LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

摘要

目前，图卷积网络（GCN）成为协同过滤的新技术。但其推荐有效性的原因尚未可知。GCN 适应推荐的现有工作缺乏对 GCN 的彻底消融分析。根据经验发现，GCNs 中最常见的两种设计特征变换和非线性激活对协同过滤的性能贡献很小。更糟的是，贡献小的情况下极大的增加了训练难度，降低推荐性能。在这项工作中，目标是简化 GCN 的设计，使其更加简洁和适合推荐。作者提出了一个名为 LightGCN 的新模型，只保留 GCN 中最重要的组件，邻域聚合用于协同过滤。具体来说，LightGCN 通过在用户——项目交互图上线性传播用户和项目嵌入来学习用户和项目嵌入，并使用在所有层学习的嵌入使用加权和作为最终嵌入。这种简单、线性和整洁的模型更容易实现和训练，在完全相同的实验设置下，比神经图协同过滤（NGCF）一种最先进的基于 GCN 的推荐模型——表现出实质性的改进。

关键词： 协同过滤，LightGCN,GCN, 随机负采样

1 引言

随着信息技术迅速发展和在线服务的普及，人们能够快速获取大量信息，这也使得人们从信息匮乏的时代跨进了“信息过载”的时代。数据的“爆炸式”增长，为人类记忆和处理信息的能力带来了极大的挑战，大量冗余信息严重干扰对有用信息的提取和利用，增加信息处理的成本。为了减轻网络上的信息过载，推荐系统已被广泛部署来执行个性化信息过滤。

推荐系统的核心是预测用户是否会与一个项目进行交互，例如，点击、评级、购买以及其他形式的交互。因此，协作过滤（CF）是实现有效个性化推荐的基本方式，其重点是利用过去的用户——项目交互来实现预测。CF 最常见是学习潜在嵌入来表示用户和项目，并基于嵌入向量执行预测。鉴于用户——项目交互图的基础上，使用用户的子图结构，更具体地说，她的单跳邻居来改进嵌入学习。

为了深化具有高跳邻居的子图结构的使用，王 [3] 等最近提出了 NGCF，并实现了 CF 的最新性能。灵感来自图卷积网络（GCN），它的传播规则：特征变换、邻域聚合和非线性激活。虽然 NGCF 已经显示出有希望的结果，但作者认为它的设计相当沉重和繁琐——没有正当理由的情况下直接从 GCN 继承，没有结合 CF 的特点进行适应性改变。具体来说，GCN 最初是为属性图上的节点分类而提出的，其中每个节点都有丰富的属性作为输入特征；而在 CF 的用户——项目交互图中，每个节点（用户或项目）仅由一个热 ID 描述，该 ID 除了是一个标识符之外，没有具体的语义。在这种情况下，给定 ID 嵌入作为输入，执行多层非线性特征转换不仅不会带来任何好处，而且会负面地增加模型训练的难度。

为了验证上述想法，作者对 NGCF 进行了消融研究。通过严格的控制实验得出结论，从 GCN 继承的两种操作——特征变换和非线性激活——对 NGCF 的有效性没有贡献。更令人惊讶的是，移除它们会显著提高精确度。受到该结果的启发，作者提出了一个名为 LightGCN 的新模型，包括 GCN 最重要的组成部分——邻域聚合——用于协同过滤。具体来说，在将每个用户（项目）与 ID 嵌入相关联之后，在用户——项目交互图上传播嵌入以细化它们。然后，将在不同传播层学习到的嵌入与加权和相结合，以获得用于预测的最终嵌入。整个模型简单实用，不仅更容易训练，而且比 NGCF 和其他最先进的方法如 Mult-VAE 获得了更好的经验性能。

2 相关工作

2.1 NGCF Brief

在初始步骤中，每个用户和项目都与 ID 嵌入相关联。设 $e_u^{(0)}$ 表示用户 u 的 ID 嵌入，$e_i^{(0)}$ 表示项目 i 的 ID 嵌入。然后 NGCF 利用 useritem 交互图将嵌入传播为：

$$e_i^{(k+1)} = \sigma(W_1 e_i^{(k)} + \sum_{i \in = N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}(W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)}) \tag{2.1}$$

$$e_u^{(k+1)} = \sigma(W_1 e_u^{(k)} + \sum_{i \in = N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}(W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)}) \tag{2.2}$$

其中 $e_u^{(k)}$ 和 $e_i^{(k)}$ 分别表示用户 u 和项目 i 在 k 层传播后的嵌入， 是非线性激活函数，N_u 表示用户 u 交互的项目集合，N_i 表示与项目 i 交互的用户集合，W_1 和 W_2 是可训练的权重矩阵，用于在每层执行特征变换。通过传播 L 层，NGCF 获得 $L+1$ 嵌入来描述用户 $e_u^{(0)}$,, $e_u^{(1)}$，...，$e_u^{(L)}$ 和一个项目 $e_i^{(0)}$,$e_i^{(1)}$,...,$e_i^{(L)}$。然后，它将这些 $L+1$ 嵌入连接起来，以获得最终的用户嵌入和项目嵌入，使用内积来生成预测分数。

2.2 ablation studies on NGCF

作者认为 GCN 最初是为属性图上的节点分类而提出的，他针对的情况是每个节点都有丰富的属性作为其输入特征；而 CF 的用户——项目交互图中，每个节点仅由一个 ID 描述，该 ID 没有具体的语义。在这种情况下，将给定 ID 嵌入作为输入，执行多层非线性特征转换不仅不会带来任何好处，而且会负面地增加模型训练的难度。针对该情况作者进行了严格的消融实验。

使用 NGCF 作者发布的代码，在相同的数据分割和评估协议上运行实验，以尽可能保持比较的公平性。实现了 NGCF 的三个简化变体：

i)NGCF-f，它去除了特征变换矩阵 W_1 和 W_2;

ii)NGCF-n，去掉了非线性激活函数 α;

iii)NGCF-fn，移除特征变换矩阵和非线性激活函数;

在表 1 中发现了 Gowalla 和 Amazon-Book 数据集上的 2 层设置的结果。可以看出，移除特征转换（即 NGCF-f）导致在所有三个数据集上对 NGCF 的一致改进。相比之下，去除非线性激活不会对精度产生太大影响。然而，如果在去除特征变换的基础上去除非线性激活

表 1. NGCF 及其三种变体的性能

	Gowalla		Amazon-Book	
	recall	ndcg	recall	ndcg
NGCF	0.1547	0.1307	0.0330	0.0254
NGCF-f	0.1686	0.1439	0.0368	0.0283
NGCF-n	0.1536	0.1295	0.0336	0.0258
NGCF-fn	0.1742	0.1476	0.0399	0.0303

(即 NGCF-fn)，性能显著提高。从这些观察中，得出结论：(1) 增加特征变换对 NGCF 有负面影响，因为在 NGCF 和 NGCF-n 模型中去除特征变换显著提高了性能；(2) 加入非线性激活对特征变换的影响较小，但对禁用特征变换的影响较小。(3) 总体而言，特征转换和非线性激活对 NGCF 有较大的负面影响，因为同时去除特征转换和非线性激活，NGCF-fn 比 NGCF 有较大的改善（相对召回率提高 9.57%）。

3 本文方法

3.1 方法概述

前一节证明了 NGCF 是协同过滤的一个繁重的 GCN 模型。在这些发现的推动下，我们设定了一个目标，通过包括 GCN 最基本的成分来开发一个简单而有效的模型。[4] [1] 在本节中，首先展示设计的（LightGCN）模型，如图 1 所示。LightGCN 是一种轻量级的图卷积网络模型，用于推荐系统。它通过直接传播用户和物品的嵌入向量来进行推荐预测。模型结构简洁，没有复杂的非线性激活函数和权重矩阵，只保留了用户和物品的 ID 嵌入，通过多层传播嵌入向量，并使用内积生成预测评分。它的设计理念是保留网络传播的简洁性和有效性，以提高推荐性能。接下来将会对 LightGCN 进行了深入分析，以展示其简单设计背后的合理性。

3.2 Light 图卷积

在 LightGCN 中，采用简单的加权和聚合器，放弃了特征变换和非线性激活的使用。LightGCN 中的图形卷积运算（又为传播规则）定义为：

$$e_u^{(k+1)} = \sum_{i \in = N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(k)} \tag{3.1}$$

$$e_i^{(k+1)} = \sum_{i \in = N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_u^{(k)} \tag{3.2}$$

$e_u^{(k)}$ 和 $e_i^{(k)}$ 分别表示用户 u 和项目 i 在 k 层传播后的嵌入，N_u 表示用户 u 交互的项目集合，N_i 表示与项目 i 交互的用户集合。$e_i^{(k+1)}$,$e_u^{(k+1)}$ 由 $e_u^{(k)}$ 和 $e_i^{(k)}$ 的信息聚合得到的。符合 GCN 的思想，聚合邻居节点的信息来学习节点的表示。

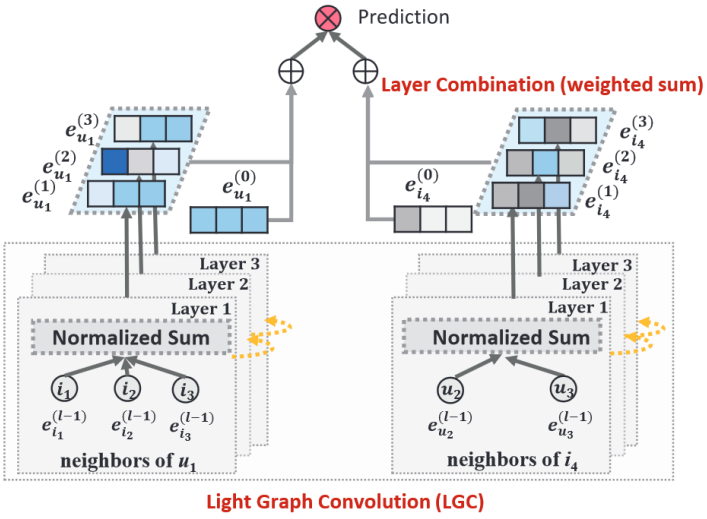


图 1. LightGCN 方法示意图

对称归一化项 $\frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}$ 遵循标准 GCN 的设计，可以避免嵌入规模随着图卷积运算的增加而增加。值得注意的是，在 LGC，只聚合连接的邻居，不进行自连接。这不同于大多数现有的图卷积运算。下一小节将介绍的层组合操作本质上捕捉到了与自连接相同的效果。因此，LGC 中不需要包含自连接。

3.3 层连接

在 LightGCN 中，唯一可训练的模型参数是第 0 层的嵌入，即所有用户的 $e_u^{(0)}$ 和所有项目的 $e_i^{(0)}$，即用户和项目的初始嵌入。当给定初始嵌入时，更高层的嵌入可以通过等式中定义的 LGC 来计算。在 k 层 LGC 之后，进一步组合在每一层获得的嵌入来形成最终的表示:

$$e_u = \sum_{k=0}^{K} \alpha_k * e_u^{(k)} \tag{3.3}$$

$$e_i = \sum_{k=0}^{K} \alpha_k * e_i^{(k)} \tag{3.4}$$

其中 $\alpha_k \geq 0$ 表示第 k 层嵌入在构成最终嵌入中的重要程度。它可以被视为要手动调整的超参数，或者被视为要自动优化的模型参数。在实验中，我们发现将 α_k 均匀地设置为 $\frac{1}{k+1}$ 通常会导致良好的性能。因此，不需要设计特殊的组件来优化 α_k，以避免不必要的复杂 LightGCN，并保持其简单性。[2]

执行层组合以获得最终表示的原因有三个。

(1) 随着层数的增加，嵌入将被过度平滑。因此，简单地使用最后一层是有问题的。

(2) 不同层的嵌入捕获不同的语义。例如，第一层对具有交互的用户和项目实施平滑，第二层平滑在交互的项目（用户）上具有重叠的用户（项目），并且更高层捕获更高阶的邻近性。因此，将它们结合起来将使表述更加全面。

(3) 将不同层的嵌入与加权和相结合捕获了具有自连接的图卷积的效果，这是 GCNs 中的一个重要技巧。

3.4 模型预测

模型预测定义为用户和项目最终表示的内积，最终的计算结果将会用作推荐生成的排名分数。计算方式为:

$$\hat{y_{ui}} = e_u^T * e_i \tag{3.5}$$

3.5 损失函数

采用贝叶斯个性化排名（BPR）损失，使得模型更倾向于正确物品的得分高于错误物品，即学习模型参数，使得在排序中用户喜欢的物品排在用户不喜欢的物品之前。同时加入的 $L2$ 正则化防止过拟合。其中 控制 $L2$ 正则化强度。

$$L_{BPR} = -\sum_{u=1}^{M}\sum_{i\in N_u}\sum_{j\notin N_u} ln\sigma(\hat{y_u}i - \hat{y_u}j) + \lambda||E^{(0)}||^2 \tag{3.6}$$

4 复现细节

4.1 与已有开源代码对比

本次复现的模型-LightGCN 模型, 该论文的作者公开了其代码，本复现报告主要以该代码为基础，并在此基础上修改原模型的损失函数，使得模型有更好的效果。

```python
def bpr_loss(self, users, pos, neg):
    (users_emb, pos_emb, neg_emb,
     userEmb0,  posEmb0, negEmb0) = self.getEmbedding(users.long(), pos.long(), neg.long())
    reg_loss = (1/2)*(userEmb0.norm(2).pow(2) +
                      posEmb0.norm(2).pow(2)  +
                      negEmb0.norm(2).pow(2))/float(len(users))
    pos_scores = torch.mul(users_emb, pos_emb)
    pos_scores = torch.sum(pos_scores, dim=1)
    neg_scores = torch.mul(users_emb, neg_emb)
    neg_scores = torch.sum(neg_scores, dim=1)
    loss = torch.mean(torch.nn.functional.softplus(neg_scores - pos_scores))
    device = world.device
    neg_labels = torch.zeros(neg_scores.size()).to(device)
    self.weight_decay = torch.tensor(0.0001)
    neg_loss2 = F.binary_cross_entropy_with_logits(neg_scores, neg_labels,
weight=torch.tensor(0.0001), reduction='none').mean(dim=-1)
    pos_labels = torch.ones(pos_scores.size()).to(device)
    pos_loss2 = F.binary_cross_entropy_with_logits(pos_scores, pos_labels,
weight=self.weight_decay, reduction='none')
    loss2 = pos_loss2 + neg_loss2 * self.weight_decay
    return loss, reg_l2_loss, loss2.sum()
class BPRLoss:
    def __init__(self,
                 recmodel: PairWiseModel,
                 config: dict):
        self.model = recmodel
        self.weight_decay = config['decay']
        self.lr = config['lr']
        self.opt = optim.Adam(recmodel.parameters(), lr=self.lr)
    def stageOne(self, users, pos, neg):
        loss, reg_l2_loss, loss2 = self.model.bpr_loss(users, pos, neg)
        reg_loss = reg_l2_loss*self.weight_decay
        lossbce = loss2
        loss = loss + reg_loss + lossbce
        self.opt.zero_grad()
        loss.backward()
        self.opt.step()
        return loss.cpu().item()
```

图 2. LightGCN 方法示意图

4.2 实验环境搭建

为了搭建实验环境，使用基于 Linux 操作系统的高性能计算机集群。在软件方面，依赖 Python 3.8 及以上版本作为主要编程语言。特定的 Python 库包括 torch,pandas,scipy，numpy，tensorboardX,scikit-learn，tqdm，这些版本与实验代码兼容。

4.3 实验数据集介绍

实验中使用的数据集为 Gowalla、Yelp2018 和 lastfm。

Gowalla 数据集是一个包含社交媒体签到数据的数据集，用于研究位置推荐和移动社交网络等领域。该数据集包含了来自 Gowalla 应用的用户签到记录，涵盖了 2009 年到 2010 年期间在美国、加拿大和欧洲的许多城市。数据集中每个签到记录包含用户 ID、位置经纬度、签到时间戳等信息。用户在 Gowalla 平台上的签到记录和社交网络关系揭示了他们在现实世界中的活动和偏好。研究者可以通过分析用户的地理位置历史、签到地点和社交连接，为用户提供个性化的地理位置推荐。这种推荐系统可以涵盖各种场景，包括餐馆、景点等，帮助用户发现附近的热门地点或符合他们兴趣的新颖场所。该数据集融合了地理信息和社交网络数据，为推荐算法提供了多维度的用户行为特征，这使得推荐系统不仅能更好地理解用户的地理位置偏好，还能考虑到社交关系的影响。

Yelp2018 数据集包含了用户对商家的详细评分和评论，以及地理位置等信息。通过分析用户行为和情感反馈，该数据集可用于建模用户偏好、实现个性化推荐，引入情感分析提高推荐精度，同时结合多模态信息如图片，使推荐更全面。这为研究者提供了深入理解用户喜好和行为模式的机会，以优化推荐算法，提升用户体验。

Lastfm 数据集提供了音乐推荐研究所需的用户听歌历史和音乐元数据。该数据集包含了用户的音乐播放历史，同时提供了歌曲和艺术家的详细信息，这使得推荐系统能够通过分析用户与歌曲之间的关联，实现更精准、个性化的音乐推荐。通过 Lastfm 数据集，研究者能够探索基于听歌历史的推荐算法，提高音乐推荐的准确性，同时通过挖掘用户与艺术家、歌曲之间的潜在关系，进一步丰富推荐系统的多样性和用户体验。

具体信息如下：

表 2. 实验数据的统计

Dataset	User	Item	Interaction	Density
Gowalla	29858	40981	1027370	0.00084
Yelp2018	31668	38048	1561406	0.00130
LastFM	1892	4489	305560	0.00620

4.4 创新点

本次实验的创新点体现在对原模型的损失优化部分。在复现原论文时，发现原模型存在到达一定 epoch 后出现 recall 和 NDCG 指标下降的问题，分析原因后发现是模型存在过拟合问题。所以针对该存在问题，本人在优化部分增加随机负采样作为辅助损失，公式定义如下所示，降低过拟合程度，更好的对模型进行优化。最终的损失由 BPR 损失和辅助损失聚合所得到，将修改后的模型命名为 NegLightGCN。复现原论文的数据集后，另外在 lastFM 数据集上运行证明了 LightGCN、NegLightGCN 的合理性。

随机负采样的主要优势在于简单而高效，相对于使用所有可能的负例，通过随机采样一小部分负例，可以显著降低训练时的计算成本。而且在推荐系统等任务中，负样本通常要远多于正样本，如果每次都考虑所有的负样本，那么计算复杂度将会很高。随机负采样能够有效地降低这种复杂度，提高训练效率。而且在很多任务中，正样本可能远远少于负样本，随机负采样可以在一定程度上缓解导致类别不平衡的问题。随机负采样引入了一些负面的例子，这有助于模型更好地理解什么是负例，提高模型的泛化能力。这在避免过拟合的同时，使模

型更具有鲁棒性。

$$L_O = -\sum_{(u,i)\in N^+} log(\sigma(e_u^T e_i)) - \sum_{(u,i)\in N^-} log(\sigma(-e_u^T e_i)) \tag{4.1}$$

$$L_{total} = L_O + L_{BPR} \tag{4.2}$$

5 实验结果分析

5.1 评估指标

使用 Recal 和 NDCG 这两个广泛使用的协同过滤指标来评估 top-K 推荐的性能。更高的召回 Recall 和 NDCG 意味着更好的性能。

5.2 结果分析

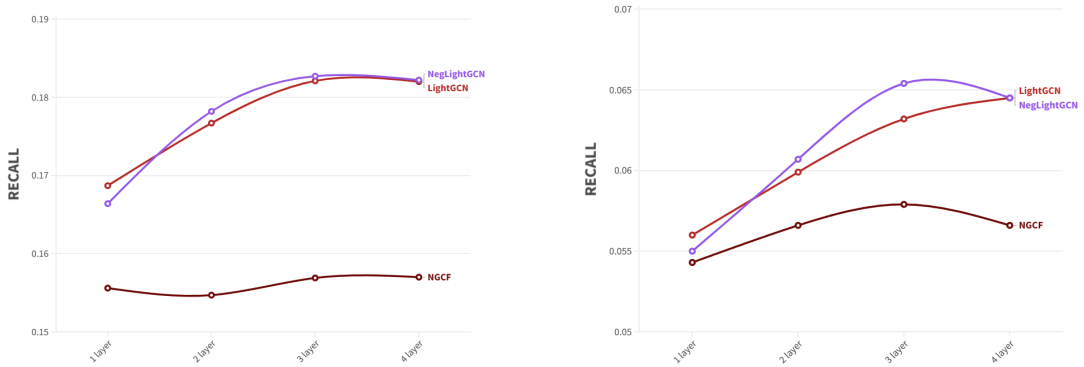
本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

表 3. NGCF、LightGCN、NegLightGCN 在相同条件下的对比

Dataset		Gowalla		Yelp2018		lastfm	
Layer	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1547	0.1315	0.0543	0.0442	-	-
	LightGCN	0.1687	0.1417	0.0560	0.0456	0.2375	0.1780
	NegLightGCN	**0.1664**	**0.1395**	**0.0550**	**0.0449**	**0.2360**	**0.1790**
2 Layers	NGCF	0.1556	0.1307	0.0566	0.0465	-	-
	LightGCN	0.1767	0.1508	0.0599	0.0496	0.2560	0.1995
	NegLightGCN	**0.1782**	**0.1513**	**0.0607**	**0.0506**	**0.2564**	**0.2003**
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	-	-
	LightGCN	0.1821	0.1545	0.0632	0.0519	0.2680	0.2096
	NegLightGCN	**0.1827**	**0.1546**	**0.0654**	**0.0534**	**0.2706**	**0.2105**
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	-	-
	LightGCN	0.1820	0.1535	0.0645	0.0530	0.2760	0.2146
	NegLightGCN	**0.1822**	**0.1534**	**0.0645**	**0.0528**	**0.2770**	**0.2156**

表 4. 在 Gowalla 和 yelp2018 上两种方法的平均提升程度

dataset	Gowalla		Yelp2018	
model	LightCGN	NegLightGCN	LightCGN	NegLightGCN
recall	14.30%	14.40%	8%	8.40%
ndcg	14%	13.40%	9%	9.2%



(a) Gowalla (b) Yelp2018

图 4. 三种模型的 RECALL 对比图.

将模型在不同数据集（Gowalla、Yelp2018、lastfm）以及不同层数条件下进行运行，记录每一次实验的衡量指标，包括 recall 和 ndcg。对 NegLightGCN、LightGCN 与 NGCF 进行了详细比较，在表 3 中记录了不同层（1 到 4）的性能，表 4 显示了 NegLightGCN、LightGCN 相比较 NGCF 的平均改善百分比。

在所有情况下，NegLightGCN、LightGCN 的表现都远远优于 NGCF。例如，在 Gowalla 上，NGCF 论文中报告的最高召回率为 0.1570，而 LightGCN 在 4 层设置下可以达到 0.1820，高出 15.92%。NegLightGCN 在 4 层设置下可以达到 0.1822，高出 16%。

平均而言，NeglightGCN 在数据集 Gowalla 的平均召回率提高了 14.4%，ndcg 提高了 13.8%，数据集 Yelp2018 的平均召回率提升了 8.4%,ndcg 提升了 9.2%, 这是相当显著的。

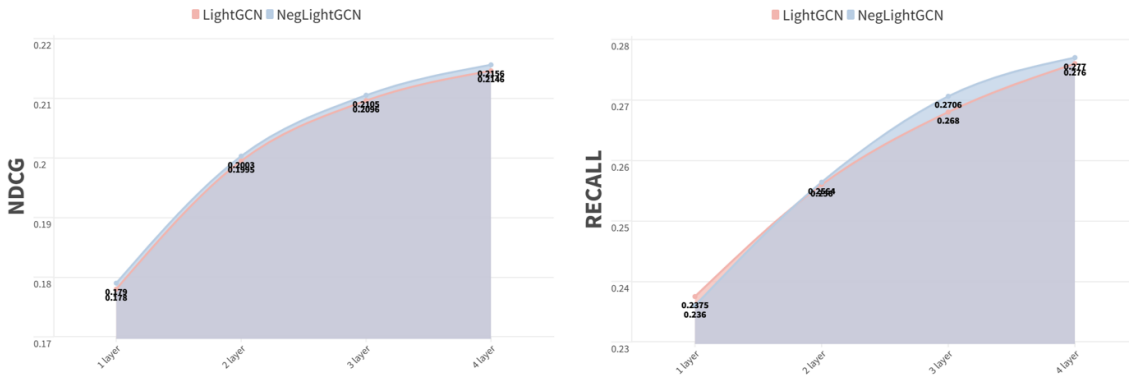


图 3. LastFM

从图 3 中可以看出，使用负采样的 NegLightGCN 比 LightGCN 表现得更好。一定程度上说明加入随机负采样对模型的过拟合程度起到了一定的缓解作用。

总体而言，实验结果显示在不同层数和数据集条件下，NegLightGCN 和 LightGCN 相对于 NGCF 在推荐系统任务中表现更为出色。对于中间层来说，NeglightGCN 有更好的表现。结果表明 NegLightGCN 更好地捕捉用户和物品之间的关系有关，从而提高了推荐的准确性和个性化程度。

6 总结与展望

在这项工作中，讨论了用于协同过滤的 GCN 的不必要的复杂设计，并进行了实证研究来证明这一论点。作者提出了 LightGCN，它由两个基本组件组成——Light Graph convertional 和层组合。在 LightGCN 中，放弃了 GCNs 中的两个标准操作——特征变换和非线性激活，降低了训练难度。在层组合中，将节点的最终嵌入构造为其在所有层上的嵌入的加权和，这被证明包含了自连接的影响，有助于控制过平滑。通过实验验证了 LightGCN 的简单性：易于训练，效率高。

相信 LightGCN 的见解对推荐模型的未来发展具有启发性。随着链接图数据在实际应用中的普及，基于图的模型在推荐中变得越来越重要,LightGCN 的思想对工业使用有实际意义。尽管该模型已经相比较之前 NGCF 有一定程度的提高，但是仍然存在地方需要改进。一个未来方向是针对个性化层组合权重 α_k，实现不同用户进行自适应的阶平滑效果。给定用户 u，项目 i 和项目 k 的因子是不对称的，这是不合理的，因为不平等地对待项目 k 和项目 i 是违反直觉的。另一个方向是在不同层嵌入时使用注意力机制。因为消息递归地将不同类型的关系组合到建模中。虽然这种协作信号理论上应该是有益的，但在消息传递公式未能捕捉到它们不同的重要性。同时堆叠多层消息传递可能会引入无信息、嘈杂或模糊的关系，这可能会在很大程度上影响训练效率和有效性。所以未来的工作会集中在灵活地调整各种关系的相对重要性方面。

参考文献

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation, 2020.

[3] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[4] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.