Simplifying Simplex Convolutional Networks

Gabriel Duarte[1] ,Tamara Arruda[2], Diego Mesquita[2] Amauri H. Souza[1],

[1] Instituto Federal do Ceará, Brasil
g.jonas.duarte@gmail.com, amauriholanda@ifce.edu.br
[2] Fundação Getulio Vargas, Brasil
{tamara.pereira, diego.mesquita}@fgv.com.br

Abstract. Topological deep learning (TDL) has recently emerged as a family of neural networks for data on topological domains (e.g., graphs and cellular complexes). In the context of graph data, TDL provides a recipe for leveraging high-order information, e.g., from cliques, to boost the expressiveness of graph neural networks. This additional power comes at a computational cost, stemming from message passing between higher-order structures. In this paper, we alleviate the computing toll of TDL by proposing `SimpleXCN`, a linear convolutional model for simplicial complexes. `SimpleXCN` derives naturally from removing non-linearities of base TDL architectures. Despite its simplicity, `SimpleXCN` remains competitive with established topological neural networks in node classification tasks. Our experiments show a reduction of approximately 25% in time and 10 times in the number of parameters, and consequently a significant reduction in memory usage of up to 50% compared to other SNNs.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**.

Keywords: Graph Neural Networks, Node Classification, Topological Deep Learning.

1. INTRODUÇÃO

Aprendizagem profunda topológica (*topological deep learning*, TDL) [Horn et al. 2022; Bodnar et al. 2021; Papamarkou et al. 2024] compreende um conjunto de modelos que incorporam conceitos de topologia em redes neurais para processamento de dados relacionais de alta ordem, como hipergrafos, complexos simpliciais e complexos celulares — ou domínios topológicos. Uma das abordagens mais populares [Papillon et al. 2023] consiste em estender o esquema de passagem de mensagem de redes neurais para grafos (*graph neural networks*, GNNs) [Kipf and Welling 2017; Wu et al. 2019] para incorporar estruturas mais complexas que interações par-a-par (e.g., arestas de um grafo). Essa classe de modelos é comumente conhecida como redes neurais topológicas (*topological neural networks*, TNNs) [Papillon et al. 2023]. Essas redes têm alcançado resultados promissores, por exemplo, na descoberta dos mecanismos de evolução do SARS-CoV-2 [Chen et al. 2020] e no desenvolvimento de novos medicamentos [Nguyen et al. 2019].

Notavelmente, [Yang et al. 2022] propuseram *simplex convolutional networks* (SCNs) e *simplicial complex convolutional networks* (SCCNs) como extensões de redes convolucionais em grafos (*graph convolutional networks*, GCNs) [Kipf and Welling 2017] para complexos simpliciais. Nesses modelos, Laplacianas de Hodge são utilizadas para definir noções de vizinhança que caracterizam o procedimento de passagem de mensagem entre elementos (e.g., vértices, arestas, faces) do complexo simplicial. Em resumo, enquanto a operação de passagem de mensagem em GNNs ocorre entre nós, TNNs (para complexos simpliciais) trocam mensagens entre símplices.

Por comportarem estruturas de alta ordem, TNNs também podem ser usadas em conjunção com operações de *lifting* para construir modelos para grafos mais expressivos que GNNs convencionais [Bodnar et al. 2021]. Essa operação de *lifting* tipicamente consiste em criar uma série de símplices (e.g., representando cliques) bem como noções de adjacência (ou estruturas de vizinhança) entre eles.

Como contrapartida ao poder representacional superior de TNNs, o *gap* de complexidade computacional entre TNNs e GNNs aumenta à medida que a ordem do complexo simplicial aumenta — decorrente da passagem de mensagens entre estruturas de ordem superior. Isso torna fundamental o desenvolvimento de TNNs eficientes e escaláveis.

Neste trabalho, apresentamos `SimpleXCN` — um modelo linear para complexos simpliciais, que reduz o custo computacional ao remover não-linearidades entre as camadas de uma SCN. Apesar de sua simplicidade, `SimpleXCN` é competitivo com TNNs populares (e.g., SCNs e SCCNs) para tarefas de classificação de nós, além de reduzir significativamente o uso de memória, o número de parâmetros e o tempo de execução por época. O código está disponível para reprodução no Github[1].

2. PRELIMINARES

Grafos. Um grafo $\mathcal{G}$ é composto por um conjunto de nós $\mathcal{V} = \{1, \ldots, n\}$ e um conjunto de arestas $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Esse relacionamento entre os nós pode ser representado por uma matriz de adjacência $A \in \mathbb{R}^{n \times n}$, onde A_{ij} é igual a 1 se $(i, j) \in \mathcal{E}$, e 0 caso contrário. Além disso, o grafo pode ter atributos associados, geralmente no nível dos nós, representados por uma matriz $X \in \mathbb{R}^{n \times d}$. Dessa forma, um grafo $\mathcal{G}$ pode também ser representado por (A, X). Denotamos por D a matriz diagonal de grau $\mathcal{G}$, onde $D_{ii} \coloneqq \sum_j A_{ij}$.

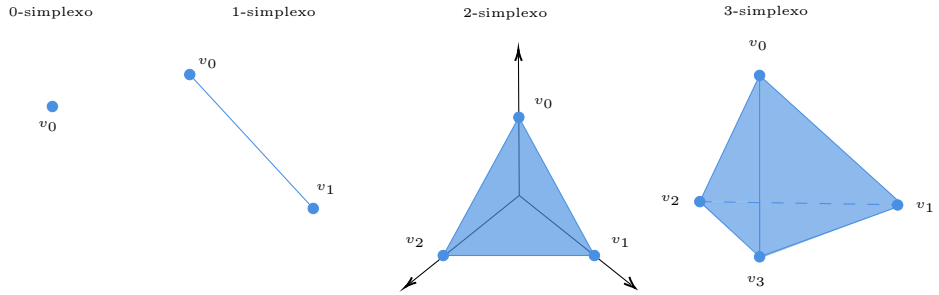

Fig. 1. Representação de simplexos para dimensões até $n = 3$ podemos ver que um 0-simplexo corresponde a um ponto, 1-simplexo um segmento de reta, 2-simplex um triangulo preenchido, 3-simplexo um tetraedro sólido.

Complexos simpliciais. Um *complexo simplicial abstrato* sobre um conjunto de vértices V é um conjunto K de subconjuntos de V (chamados *símplices* ou simplexos) tal que, para todo $\sigma \in K$ e todo $\tau \subset \sigma$ não-vazio, temos que $\tau \in K$. Seja σ um símplice, então seus subconjuntos não-vazios $\tau \subset \sigma$ são chamados de *faces*, e σ é um *coface* de τ. A dimensão de um símplice é igual à sua cardinalidade menos 1, e a dimensão de um complexo simplicial é a dimensão máxima de seus símplices. Denotamos por $K_{[i]}$ o subconjunto de simpliciais de dimensão i de K. Aqui, representamos símplices usando colchetes. Por exemplo, $K = \{[0], [1], [0, 1]\}$ denota um complexo

[1]Código disponível no GitHub: `https://github.com/gjduart/SimpleXCN`

simplicial de dimensão 1 sobre $V = \{0, 1\}$, e os símplices de dimensão 0 dados por [0] e [1] são as faces do símplice $[0, 1]$. A Figura 1 ilustra geometricamente a noção de símplices.

2.1 Elevando grafos a complexos simpliciais

Uma forma simples de elevar o grafo a um complexo simplicial (*lifting*) é criando um complexo de cliques. O complexo de cliques de um grafo $\mathcal{G}$ é o complexo simplicial K onde, se os nós $v_0, \ldots, v_k$ formam um clique em $\mathcal{G}$, então o símplice $[v_0, \ldots, v_k]$ está em K. Há outras maneiras de realizar *lifting* como a estratégia k-hop encontrada em [Kahle 2007].

Para um complexo simplicial finito K, dois d-símplices distintos σ_1 e σ_2 são superiormente adjacentes se ambos são faces de um mesmo $(d+1)$-símplice τ em K. Similarmente, eles são inferiormente adjacentes se ambos contêm algum $(d-1)$-símplice não-vazio η em K como face. Denotamos também por N_r a quantidade de símplices de dimensão r em K.

Uma noção instrumental para a construção de TNNs é a de matrizes de incidência, uma vez que elas são utilizadas para definir estruturas de vizinhança e Laplacianas de Hodge. Formalmente, para toda dimensão (ou rank) $r > 1$, $B_r \in \{-1, 0, 1\}^{N_{r-1} \times N_r}$ é tal que $(B_r)_{ij}$ é ± 1 se o (r)-símplice j é coface do $(r-1)$-símplice i e 0 caso contrário. A escolha do sinal das entradas não-zero é arbitrária e tipicamente dependente da aplicação.

***Lifting* de atributos.** Em geral, atributos são definidos somente para os nós do grafo. Então na elevação de grafos para complexos simpliciais, precisamos definir como os atributos dos nós serão propagados para obtermos atributos dos símplices de dimensão maior que 0. Definimos a matriz de atributos para os símplices de dimensão r recursivamente como:

$$\tilde{X}_r = \begin{cases} B_r^T \tilde{X}_{r-1} & \text{se } r > 1 \\ X & \text{se } r = 0 \end{cases} \tag{1}$$

onde $X \in \mathbb{R}^{N_0 \times d}$ é a matriz de atributos originais dos nós.

2.2 Redes neurais para grafos

Redes neurais para grafos (*graph neural networks*, GNNs) [Scarselli et al. 2009] têm se tornado a abordagem padrão para aprendizagem de representações em grafos. Na abordagem mais popular, GNNs empregam uma sequência de etapas de passagem de mensagens, onde cada nó v agrega mensagens de seus vizinhos $\mathcal{N}(v) = \{u : (v, u) \in \mathcal{E}\}$ e usa o vetor resultante para atualizar sua própria representação. Em particular, começando com $h_v^{(0)} = x_v$ para todo $v \in \mathcal{V}$, as GNNs aplicam recursivamente a regra de atualização

$$h_v^{(\ell+1)} = \mathrm{Upd}_\ell \left(h_v^{(\ell)}, \mathrm{Agg}_\ell(\{\!\{ h_u^{(\ell)} : u \in \mathcal{N}(v) \}\!\}) \right),$$

onde $\{\!\{\cdot\}\!\}$ denota um multiconjunto, Agg_ℓ é uma função invariante à ordem e Upd_ℓ é uma função de atualização arbitrária (normalmente parametrizada usando um perceptron multicamadas). Usando notação matricial, cada camada de uma GNN é uma função definida como $H^{(\ell)} = f(H^{(\ell-1)}, A)$, com $H^{(0)} = X$ e $H^{(L)}$ como a saída após L camadas. A principal diferença entre os modelos está na definição da função f.

Simple graph convolutions. O modelo *simple graph convolution* (SGC) [Wu et al. 2019] consiste de uma simplificação da rede convolucional para grafos (*graph convolutional network*, GCN) [Kipf and Welling 2017]. Utilizando a matriz de adjacência normalizada adicionada de

self-loops, i.e., $\widetilde{A} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}$, uma GCN aplica recursivamente:

$$H^{(\ell)} = \phi\left(\widetilde{A}H^{(\ell-1)}\Theta^{(\ell)}\right), \tag{2}$$

$$= \phi\left(\widetilde{A}\phi\left(\cdots\phi\widetilde{A}\left(\widetilde{A}X\Theta^{(1)}\right)\Theta^{(2)}\ldots\right)\Theta^{(\ell)}\right), \tag{3}$$

onde $H^{(0)} = X$, $\phi(\cdot)$ é uma função de ativação não linear, por exemplo, uma unidade linear retificada (*rectified linear unit*, ReLU), e $\Theta^{(\ell)} \in \mathbb{R}^{d^{(\ell-1)}\times d^{(\ell)}}$ denota os parâmetros do modelo.

Para reduzir o custo computacional de GCNs, o modelo SGC remove as não linearidades entre as camadas da GCN. Como resultado, é possível colapsar os parâmetros do modelo em uma única matriz $\Theta = \Theta^{(1)}\Theta^{(2)}\cdots\Theta^{(L)}$. Desta forma a função de atualização para calcular a matriz de representações do SGC é definida como:

$$H = \widetilde{A}^{L}X\Theta. \tag{4}$$

Surpreendentemente, o modelo SGC mostrou desempenho igual ou superior às GCNs para a tarefa de classificação de nós, mas é mais eficiente e possui menos parâmetros ajustáveis.

2.3 Redes neurais topológicas

Apesar do sucesso das GNNs, o poder das GNNs é limitado pelo teste de isomorfismo de grafos de Weisfeiler-Lehman [Weisfeiler and Leman 1968]. Como resultado, GNNs são incapazes de detectar estruturas topológicas como triângulos ou cliques. Nesse contexto, redes neurais topológicas [Papillon et al. 2023] surgem como uma alternativa promissora para melhorar o poder representacional de GNNs.

Simplex Convolutional Network (SCN, [Yang et al. 2022]). O SCN é um método de aprendizado de representação que usa complexos simpliciais para modelar interações de ordem superior. Ele lida com dependências de alta dimensão, melhorando a generalização em tarefas de classificação e superando as limitações das GNNs em capturar essas interações complexas.

Para $r > 0$, seja $\Delta_r = B_{r-1}^T B_{r-1} + B_r B_r^T$ a Laplaciana de Hodge de dimensão r. Então, a matriz de adjacência correspondente com *self-loops* adicionados é dada por $A_r = \Delta_r + 2I_{N_r}$. Também definimos as matrizes de adjacência normalizadas como $\tilde{A}_r = D_r^{-1/2}A_r D_r^{-1/2}$ para $r = 0\ldots R$, onde $(D_r)_{ii} = \sum_j (A_r)_{ij}$ é a matriz diagonal de graus para os r-símplices. Na camada ℓ e para a dimensão i, a saída da camada do SCN é dada por:

$$H_r^{(\ell)} = \phi(\tilde{A}_r H_r^{(\ell-1)} W_r^{(\ell)}) \tag{5}$$

com $H_r^{(0)} = X_r$. Para a tarefa de classificação de nós, é aplicada uma camada de *readout* feita recursivamente sobre a saída calculada para cada dimensão. Dado $Z_r = H_r^{(L)}$ realizamos o *readout* de dimensões superiores para inferiores da seguinte forma:

$$Z_{r-1} = \left[\phi(B_r Z_r \Theta_r^{(1)}) \parallel H_{r-1}^{(L)}\right]\Theta_r^{(2)} \quad \text{para } 1 \le r \le R \tag{6}$$

onde $\Theta_r^{(1)}$ e $\Theta_r^{(2)}$ são matrizes de pesos treináveis e $\phi(\cdot)$ uma função de ativação não linear. Por fim, a predição para nós é dada por

$$Y_0 = \text{softmax}(Z_0\Theta_0) \tag{7}$$

Em que Θ_0 mapeia os vetores-linha de Z_0 para $\mathbb{R}^{C_0}$ com C_0 denotando o número de classes para os 0-símplices.

No mesmo trabalho, [Yang et al. 2022] também propõem uma variante chamada de simplicial complex convolutional networks (SCCN) que considera troca de mensagem entre símplices de dimensões adjacentes, como em [Hajij et al. 2022]. Nesse caso, SCCN utiliza uma matriz de adjacência completa A dada por

$$A = \begin{bmatrix} \alpha A_0 & \beta B_0 & 0 \\ \beta B_0^T & \alpha A_1 & \beta B_1 \\ 0 & \beta B_1^T & \alpha A_2 \end{bmatrix} \tag{8}$$

A matriz A tem as matrizes de r-adjacência A_r na diagonal principal e as matrizes de incidência B_r nas diagonais superiores e inferiores. As variáveis α e β são parâmetros ajustáveis para diferentes conexões. A matriz A_r captura relações dentro da mesma dimensão, enquanto B_r mostra conexões entre um r-simplex e um $(r-1)$-simplex que é uma face do r-simplex. O SCCN usa a operação convolucional na matriz de adjacência completa A conforme a Equação 5.

3. SIMPLEXCN

Nesta seção, descreveremos uma versão linear do SCN, chamada de *simple simplex convolutional network* ou SimpleXCN. Simplificando o SCN, SimpleXCN remove as não-linearidades entre as camadas intermediárias e colapsa as matrizes de pesos em uma única matriz Θ_r para cada dimensão do complexo simplicial — reduzindo substancialmente o número de parâmetros a serem aprendidos. Além disso, tratamos as matrizes utilizadas na propagação como hiper-parâmetros. Ou seja, assuma que S_r é uma matriz de propagação (ou de difusão) de dimensão r. Então, podemos escolher $S_r = \Delta_r$ (Laplaciana de Hodge), $S_r = A_r = \Delta_r + 2I_{N_r}$ (adjacência) ou $S_r = \tilde{A}_r = D_r^{-1/2} A_r D_r^{-1/2}$ (adjacência normalizada).

Para obtermos as representações dos símplices de dimensão r, primeiro calculamos o resultado $S_{r'}^L X_{r'}$ de L passos de difusão de $\tilde{X}_{r'}$ para toda ordem r', incluindo r. Então, concatenamos às *features* $\left(S_r^L X_r\right)_\sigma$ do r-símplice σ pós-difusão as somas das features de todas as cofaces de σ, também após difusão e separadas por ordem. Finalmente, obtemos os logits para classificação do r-símplice σ usando uma transformação linear Θ_r. O processo pode ser descrito sucintamente para todos os símplices de dimensão r como:

$$Z_r = \left[\left(\Big\|_{m=r+1}^{R} \left(\prod_{j=r+1}^{m} B_j \right) S_m^L X_m \right) \Big\| S_r^L X_r \right] \Theta_r, \tag{9}$$

onde R é dimensão do maior símplice do complexo simplicial. Para classificação de nós, computamos $Y_0 = \text{softmax}(Z_0 \Theta_0)$.

4. EXPERIMENTOS

Nesta seção, detalhamos as configurações experimentais, as etapas de pré-processamento aplicadas, as configurações da arquitetura das redes neurais, a métrica de avaliação e os conjuntos de dados utilizados. Nossa avaliação empírica se concentra no desempenho de redes neurais simpliciais para classificação de nó. Discutimos os resultados para testes como de acurácia, desempenho computacional de forma empírica e teste de hipótese de Wilcoxon.

4.1 Conjuntos de Dados

Para análise foram considerados quatro bancos de dados comumente usados em benchmarks populares: Cora, Citeseer, Pubmed e DBLP [Yang et al. 2016] [Bojchevski and Günnemann

2018] que consistem em conjuntos de dados que representam redes de citação clássicas.

Além disso, utilizamos quatro bancos de dados com heterofilia: *Amazon-ratings*, onde os nós são produtos conectados por co-compra; *Minesweeper*, um grafo 100x100 onde cada nó está ligado a oito vizinhos e a tarefa é identificar minas; e *Questions*, um site de perguntas e respostas com nós representando usuários e arestas conectando usuários que responderam perguntas uns dos outros [Platonov et al. 2023].

Dataset	#Vertices	#Arestas	#Features	#Classes	Train%	Val%	Test%
Cora	$\sim 2,7$K	~ 10K	$1,433$	7	50	25	25
Citeseer	$\sim 3,3$K	~ 9K	$3,703$	6	50	25	25
Pubmed	$\sim 19,7$K	~ 88K	500	3	50	25	25
DBLP	$\sim 17,7$K	~ 105K	1,639	4	50	25	25
Amazon-ratings	$\sim 24,5$K	~ 93K	300	5	50	25	25
Minesweeper	$10,0$K	~ 39K	7	2	50	25	25
Questions	$\sim 48,9$K	$\sim 153,5$K	301	2	50	25	25

Tabela I. Propriedades dos conjuntos de dados. Apresentamos a quantidade aproximada de nós, arestas, características e classes para cada conjunto de dados. Também descrevemos como os dados são distribuídos entre os conjuntos de teste, validação e treinamento. A proporção de divisão para cada conjunto de dados: metade dos dados é usada para treinamento, 25% é usada para validação, e 25% é usada para teste.

4.2 Configuração Experimental

Para a comparação de desempenho, utilizamos os modelos SCN e SCCN [Yang et al. 2022], todos com pelo menos duas camadas de convolução. Para os modelos da literatura, realizamos um *gridsearch* para encontrar os melhores hiperparâmetros. No `Simple`XCN, usamos o Optuna [Akiba et al. 2019] para ajustar o grau da propagação (2 a 7), a taxa de aprendizado (0.001 a 0.01), o *weight decay* (0, 5e-4, 5e-6), e a ordem do *lifting* (2), considerando apenas triângulos.

Para as implementações, usamos o *framework* TopoBenchmarkX [PyT-Team 2024], que fornece todos os insumos necessários para calcular o complexo simplicial através da elevação por clique e da elevação de características via soma projetiva.

Executamos cada conjunto de dados por 5 vezes. Dividimos na proporção de 50% dos pontos de dados para treinamento, 25% para validação e 25% para teste. A média e o desvio padrão de cada métrica de desempenho são calculados para cada conjuntos de dados, os resultados estão na Tabela II. Podemos ver que não há clareza de qual é a melhor arquitetura dentre essas. Há uma alternância entre os métodos sob qual é o melhor e o segundo melhor em cada banco. Onde o `Simple`XCN se mantém competitivo.

	SCN	SCCN	`Simple`XCN
Cora	86.3 ± 1.2	85.9 ± 0.3	86.1 ± 0.6
Citeseer	73.3 ± 1.7	73.0 ± 1.1	73.9 ± 1.6
PubMed	87.2 ± 0.4	86.5 ± 0.6	86.9 ± 0.7
DBLP	82.0 ± 0.2	82.9 ± 0.9	81.9 ± 0.3
Amazon	46.9 ± 0.3	44.7 ± 0.9	42.4 ± 0.7
Minesweeper	89.5 ± 0.3	89.8 ± 0.3	87.4 ± 0.3
Questions	65.1 ± 2.4	68.5 ± 2.5	68.0 ± 0.7

Tabela II. A comparação de várias arquiteturas em múltiplos bancos de dados. Cada entradas da tabela apresenta a média e o desvio-padrão das acurácias. Apesar de não ter uma melhora significativa o `Simple`XCNé competitivo comparativamente com as outras duas arquiteturas, embora seja bem mais simples.

	SCN	SCCN	SimpleXCN
DBLP	0.49 seg/it	0.60 seg/it	**0.47** seg/it
PubMed	0.11 seg/it	0.11 seg/it	0.11 seg/it
Questions	1.41 seg/it	1.52 seg/it	**0.28** seg/it
Amazon	0.27 seg/it	0.26 seg/it	**0.21** seg/it

Tabela III. Comparação de tempo por iteração entre as arquiteturas. Foram realizados testes em 4 conjuntos de dados, e são exibidos o tempo médio para cada arquitetura em 10 iterações. Nota-se que houve uma redução significativa no tempo de execução do conjunto Questions.

4.3 Teste empírico de desempenho computacional

Os experimentos foram executados em uma GPU RTX4060. Para o teste de peformance medimos o pico de memória alocada, número de parâmetros e o tempo médio para cada arquitetura. Estes resultados são mostrados nas tabelas IV e III.

	SCN	SCCN	SimpleXCN
DBLP	173 K / 1.29 GB	178 K / 1.29 GB	**22.9 K / 1.11 GB**
PubMed	64.3 K/ 0.46 GB	69.3 K / 0.46 GB	**5.5 K / 0.31 GB**
Questions	45.2 K / 4.98 GB	50.1 K / 4.98 GB	**2.4 K / 0.61 GB**
Amazon	45.2 K/ 0.99 GB	50.1 K / 0.98 GB	**5.1 K / 0.40 GB**

Tabela IV. O número total de parâmetros para cada arquitetura e o pico de memória alocada para sua execução. Podemos ver uma redução clara que chega a ser de 10 vezes no número de parâmetros e, consequentemente, uma redução significativa no uso de memória que chega a ser de 50%.

		Cora	Citeseer	PubMed	DBLP	Amazon	Questions	Minesweeper
SCN	Statistic	3.0	5.0	2.0	7.0	0.0	2.0	0.0
	p-value	0.4652	0.6250	0.1875	1.0000	0.0625	0.1875	0.0625
SCCN	Statistic	3.0	7.0	5.0	2.0	0.0	6.0	0.0
	p-value	0.4652	1.0000	0.6250	0.1875	0.0625	0.8125	0.0625

Tabela V. Apresenta os resultados do teste de Wilcoxon aplicado a dois métodos distintos, SCN e SCCN, avaliados em diversos datasets: Cora, Citeseer, PubMed, DBLP, Amazon, Question e Minesweeper. Este teste foi utilizado para verificar se existem diferenças estatisticamente significativas entre as condições comparadas.

Os resultados dos testes de Wilcoxon [Wilcoxon 1945] para ambos os métodos SCN e SCCN indicam que, em geral, não há diferenças estatisticamente significativas entre as condições comparadas nos datasets analisados. A maioria dos p-valores está acima de 0.05, sugerindo que as diferenças observadas podem ser atribuídas ao acaso e não representam um efeito real significativo. Apenas alguns datasets apresentaram p-valores próximos ao limiar de significância, indicando uma tendência marginal para diferenças, mas sem evidência suficiente para rejeitar a hipótese nula com um nível de confiança padrão.

5. CONCLUSÃO E TRABALHOS FUTUROS

O SimpleXCN apresenta-se como uma simplificação eficaz do SCN para classificação de nós, removendo não linearidades entre as camadas e utilizando uma única matriz de pesos. Apesar das simplificações, o SimpleXCN demonstrou-se competitivo em diversos conjuntos de dados, mantendo um desempenho próximo ou equivalente às arquiteturas mais complexas como SCN e

SCCN. Os resultados experimentais indicam que, embora não haja uma arquitetura claramente superior, o `SimpleXCN` oferece uma alternativa mais eficiente, tanto em termos de desempenho computacional quanto de precisão, especialmente em contextos onde a simplicidade e a eficiência são priorizadas. As análises estatísticas sugerem que as diferenças entre os métodos não são estatisticamente significativas na maioria dos casos, reforçando a viabilidade do `SimpleXCN` como uma solução prática e eficiente para a classificação de nós em complexos simpliciais.

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. pp. 2623–2631, 2019.

Bodnar, C., Frasca, F., Wang, Y., Otter, N., Montufar, G. F., Lió, P., and Bronstein, M. Weisfeiler and Lehman go topological: Message passing simplicial networks. In *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang (Eds.). Proceedings of Machine Learning Research, vol. 139. PMLR, pp. 1026–1037, 2021.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

Chen, J., Wang, R., Wang, M., and Wei, G.-W. Mutations strengthened sars-cov-2 infectivity. *Journal of Molecular Biology* 432 (19): 5212–5226, 2020.

Hajij, M., Zamzmi, G., Papamarkou, T., Miolane, N., Guzmán-Sáenz, A., and Ramamurthy, K. N. Higher-order attention networks. *arXiv preprint arXiv:2206.00606* 2 (3): 4, 2022.

Horn, M., De Brouwer, E., Moor, M., Moreau, Y., Rieck, B., and Borgwardt, K. Topological graph neural networks. In *International Conference on Representation Learning (ICLR)*, 2022.

Kahle, M. The neighborhood complex of a random graph. *Journal of Combinatorial Theory, Series A* 114 (2): 380–387, 2007.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Nguyen, D. D., Cang, Z., Wu, K., Wang, M., Cao, Y., and Wei, G.-W. Mathematical deep learning for pose and binding affinity prediction and ranking in d3r grand challenges. *Journal of Computer-Aided Molecular Design* 33 (1): 71–82, 2019.

Papamarkou, T., Birdal, T., Bronstein, M., Carlsson, G., Curry, J., Gao, Y., Hajij, M., Kwitt, R., Liò, P., Lorenzo, P. D., Maroulas, V., Miolane, N., Nasrin, F., Ramamurthy, K. N., Rieck, B., Scardapane, S., Schaub, M. T., Veličković, P., Wang, B., Wang, Y., Wei, G.-W., and Zamzmi, G. Position: Topological deep learning is the new frontier for relational learning. In *International Conference on Machine Learning (ICML)*, 2024.

Papillon, M., Sanborn, S., Hajij, M., and Miolane, N. Architectures of topological deep learning: A survey on topological neural networks. *ArXiv e-prints*, 2023.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. A critical look at evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

PyT-Team. Topobenchmarkx, 2024.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks* 20 (1): 61–80, 2009.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series* 2 (9): 12–16, 1968.

Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin* 1 (6): 80–83, 1945.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov (Eds.). Proceedings of Machine Learning Research, vol. 97. PMLR, pp. 6861–6871, 2019.

Yang, R., Sala, F., and Bogdan, P. Efficient representation learning for higher-order data with simplicial complexes. In *Proceedings of the First Learning on Graphs Conference*, B. Rieck and R. Pascanu (Eds.). Proceedings of Machine Learning Research, vol. 198. PMLR, pp. 13:1–13:21, 2022.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, pp. 40–48, 2016.