Neural Graph Collaborative Filtering

摘要

现代推荐系统的核心是用户和商品的嵌入。从早期的矩阵分解法到最近的基于深度学习的方法，现有的研究成果通常通过映射用户（或商品）的现有特征（如 ID 和属性）来获取用户（或商品）的嵌入。但是，这种方法的一个固有缺点是，潜在于用户项目交互中的协作信号没有被编码到嵌入过程中，所以其所产生的嵌入可能不足以捕捉协同过滤的效果。因此，本文提出将用户物品交互的二部图结构整合到嵌入过程中，开发了一种新的推荐框架——基于图网络的协同过滤 (Neural Graph Collaborative Filtering, NGCF)。它利用用户-项目图结构传播嵌入信息，这可以对用户-项目图中的高阶连接的表达进行建模，有效地将协作信号以显式的方式注入到嵌入过程中。实验证明 NGCF 在几个最先进的模型上取得了显著改进，进一步分析验证了嵌入传播对于学习更好的用户和项目表示的重要性，证明了 NGCF 的合理性和有效性。

关键词： 协同过滤；图神经网络；推荐系统

1 相关工作

1.1 传统的推荐方法

个性化推荐系统在电子商务、广告和社交媒体等在线服务中广泛应用，其核心在于基于用户的历史交互行为（购买和点击）来估计用户对商品的购买和采纳概率。协同过滤（CF）是一种常见的方法，假设行为相似的用户在物品上具有相似的偏好 [1, 2]。学习型 CF 模型通常包含两个关键组成部分：1）嵌入，用于将用户和物品转换为向量表示；2）交互建模，一种基于嵌入的方法，用于重构历史交互。比如说，矩阵分解 (MF) 直接将用户 ID 或项目 ID 作为向量嵌入，并使用内积运算来表示用户-物品的交互 [3]。协同深度学习通过整合从物品的附加信息中学到的深层表示，扩展了矩阵分解的嵌入函数 [6]；神经协同过滤模型则用非线性神经网络替代了矩阵分解的内积交互函数 [2]；基于翻译的协同过滤模型则使用欧氏距离度量作为交互函数 [5]，等等。

尽管这些方法已经证明是有效的，但是作者还是认为其协同过滤嵌入还是不足，问题的关键在于嵌入函数无法清晰地编码关键的协同信号。所以这就导致无法有效地捕捉用户与物品之间的行为相似信号。比如大多数现有方法仅仅使用描述性特征（如 ID 和属性）构建嵌入函数，而不考虑用户-项目交互，且这些交互也仅用于定义模型训练的目标函数 [4]。因此，当嵌入信息不能呗充分的捕获协同信息时，这个方法就会被迫依赖交互函数来弥补嵌入的不足 [2]. 另一方面，虽然将用户-项目的交互信息聚合到嵌入函数中时很有效的，但是并不容易

做好。因为在实际的应用程序中，交互规模会达到百万级别甚至更大，这使得提取所需的协同信号变得困难。

1.2 基于图网络的协同过滤

本文的解决方法是，提出一种新的推荐框架——基于图网络的协同过滤（Neural Graph Collaborative Filtering，NGCF），它可以通过在嵌入过程中建模用户-项目交互的高阶连通来显式地注入协同信号。简单来说，高阶连通性考虑了不仅仅是用户和物品之间直接的互动，还包括更深层次、更复杂的关系，这有助于更全面地理解用户和物品之间的联系。这种显式的建模方法旨在确保模型能够更好地捕捉用户和项目之间的协同关系，而不仅仅依赖于描述性特征构建的嵌入。通过这种方式，NGCF 可以更全面地考虑用户-项目的交互信号，从而提高了推荐系统的性能。

2 本文方法

2.1 本文方法概述

NGCF 的模型架构如图 1所示。该框架包括三个部分：（1）提供用户嵌入和物品嵌入初始化的嵌入层；（2）通过注入高阶连接关系对嵌入进行细化的多个嵌入传播层；（3）从不同传播层聚合细化的嵌入并输出用户-物品对的关联分数的预测层。

下面来分别介绍一下每一层的设计。

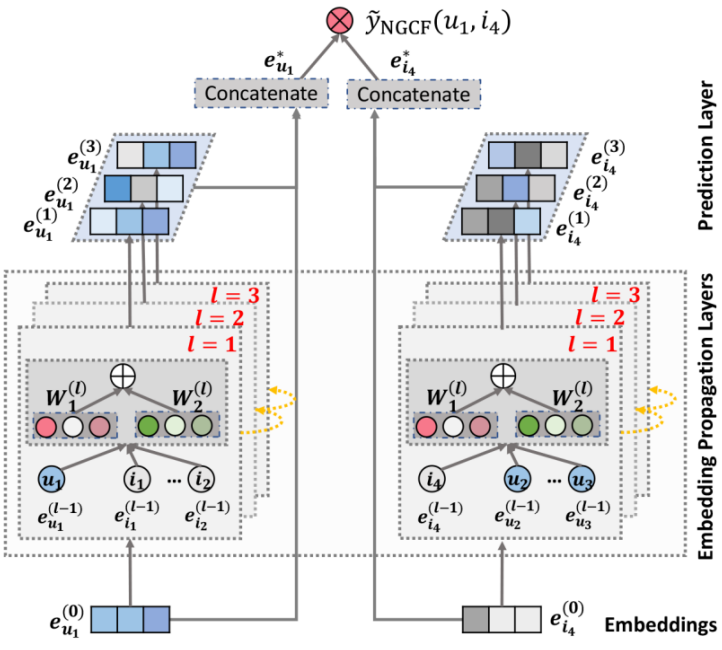

图 1. NGCF 模型架构的说明 (带箭头的线表示信息流)。用户 u_1(左) 和项目 i_4(右) 的表示通过多重嵌入传播层进行细化，将这些层的输出连接起来，做出最终的预测

2.2 嵌入层

嵌入层是推荐系统中的一个关键组件，用于将用户和物品表示为嵌入向量，这些向量捕捉了它们在潜在空间中的特征。

- 嵌入表示：使用嵌入向量 $e_u \sqsubseteq R_d$ 表示用户 u 和 $e_i \sqsubseteq R_d$ 表示物品 i, 其中 d 是嵌入大小，可以将这过程看作构建一个参数矩阵 E，作为嵌入查找表，其中包含用户嵌入和物品嵌入。
- 嵌入表格: 这个嵌入表格作为用户嵌入和物品嵌入的初始状态，将在端到端的方式进行优化。
- 嵌入优化：在传统的推荐模型中，比如矩阵分解（MF）和神经协同过滤，用户和物品通常被简单地表示为一个固定的标识向量。然而，这里采用了一种不同的方法。相当于把用户和物品的特征信息放在一个表格里，这个表格会在用户和物品之间的互动图上进行一种“传播”过程。这个传播过程并不是简单地将用户和物品的信息直接送到预测模型，而是通过一个优化过程不断地改善这些特征向量。这就好比把用户和物品的关系考虑得更全面，而不仅仅是简单地用静态的特征向量表示。通过这种方式，模型能够更灵活地适应用户和物品之间复杂的关系，提高了推荐系统的效果。简而言之，这种方法通过优化用户和物品的特征向量，引入协同信号，从而让推荐更加智能和准确。

2.3 嵌入传播层

2.3.1 一阶传播

消息构建：消息构建的目标是通过相互作用的物品提供直接证据来捕捉用户的偏好。首先，对所连接的用户-物品对 (u,i) 进行定义为物品 i 到用户 u 的消息 $m_{u\leftarrow i}$ 。再使用消息编码函数 $f(\cdot)$，该函数将物品 i 的嵌入 e、用户 u 的嵌入 e_u 以及系数 p_{u_i} 作为输入，输入消息 m_{u_i}，表示要传播的信息 $m_{u\leftarrow i} = f(e_i, e_u, p_{ui})$。消息编码函数的具体实现如下所示：

$$m_{u\leftarrow i} = \frac{1}{\sqrt{\mid N_u \mid\mid N_i \mid}}(W_1 e_i + W_2(e_i \odot e_u) \tag{1}$$

其中 W1 和 W2 是可以训练的权重矩阵，d' 是变换的大小。表示逐元素乘法，用这种方式将物品 i 和用户 i 之间的交互编码传播到消息中，使得消息依赖于他们之间的关联。

消息聚合：在聚合阶段，从用户 u 的领域中汇总传播的消息，以得到更丰富的用户表示。这里定义了聚合函数，这是一种数学操作，它的目的是将来自用户邻居的传播消息合并起来，从而得到一个更综合的用户表示。具体来说，这个操作使用了一种叫做 LeakyReLU 的激活函数。想象一下，每个邻居都向用户发送一条信息，而 LeakyReLU 的作用就像是在合并这些信息时加入了一种特殊的处理。LeakyReLU 允许信息同时包含正向信号和小的负向信号。这就好比是，我们不仅关心积极的信息，还对稍微不太积极的信息也进行考虑。这种处理方式使得模型能够更灵活地捕捉用户复杂的特征，而不会完全忽略一些不太重要但仍有用的信息。首先，嵌入传播层的优势在于明确地利用一阶连接信息来关联用户和物品的表示。

$$e_u^{(1)} = LeakyReLU(m_{u\leftarrow u} + \sum_{i\in N_u} m_{u\leftarrow i}) \tag{2}$$

其中，$e_u^{(1)}$ 表示用户 u 在第一层嵌入传播之后得到的表示。出来从邻居 N_u 传播过来的消息之外，还考虑了用户 u 的自连接来保留原始特征的信息 $m_u^{(u)} = W_1 e_u$。

2.3.2 高阶传播

通过第一阶连接建模增强表示后，我们可以堆叠更多的嵌入传播层以探索高阶连接信息。这种高阶连接对于编码协同信号以估计用户和物品之间的相关性得分至关重要。高阶连接性在推荐系统中的优势体现在更全面的关系捕捉、协同信号注入和特征逐层丰富性上。通过多层嵌入传播，模型能够更深入地理解用户和物品之间的复杂关系，提高协同信号的把握，同时逐渐丰富特征表示，从而提升推荐模型的性能。通过堆叠 l 个嵌入传播层，用户（和物品）能够接收来自其 l 跳邻居的传播消息。用户 u 的表示可以递归地表示为：

$$e_u^{(l)} = LeakyReLU(m_{u\leftarrow u}^{(l)} + \sum_{i\in N_u} m_{u\leftarrow i}^{(l)}) \tag{3}$$

$$\begin{cases} m_{u\leftarrow u}^{(l)} = p_{ui}(W_1^{(l)} e_i^{(l-1)} + W_2^{(l)}(e_i^{(l-1)} \odot e_u^{(l-1)})) \\ m_{u\leftarrow i}^{(l)} = W_1^{(l)} e_i^{(l-1)} \end{cases} \tag{4}$$

其中，$W_1^{(l)}$，$W_2^{(l)}$ 是训练的转换矩阵。$e^{(l-1)_i}$ 是从前一次消息传递步骤中生成的物品表示，记忆了其 $l-1$ 跳邻居的信息，并进一步对第 l 层用户 u 的表示做出贡献。类似地，我们可以在第 l 层获得物品 i 的表示。这个高阶传播的过程能够捕捉协同信号，例如 $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$，如图 2 所示。此外，来自 i_4 的消息在 $e_{u_1}^{(3)}$ 中被明确地编码（由红色线表示）。因此，堆叠多个嵌入传播层能够无缝地注入协同信号到表示学习过程中。

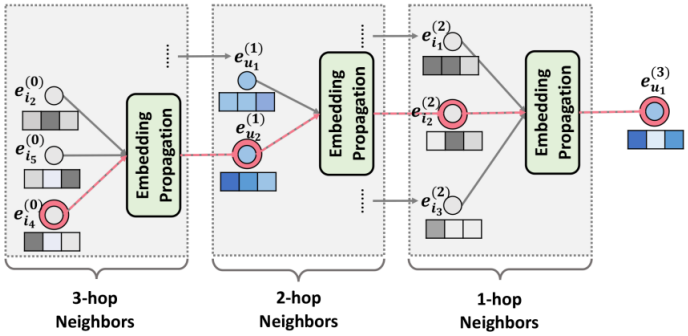

图 2. 用户 u_1 的三阶嵌入传播图

2.4 模型预测

这一部分描述了一个推荐系统模型的预测过程。模型经过多层次的信息传播，为每个用户和物品得到多个不同层次的表示，即

$$e_u^* = e_u^{(0)} \parallel \ldots \parallel e_u^{(L)}, \quad e_i^* = e_i^{(0)} \parallel \ldots \parallel e_i^{(L)} \tag{5}$$

这些表示对应于网络中不同层的输出，捕捉了输入在不同层次上的抽象表示。且由于每个层次关注的是数据的不同抽象层次，表示中的信息也反映了用户喜好的多个方面。这样的设计使得模型能够更全面地理解和捕捉用户的偏好。为了得到最终的用户和物品的表达，作者采用了简单的拼接操作，将不同层次的表示连接在一起。这种方法不仅丰富了最初的表达，还通过调整传播层次的数量来控制信息传播的范围。与其他聚合方式相比，作者选择拼接的优势在于其简单性，而且在图神经网络领域的研究中已被证明效果不错。最后，作者使用内积

操作来估计用户对目标物品的偏好，强调了模型设计的简洁性，最终预测结果表达式如公式(6) 所示。虽然内积是一种简单的交互方式，但作者强调了在未来的研究中可以尝试更复杂的交互函数，如基于神经网络的方法。

$$\hat{y}_{NGCF(u,i)} = e_u^* \perp e_i^* \tag{6}$$

2.5 模型优化

在这一部分，描述了模型是如何学习其参数以及如何应对过拟合问题的。首先，通过采用一个称为 BPR 的损失函数 (如公式（7）所示)，该损失函数考虑了用户与物品之间交互的相对顺序，通过优化这个目标函数来训练模型。

$$Loss = \sum_{(u,i,j)\in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \| \Theta \|_2^2 \tag{7}$$

这个损失函数的设计有助于模型更好地预测用户对物品的喜好。同时，为了防止模型在训练中学到过于复杂的规律，作者采用了一种称为小批量 Adam 的优化算法。此外，为了防止过拟合，采用了两种 dropout 技术，分别是 message dropout 和 Node dropout。这些技术通过在训练过程中随机地丢弃一些信息，使得模型在学习时更加鲁棒，不容易被训练数据中的噪声干扰。

在考虑模型大小的时候，虽然模型在每个传播层都有一个嵌入矩阵，但是引入的额外参数非常有限，仅包括两个权重矩阵。相较于其他推荐模型，这样的设计使得模型在参数数量上更加高效，特别是在嵌入层的设计上，增加了模型的复杂度但成本几乎可以忽略。

3 复现内容

3.1 创新点——尝试加入注意力机制

在协同过滤中加入注意力机制，能在处理用户-物品交互数据时，允许模型对不同的用户或物品分配不同的权重，有助于捕捉数据中的个性化信息和重要关系。除此之外，在协同过滤中，用户-物品矩阵往往是非常稀疏的，即大多数用户-物品对没有交互。注意力机制可以帮助模型集中注意力在更有意义的用户-物品对上，从而减轻稀疏性问题。而且，注意力权重可以提供对模型预测的一定解释性。通过观察哪些用户或物品在预测中获得更高的注意力权重，你可以更好地理解模型是如何进行预测的。以下是我对加入注意力机制的构思：假设有一个用户-物品交互图 $G = (V, E)$，其中 V 是节点集合，E 是边集合。每个节点表示用户或物品，每条边表示用户-物品的交互。

首先，定义注意力权重的计算方式。对于节点 i 和节点 j 之间的注意力权重 α_{ij}，可以使用如下公式：

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[W \cdot e_i || W \cdot e_j]))}{\sum_{k\in N_i} \exp(\text{LeakyReLU}(\mathbf{a}^T[W \cdot e_i || W \cdot e_k]))} \tag{8}$$

其中，e_i 和 e_j 是节点 i 和节点 j 的嵌入向量，N_i 是节点 i 的邻居节点集合，W 是权重矩阵，$\mathbf{a}$ 是注意力机制中的可学习参数。

然后，根据注意力权重，更新节点 i 的表示：

$$h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij}(W \cdot e_j)\right) \tag{9}$$

其中，σ 是激活函数（例如 LeakyReLU）。

最后，可以使用更新后的节点表示 h_i' 进行预测或其他任务。

3.2 与已有开源代码对比

该论文在 GitHub 上发表了 TensorFlow 版本的源代码，但是由于原 Tensor Flow 版本代码所需的配置环境较低，安装起来比较麻烦且问题繁琐，因此在复现时本文使用的是更新改进后的 Pytorch 版本代码。

3.3 实验环境搭建

3.3.1 硬件环境描述

使用一台配备 NVIDIA 525.60.11 RTX 8000、16GB 内存的工作站进行实验。

3.3.2 软件环境描述

在 Ubuntu 20.04 操作系统上，使用 Python 3.7 编写代码，借助 Pytorch 1.1 进行深度学习模型的实现。所需的安装包有 numpy 1.18 、scipy 1.3.2 、sklearn 0.2。

3.3.3 数据集描述

在评估 NGCF 的效果时，使用了三个公开可访问的基准数据集：Gowalla、Yelp2018 和 Amazon-book。这些数据集在领域、大小和稀疏性方面各不相同。

Gowalla：这是从 Gowalla 获取的签到数据集，用户通过签到分享他们的位置。为了确保数据集的质量，我们采用了 10-core 设置，（即至少有十次用户与物品的互动或交互记录）这一设置的目的是为了确保在建模用户和物品之间的关系时有足够的数据可用，以提高模型的质量和泛化能力。

Yelp2018：该数据集源自 2018 年 Yelp 挑战赛。其中，本地的企业如餐馆和酒吧被视为物品。我们同样采用了 10-core 设置，以确保数据质量。

Amazon-book： Amazon-review 是用于产品推荐的广泛使用的数据集，我们从中选择了 Amazon-book。同样，我们使用 10-core 设置，以确保每个用户和物品都有至少十个交互。

对于每个数据集，我们随机选择每个用户 80% 的历史交互作为训练集，将其余的作为测试集。从训练集中，我们随机选择 10% 的交互作为验证集，以调整超参数。对于每个观察到的用户-物品交互，我们将其视为正样本，然后采用负采样策略，将其与用户之前未消费过的一个负样本物品配对。

3.3.4 参数设置

嵌入大小：对于所有模型，嵌入大小都被固定为 64。HOP-Rec 参数设置：对于 HOP-Rec 模型，搜索了随机游走步数（random walks）的范围为 {1, 2, 3}，并在学习率中进行调优，范围为 {0.025, 0.020, 0.015, 0.010}。

	Gowalla		Yelp2018		Amazon-Book	
	reacall	ndcg	reacall	ndcg	reacall	ndcg
Ngcf-1	0.1523	0.1318	0.0491	0.0423	0.0311	0.0213
Ngcf-2	0.1416	0.1307	0.0566	0.0465	0.0330	0.0254
Ngcf-3	0.1569	0.1316	0.0555	0.0467	0.0301	0.0224
Ngcf-4	0.1538	0.1332	0.0529	0.0434	0.0303	0.0258

表 1. 嵌入传播层数 (l) 的影响

优化器选择：除了 HOP-Rec 外，其他所有模型都使用 Adam 优化器，其中 batch size 固定为 1024。

超参数调优：采用网格搜索进行超参数调优，包括学习率在 {0.0001, 0.0005, 0.001, 0.005} 中选择，L2 归一化系数在 $\{10^{-5}, 10^{-4}, ..., 10^{1}, 10^{2}\}$ 中搜索，以及节点丢弃率和消息丢弃率在 {0.0, 0.1, ..., 0.8} 中选择。

节点丢弃技术：对于 GC-MC 和 NGCF，采用节点丢弃技术，其中节点丢弃率在 {0.0, 0.1, ..., 0.8} 中调整。

参数初始化：使用 Xavier 初始化器对模型参数进行初始化。

早停策略：实施了早停策略，即如果在验证数据上的 recall@20 连续 50 个 epoch 不增加，则提前停止训练。

NGCF 深度设置：为了建模第三阶连接中编码的协同过滤信号，将 NGCF 模型的深度 L 设置为三。

默认设置：节点丢弃率为 0.0，消息丢弃率为 0.1

3.3.5 代码链接：

https://github.com/xiangwang1223/neural_graph_collaborative_filtering

4 实验结果分析

由于嵌入个传播层在 NGCF 中起着至关重要的作用，因此需要探究层数对该模型的性能影响。实验结果如下表所示，与原本中的实验结果相差不大：

改变 NGCF 模型的层数对推荐性能有影响。增加层数有助于更好地理解用户之间的相似性和协同信号，从而提高推荐准确性。然而，增加层数并不总是好的，过多可能导致在某些情况下的过拟合问题。在实验中，发现选择三个传播层在不同数据集上表现最好，显示出 NGCF 相对于其他方法的优越性。因此，选择适当的层数是一个需要谨慎考虑的关键因素。

5 总结与分析

本文改进了推荐系统，使其更能理解和利用用户与物品之间的合作关系。所提出的 NGCF 可以显式地将 User-Item 的高阶交互编码进 Embedding 中来提升 Embedding 的表示能力，考

虑了用户和物品之间更复杂的关联关系进而提升整个推荐效果。NGCF 的关键就在于 Embedding Propagation Layer 来学习 User 和 Item 的 Embedding，后面的预测部分只是简单的内积。可以说，NGCF 较好地解决了协同过滤算法的第一个核心问题。

未来，我们计划继续改进这个方法，使其更加灵活和适应各种不同的情况。在本文中，Embedding Propagation 实际上没有考虑邻居的重要性，如果可以像 Graph Attention Network 在传播聚合过程中考虑邻居重要性的差异，NGCF 的效果应该可以进一步提升。此外，本方法在处理真实世界中常见的数据稀疏性问题时仍然存在挑战，由于用户-物品矩阵通常是非常稀疏的，这可能导致模型性能下降。

所以接下来我们可以从其他方面考虑改进。例如，协同过滤中是否有不必要的复杂设计影响其训练效果？我们如何简化它的设计呢？在用户和物品的信息处理中是否可以引入一些新的技术，以增强模型的鲁棒性？那么引进什么技术最合适呢？若根据实际场景设计 NGCF，通过进一步引入更多的个性化元素，比如用户的兴趣变化、购买历史、社交网络关系等，能否帮助了系统更全面地了解用户的喜好呢？若引入多模态模型来增加侧信息，设计更有效的特征提取方法，是否更有利于信息整合呢.....

综合而言，我们将继续通过不断尝试和实验，根据具体应用需求和数据特点，对推荐系统进行改进。我们期望这些研究方向能够进一步提升个性化推荐系统的效果和解释性能，为推荐系统领域的发展做出更有价值的贡献。

参考文献

[1] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[5] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 2018 world wide web conference*, pages 729–739, 2018.

[6] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244, 2015.