa seguinte. Na etapa de seleção de variável, eliminam-se as amostras sem importância para a predição. Finalmente, na etapa de aprendizagem da estrutura, geram-se explicações a partir de uma rede bayesiana aprendida com base no conjunto de amostras retornado da etapa anterior.

Figura 7 – Arquitetura do modelo explicador PGM-Explainer.



Fonte: Adaptada de Vu e Thai (2020).

3.2 Conclusão

Devido às estruturas complexas e não interpretáveis da maioria das redes neurais para grafos, os modelos explicadores foram pensados com o propósito de distinguir quais informações do grafo são relevantes para as predições de uma GNN. Neste capítulo foram apresentadas as principais características dos modelos explicadores para GNNs, seus objetivos, as diferentes abordagens e os principais métodos existentes na literatura. Todos os métodos apresentados aqui são utilizados para comparação com o modelo proposto neste trabalho.

4 *Distill n' Explain*: explicando redes neurais para grafos usando substituto simples

Este capítulo apresenta as principais contribuições desta dissertação, um novo explicador para GNNs que gera explicações de maneira simples e rápida. O explicador compreende duas etapas: destilação de conhecimento e extração de explicação. Durante a etapa de destilação, uma GNN linear é usada para aproximar as predições da GNN a ser explicada. Assumindo que a aproximação é suficientemente boa, ao invés de explicar o modelo original, aplica-se um método para obter explicações das predições do modelo aproximador. Para obter explicações, explora-se a natureza linear do modelo aproximador e dois procedimentos simples foram propostos. O primeiro consiste em resolver um problema convexo. O segundo classifica os nós com base em uma simples decomposição de predições em termos aditivos.

Notação: Considere que Φ denota a GNN a ser explicada, de forma que Φ fornece predições para todos os nós do grafo $\mathcal{G}$, i.e., $\hat{Y}^{(\Phi)} = \Phi(\mathcal{G})$, onde $\hat{Y}^{(\Phi)}$ denota as predições de classe para todos os nós do grafo $\mathcal{G}$. Dessa forma, dada a GNN Φ e suas predições $\hat{Y}^{(\Phi)}$ a serem explicadas, o interesse está em encontrar um modelo explicador capaz de identificar as informações mais influentes para a predição $\hat{Y}_i^{(\Phi)}$ de Φ para o nó i . Neste trabalho, considera-se a GNN Φ um modelo caixa-preta, ou seja, o modelo explicador não tem acesso à nenhuma informação interna, como parâmetros ou representações ocultas do modelo, mas apenas à entrada e à saída da GNN.

4.1 Destilação de conhecimento

Apesar do sucesso dos modelos de aprendizado profundo, a maioria possui alta complexidade devido à necessidade de ajustar um grande número de parâmetros. Com isso em mente, surgiu a abordagem chamada de destilação do conhecimento (HINTON; VINYALS; DEAN, 2015), na qual o objetivo é destilar o conhecimento de uma rede complexa por meio de um modelo simples e possivelmente mais interpretável. O uso da destilação do conhecimento para modelos de redes neurais para grafos tem sido abordado em trabalhos recentes que validam a eficiência dessas estratégias (YANG et al., 2020; ZHOU et al., 2021).

Neste trabalho, é utilizado o modelo SGC (WU et al., 2019) para aproximar as predições obtidas com a GNN Φ a ser explicada. Conforme discutido no Capítulo 2, o SGC é um modelo linear, computacionalmente mais eficiente, interpretável e com menos parâmetros.

Formalmente, o modelo destilador (SGC), aqui denotado por Ψ , recebe o grafo de entrada $\mathcal{G} = (A, X)$ e fornece predições de classe $\hat{Y}^{(\Psi_\Theta)} = \text{softmax}(\tilde{A}^L X \Theta)$, onde Θ é a matriz de parâmetros do modelo SGC, L é o número de camadas da GNN Φ , e a função softmax atua nas linhas da matriz $\tilde{A}^L X \Theta$.

O processo de destilação consiste em ajustar os parâmetros Θ do modelo SGC Ψ para que ele aproxime as predições da rede a ser explicada Φ . Isso pode ser alcançado através da minimização da divergência Kullback-Leibler KL entre as predições de Φ e Ψ_Θ . Considere que $\hat{Y}_i^{(\Psi_\Theta)}$ e $\hat{Y}_i^{(\Phi)}$ denotam as predições de classe para o nó i dos modelos Ψ_Θ e Φ , respectivamente. O processo de destilação é então dado por

$$\min_{\Theta} \left\{ \text{KL} \left(\hat{Y}^{(\Phi)}, \hat{Y}^{(\Psi_\Theta)} \right) := \sum_{i \in V} \sum_c \hat{Y}_{ic}^{(\Phi)} \log \frac{\hat{Y}_{ic}^{(\Phi)}}{\hat{Y}_{ic}^{(\Psi_\Theta)}} \right\}, \quad (4.1)$$

que é equivalente à entropia cruzada entre $\hat{Y}^{(\Phi)}$ e $\hat{Y}^{(\Psi_\Theta)}$. Observe que minimizar essa perda não requer retropropagação por meio da GNN original Φ , apenas por meio do substituto Ψ . Também não é exigido nenhum conhecimento sobre a arquitetura do modelo Φ .

4.2 Obtendo explicações

Para obter uma explicação para uma determinada predição $\hat{Y}_i^{(\Psi_\Theta)}$, é necessário identificar um subgrafo de $\mathcal{G}$ contendo os nós que mais influenciam essa predição. Uma explicação $\mathcal{E}$ é definida como um vetor n -dimensional de pontuações de importância (maior significa mais relevante), um para cada nó no conjunto de vértices V . Duas estratégias para calcular $\mathcal{E}$ foram propostas neste trabalho.

4.2.1 Encontrando $\mathcal{E}$ via otimização

Para explicar Ψ como um substituto de Φ , o problema é formulado em encontrar a explicação $\mathcal{E}$ tratando-a como um vetor de pesos 0-1 e minimizando a norma L_2 ao quadrado entre os *logits* associados a $\hat{Y}_i^{(\Psi_\Theta)}$ e aqueles do grafo com atributos de nó mascarados por $\mathcal{E}$:

$$\min_{\mathcal{E} \in \{0,1\}^n} \| \tilde{A}_i^L \text{diag}(\mathcal{E}) X \Theta - \tilde{A}_i^L X \Theta \|_2^2, \quad (4.2)$$

onde $\tilde{A}_i^L$ denota a i -ésima linha da matriz $\tilde{A}^L$. Observe que a formulação em Equação 4.2 tem um grande problema: ela admite a solução trivial $\mathcal{E} = [1, 1, \dots, 1]$. Para contornar o problema e simultaneamente evitar a otimização binária, o espaço de busca $\{0, 1\}^n$ é substituído por $(n - 1)$ -simplex $\Delta = \{r \in \mathbb{R}^n : \sum_i r_i = 1, \forall_i r_i \geq 0\}$. Implementando essa mudança e reorganizando os cálculos, $\mathcal{E}$ é encontrado por:

$$\min_{\mathcal{E} \in \Delta} \left\| \tilde{A}_i^L (\text{diag}(\mathcal{E}) - I_n) X \Theta \right\|_2^2. \quad (4.3)$$

Como a camada de uma GNN convencional usa informações de primeira ordem (ou seja, a vizinhança imediata) para atualizar as representações dos nós, apenas os nós à distância $\leq L$ influenciam a predição uns dos outros em uma GNN com profundidade L . Portanto, $\mathcal{G}_{\mathcal{E}}$ deve ser um subgrafo do grafo induzido por i e sua L -vizinhança em $\mathcal{G}$. Por esta razão, os nós fora desta vizinhança são mascarados, mantendo sua importância em zero. Para facilitar a implementação, a Equação (4.3) no $\mathbb{R}^n$ irrestrito é resolvido mapeando vetores de $\mathbb{R}^n$ para Δ usando a função softmax.

Após a otimização de $\mathcal{E}$, os valores em $\mathcal{E}$ servem como um ranking para a importância de cada nó. Quando utilizado essa estratégia resolvendo a equação 4.3, o método é denominado como DnX.

4.2.2 Encontrando $\mathcal{E}$ via decomposição linear

Devido a natureza linear de Ψ é possível encontrar o vetor $\mathcal{E}$ simplesmente via decomposição linear. Seja Z_i o vetor *logit* associado à predição $\hat{Y}_i^{(\Psi_{\Theta})}$, pode-se decompor Z_i em uma soma de n termos, um para cada nó em V (mais o b):

$$\tilde{A}_{i1}^L X_1 \Theta + \tilde{A}_{i2}^L X_2 \Theta + \dots + \tilde{A}_{in}^L X_n \Theta + b = Z_i. \quad (4.4)$$

Portanto, pode-se medir a contribuição de cada nó para a predição como sua projeção escalar em $Z_i - b$:

$$\mathcal{E}_j := \tilde{A}_{ij}^L X_j \Theta (Z_i - b)^{\top} \quad (4.5)$$

Quando utilizado essa estratégia em vez de resolver a equação 4.3, o método é denominado como FastDnX.

4.3 Análise teórica

Nesta seção, são discutidos os aspectos teóricos e computacionais do método proposto. Primeiro, são fornecidos resultados teóricos que sustentam a hipótese de que boas explicações de um substituto global Ψ também caracterizam boas explicações de Φ — em termos de fidelidade. Em seguida, é discutida a convexidade do problema de otimização que o DnX resolve para extrair explicações. As provas estão dispostas no Apêndice A.

Seja $\mathcal{G}_u$ o subgrafo de $\mathcal{G}$ induzido pela vizinhança L -hop em torno do nó u . Uma explicação $\mathcal{E}_u$ para um nó u é fiel em relação a Φ se: i) Φ produz aproximadamente as mesmas predições para u independentemente de usar $\mathcal{E}_u$ para pesar os nós de $\mathcal{G}_u$ ou não; e ii) o mesmo vale para pequenas perturbações de $\mathcal{G}_u$. Uma perturbação $\mathcal{G}'_u$ de $\mathcal{G}_u$ pode ser definida adicionando ruído aos atributos de u ou religando aleatoriamente as

arestas incidentes do nó u (AGARWAL; ZITNIK; LAKKARAJU, 2022). Neste trabalho, considera-se perturbações sobre atributos de nós. Mais precisamente, isso implica que $V(\mathcal{G}'_u) = V(\mathcal{G}_u)$, $E(\mathcal{G}'_u) = E(\mathcal{G}_u)$, e que os atributos são corrompidos por ruído, ou seja, $X'_i = X_i + \epsilon_i$ para $i \in V(\mathcal{G}_u)$ e $\epsilon_i \in \mathbb{R}^d$.

Definição 1 (Fidelidade). *Dado um conjunto $\mathcal{K}$ de perturbações de $\mathcal{G}_u$, uma explicação $\mathcal{E}_u$ é fiel para um modelo f se*

$$\frac{1}{|\mathcal{K}| + 1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|f(\mathcal{G}'_u) - f(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \delta,$$

onde $\mathcal{G}'_u$ é uma versão possivelmente perturbada de $\mathcal{G}_u$, t é uma função que aplica a explicação $\mathcal{E}_u$ ao grafo $\mathcal{G}'_u$, e δ é uma pequena constante (AGARWAL; ZITNIK; LAKKARAJU, 2022).

Lema 1 fornece um limite superior para a fidelidade de $\mathcal{E}_u$ em relação ao modelo substituto Ψ . Teorema 1 estende este resultado para obter um limite para $\mathcal{E}_u$ em relação ao modelo que originalmente queremos explicar, ou seja, Φ .

Lema 1 (Fidelidade em relação a Ψ). *Dado um nó u e um conjunto $\mathcal{K}$ de perturbações, a fidelidade da explicação $\mathcal{E}_u$ em relação à predição $Y_u^{(\Psi\Theta)}$ do nó u é limitado da seguinte forma:*

$$\frac{1}{|\mathcal{K}| + 1} \sum_{\substack{\mathcal{G}'_u \in \\ \mathcal{K} \cup \{\mathcal{G}_u\}}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \gamma \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2,$$

onde $\mathcal{G}'_u$ é uma versão possivelmente perturbada de $\mathcal{G}_u$, t é uma função que aplica a explicação $\mathcal{E}_u$ ao grafo $\mathcal{G}'_u$, γ é uma constante que depende dos pesos do modelo Θ , atributos de nó X e perturbação ϵ . Além disso, $\Delta_{\mathcal{E}_u} \tilde{A}_u^L$ é a u -ésima linha da diferença da matriz de adjacência normalizada elevada a ℓ -ésima potencia $\tilde{A}^L$ antes e após aplicar a explicação $\mathcal{E}_u$.

Esboço da prova. Primeiro é mostrado que

$$\|\Psi(\mathcal{G}_u) - \Psi(t(\mathcal{G}_u, \mathcal{E}_u))\|_2 \leq \|X\Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2$$

usando a continuidade de Lipschitz da função softmax e a propriedade submultiplicativa da norma matricial L_2 . Repetindo para $\mathcal{G}'_u \in \mathcal{K}$, tomado a média em $\mathcal{K} \cup \{\mathcal{G}_u\}$ e isolando $\left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 = \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2$. A prova completa está disponível em Apêndice A. $\square$

Teorema 1 (Fidelidade em relação a Φ). *Sob as mesmas premissas do Lema 1 e assumindo que o erro de destilação L_2 é limitado por α , a fidelidade da explicação $\mathcal{E}_u$ para a predição do nó u do modelo original Φ é limitada da seguinte forma:*

$$\frac{1}{|\mathcal{K}| + 1} \sum_{\substack{\mathcal{G}'_u \in \\ \mathcal{K} \cup \{\mathcal{G}_u\}}} \|\Phi(\mathcal{G}'_u) - \Phi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \gamma \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 + 2\alpha.$$

Observe que Teorema 1 estabelece um limite de fidelidade que depende diretamente do erro de destilação α . Importante, quando Ψ é uma aproximação perfeita de Φ , recupera-se o limite superior no RHS do Lema 1.

Note que o Teorema 1 de Agarwal, Zitnik e Lakkaraju (2022) cobre um limite superior para a fidelidade dos métodos de explicação para GNN. No entanto, eles não cobrem o caso em que a explicação é um subconjunto ponderado de nós na vizinhança L -hop de u , como no método proposto.

Para completar, o Lema 1 e Teorema 1 são estendidos para explicar a natureza (muitas vezes) probabilística do ruído, ou seja, para o caso em que ϵ_i são variáveis aleatórias.

Lema 2 (Probabilidade vinculada à fidelidade *w.r.t.* Ψ). *Dado um nó u e um conjunto $\mathcal{K}$ de perturbações e assumindo que as perturbações são i.i.d. com distribuição $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, a fidelidade da explicação $\mathcal{E}_u$ em relação à predição $Y_u^{(\Psi_\Theta)}$ do nó u é limitado em probabilidade como segue:*

$$\mathbb{P} \left(\frac{1}{|\mathcal{K}| + 1} \sum_{\substack{\mathcal{G}'_u \in \\ \mathcal{K} \cup \{\mathcal{G}_u\}}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \xi \right) \geq F_{\chi^2_{|\mathcal{K}|nd}} \left(\frac{\xi - \gamma_1 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2}{\gamma_2 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 \sigma^2} \right)$$

onde γ_1 é uma constante que depende dos pesos do modelo Θ e características do nó X , γ_2 é uma constante que depende dos pesos do modelo Θ e $F_{\chi^2_{|\mathcal{K}|nd}}$ é a FDA de uma variável aleatória qui-quadrada com $|\mathcal{K}| \times n \times d$ graus de liberdade onde (n, d) são as dimensões de linha e coluna de X .

Teorema 2 (Probabilidade vinculada à fidelidade *w.r.t.* Φ). *Sob as mesmas suposições de Lema 2 e assumindo que o erro de destilação L_2 é limitado por α , a fidelidade da explicação $\mathcal{E}_u$ para a predição do nó u do modelo original Φ é limitada em probabilidade como segue:*

$$\mathbb{P} \left(\frac{1}{|\mathcal{K}| + 1} \sum_{\substack{\mathcal{G}'_u \in \\ \mathcal{K} \cup \{\mathcal{G}_u\}}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \xi \right) \geq F_{\chi^2_{|\mathcal{K}|nd}} \left(\frac{\xi - \gamma_1 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 - 2\alpha}{\gamma_2 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 \sigma^2} \right)$$

Para o Lema 2 e o Teorema 2, quando a variância σ^2 se aproxima de zero, ξ abandona sua natureza aleatória e a probabilidade no RHS converge para um. Note que os numeradores no RHS devem ser não negativos.

Lembre-se de que o *pipeline* do DnX/FastDnX envolve duas etapas: destilação do modelo (Equação 4.1) e extração da explicação (Equação 4.3). O primeiro é feito apenas uma vez para aprender o substituto Ψ . Este último, no entanto, deve ser executado para cada nó cuja predição deve ser explicada. Então, o custo da etapa de extração pode se tornar uma preocupação genuína do ponto de vista prático, especialmente para DnX, o

que implica resolver um problema de otimização repetidamente. Felizmente, o cenário de perdas do problema de extração depende apenas da forma de Ψ , e não da GNN original Φ como no GNNExplainer. Como Ψ é um SGC, a Equação 4.3 é um problema convexo (Teorema 3) os ótimos globais são alcançados usando, por exemplo, algoritmos baseados em gradiente.

Teorema 3 (Convexidade de DnX). *O problema de otimização de Equação 4.3 é convexo.*

4.4 Conclusão

Neste capítulo, a proposta de gerar explicações para predições de redes neurais para grafos de forma simples e rápida utilizando a técnica de destilação foi formalizada. Em seguida foi apresentado o método DnX — um modelo simples para geração de explicações para predições de GNNs que possui duas etapas. A primeira etapa consiste em destilar o conhecimento da GNN a ser explicada Φ , que normalmente é um modelo complexo, para um modelo linear Ψ . Já a segunda etapa busca aprender um vetor de importância $\mathcal{E}$ para os nós do grafo, com o intuito de distinguir as informações relevantes para as predições da GNN. Além disso, também foi apresentada uma variação do modelo DnX, que em vez de usar uma otimização para encontrar $\mathcal{E}$ aplica apenas uma decomposição linear, essa nova estratégia é chamada de FastDnX.

5 Experimentos

Neste capítulo são descritos os conjuntos de dados, as configurações dos experimentos e os resultados alcançados. Nos experimentos, são utilizados conjuntos de dados do mundo real e sintéticos, foram treinadas quatro GNNs com diferentes arquiteturas para o problema de classificação de nós. Mesmo o DnX e o FastDnx serem originalmente projetados para gerar explicações em nível de nó, para uma comparação justa com algumas linhas de base que geram somente explicações em nível de arestas foram feitas adaptações para os modelos gerarem explicações também a nível de arestas. Os resultados alcançados foram comparados com outros três explicadores GNNExplainer (YING et al., 2019), PGExplainer (LUO et al., 2020) e PGMEExplainer (VU; THAI, 2020). Todos os experimentos foram implementados usando PyTorch (PASZKE et al., 2019) e Torch Geometric (FEY; LENSSEN, 2019).

5.1 Conjuntos de dados

Dados sintéticos. Seis conjuntos de dados sintéticos amplamente usados para avaliar explicações de GNNs foram considerados para os experimentos: BA-House-Shapes, BA-Community, BA-Grids, Tree-Cycles, Tree-Grids e BA-Bottle-Shaped. Esses conjuntos de dados estão disponíveis em (YING et al., 2019) e (VU; THAI, 2020).

Cada conjunto de dados consiste em um único grafo composto por várias cópias do mesmo *motif* (i.e., subgrafo com um padrão estrutural específico) conectado a subgrafos base. Para os conjuntos de dados BA-House-Shapes, BA-Community, BA-Grids e BA-Bottle-Shaped os subgrafos base são gerados aleatoriamente usando o modelo Barabási-Albert (BA) (BARABÁSI; ALBERT, 1999). Para os conjuntos de dados Tree-Cycles e Tree-Grids, os subgrafos base são uma árvore binária balanceada de 8 níveis. A classe de cada nó do *motif* depende apenas dos elementos pertencentes ao mesmo *motif* e, consequentemente, a explicação associada a qualquer nó deve incluir apenas elementos do *motif*, ou seja, não pode incluir nós base. Portanto, os subgrafos base denotam informações irrelevantes para a predição de qualquer nó. Os detalhes dos conjuntos de dados sintéticos são apresentados na Tabela 1, incluindo número de classes, número de nós, e tipo dos atributos dos nós.

Dados reais. Além dos conjuntos de dados sintéticos foram utilizados dois conjuntos de dados reais nos experimentos: Bitcoin-Alpha e Bitcoin-OTC (KUMAR et al., 2016; KUMAR et al., 2018). Esses conjuntos denotam redes em que cada nó corresponde a uma conta de usuário que negocia Bitcoin e cada aresta (u, v) (entre os usuários u e v) denota a grau de confiabilidade atribuído por u a v . Com isso, cada aresta possui um *score*

Tabela 1 – Detalhes dos conjuntos dados sintéticos usados nos experimentos.

Conjunto de dados	Base	Motifs	Atributos dos nós	Classes
BA-House-Shapes	300-nós grafos BA	80× 5-nós formato de casa	Constante	4
BA-Community	350-nós grafos BA	100 5-nós formato de casa	Gerado a partir dos rótulos	8
BA-Grids	300-nós grafos BA	80× 9-nós formato de grade	Constante	2
Tree-Cycles	Árvore com altura 8	60 9-nós formato de ciclo	Constante	2
Tree-Grids	Árvore com altura 8	80 9-nós em formato de grade	Constante	2
BA-Bottle-Shaped	300-nós grafos BA	80× 5-nós formato de garrafa	Constante	4

associado, denotando o grau de confiabilidade. As redes Bitcoin-Alpha e Bitcoin-OTC possuem 3783 e 5881 contas, respectivamente. Os membros da plataforma classificam os outros membros em uma escala de -10 (desconfiança total) a +10 (confiança total). Cada conta é rotulada como confiável ou não confiável com base nas classificações dos outros membros. Além disso, as contas possuem atributos que capturam informações de saída da conta, como a taxa média ou o número normalizado de votos que a conta fez. As explicações alvo para cada nó são fornecidas por especialistas.

5.2 Baselines

Os modelos DnX e FastDnX são comparados com três explicadores: GNNExplainer (YING et al., 2019), PGMEExplainer (VU; THAI, 2020) e PGExplainer (LUO et al., 2020). Para garantir uma comparação válida, as configurações dos experimentos deste trabalho seguem de perto as diretrizes e a configuração de avaliação dos trabalhos originais. Assim como os explicadores GNNExplainer, PGMEExplainer e PGExplainer foi gerado explicações para uma GCN (KIPF; WELLING, 2017) de 3 camadas com a função de ativação *ReLU*. Mas além da GCN também foram considerados três arquiteturas adicionais: *Graph isomorphism network* (GIN) (XU et al., 2019), *Gated graph sequence neural networks* (GATED) (LI et al., 2016) e *GNN with convolutional ARMA filters* (ARMA) (BIANCHI et al., 2022). Isso permite avaliar a robustez e o desempenho dos explicadores em GNNs de diferentes complexidades.

5.3 Configurações dos experimentos

Os conjuntos de dados são divididos em treinamento/validação/teste com porcentagem de 80/10/10%, respectivamente. Todas as GNNs possuem 3 camadas e são treinadas por 1000 épocas usando a parada antecipada se a acurácia de validação não melhorar em 100 épocas consecutivas. Para o treinamento de todas as GNNs foi usado o otimizador Adam (KINGMA; BA, 2015) com uma taxa de aprendizado de 0,01 com *weight decay* de $5,0 \times 10^{-4}$.

Para a fase de destilação do DnX, foi utilizado o modelo SGC (WU et al., 2019) com 3 camadas. Durante a destilação do modelo, as predições de todos os nós são usadas para

que o modelo do destilador se ajuste melhor ao modelo a ser explicado. Para a otimização, foi usado o AdamW (LOSHCHILOV; HUTTER, 2019) com uma taxa de aprendizado de 0,1 com *weight decay* de $5,0 \times 10^{-6}$ e 10000 épocas. Para os conjuntos de dados sintéticos, que não contam com atributos de nó, o SGC foi treinado com vetor one-hot do grau como atributos de nós.

Todas as implementações foram desenvolvidas usando as bibliotecas PyTorch (PASZKE et al., 2019) e Torch Geometric (FEY; LENSSEN, 2019). Todos os experimentos foram executados usando um computador com CPU Intel Xeon 2,20 GHz e 13 GB de RAM DDR5.

5.3.1 Explicações a nível de nós e arestas.

O DnX e o FastDnX foram originalmente projetados para gerar explicações em nível de nó como PGMEExplainer, mas algumas linhas de base, como PGExplainer e GNNExplainer, fornecem explicações em nível de arestas. Para uma comparação justa com essas linhas de base, foi utilizado um procedimento para converter explicações de nível de arestas para explicações de nível de nó (e vice-versa). Para converter as pontuações de nós em pontuações de aresta, as pontuações dos terminais de cada aresta são somados, ou seja, uma aresta (u, v) obtém pontuação $s_{u,v} = s_u + s_v$ onde s_u e s_v são pontuações de nós. Para nós, dizemos $s_u = |\mathcal{N}_u|^{-1} \sum_{j \in \mathcal{N}_u} s_{u,j}$ onde $\mathcal{N}_u$ é a vizinhança de u .

5.4 Resultados

Nesta seção, é mostrado o desempenho dos modelos no qual queremos gerar explicações, em seguida são fornecidos os resultados do modelo destilador Ψ para todas as GNNs. Por fim, são apresentados os resultados de explicação para nível de nó e aresta dos métodos propostos que são comparados com os resultados dos principais explicadores da literatura.

Resultados de classificação. O desempenho das GNNs que queremos explicar para os conjuntos de dados sintético e reais é mostrado na Tabela 2. A Tabela 2 reporta a média e o desvio padrão calculados após 10 rodadas independentes. É importante destacar que a acurácia alcançou valores $\geq 94\%$ para a maioria das combinações de dados/modelos.

Tabela 2 – Performance (média e desvio padrão da acurácia) para tarefa de classificação de nós dos modelos a serem explicados.

	BA-House	BA-Community	BA-Grids	Tree-Cycles	Tree-Grids	BA-Bottle	Bitcoin-Alpha	Bitcoin-OTC
GCN	97.9 $\pm$ 1.6	85.6 $\pm$ 1.7	99.9 $\pm$ 0.2	97.8 $\pm$ 1.2	88.9 $\pm$ 1.8	99.0 $\pm$ 0.1	93.3 $\pm$ 0.4	93.2 $\pm$ 0.6
ARMA	98.1 $\pm$ 2.3	92.8 $\pm$ 2.3	99.5 $\pm$ 0.6	96.9 $\pm$ 1.5	92.4 $\pm$ 2.3	99.6 $\pm$ 0.9	93.6 $\pm$ 1.4	92.9 $\pm$ 1.1
GATED	98.0 $\pm$ 1.3	92.3 $\pm$ 2.7	99.9 $\pm$ 0.2	97.8 $\pm$ 2.2	94.4 $\pm$ 3.0	99.4 $\pm$ 0.6	94.4 $\pm$ 1.3	93.8 $\pm$ 0.7
GIN	95.6 $\pm$ 5.0	87.0 $\pm$ 1.5	99.4 $\pm$ 0.6	97.8 $\pm$ 1.1	91.4 $\pm$ 2.6	98.8 $\pm$ 1.0	93.4 $\pm$ 1.0	92.6 $\pm$ 1.1

Resultados da destilação. Os resultados alcançados com o modelo linear Ψ ao destilar os modelos a serem explicados GCN, ARMA, GATED e GIN para todos os conjuntos de dados são mostrados na Tabela 3. A métrica de avaliação utilizada foi a acurácia entre o modelo a ser predito e o modelo destilador. A Tabela 3 reporta a média e o desvio padrão calculados após 10 rodadas independentes. De modo geral, os resultados mostram que o destilador Ψ consegue se aproximar bem do modelo a ser explicado Φ , visto que alcança uma acurácia superior a 86% em todos os conjuntos de dados. A Tabela 3 também mostra o tempo consumido na etapa de destilação.

Tabela 3 – Acurácia de destilação e tempo decorrido do destilador Ψ para os modelos GCN, ARMA, GATED e GIN. Para todos os casos, a acurácia de Ψ apresentou resultados maiores que 86% e o destilador consegue aprender em menos de um minuto.

Dataset	GCN		ARMA		GATED		GIN	
	Acurácia	Tempo (s)	Acurácia	Tempo (s)	Acurácia	Tempo (s)	Acurácia	Tempo (s)
BA-House	94.2 $\pm$ 1.2	13.996	97.7 $\pm$ 0.01	7.254	98.0 $\pm$ 0.01	5.954	93.4 $\pm$ 0.46	11.461
BA-Community	86.6 $\pm$ 0.1	16.447	96.7 $\pm$ 0.03	22.848	92.1 $\pm$ 0.04	27.847	90.1 $\pm$ 0.13	24.204
BA-Grids	99.9 $\pm$ 0.1	2.721	100.0 $\pm$ 0.00	2.701	100.0 $\pm$ 0.00	2.679	100.0 $\pm$ 0.00	3.884
Tree-Cycles	97.7 $\pm$ 0.2	3.820	98.8 $\pm$ 0.03	6.331	100.0 $\pm$ 0.00	2.679	95.8 $\pm$ 0.02	3.335
Tree-Grids	98.0 $\pm$ 0.2	3.803	91.7 $\pm$ 0.15	4.152	99.4 $\pm$ 0.04	6.643	88.1 $\pm$ 0.06	5.198
BA-Bottle	98.5 $\pm$ 0.2	3.181	100.0 $\pm$ 0.00	6.312	100.0 $\pm$ 0.00	6.171	100.0 $\pm$ 0.01	6.970
Bitcoin-Alpha	90.4 $\pm$ 0.1	28.317	91.3 $\pm$ 0.01	28.613	91.3 $\pm$ 0.02	26.611	95.5 $\pm$ 0.03	44.359
Bitcoin-OTC	89.1 $\pm$ 0.2	32.414	91.2 $\pm$ 0.03	39.832	91.1 $\pm$ 0.01	42.626	89.8 $\pm$ 2.04	55.860

Figura 8 – Matriz de confusão do processo de destilação do modelo GCN para o conjunto de dados BA-Community. Classe 1 corresponde aos nós base. O destilador Ψ classifica incorretamente muitos nós do *motif*, mas ele é capaz de prever corretamente quase todos os nós base.

BA-Community

True labels	1	300	0	0	0	0	0	1	0
	2	0	149	15	0	0	0	0	0
	3	0	12	155	0	1	0	0	0
	4	0	6	58	3	0	1	0	0
	5	0	0	0	0	301	0	0	0
	6	0	3	0	0	0	154	10	0
	7	0	0	3	0	0	14	128	3
	8	0	0	0	0	0	9	58	16
		1	2	3	4	5	6	7	8
Predicted labels									

Curiosamente, embora BA-Community seja o conjunto de dados com a menor precisão de destilação (86,6%) quando Ψ destila a rede GCN, os explicadores DnX e FastDnX alcançam resultados significativamente melhores do que o estado da arte anterior (como podemos observar na Tabela 4). A justificativa para esses resultados contra-intuitivos

é que o destilador pode diferenciar entre nós do *motif* e nós base, e isso é suficiente para obter boas explicações - já que o conjunto de avaliação compreende apenas nós do *motif*. Mais concretamente, a matriz de confusão em Figura 8 revela que, apesar da baixa precisão da destilação, o modelo substituto Ψ prevê corretamente os nós base (classes 1 e 5). Portanto, Ψ atinge alta precisão para o problema de classificação binária de distinguir nós do *motif* e base, apoiando nossa hipótese.

Resultados de explicação a nível de nó. Os resultados para a geração de explicações dos conjuntos de dados sintéticos são apresentados na Tabela 4. A métrica utilizada foi a acurácia, pois as explicações verdade para esses conjuntos de dados são claras. Os resultados do DnX e FastDnX são comparados com os resultados dos explicadores GNNExplainer (YING et al., 2019), PGMEExplainer (VU; THAI, 2020) e PGExplainer (LUO et al., 2020). As explicações são geradas para todos os nós dos *motifs* no grafo de entrada. Todos os explicadores são programados para retornar os n principais nós importantes para a predição de destino, onde n é o número de nós no *motif* do conjunto de dados.

Tabela 4 – Performance (acurácia) de explicação a nível de nó dos métodos explicadores (i.e., as explicações são dadas por um conjunto de nós). Os números **Azuis** e **Verdes** denotam o melhor e o segundo melhor método, respectivamente. A média e o desvio padrão são calculados após 10 rodadas independentes do processo de geração de explicação, a destilação não inclusa. Como as explicações do FastDnX são determinísticas o desvio padrão foi marcado com não aplicável (NA). Na maioria dos casos, FastDnX alcança o melhor desempenho.

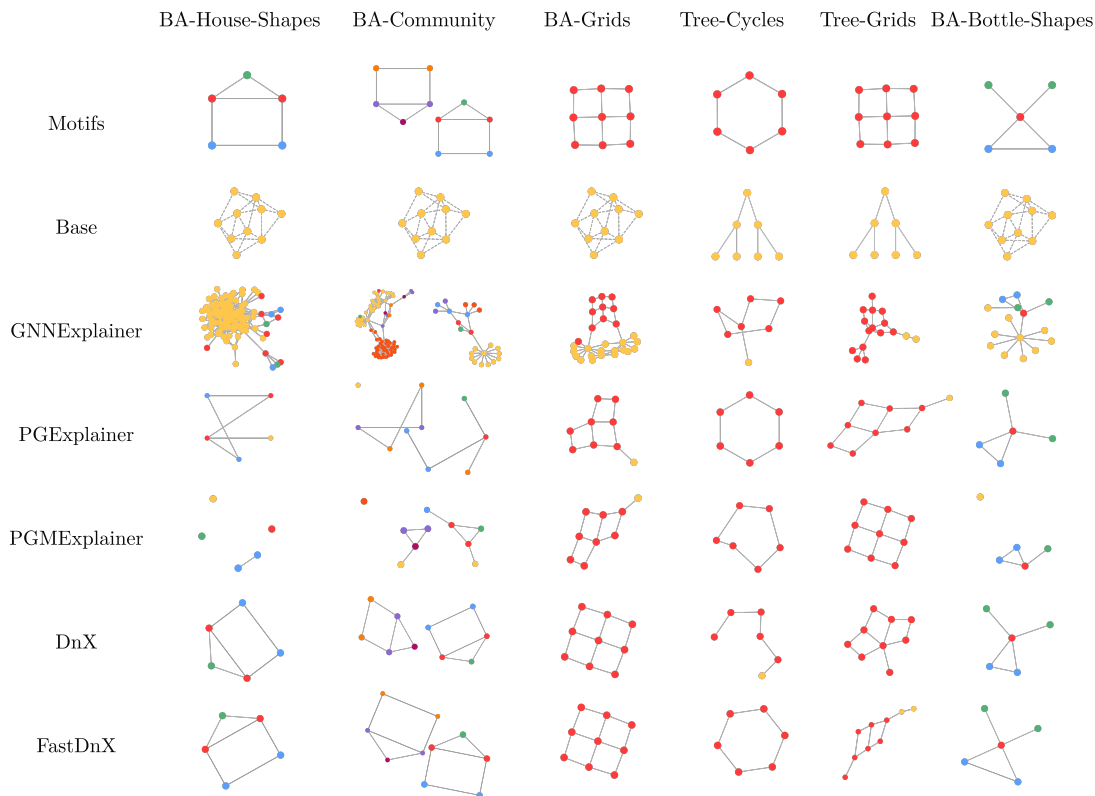
Modelo	Explicador	BA-House	BA-Community	BA-Grids	Tree-Cycles	Tree-Grids	BA-Bottle
GCN	GNNExplainer	77.5 $\pm$ 1.2	64.7 $\pm$ 1.0	89.2 $\pm$ 2.0	77.2 $\pm$ 9.0	71.1 $\pm$ 1.0	73.3 $\pm$ 3.0
	PGExplainer	95.0 $\pm$ 1.1	70.6 $\pm$ 2.0	86.2 $\pm$ 9.0	92.4 $\pm$ 5.2	76.7 $\pm$ 1.2	98.2 $\pm$ 3.0
	PGMEExplainer	97.9 $\pm$ 0.9	92.2 $\pm$ 0.2	88.6 $\pm$ 0.9	94.1 $\pm$ 0.8	86.8 $\pm$ 2.0	97.5 $\pm$ 1.5
	DnX	97.7 $\pm$ 0.2	94.6 $\pm$ 0.1	89.8 $\pm$ 0.1	83.3 $\pm$ 0.4	80.2 $\pm$ 0.1	99.6 $\pm$ 0.1
	FastDnX	99.6 $\pm$ NA	95.4 $\pm$ NA	93.9 $\pm$ NA	87.3 $\pm$ NA	85.0 $\pm$ NA	99.8 $\pm$ NA
ARMA	GNNExplainer	80.9 $\pm$ 1.2	78.5 $\pm$ 1.0	87.3 $\pm$ 1.3	77.7 $\pm$ 1.0	79.3 $\pm$ 1.1	84.3 $\pm$ 1.3
	PGExplainer	91.4 $\pm$ 0.1	72.1 $\pm$ 0.1	83.8 $\pm$ 1.0	92.6 $\pm$ 2.1	85.1 $\pm$ 0.1	97.0 $\pm$ 1.1
	PGMEExplainer	99.3 $\pm$ 0.2	67.5 $\pm$ 0.8	86.8 $\pm$ 0.3	95.0 $\pm$ 0.2	90.6 $\pm$ 0.3	99.7 $\pm$ 0.1
	DnX	98.1 $\pm$ 0.2	92.7 $\pm$ 0.2	90.8 $\pm$ 0.1	83.5 $\pm$ 0.4	79.6 $\pm$ 0.3	96.9 $\pm$ 0.2
	FastDnX	100.0 $\pm$ NA	95.2 $\pm$ NA	94.7 $\pm$ NA	87.1 $\pm$ NA	87.7 $\pm$ NA	99.9 $\pm$ NA
GATED	GNNExplainer	79.7 $\pm$ 1.0	68.8 $\pm$ 1.0	91.4 $\pm$ 3.0	85.2 $\pm$ 2.0	73.2 $\pm$ 4.0	70.0 $\pm$ 2.0
	PGExplainer	96.1 $\pm$ 4.1	70.9 $\pm$ 3.0	90.7 $\pm$ 1.0	91.7 $\pm$ 7.0	83.7 $\pm$ 1.5	98.7 $\pm$ 0.1
	PGMEExplainer	98.6 $\pm$ 0.1	69.4 $\pm$ 0.5	86.8 $\pm$ 0.3	94.1 $\pm$ 0.2	90.1 $\pm$ 0.2	98.3 $\pm$ 0.2
	DnX	98.3 $\pm$ 0.1	91.1 $\pm$ 0.1	90.8 $\pm$ 0.1	85.0 $\pm$ 0.3	82.1 $\pm$ 0.2	98.0 $\pm$ 0.2
	FastDnX	99.6 $\pm$ NA	93.5 $\pm$ NA	94.0 $\pm$ NA	76.8 $\pm$ NA	86.8 $\pm$ NA	98.0 $\pm$ NA
GIN	PGMEExplainer	60.2 $\pm$ 0.2	84.5 $\pm$ 0.3	68.4 $\pm$ 0.2	89.3 $\pm$ 0.2	85.0 $\pm$ 0.5	55.7 $\pm$ 0.4
	DnX	99.0 $\pm$ 0.1	94.0 $\pm$ 0.2	91.1 $\pm$ 0.1	84.1 $\pm$ 0.3	77.3 $\pm$ 0.2	95.3 $\pm$ 0.2
	FastDnX	99.6 $\pm$ NA	94.7 $\pm$ NA	93.9 $\pm$ NA	75.2 $\pm$ NA	76.5 $\pm$ NA	99.1 $\pm$ NA

No geral, FastDnX é o método com melhor desempenho para todas as arquiteturas de rede (GCN, ARMA, GATED e GIN) em todos os conjuntos de dados, exceto Tree-Cycles e Tree-Grids. Para conjunto de dados Tree-Grids, o FastDnX fica em segundo lugar nas arquiteturas GCN, ARMA e GATED, enquanto o PGMEExplainer obtém as mais

altas precisões. Também foi observado que, embora o DnX seja geralmente melhor que o GNNExplainer e o PGExplainer, seu desempenho supera o FastDnX apenas em 12,5% dos casos. GNNExplainer e PGExplainer não aparecem na comparação para GIN, pois requerem propagação de máscaras de arestas, e o Torch Geometric não oferece suporte a recursos de arestas para o GIN.

A Figura 9 mostra os subgrafos dos *motifs* e base além das explicações fornecidas por diferente explicadores para os conjuntos de dados sintéticos ao explicar o modelo GCN. No geral, o DnX e o FastDnX são os modelos que melhor capturam a estrutura do *motif* em BA-House, BA-Community, BA-Grids e BA-Bottle, validando sua maior acurácia nesses conjuntos de dados.

Figura 9 – Ilustração mostra como cada conjunto de dados sintéticos são montados com seus *motifs* e nós base. Os rótulos dos nós são representados por cores. Além disso, para avaliar como cada explicador se comporta, são mostrados exemplos de explicações extraídas do GNNExplainer, PGMExplainer, PGExplainer, DnX e FastDnX para a mesma predição do modelo GCN em cada conjunto de dados.



A Tabela 5 relata o desempenho de todos os explicadores nos conjuntos de dados reais Bitcoin-Alpha e Bitcoin-OTC. Seguindo o trabalho anterior (VU; THAI, 2020), foi utilizado a precisão média (AP) como métrica de avaliação, ou seja, a porcentagem dos top- k nós obtidos de cada explicador que estão corretos, calculados sobre todos os nós a

serem explicados. As explicações foram geradas sobre GNNs de 3 camadas e as explicações são buscadas na vizinhança de 3-hop. O DnX é o método de melhor desempenho e sua variante rápida FastDnX é a segunda melhor em todos os experimentos. A diferença de precisão entre o DnX e a melhor linha de base é de pelo menos 14% para Bitcoin-Alpha e 11% para Bitcoin-OTC. No geral, o DnX supera o GNNExplainer e o PGMEExplainer por uma grande margem.

Tabela 5 – Performance (média da precisão) para explicações a nível de nó dos conjuntos de dados do mundo real para os primeiros 500 nós. O DnX é o método com melhor desempenho e sua variante FastDnX é a segunda melhor em todos os experimentos.

Modelo	Explicador	Bitcoin-Alpha			Bitcoin-OTC		
		top 3	top 4	top 5	top 3	top 4	top 5
GCN (3-hop)	GNNEx	80.1	74.9	70.9	82.4	79.6	70.6
	PGEx	81.5	78.1	69.5	78.5	74.5	67.4
	PGMEx	67.0	59.8	51.8	63.0	55.2	47.4
	DnX	95.8	91.9	87.9	94.8	91.4	86.3
	FastDnX	89.8	85.2	80.2	88.0	83.0	78.8
ARMA (3-hop)	GNNEx	80.9	78.9	74.5	73.2	69.5	64.0
	PGEx	72.5	67.8	65.0	69.7	68.7	61.45
	PGMEx	73.8	66.4	58.3	69.1	62.8	54.9
	DnX	95.0	90.1	84.9	93.6	88.1	83.2
	FastDnX	90.2	85.8	81.0	87.7	83.3	79.2
GATED (3-hop)	GNNEx	80.1	75.1	769.6	75.9	70.9	66.0
	PGEx	75.4	74.8	67.8	72.5	70.5	65.1
	PGMEx	80.2	76.5	72.4	77.5	72.1	67.3
	DnX	94.4	89.6	84.4	93.4	88.9	83.5
	FastDnX	89.6	85.3	80.0	88.0	83.4	79.0
GIN (3-hop)	PGM-Ex	58.7	49.2	40.8	57.6	47.8	40.2
	DnX	94.3	88.9	83.2	94.0	89.2	83.6
	FastDnX	85.0	80.4	74.6	82.6	77.1	71.3

Os resultados relatados para o PGMEExplainer ao gerar explicações para a rede GCN na Tabela 5 divergem dos resultados apresentados em seu artigo pois, durante a execução dos experimentos, foi notado que o protocolo de avaliação empregado pelo Vu e Thai (2020) para obter explicações para uma GCN de 3 camadas considera apenas nós candidatos de 1 *hop* durante a fase de explicação. Isso implica que alguns nós potencialmente relevantes são descartados. A Tabela 6 mostra os resultados para configurações de 1 *hop* do modelo GCN. Analisando os resultados da Tabela 5 e Tabela 6 é possível notar que o desempenho do PGMEExplainer cai consideravelmente ao passar de 1 para 3 *hop*. Mesmo com essa configuração o DnX e FastDnX conseguem melhores resultados que o PGMEExplainer.

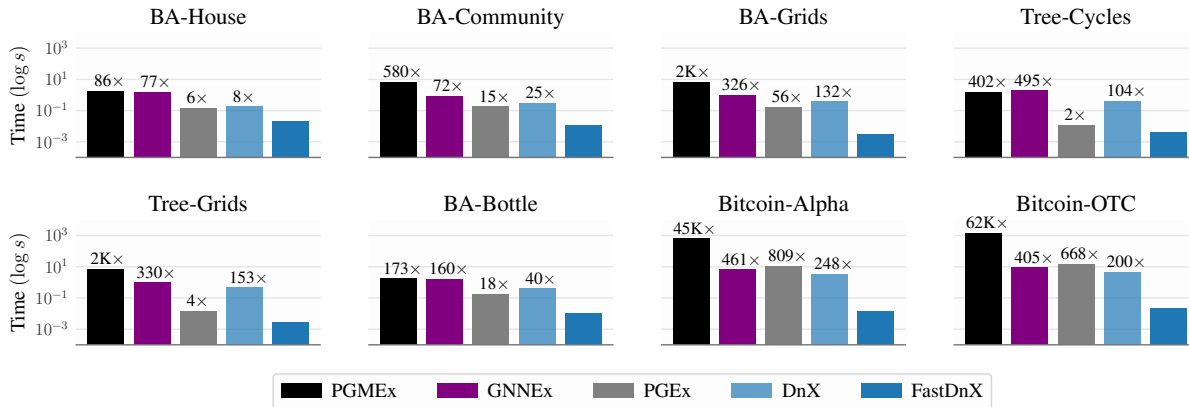
Comparação de tempo. Para demonstrar a eficiência computacional de DnX/FastDnX, a Figura 10 mostra o tempo que cada método leva para explicar uma única predição GCN. Para uma comparação justa, também foi considerado a etapa de destilação do DnX/FastDnX. Em particular, foi adicionado uma fração - um sobre o número total de nós que desejamos explicar - do tempo de destilação ao tempo que o DnX e FastDnX

Tabela 6 – Performance (média da precisão) de explicação a nível de nó para os conjunto de dados reais, buscando os nós apenas na vizinhança de 1-hop, mesmo protocolo utilizado por Vu e Thai (2020). Os explicadores DnX e FastDnX conseguem melhores resultados que todos os outros explicadores.

Modelo	Explicador	Bitcoin-Alpha			Bitcoin-OTC		
		top 3	top 4	top 5	top 3	top 4	top 5
GCN (1-hop)	GNNEx	86.3	85.2	81.2	83.3	81.7	77.0
	PGEx	83.5	83.6	79.5	79.9	80.1	76.6
	PGMEx	87.3	85.7	84.8	83.3	81.7	80.8
	DnX	92.2	89.5	88.4	89.4	86.6	84.7
	FastDnX	89.4	87.8	86.8	87.7	85.1	83.4

realmente levam para gerar uma explicação. Notavelmente, tanto o DnX quanto o FastDnX são consistentemente muito mais rápidos que o GNNExplainer e o PGMEExplainer. Por exemplo, FastDnX é mais de quarenta mil vezes mais rápido que PGMEExplainer em Bitcoin-Alpha e Bitcoin-OTC.

Figura 10 – Comparação de tempo. Os gráficos de barras mostram o tempo médio que cada método leva para explicar uma predição do GCN. FastDnX é consistentemente o método mais rápido, muitas vezes por uma grande margem. Para os conjuntos de dados com maior grau médio (conjuntos de dados Bitcoin), o FastDnX é 4 ordens de magnitude mais rápido que o PGMEExplainer e 2 ordens mais rápido que os outros métodos.



Resultados de explicação a nível de aresta. Os resultados alcançados para explicações de nível de aresta para os conjuntos de dados sintéticos são apresentados na Tabela 7. Todos os experimentos foram repetidos dez vezes e calculado a média e o desvio padrão. Para todos os conjuntos de dados, medimos o desempenho em termos de AUC, seguindo (LUO et al., 2020). Assim como os resultados da Tabela 4 na maioria dos casos, FastDnX é o melhor ou o segundo melhor explicador para todos os modelos e conjuntos de dados. Para GCN e GATED, o PGExplainer produz os melhores resultados. No geral, tanto o DnX quanto o FastDnX superam o GNNExplainer e o PGMEExplainer. Notavelmente, o desempenho do FastDnX e do DnX é estável, com pequenas flutuações dependendo do modelo explicado. O mesmo não é verdade para os métodos concorrentes, por exemplo,

o PGExplainer perde mais de 15% AUC para o BA-Community (em comparação com a GCN e ARMA).

Tabela 7 – Performance (media e desvio padrão da AUC) para explicações a nível de arestas dos modelos explicadores para os conjuntos de dados sintéticos. Os números **Azuis** e **Verdes** denotam o melhor e o segundo melhor resultados respectivamente. O FastDnX é o melhor ou o segundo melhor explicador para todos os modelos e conjuntos de dados. Para GCN e GATED, o PGExplainer produz os melhores resultados. No geral, tanto o Dnx quanto o FastDnX superam o GNNEExplainer e o PGMEExplainer.

Modelo	Explicador	BA-House	BA-Community	BA-Grids	Tree-Cycles	Tree-Grids	BA-Bottle
GCN	GNNEExplainer	82.4 ± 6.0	71.1 ± 6.0	80.9 ± 1.0	58.4 ± 1.0	53.9 ± 2.0	82.8 ± 1.0
	PGExplainer	99.9 ± .01	99.9 ± .01	94.1 ± 9.0	92.3 ± 5.0	79.4 ± 2.0	99.9 ± .01
	PGMEExplainer	56.2 ± 0.3	53.0 ± 0.4	68.9 ± 0.2	62.1 ± 0.2	66.3 ± 0.4	53.4 ± 0.5
	DnX	95.7 ± .09	87.5 ± .09	97.3 ± .03	80.5 ± .02	72.6 ± 0.1	94.5 ± 0.1
	FastDnX	99.4 ± NA	93.0 ± NA	96.7 ± NA	89.1 ± NA	77.4 ± NA	99.5 ± NA
ARMA	GNNEExplainer	80.8 ± 1.0	75.1 ± 1.0	69.4 ± 0.1	59.2 ± 2.0	59.9 ± 1.0	85.8 ± .01
	PGExplainer	93.3 ± 1.0	82.7 ± .01	93.1 ± .01	92.4 ± 2.0	84.0 ± .03	98.0 ± .02
	PGMEExplainer	82.2 ± 0.2	49.6 ± 0.7	40.4 ± 0.2	64.7 ± 0.2	69.0 ± 0.3	63.1 ± 0.3
	DnX	95.7 ± .06	87.9 ± .07	97.3 ± .04	80.4 ± 0.3	72.8 ± 0.1	94.4 ± .08
	FastDnX	99.7 ± NA	96.8 ± NA	97.9 ± NA	87.1 ± NA	78.4 ± NA	99.7 ± NA
GATED	GNNEExplainer	79.1 ± 2.0	59.2 ± 3.0	69.4 ± 2.0	62.6 ± 2.0	53.6 ± 1.0	73.9 ± 2.0
	PGExplainer	99.9 ± 2.0	99.1 ± 2.0	99.8 ± 1.0	76.5 ± 9.0	97.0 ± 3.0	99.9 ± 0.1
	PGMEExplainer	49.3 ± 0.4	47.2 ± 0.5	46.2 ± 0.3	45.8 ± 0.6	53.1 ± 0.8	49.1 ± 0.7
	DnX	95.8 ± .07	84.4 ± .09	97.3 ± .02	81.1 ± 0.1	72.8 ± 0.1	94.4 ± .09
	FastDnX	99.5 ± NA	93.7 ± NA	96.6 ± NA	89.2 ± NA	78.6 ± NA	99.6 ± NA
GIN	PGMEExplainer	52.4 ± 0.3	52.0 ± 0.4	52.2 ± 1.0	67.4 ± 0.2	63.9 ± 0.2	53.3 ± 0.4
	DnX	95.7 ± 0.1	87.2 ± 0.1	97.3 ± 0.1	81.8 ± 0.2	72.0 ± 0.1	94.2 ± 0.1
	FastDnX	97.9 ± NA	91.0 ± NA	97.1 ± NA	86.3 ± NA	72.3 ± NA	97.0 ± NA

Já a Tabela 8 mostra os resultados a nível de arestas para os conjuntos de dados reais, assim como os resultados a nível de nó o FastDnX e DnX são os melhores explicadores para esses conjuntos de dados.

5.5 Discussões

5.5.1 Os *benchmarks* são simples demais?

Dado que o DnX/FastDnX muitas vezes alcança um desempenho notável ao explicar substitutos simples, surge uma pergunta natural: *esses benchmarks (conjuntos de dados) populares para explicações de GNNs são muito simples?* Como esses benchmarks dependem de explicações *ground-truth* de modelo agnósticos, agora investigamos vieses indutivos por trás dessas explicações e mostramos que eles podem ser facilmente capturados.

A Figura 11 informa a distribuição de graus dos nós dos *motifs* e base para todos os conjuntos de dados sintéticos. Lembre-se de que, por design, as explicações verdade são sempre dadas por nós dos *motifs*. Observe também que a distribuição dos nós dos *motifs*

Tabela 8 – Performance (AUC) para explicações a nível de aresta dos conjuntos de dados reais para os primeiros 500 nós. O FastDnX é o método com melhor desempenho e o DnX é a segunda melhor em todos os experimentos.

Modelo	Explicador	Bitcoin-Alpha	Bitcoin-OTC
GCN (3-hop)	GNNex	94.0	97.3
	PGEx	59.7	54.2
	PGMEx	75.8	52.7
	DnX	97.1	98.1
	FastDnX	97.3	99.1
ARMA (3-hop)	GNNex	79.6	92.3
	PGEx	47.8	45.4
	PGMEx	76.2	73.8
	DnX	96.4	98.7
	FastDnX	97.4	99.1
GATED (3-hop)	GNNex	92.5	93.6
	PGEx	34.7	33.7
	PGMEx	86.3	83.1
	DnX	96.3	98.6
	FastDnX	97.2	99.1
GIN (3-hop)	PGMEx	52.8	76.1
	DnX	96.4	98.7
	FastDnX	97.5	98.9

e nós base quase não tem sobreposição para a maioria dos conjuntos de dados (exceto Tree-Cycles & Tree-Grids). Assim, qualquer explicador capaz de alavancar informações de grau obteria alta precisão.

Para tornar isso mais concreto, propomos um "explicador" de linha de base muito simples que gera explicações com base na matriz de adjacência normalizada. Em particular, definimos a importância do nó j para a predição do nó i como a entrada (i, j) de $\tilde{A}^L$, com $L = 3$. Com esta linha de base simples, obtemos os seguintes valores de precisão: 99,9% (BA-House), 98,1% (BA-Community), 99,9% (BA-Grids), 95,9% (Tree-Cycles), 90,4% (Tree-Grids) e 99,9% (BA-Bottle). Notavelmente, essa linha de base obtém melhores resultados que todos os demais métodos explicadores para GCN na Tabela 4.

Faber, Moghaddam e Wattenhofer (2021) também levantaram questões em relação a esses *benchmarks*, propondo conjuntos de dados alternativos. Foi realizado experimentos executando o FastDnX para explicar um modelo GCN de 2 camadas para dois de seus conjuntos de dados propostos (*Community* e *Negative Evidence*) e os resultados de precisão alcançados são notavelmente bons: 94,0% e 99,5%, respectivamente. Além disso, simplesmente classificar os nós com base nas entradas de $\tilde{A}^L$ ($L = 2$) atinge uma precisão de 93,0% (*Community*) e 99,6% (*Neg. evidence*).

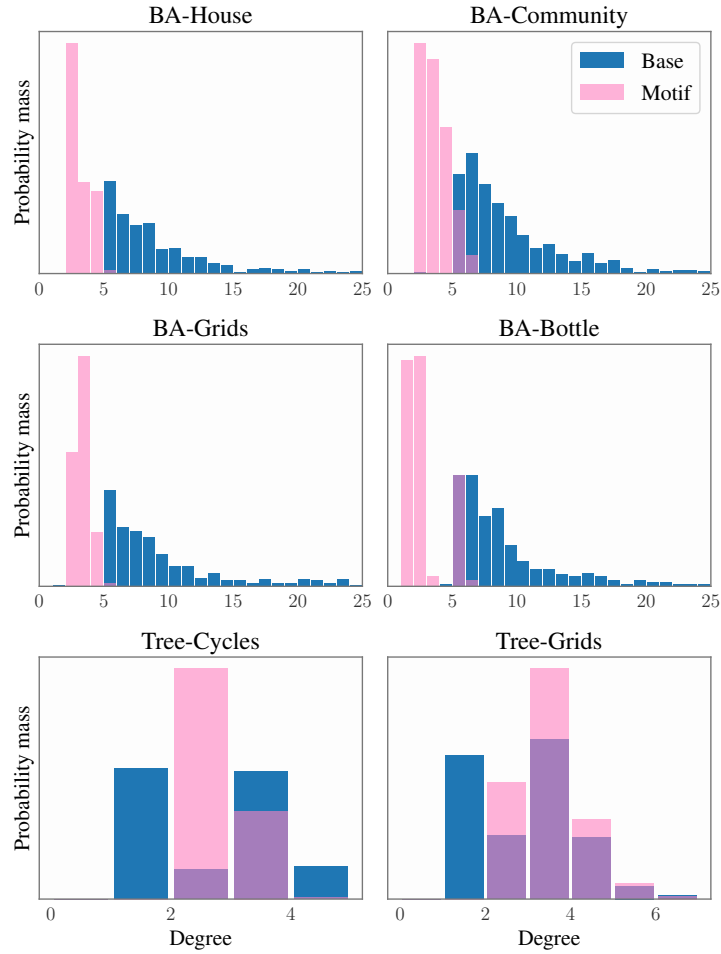


Figura 11 – Distribuição de graus dos nós do *motif* e base. Embora seja possível distinguir em geral os nós do *motif* e base das informações de grau em conjuntos de dados baseados em BA, há uma sobreposição significativa em Tree-Cycles and Tree-Grids.

5.5.2 Limitações

Embora GNNs simples (como SGC) tenham mostrado um bom desempenho em tarefas de classificação a nível de nó, eles não conseguem rivalizar com GNNs recentes para tarefas de predição a nível de grafo (HUANG et al., 2021; WU et al., 2019). Naturalmente, não é esperado que o DnX e o FastDnX funcionassem bem para explicar as predições em nível de grafo. No entanto, os métodos propostos podem ser facilmente estendidos para usar GNNs lineares mais poderosos que incorporam diferentes tipos de operadores de difusão (ROSSI et al., 2020), ou usar conexões residuais de longo alcance (CHEN et al., 2020).

5.6 Conclusão

Neste capítulo foi apresentada a metodologia empregada durante os experimentos para avaliação dos modelos propostos DnX e FastDnX, bem como a descrição dos conjuntos

de dados sintéticos e do mundo real utilizados para os experimentos. A proposta foi avaliada utilizando as métricas de avaliação acurácia, precisão e AUC. Os experimentos comparam o desempenho do DnX e FastDnX com outros três modelos explicadores presentes na literatura GNNExplainer, PGExplainer e PGMEExplainer . Os resultados alcançados reforçam a eficiência do modelo proposto, visto que DnX e FastDnX são modelos bem menos complexos que os outros explicadores, mas que mostrou resultados competitivos com um custo de tempo significativamente menor.

6 Conclusão

Este trabalho teve foco em encontrar formulações mais simples de modelos explicadores para redes neurais para grafos e foi proposto o DnX, um método simples e intuitivo de duas etapas para explicação de GNNs. O DnX busca encontrar subestruturas do grafo de entrada que são mais relevantes para a predição da GNN. Para isso, a GNN a ser explicada é destilada por uma GNN mais simples e mais interpretável, que serve como um substituto global. Em seguida, aproveitando a estrutura simples do substituto resolve-se um problema convexo para extrair explicações. Além do DnX, foi proposto também o FastDnX que aproveita a estrutura linear do modelo substituto e extrai explicações usando decomposição linear.

Para avaliar a proposta foram realizadas simulações computacionais com conjuntos de dados sintéticos e reais e os resultados comparados com outros três modelos explicadores da literatura. Além da GCN frequentemente usada em experimentos de explicabilidade, foram consideradas mais três GNNs com diferentes arquiteturas, o que permite avaliar a robustez e o desempenho dos explicadores em GNNs de diferentes complexidades.

A acurácia e a precisão alcançadas nos resultados para o DnX e FastDnX assumiram valores significativos, consequentemente, os métodos conseguem gerar boas explicações para as predições, superando técnicas anteriores em uma variedade de conjuntos de dados. Os resultados apresentados demonstraram a viabilidade da proposta, uma vez que o DnX e o FastDnX mostram desempenho no mínimo equivalente aos alcançados através do PGMEExplainer, GNNExplainer e PGExplainer.

Foram fornecidos também resultados teóricos que provam a estrutura convexa do problema de otimização da etapa de extração de explicação do método DnX, além de confirmar a teoria de que boas explicações de uma GNN substituta também caracterizam boas explicações da GNN que queremos explicar. Por fim, foi feita uma análise no desempenho computacional e, notavelmente, o design simples do FastDnX permite executar pelo menos 200 vezes mais rápido do que as linhas de base relevantes em tarefas do mundo real.

Referências

- AGARWAL, C.; ZITNIK, M.; LAKKARAJU, H. Probing GNN Explainers: A Rigorous Theoretical and Empirical Analysis of GNN Explanation Methods. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. [S.l.: s.n.], 2022. Citado 2 vezes nas páginas 44 e 45.
- BARABÁSI, A.; ALBERT, R. Emergence of scaling in random networks. *Science*, v. 286, n. 5439, 1999. Citado na página 47.
- BIANCHI, F. M. et al. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 44, n. 7, 2022. Citado na página 48.
- BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006. Citado na página 25.
- BRONSTEIN, M. M. et al. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, v. 34, n. 4, p. 18–42, 2017. Citado na página 25.
- CAO, Q. et al. Vggface2: A dataset for recognising faces across pose and age. In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. [S.l.]: IEEE Computer Society, 2018. Citado na página 25.
- CHEN, M. et al. Simple and deep graph convolutional networks. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2020. Citado na página 57.
- CHEN, X. et al. Multi-view 3d object detection network for autonomous driving. In: . [S.l.: s.n.], 2017. p. 6526–6534. Citado na página 25.
- FABER, L.; MOGHADDAM, A. K.; WATTENHOFER, R. When comparing to ground truth is wrong: On evaluating gnn explanation methods. In: *Conference on Knowledge Discovery & Data Mining (KDD)*. [S.l.: s.n.], 2021. Citado na página 56.
- 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. Citado 2 vezes nas páginas 47 e 49.
- FONG, R.; VEDALDI, A. Interpretable explanations of black boxes by meaningful perturbation. In: . [S.l.: s.n.], 2017. p. 3449–3457. Citado na página 25.
- GAO, B.; PAVEL, L. On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning. *ArXiv:1704.00805*, 2018. Citado na página 67.
- GEIJN, R. V.; MYERS, M. *Advanced Linear Algebra Foundations to Frontiers*. Austin, Texas: open EdX Publisher, 2022. Citado na página 67.
- HAMILTON, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan and Claypool, v. 14, n. 3, p. 1–159, 2020. Citado 2 vezes nas páginas 29 e 31.

- HAMILTON, W. L.; YING, R.; LESKOVEC, J. Inductive representation learning on large graphs. In: *NIPS*. [S.l.: s.n.], 2017. Citado na página 31.
- HENNI, K.; MEZGHANI, N.; GOUIN-VALLERAND, C. Unsupervised graph-based feature selection via subspace and pagerank centrality. *Expert Systems with Applications*, 2018. Citado na página 25.
- HINTON, G. E.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*, 2015. Citado na página 41.
- HUANG, Q. et al. Combining label propagation and simple models out-performs graph neural networks. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2021. Citado na página 57.
- KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2015. Citado na página 48.
- KIPF, T. N.; WELING, M. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016. Citado na página 31.
- KIPF, T. N.; WELING, M. Semi-supervised classification with graph convolutional networks. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 31, 32 e 48.
- KUMAR, S. et al. Rev2: Fraudulent user prediction in rating platforms. In: ACM. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. [S.l.], 2018. p. 333–341. Citado na página 47.
- KUMAR, S. et al. Edge weight prediction in weighted signed networks. In: IEEE. *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. [S.l.], 2016. p. 221–230. Citado na página 47.
- LI, Y. et al. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)*, 2016. Citado na página 48.
- LIAO, R. et al. Efficient graph generation with graph recurrent attention networks. In: *NeurIPS*. [S.l.: s.n.], 2019. Citado na página 31.
- LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2019. Citado na página 49.
- LUCIC, A. et al. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. [S.l.: s.n.], 2022. Citado na página 26.
- LUO, D. et al. Parameterized explainer for graph neural network. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. Citado 7 vezes nas páginas 26, 35, 37, 47, 48, 51 e 54.
- MA, Z. et al. Path integral based convolution and pooling for graph neural networks. In: *NeurIPS*. [S.l.: s.n.], 2020. Citado na página 31.
- MARGARITIS, D.; THRUN, S. Bayesian network induction via local neighborhoods. In: SOLLA, S.; LEEN, T.; MÜLLER, K. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.]: MIT Press, 1999. v. 12. Citado na página 38.

- PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. 2019. Citado 2 vezes nas páginas 47 e 49.
- ROSSI, E. et al. SIGN: scalable inception graph neural networks. *Workshop on Graph Representation Learning and Beyond (ICML)*, 2020. Citado na página 57.
- SALLAB, A. E. et al. Deep reinforcement learning framework for autonomous driving. *S&T Electronic Imaging, Autonomous Vehicles and Machines 2017*, abs/1704.02532, 2017. Citado na página 25.
- SCARSELLI, F. et al. The graph neural network model. *IEEE Transactions on Neural Networks*, v. 20, p. 61–80, 2009. Citado na página 25.
- SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. Citado na página 25.
- STRUMBELJ, E.; KONONENKO, I.; ROBNIK-SIKONJA, M. Explaining instance classifications with interactions of subsets of feature values. *Data Knowl. Eng.*, v. 68, p. 886–904, 2009. Citado 2 vezes nas páginas 25 e 35.
- VEDALDI, A. M. e A. Salient deconvolutional networks. In: *ECCV*. [S.l.: s.n.], 2016. Citado na página 35.
- VELIČKOVIĆ, P. et al. Graph Attention Networks. *International Conference on Learning Representations (ICLR)*, 2018. Citado na página 31.
- VU, M.; THAI, M. T. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. Citado 11 vezes nas páginas 17, 26, 35, 38, 39, 47, 48, 51, 52, 53 e 54.
- WANG, X. et al. Towards multi-grained explainability for graph neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2021. Citado na página 26.
- WEISFEILER, B.; LEHMAN, A. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, v. 2, n. 9, p. 12–16, 1968. Citado na página 32.
- WU, F. et al. Simplifying graph convolutional networks. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2019. Citado 6 vezes nas páginas 26, 31, 33, 41, 48 e 57.
- XU, K. et al. How powerful are graph neural networks? *International Conference on Learning Representations (ICLR)*, 2019. Citado 3 vezes nas páginas 31, 32 e 48.
- XU, Y. L.; KONSTANTINIDIS, K.; MANDIC, D. P. Multi-graph tensor networks. In: *the First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. Citado na página 25.
- YANG, Y. et al. Distilling knowledge from graph convolutional networks. p. 7074–7083, 2020. Citado na página 41.

- YING, Z. et al. Gnnexplainer: Generating explanations for graph neural networks. In: WALLACH, H. et al. (Ed.). *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2019. Citado 7 vezes nas páginas 26, 35, 36, 37, 47, 48 e 51.
- YUAN, H. et al. XGNN: Towards model-level explanations of graph neural networks. In: *International Conference on Knowledge Discovery & Data Mining (KDD)*. [S.l.: s.n.], 2020. Citado na página 26.
- YUAN, H. et al. Xgnn: Towards model-level explanations of graph neural networks. *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020. Citado na página 35.
- YUAN, H. et al. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 2022. Citado na página 26.
- YUAN, H. et al. On explainability of graph neural networks via subgraph explorations. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2021. Citado na página 26.
- ZEILER, M.; FERGUS, R. Visualizing and understanding convolutional neural networks. In: . [S.l.: s.n.], 2013. v. 8689. ISBN 978-3-319-10589-5. Citado na página 35.
- ZHANG, M.; CHEN, Y. Link prediction based on graph neural networks. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2018. (NIPS'18), p. 5171–5181. Citado na página 31.
- ZHOU, S. et al. Distilling holistic knowledge with graph neural networks. p. 10387–10396, October 2021. Citado na página 41.

Apêndices

APÊNDICE A – Provas Matemáticas

Prova do Lema 1. Para o nó u , sabe-se, por definição, que

$$\begin{aligned} & \|\Psi(\mathcal{G}_u) - \Psi(t(\mathcal{G}_u, \mathcal{E}_u))\|_2 \\ &= \left\| \sigma \left(\tilde{A}^L X \Theta \right)_u - \sigma \left(\tilde{E}^L X \Theta \right)_u \right\|_2 \\ &= \left\| \sigma \left(\tilde{A}_u^L X \Theta \right) - \sigma \left(\tilde{E}_u^L X \Theta \right) \right\|_2, \end{aligned}$$

se chamarmos σ a função softmax e $\tilde{E}^L$ a matriz de adjacência normalizada elevada a potência L $\tilde{A}^L$ depois de aplicar a explicação $\mathcal{E}_u$. Porque softmax é uma função Lipschitz contínua com constante de Lipschitz 1 em relação à norma $\|\cdot\|_2$ (GAO; PAVEL, 2018) e considerando a propriedade submultiplicativa da norma da matriz induzida (GEIJN; MYERS, 2022),

$$\begin{aligned} & \|\Psi(\mathcal{G}_u) - \Psi(t(\mathcal{G}_u, \mathcal{E}_u))\|_2 \\ &= \left\| \sigma \left(\tilde{A}_u^L X \Theta \right) - \sigma \left(\tilde{E}_u^L X \Theta \right) \right\|_2 \\ &\leq \|X \Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2. \end{aligned}$$

Da mesma forma, para uma perturbação $\mathcal{G}'_u$ de $\mathcal{G}_u$ dada por Σ'_u sendo adicionada a X ,

$$\|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \leq \|(X + \Sigma'_u) \Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2.$$

Calculando a média $\mathcal{K} \cup \{\mathcal{G}_u\}$:

$$\begin{aligned} & \frac{1}{|\mathcal{K}| + 1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\ &\leq \frac{1}{|\mathcal{K}| + 1} \left[\|X \Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 + \sum_{\mathcal{G}'_u \in \mathcal{K}} \|(X + \Sigma'_u) \Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 \right] \\ &\leq \|\Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 \frac{1}{|\mathcal{K}| + 1} \left(\|X\|_2 + \sum_{\mathcal{G}'_u \in \mathcal{K}} \|X + \Sigma'_u\|_2 \right) \\ &\leq \|\Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 \max \left(\{\|X\|_2\} \cup \{\|X + \Sigma'_u\|_2\}_{\mathcal{G}'_u \in \mathcal{K}} \right) \\ &= \gamma_{X, \Sigma} \|\Theta\|_2 \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 \\ &= \gamma \left\| \tilde{A}_u^L - \tilde{E}_u^L \right\|_2 \\ &= \gamma \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2. \end{aligned}$$

Quando Σ é limitado, a constante γ pode não depender de Σ . □

Prova do Teorema 1. Nós sabemos que

$$\begin{aligned} & \|\Phi(\mathcal{G}'_u) - \Phi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\ &= \|\Phi(\mathcal{G}'_u) - \Psi(\mathcal{G}'_u) + \Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u)) + \Psi(t(\mathcal{G}'_u, \mathcal{E}_u)) - \Phi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2. \end{aligned}$$

Então, usando a desigualdade triangular e o Lema 1,

$$\begin{aligned}
& \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Phi(\mathcal{G}'_u) - \Phi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \leq \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Phi(\mathcal{G}'_u) - \Psi(\mathcal{G}'_u)\|_2 \\
& \quad + \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \quad + \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(t(\mathcal{G}'_u, \mathcal{E}_u)) - \Phi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \leq \gamma \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 + 2\alpha.
\end{aligned}$$

□

Prova do Lema 2. Da prova do Lema 1, nós sabemos isso

$$\begin{aligned}
& \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \leq \|\Theta\|_2 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 \frac{1}{|\mathcal{K}|+1} \left(\|X\|_2 + \sum_{\mathcal{G}'_u \in \mathcal{K}} \|X + \Sigma'_u\|_2 \right).
\end{aligned}$$

Usando a desigualdade triangular, podemos escrever

$$\begin{aligned}
& \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \leq \|\Theta\|_2 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 \|X\|_2 + \|\Theta\|_2 \left\| \Delta_{\mathcal{E}_u} \tilde{A}_u^L \right\|_2 \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K}} \|\Sigma'_u\|_2.
\end{aligned}$$

Podemos trabalhar na soma de perturbações:

$$\begin{aligned}
\sum_{\mathcal{G}'_u \in \mathcal{K}} \|\Sigma'_u\|_2 & \leq \sum_{\mathcal{G}'_u \in \mathcal{K}} \|\Sigma'_u\|_F \\
& \leq \sum_{\mathcal{G}'_u \in \mathcal{K}} \|\Sigma'_u\|_F^2 \\
& = \sum_{\mathcal{G}'_u \in \mathcal{K}} \sum_{i,j} (\Sigma'_{u,ij})^2 \\
& = \sum_{i=1}^{|\mathcal{K}|nd} \epsilon_i^2 \\
& = \sigma^2 \sum_{i=1}^{|\mathcal{K}|nd} Z_i^2 \\
& = \sigma^2 Q,
\end{aligned}$$

com

$$\bullet \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2);$$

- $Z_i \sim \mathcal{N}(0, 1)$;
- $Q \sim \chi^2_{|K|nd}$;
- $(n, d) = \dim(X)$.

Se α_p é o p -percentil de Q , então a probabilidade da última desigualdade é p :

$$\begin{aligned}
& \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K} \cup \{\mathcal{G}_u\}} \|\Psi(\mathcal{G}'_u) - \Psi(t(\mathcal{G}'_u, \mathcal{E}_u))\|_2 \\
& \leq \|\Theta\|_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \|X\|_2 + \|\Theta\|_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \frac{1}{|\mathcal{K}|+1} \sum_{\mathcal{G}'_u \in \mathcal{K}} \|\Sigma'_u\|_2 \\
& \leq \gamma_1 \left\| \Delta \tilde{A}_u^L \right\|_2 + \|\Theta\|_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \frac{1}{|\mathcal{K}|+1} \sigma^2 Q \\
& \leq \gamma_1 \left\| \Delta \tilde{A}_u^L \right\|_2 + \|\Theta\|_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \frac{1}{|\mathcal{K}|+1} \sigma^2 \alpha_p \\
& = \gamma_1 \left\| \Delta \tilde{A}_u^L \right\|_2 + \gamma_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \sigma^2 \alpha_p
\end{aligned}$$

Observe que, quando σ se aproxima de zero, a característica probabilística do limite torna-se desprezível.

Finalmente, se o limite for ξ , então a probabilidade é

$$p = F_{\chi^2_{|K|nd}} \left(\frac{\xi - \gamma_1 \left\| \Delta \tilde{A}_u^L \right\|_2}{\gamma_2 \left\| \Delta \tilde{A}_u^L \right\|_2 \sigma^2} \right),$$

$F_{\chi^2_{|K|nd}}$ sendo o c.d.f. de Q , dado que o numerador é positivo. $\square$

Prova de Teorema 3. A função objetivo do problema em Equação 4.3 pode ser escrito como

$$\begin{aligned}
f(\mathcal{E}) &= \left\| \tilde{A}_i^L \text{diag}(\mathcal{E}) X \Theta - \tilde{A}_i^L X \Theta \right\|_2^2 \\
&= \left(\tilde{A}_i^L \text{diag}(\mathcal{E}) X \Theta - \tilde{A}_i^L X \Theta \right) \cdot \\
&\quad \cdot \left(\tilde{A}_i^L \text{diag}(\mathcal{E}) X \Theta - \tilde{A}_i^L X \Theta \right)^\top \\
&= \mathcal{E}^\top \text{diag} \left[\left(\tilde{A}_i^L \right)^\top \right] X \Theta \Theta^\top X^\top \text{diag} \left[\left(\tilde{A}_i^L \right)^\top \right] \mathcal{E} \\
&\quad - 2 \mathcal{E}^\top \text{diag} \left[\left(\tilde{A}_i^L \right)^\top \right] X \Theta \Theta^\top X^\top \left(\tilde{A}_i^L \right)^\top \\
&\quad + \left\| \tilde{A}_i^L X \Theta \right\|_2^2 \\
&= \frac{1}{2} \mathcal{E}^\top Q \mathcal{E} + \mathcal{E}^\top c + \delta.
\end{aligned}$$

Observe também que

$$Q = P^\top P \text{ com } P = \sqrt{2} \Theta^\top X^\top \text{diag} \left[\left(\tilde{A}_i^L \right)^\top \right],$$

assim Q é simétrico e semidefinido positivo. Como tanto a função objetivo quanto o conjunto factível são convexos, o problema de otimização também é convexo. $\square$