Redes Neurais de Grafos para Sistemas de Recomendação

Matheo Angelo Pereira Dantas

Feito como parte do Trabalho Final
da disciplina SCC0284 - Sistemas
de Recomendação do ICMC-USP

Sumário

1 Introdução

1.1 Motivação

Com o advento da era digital, as pessoas agora têm muito mais possibilidades de acesso a informação e diferentes formas de mídia. Entretanto, ao mesmo tempo que traz uma liberdade maior, esse crescimento exponencial de dados disponíveis é acompanhada de uma grande sobrecarga de informação para os usuários: com cerca de 10.000 filmes na Netflix, 12 milhões de produtos na Amazon e 10 bilhões de vídeos no YouTube, é logisticamente impossível para as pessoas procurar por todas as informações disponíveis na *internet* para encontrarem o que querem.

A área de Sistemas de Recomendação foi criada para lidar com esse desafio. Os Sistemas de Recomendação usam algoritmos para explorar padrões no comportamento dos usuários e suas preferências, e a partir disso, buscar de forma automatizada conteúdos que possam melhor suprir as suas necessidades.

Ao mesmo tempo, os modelos de Redes Neurais de Grafos vêm evoluindo rapidamente e encontrado aplicações de estado-da-arte em diversas áreas do conhecimento, como descoberta de medicamentos[1], táticas esportivas[2] e, é claro, Sistemas de Recomendação[3, 4]. Ao contrário de técnicas tradicionais, que funcionam a partir de comparações simples entre usuários e/ou itens, essas redes se destacam na área de Sistemas de Recomendação porque conseguem explorar padrões de conectividade mais complexos nos dados.

1.2 O que esperar deste material

Este material busca fornecer uma introdução às principais aplicações de Redes Neurais de Grafos (GNNs) em Sistemas de Recomendação, com foco em métodos de filtragem colaborativa e na arquitetura LightGCN. Aqui, fornecemos os principais conceitos de Sistemas de Recomendação, bem como de Aprendizado de Máquina em Grafos, necessários para o estudo do tema.

1.2.1 Conhecimentos prévios esperados

O material busca introduzir conceitos novos do zero sempre que possível, mas assume-se que a pessoa que utiliza o material tem familiaridade com metodologias do Aprendizado de Máquina no geral, bem como conteúdos básicos de matemática do Ensino Superior (funções, noções básicas de Probabilidade, Álgebra Linear, etc.),

1.3 Definições básicas de grafos

Um grafo é definido como um objeto $G = (V, E)$, onde V é o conjunto de pontos do grafo (chamados de "nós" ou "vértices") e $E \subseteq (V \times V)$ é o conjunto de conexões do grafo, chamadas de "arestas", dadas por pares de vértices. Esses pares podem ser ordenados onde dizemos que o grafo é direcionado, e caso contrário (onde a ordem não importa), dizemos que o grafo é não direcionado, como ilustrado na Figura 1. Aqui, vamos lidar com grafos não-direcionados.

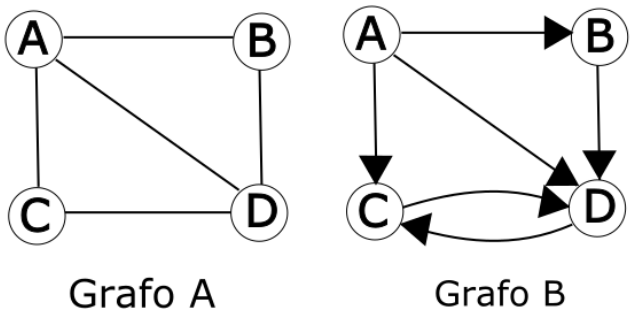
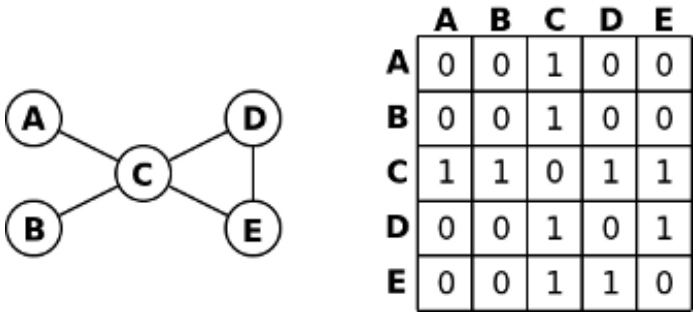


Figura 1: Comparação entre um grafo não-direcionado (Grafo A) e um grafo direcionado (Grafo B). Enquanto no Grafo A os pontos são conectados por linhas simples, no Grafo B, cada conexão é uma seta que sai de um ponto e vai para outro.

Quando um nó v está conectado a outro nó u, dizemos que v é vizinho de u, ou adjacente a u. Essas adjacências podem ser representadas computacionalmente de diversas formas, sendo uma das principais a matriz de adjacências. A matriz de adjacências de G é definida na forma $A_{|V|\times|V|}$, onde $a_{i,j} = 1$ se há uma conexão entre os nós i e j, e $a_{i,j} = 0$ caso contrário, como ilustrado na Figura 2.



	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

Figura 2: Exemplo de uma matriz de adjacência em um grafo não-direcionado.

1.4 Grafos em Sistemas de Recomendação

Grafos podem ser naturalmente usados para representar dados em sistemas de recomendação. Eles podem aparecer de diversas formas, mas a mais comum é o grafo usuário-item.

Como ilustrado na Figura 3, o grafo usuário-item é um grafo bipartido, ou seja, podemos particionar o conjunto de nós V em duas partes U, I de forma que as arestas $A \subseteq (U \times I)$ sempre ligam os nós de U aos nós de I. Mais especificamente, os nós do grafo usuário-item são dados pelo conjunto U de usuários e I de itens, de forma que uma ligação (u, i) no grafo indica que o usuário u interagiu com o item i. Assim, o Sistema

de Recomendação visa resolver um problema de **predição de ligações**: sabemos quais usuários interagiram com quais itens, e o objetivo é descobrir quais ligações do grafo estão "faltando" e, a partir disso, encontrar potenciais itens nos quais cada usuário pode se interessar futuramente.

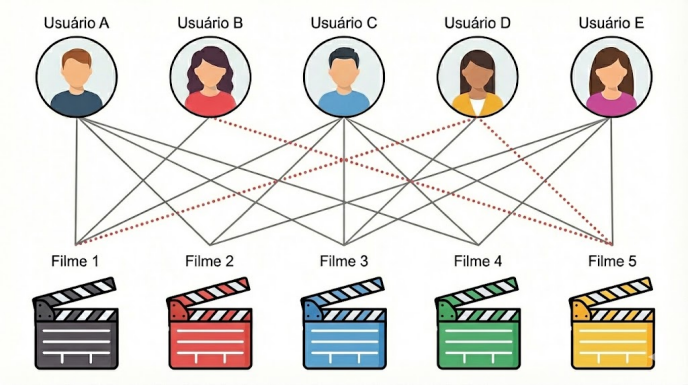


Figura 3: Exemplo de um grafo usuário-item, onde os itens são filmes. Sabemos os filmes que cada usuário assistiu, através das arestas pretas do grafo, e queremos sugerir filmes novos que os usuários talvez queiram assistir, prevendo, por exemplo, se as arestas pontilhadas vermelhas podem ser arestas faltantes.

Quando estamos trabalhando com grafos bipartidos, também é comum termos as adjacências do grafo representadas como uma matriz $A_{|U|\times|I|}$, onde $a_{u,i}$ indica se o usuário da linha u interagiu com o item da coluna i (com valor 1 se há interação, e 0 caso contrário).

1.5 Organização do Texto

A seguir, detalhamos como serão estruturadas as próximas seções do texto. Primeiro, estudaremos alguns conceitos iniciais de Sistemas de Recomendação e as principais técnicas baseadas em padrões de conectividade, com foco na ideia de Filtragem Colaborativa, que será aprofundada mais adiante, além de métricas de desempenho e funções-objetivo utilizadas na otimização matemática desses sistemas. Em seguida, entraremos na parte dedicada às aplicações com Redes Neurais de Grafos, onde serão estudadas arquiteturas específicas da área de Sistemas de Recomendação e como elas conseguem superar certas limitações de outros métodos. Por fim, haverá uma seção com exercícios teóricos e um link para um tutorial prático, juntamente com outros materiais sugeridos.

2 Técnicas de Sistemas de Recomendação

2.1 Filtragem Colaborativa

A Filtragem Colaborativa (FC) é a abordagem mais conhecida em Sistemas de Recomendação. A FC explora o "conhecimento da multidão", de forma que o histórico de interações entre usuários e itens no sistema, por si só, já possui padrões que podem ser usados para gerar recomendações de itens novos para os usuários. Para isso, esses modelos buscam recomendar itens similares a usuários com históricos de interação similares, como ilustrado na Figura 4.

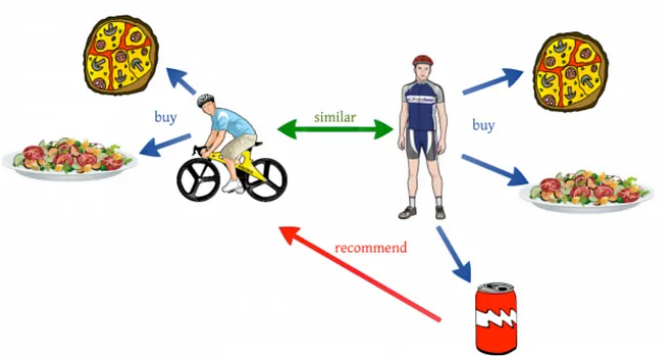


Figura 4: Exemplo de uma recomendação baseada em filtragem colaborativa. Na figura, vemos que os dois usuários têm em comum o interesse em pizza salada, então supomos que são usuários com preferências parecidas. Assim, dado que o usuário da direita também comprou uma lata de refrigerante, o sistema reconhece o refrigerante como um interesse em potencial do usuário da esquerda e faz a recomendação.

2.1.1 Modelos de fatoração de matrizes

Uma técnica de filtragem colaborativa particularmente importante de se entender antes de nos aprofundarmos nas aplicações com GNNs é a fatoração de matrizes.

A Fatoração de Matrizes é uma técnica de filtragem colaborativa que projeta usuários e itens em um espaço de características latentes compartilhado de dimensão K, onde K é muito menor que o número de usuários ou itens. A premissa é que as interações observadas podem ser explicadas por fatores ocultos (como a preferência de um usuário por um gênero específico e a presença desse gênero no item).

O modelo associa a cada usuário u um vetor $\mathbf{p}_u \in \mathbb{R}^K$ e a cada item i um vetor $\mathbf{q}_i \in \mathbb{R}^K$. A matriz de interações original R é então aproximada pelo produto dessas matrizes de baixa dimensão:

$$R \approx P \times Q^T$$

A estimativa da preferência ou nota $\hat{r}_{ui}$ de um usuário u por um item i é calculada

através do produto escalar entre seus respectivos vetores latentes:

$$\hat{r}_{ui} = \mathbf{p}_u \cdot \mathbf{q}_i = \sum_{f=1}^{K} p_{u,f} \cdot q_{i,f}$$

Valores altos nesse produto escalar indicam que os vetores apontam para direções similares no espaço latente, sugerindo alta compatibilidade entre o usuário e o item.

2.2 Métricas de avaliação

Nesta seção, definimos as métricas utilizadas para avaliar a qualidade das recomendações geradas. Consideramos um cenário de recomendação top-K, onde para cada usuário u, o modelo gera uma lista ordenada de itens recomendados $\hat{R}_u(K)$ de tamanho K. Seja R_u o conjunto de itens relevantes (ground-truth) para o usuário u.

2.2.1 Recall@K e Precision@K

A precisão e a revocação (recall) são métricas fundamentais para avaliar a relevância dos itens recuperados. A **Precision@K** mensura a proporção de itens recomendados no top-K que são, de fato, relevantes:

$$\text{Precision@}K(u) = \frac{|\hat{R}_u(K) \cap R_u|}{|\hat{R}_u(K)|} = \frac{|\hat{R}_u(K) \cap R_u|}{K}$$

Por outro lado, o **Recall@K** mensura a capacidade do modelo de encontrar os itens relevantes disponíveis no conjunto de teste:

$$\text{Recall@}K(u) = \frac{|\hat{R}_u(K) \cap R_u|}{|R_u|}$$

Enquanto a precisão foca na qualidade da lista apresentada, o recall foca na cobertura dos interesses do usuário.

2.2.2 NDCG (Normalized Discounted Cumulative Gain)

O *Normalized Discounted Cumulative Gain* (NDCG) é uma métrica que, diferentemente da precisão e do recall, leva em consideração a **posição** dos itens relevantes na lista recomendada. Assume-se que itens relevantes que aparecem no topo da lista são mais valiosos.

O DCG no corte K é definido como:

$$\text{DCG@}K(u) = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Onde rel_i indica a relevância do item na posição i (geralmente $rel_i = 1$ se o item é relevante e 0 caso contrário). Para normalizar o valor entre $[0, 1]$, dividimos pelo Ideal DCG (IDCG), que representa a ordenação perfeita dos itens relevantes:

$$\text{NDCG@}K(u) = \frac{\text{DCG@}K(u)}{\text{IDCG@}K(u)}$$

2.2.3 BPR (Bayesian Personalized Ranking)

As funções citadas anteriormente, como Recall@K e NDCG, não são diferenciáveis, então não podemos utilizá-las diretamente como objetivo de treinamento em modelos que são treinados por otimização de gradientes (como as redes neurais, que veremos mais adiante). Em geral, a alternativa mais adequada no contexto de Sistemas de Recomendação é o **BPR (Bayesian Personalized Ranking)**.

O BPR é uma função de custo projetada para otimizar o *ranking* de itens por meio de comparações por pares (*pairwise ranking*). O objetivo é garantir que itens com os quais o usuário interagiu recebam pontuações maiores do que itens desconhecidos.

Matematicamente, definimos o conjunto de todos os usuários como U e o conjunto de todos os itens como I. Para cada usuário $u \in U$, denotamos I_u^+ como o conjunto de itens positivos (observados) e $I_u^- = I \setminus I_u^+$ como o conjunto de itens negativos (não observados).

A função de perda percorre todos os usuários, todos os seus itens positivos e compara cada um com os itens negativos, buscando maximizar a diferença de pontuação $\hat{x}_{ui} - \hat{x}_{uj}$. A formulação explícita com três somatórios aninhados é dada por:

$$L_{BPR} = -\sum_{u \in U} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \ln \sigma(\hat{x}_{ui} - \hat{x}_{uj}) + \lambda ||\Theta||^2$$

Onde $\sigma(z) = \frac{1}{1+e^{-z}}$ converte a diferença de pontuação em uma probabilidade, e $\lambda||\Theta||^2$ é o termo de regularização (baseado na Regressão Ridge[5]).

Essa estrutura força o modelo a aprender que, para qualquer par (i, j) onde i foi consumido e j não, a desigualdade $\hat{x}_{ui} > \hat{x}_{uj}$ deve ser verdadeira.

3 Redes Neurais de Grafos para Sistemas de Recomendação

3.1 Introdução às Redes Neurais de Grafos

As Redes Neurais de Grafos, também chamadas de GNNs (do inglês, *Graph Neural Networks*), são um tipo de modelo de Aprendizado de Máquina que tem se desenvolvido rapidamente em aplicações não apenas de Sistemas de Recomendação, como em diversas outras áreas. As Redes Neurais Profundas, de modo geral, têm tido sucesso em tarefas que usam dados complexos, como imagens e textos, pois podem aproximar funções complexas a partir da composição de funções mais simples, as camadas da rede neural, e analisar diferentes níveis de abstração. As GNNs, em particular, recebem como entrada um grafo: um conjunto de pontos (nós) e um conjunto de conexões entre pares de nós (arestas).

As GNNs operam usando um mecanismo de "passagem de mensagem" para codificar a estrutura de conectividade local de cada nó, conforme ilustrado na Figura 5. Durante cada iteração de passagem de mensagens i a partir da vizinhança do grafo $\mathcal{A}(u)$, em uma GNN, um vetor de características ocultas $\mathbf{h}_u^{(i)}$ correspondente a cada nó $u \in \mathcal{V}$ é atualizado de acordo com as informações agregadas da vizinhança:

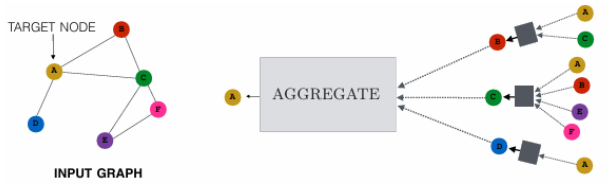


Figura 5: Ilustração do mecanismo de passagem de mensagem em uma GNN. Na figura, o nó A recebe mensagens dos seus vizinhos B, C e D, que por sua vez, receberam mensagens das suas respectivas vizinhanças. Assim, com 2 camadas de passagem de mensagem, o nó A guarda informação de até 2 nós de distância. Fonte: Livro Graph Representation Learning [7].

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right)\end{aligned}$$

onde UPDATE e AGGREGATE são funções diferenciáveis arbitrárias (ou seja, camadas de rede neural) e $\mathbf{m}_{\mathcal{N}(u)}$ é a "mensagem" que é agregada da vizinhança $\mathcal{N}(u)$ do nó u, ou seja, o conjunto de nós diretamente ligados ao nó u por arestas do grafo. Os vetores iniciais em $k = 0$ são definidos como as características de entrada para todos os nós, ou seja, $\mathbf{h}_u^{(0)} = \mathbf{x}_u, \forall u \in \mathcal{V}$.

Note que como a função AGGREGATE recebe um *conjunto* como entrada, GNNs definidas desta forma (no caso da classificação de nós) são equivariantes a permutações[6].

Temos como um exemplo bastante popular de arquitetura a Graph Convolutional Network (GCN)[8]. A GCN surgiu dentro de uma linha de pesquisa que visava adaptar as convoluções das Convolutional Neural Networks (CNNs)[9] usando filtros laplacianos no domínio de frequência do grafo. Sua equação de passagem de mensagem, na forma matricial, é dada por:

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

Onde $H^{(l)}$ é a matriz de representações latentes dos nós na L-ésima camada de GNN, $\tilde{A}$ é a matriz de adjacência do grafo com *self-loops* (ou seja, cada nó é artificialmente ligado a si mesmo), $\tilde{D}$ é a matriz diagonal com os graus dos nós em $\tilde{A}$, $W^{(l)}$ é a matriz de pesos aplicada na passagem de mensagem, e σ é uma função de ativação não-linear aplicada elemento-a-elemento (por exemplo, a ReLU - Rectified Linear Unit[10]).

Também podemos expressar a equação da passagem de mensagem para cada nó:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}^{(l)} \mathbf{h}_j^{(l)}\right)$$

Onde $h_i^{(l)}$ é o valor de $H^{(l)}$ na i-ésima linha, d_i é o grau do nó i, e $\mathcal{N}(i)$ é o conjunto de nós vizinhos de i.

Os valores das matrizes $W^{(l)}$ são otimizados em uma função objetivo diferenciável (como, no nosso caso, o BPR) usando métodos de otimização não-linear próprios para treinamento de redes neurais, como o Adam[11]. Os gradientes são computados através do algoritmo Backpropagation[12].

3.2 NGCF (Neural Graph Collaborative Filtering)

O NGCF[13] foi a primeira arquitetura de GNN a ser diretamente usada para a Filtragem Colaborativa. Foi criado a fim de permitir com que a Filtragem Colaborativa, que já era bem-sucedida mesmo com *embeddings* "rasos" de fatoração de matriz e métodos similares, pudesse também capturar padrões de conectividade mais complexos nas interações entre usuários e itens.

Inicialmente, o NGCF cria *embeddings* "rasos" dos usuários e itens como *features* de nós. Mas antes de aplicar a similaridade de produto interno para reconstruir a matriz de interações, como na fatoração de matrizes, o modelo calcula os fatores latentes usando camadas de GNN:

$$e_u^{(k+1)} = \sigma\left(W_1 e_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)})\right)\right)$$

$$e_i^{(k+1)} = \sigma\left(W_1 e_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)})\right)\right)$$

Onde $\mathcal{N}_u$ é o conjunto de todos os vizinhos do usuário u (itens curtidos por u); $\mathcal{N}_i$ é o conjunto de todos os vizinhos do item i (usuários que curtiram i); $e_u^{(k)}$ é o embedding do usuário na k-ésima camada; $e_i^{(k)}$ é o embedding do item na k-ésima camada; e W_1, W_2 são matrizes de pesos treináveis. Além disso, $\odot$ simboliza a multiplicação elemento-a-elemento de dois vetores de mesma dimensão, operação conhecida como Produto de Hadamard.

3.3 LightGCN

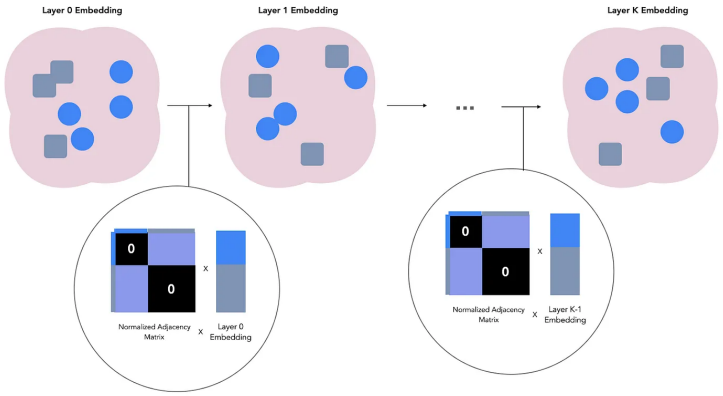


Figura 6: Ilustração das camadas de propagação do LightGCN. Fonte: "LightGCN with PyTorch Geometric".

A LightGCN[3] é uma adaptação do NGCF que, ao mesmo tempo em que é mais simples, atinge um desempenho melhor, sendo o atual estado-da-arte em Filtragem Colaborativa baseada em GNNs. Na LightGCN, a não-linearidade é removida e os pesos da passagem de mensagem também, sendo os embeddings de usuário e item os únicos pesos treináveis da rede. Assim, a passagem de mensagem se resume a calcular iterativamente as médias de *embedding* na vizinhança de cada nó:

$$e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(k)}, \qquad e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k)}$$

Como ilustrado na Figura 6, isso é equivalente a multiplicar a matriz de embeddings E pela matriz de adjacência normalizada sucessivas vezes:

$$E^{(k+1)} = \left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) E^{(k)}$$

Por fim, o embedding final de cada nó é uma média ponderada dos embeddings obtidos a cada camada de passagem de mensagem:

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^{(k)}, \qquad e_i = \sum_{k=0}^{K} \alpha_k e_i^{(k)} \tag{1}$$

Onde e_u e e_i são as representações finais (embeddings) do usuário e do item, respectivamente; α_k denota o peso de importância da k-ésima camada (hyperparâmetro ou parâmetro treinável); e K é o número total de camadas de propagação. Na versão padrão do LightGCN, usa-se $\alpha_k = \frac{1}{K+1}$ em todos os α_k (ou seja, fazemos uma média simples de todas as camadas).

3.4 PinSage

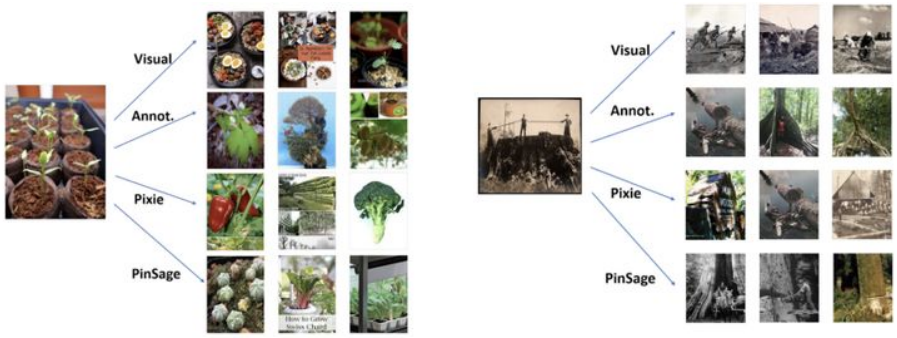


Figura 7: Recomendações de imagens a partir de uma dada imagem, em sistemas do Pinterest. Cada seta indica um algoritmo de recomendação diferente, usando a imagem à esquerda para recomendar as imagens à direita. Figura retirada do artigo original do PinSage[4].

Como uma leitura complementar para aprofundamento e uma curiosidade bastante interessante, sugerimos o artigo original do modelo PinSage[4]. O PinSage é um Sistema de Recomendação usado no Pinterest, rede social focada em pesquisas por imagens onde usuários organizam fotos de interesse em álbuns: itens são *pins* (“alfinetes”) e pastas são *boards* (quadros).

Diferentemente dos modelos abordados neste material, é um modelo de filtragem híbrida, ou seja, combina os padrões de interação entre itens e usuários ou pastas (o que é utilizado na filtragem colaborativa) juntamente com *features* dos usuários e itens (filtragem baseada em conteúdo). O resultado é um algoritmo inteligente que, ao mesmo tempo em que consegue extrair características sofisticadas das imagens com técnicas de Visão Computacional e Processamento de Linguagem Natural, consegue interpretá-las de forma mais correta usando o contexto das imagens que tendem a aparecer nos mesmos álbuns.

O PinSage também é uma das aplicações de GNNs de maior escala na atualidade, sendo aplicado originalmente em um grafo com 2 bilhões de *pins*, 1 bilhão de *boards* e 18 bilhões de arestas. Assim, o modelo também explora técnicas avançadas de escalabilidade, como o treinamento em *mini-batches* e métodos otimizados de amostragem negativa.

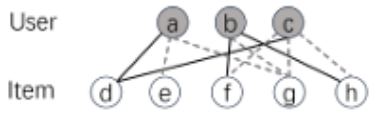
4 Exercícios

4.1 Exercícios teóricos sobre o LightGCN

Nesta subseção, temos exercícios sobre LightGCN obtidos das listas de exercícios da disciplina *CS224W - Machine Learning on Graphs*, oferecida na Universidade de Stanford. Ao lado de cada nome de exercício, há o oferecimento da disciplina de onde o exercício foi obtido (apesar de os exercícios daqui terem adaptações ao material utilizado diretamente no curso). O material completo de todos os oferecimentos da disciplina está disponível publicamente na internet[1].

Exercício 1 - Tarefa de Recomendação *(Homework 3, Fall 2023)*

Recebemos o seguinte grafo bipartido. Arestas sólidas representam interações usuário-item já observadas, enquanto arestas pontilhadas denotam o conjunto de interações positivas no futuro.



Os embeddings unidimensionais para os nós estão listados na tabela a seguir:

a	b	c	d	e	f	g	h
0.3	0.96	0.7	0.6	0.4	0.8	0.7	0.64

Recebemos um modelo de recomendação que usa a distância L2 (menor distância significa maior recomendação) entre embeddings de usuário e item para calcular as pontuações.

User	d	e	f	g	h	P_u	R_u	Recall@2
a								
b								
c								

Calcule a pontuação Recall@2 para os usuários a, b, c. Para cada usuário, escreva explicitamente os itens positivos P_u e os itens recomendados R_u, como conjuntos da forma {X, Y, Z}. Por favor, exclua itens já interagidos em R_u, pois estamos interessados apenas em recomendar itens ainda não interagidos. **Preencha suas respostas na tabela a seguir, onde as primeiras 5 colunas representam as distâncias usuário-item (para d, e, f, g, h). Qual é o Recall@2 final?**

[1] https://web.stanford.edu/class/cs224w/

Exercício 2 - Autoconexão *(Homework 3, Fall 2025)*

Denotamos o embedding de um item i na camada k como $e_i^{(k)}$ e o de um usuário u como $e_u^{(k)}$.

A operação de convolução em grafos (também conhecida como regra de propagação) no LightGCN é definida como:

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(k)} \quad \text{e} \quad e_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e_u^{(k)}$$

O termo de normalização simétrica segue o design do GCN padrão, o que evita que a escala dos embeddings aumente com as operações de convolução em grafos.

No entanto, pelas equações acima, podemos ver que no LGCN apenas agregamos os vizinhos conectados e não integramos o próprio nó alvo (ou seja, não há autoconexão), diferentemente da maioria das operações de convolução em grafos existentes, que normalmente agregam mensagens dos vizinhos junto com a autoconexão. Isso acontece porque o LightGCN usa outras operações que têm o mesmo efeito.

Qual operação do LightGCN captura o mesmo efeito que a autoconexão, e por quê?

Exercício 3 - Relação com APPNP *(Homework 3, Fall 2025)*

Existe um trabalho[14] que conecta GCN com Personalized PageRank, onde os autores propõem uma variante do GCN chamada APPNP que pode propagar longo alcance sem o risco de *oversmoothing*. O *oversmoothing*[15] ("suavização excessiva") é um problema que afeta GNNs fazendo com que as *features* de nós fiquem cada vez mais parecidas ao longo das iterações de passagem de mensagem, deteriorando a capacidade da GNN de distinguir nós de classes diferentes ou com propriedades diferentes.

Inspirado pelo design de teletransporte no modelo Personalized PageRank[16], onde o passeio aleatório no grafo pode envolver um "teletransporte" aleatório para qualquer nó do grafo com probabilidade β, o APPNP complementa cada camada de propagação com as características iniciais (i.e., os embeddings da camada 0), o que pode equilibrar a necessidade de preservar a localidade (i.e., ficar perto do nó raiz para aliviar o *oversmoothing*) e aproveitar a informação de uma grande vizinhança.

A camada de propagação no APPNP é definida como:

$$E^{(k+1)} = \beta E^{(0)} + (1 - \beta)\tilde{A}E^{(k)}$$

onde β é chamado de "probabilidade de teletransporte" para controlar a retenção das características iniciais na propagação, e $\tilde{A}$ denota a matriz de adjacência normalizada.

Alinhando com a Equação (1), podemos ver que, definindo α_k adequadamente, o LightGCN pode recuperar totalmente o embedding de predição usado pelo APPNP. Dessa forma, o LightGCN compartilha a força do APPNP em combater o *oversmoothing*; definindo α corretamente, o LightGCN permite usar um K grande para modelagem de longo alcance com *oversmoothing* controlável.

Expresse os embeddings da camada K, $E^{(K)}$, do APPNP como uma função dos embeddings iniciais $E^{(0)}$ e da matriz de adjacência normalizada A.

4.2 Outros materiais

Tutorial prático de LightGCN: usando como base um material preparado por alunos de Stanford, foi criado um tutorial prático em Português no Google Colab, aplicando o LightGCN em um conjunto de dados real para fazer recomendações de filmes. Clique aqui para acessar e executar o código.

Videoaulas: para quem preferir assistir videoaulas ao invés de ler o material escrito pode estudar por uma playlist criada no YouTube para este trabalho. Também há uma aula da Universidade de Stanford (em Inglês) que foi usada como base para a criação deste material.

Referências

[1] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, *et al.*, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688–702, 2020.

[2] Z. Wang, P. Veličković, D. Hennes, N. Tomašev, L. Prince, M. Kaisers, Y. Bachrach, R. Elie, L. K. Wenliang, F. Piccinini, *et al.*, "Tacticai: an ai assistant for football tactics," *Nature communications*, vol. 15, no. 1, p. 1906, 2024.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

[4] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

[5] IBM, "O que é regressão ridge?." https://www.ibm.com/br-pt/think/topics/ridge-regression, n.d. Acessado em: 27 nov. 2025.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[7] W. L. Hamilton, *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[8] T. Kipf, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, (Haifa, Israel), pp. 807–814, 2010.

[11] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[13] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.

[14] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[15] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[16] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford infolab, 1999.