涉密论文 □　公开论文 □

浙 江 大 學

本科生毕业论文

题目　基于图模型的推荐纠偏方法探究

姓名与学号　王海纳 3200104934

指导教师　陈佳伟

年级与专业　2020级计算机科学与技术

所在学院　计算机科学与技术学院

递交日期　2024-6-11

浙江大学本科生毕业论文（设计）承诺书

1. 本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2. 本人在毕业论文（设计）中除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得**浙江大学**或其他教育机构的学位或证书而使用过的材料。

3. 与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

4. 本人承诺在毕业论文（设计）工作过程中没有伪造数据等行为。

5. 若在本毕业论文（设计）中有侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

6. 本人完全了解**浙江大学**有权保留并向有关部门或机构送交本论文（设计）的复印件和磁盘，允许本论文（设计）被查阅和借阅。本人授权**浙江大学**可以将本论文（设计）的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编本论文（设计）。

作者签名：　　导师签名：

签字日期：　　年　月　日　　签字日期　　年　月　日

致 谢

感谢陈佳伟作为导师兼师兄为我的论文选题、实验以及论文的写作上提供的所有指导和帮助。分配论文题目时，他曾对我说过，虽然算法竞赛很难，但科研所获得的正反馈周期依旧远比竞赛要长得多；我的毕设课题很好的印证了这一点。无论是论文学习还是进行实验上，我都经历了一个逐渐深入的学习过程。起初，当我发现模型代码可以直接开源获取，而我们的猜想听上去又十分合理时，论文在我眼前似乎成了唾手可得之物。可当我试图跑通代码时，便发现几乎所有的知识都需要靠试错、摔跟头得来。在开始的一周内，我因为没有理解传统的 recall 和有偏估计的推荐场景下的 recall，对着几乎照抄源码但和预期大相径庭的结果摸不着头脑，最后才发现是自己犯了概念性错误；在课题进行的过程中的每周组会，更是时常遇到一个我冥思苦想，卡住数日的低级错误被师兄三言两语点破，或是囫囵吞枣一篇论文后出现因为未读透论文而犯错的情况。作为我的导师兼王灿老师实验室的学长，佳伟师兄虽然日程十分忙碌，但仍然会坚持和所有的学生每周独自开会，对于我所有的失误他都会心平气和的点出。

感谢林思仪学姐为我的论文提供的指导或帮助。她是陈老师实验室里研究曝光偏差方向的学姐，在对曝光偏差的纠偏这一方向上已有发表论文。在我进行项目研究的过程中，她总能在我遇到困难时提供帮助，也在研究自己的课题之余帮我验证了论文中我起初难以置信的结果，比如改变曝光 lightGCN 层数时效果先减后增、ExpoMF 这个基线模型比较一般的情况。我在毕业论文写作的过程中，无论结构和内容上都对她的 ReCRec 论文有所借鉴。遗憾的是这篇论文目前仍处于小修阶段，因此无法被加入引用。

感谢王灿老师作为任课老师兼 ACM 集训队教练在我本科四年给予我的帮助。我和他的共同经历自我来浙大前一年始，还将延续到毕业之后；作为一个本科学生这样的缘分与经历恐怕不会有许多同学有，也早已成为我最珍贵的回忆之一。在他的指导下，我在 ACM 集训队中取得了一些成绩，也在他的课堂上学到了很多知识。在我本科四年的学习生涯中，他是我最信任的老师之一。

此外，还要感谢浙江大学所提供的良好学习氛围。

求学之路从来遍布荆棘，每个难以入眠的夜晚里，我时常会回忆起关心自己的人和曾经的美好。我即将离开这里，走向充满未知的未来；相信这四年中太多的人和事还会在我的记忆中反复被唤起，给彼时的我带来一丝温暖。

摘 要

推荐系统近年来受到学术界和工业界的广泛关注。现代的推荐系统往往建立在隐式反馈数据 (如用户点击、用户搜索) 之上，它们相比显式反馈数据的收集更为真实。但是，处理隐式反馈是更具有挑战性的，因为它们之中存在着内在的曝光偏差. 隐式反馈是偏好和曝光的共同产物，其受用户对物品曝光程度的影响显著，不能准确反映用户偏好。现有的对抗曝光偏差的方法主要是降低对于未点击数据的置信度、采用曝光模型，或使用基于倾向性模型进行杠杆倾向打分。这些方法常常导致估计存在偏差，未能产生最优结果。

为了更好的纠正曝光偏差，笔者在导师的指导下发掘了用户偏好和用户曝光可能具有的独特之处，并针对地设计并编程了使用两个分别期望训练出偏好和曝光的图卷积神经网络模型进行联合训练，并通过理论和实验分析证明了其能在一定程度上纠正曝光偏差，相比原有的模型训练更有效。

关键词：

纠偏、推荐、图卷积神经网络系统、隐式反馈

Abstract

Recommender systems have received widespread attention from academia and industry in recent years. Modern recommendation systems are often built on implicit feedback data (such as user clicks, user searches), which are more realistic than the collection of explicit feedback data. However, dealing with implicit feedback is more challenging because of the inherent exposure bias in them. Implicit feedback is a joint product of preference and exposure, and it is significantly affected by the user's exposure to the item. Does not accurately reflect user preferences. Existing methods to combat exposure bias mainly include reducing the confidence in unclicked data, adopting exposure models, or using propensity-based models for leverage propensity scoring. These methods often lead to biased estimates and suboptimal results.

In order to better correct the exposure bias, under the guidance of the instructor, the author discovered the possible uniqueness of user preference and user exposure, and specifically designed and programmed the use of two graph convolutional neural networks that are expected to train preference and exposure respectively. The network model was jointly trained, and theoretical and experimental analysis proved that it can correct the exposure bias to a certain extent, which is more effective than the original model.

Keywords:

Debias, Recommendation, Graph Convolutional Network, Implicit Feedback

目录

第一部分　毕业论文

第二部分　毕业论文开题报告

第一部分

毕业论文

1 绪论

1.1 项目背景

随着信息技术的发展，自媒体凭借其交互性、自由性和时效性的特征使得传媒生态发生了前所未有的转变，用户可以通过自媒体平台获取到更多、更开放的信息，而自媒体平台也可以通过用户的行为数据更好地了解用户的需求，从而提供更好的服务。在信息极易被制造、发布并获取的同时，信息过载问题也日益严重。推荐系统是微博、微信朋友圈等自媒体平台的核心技术之一，通过分析用户的行为数据，通过为用户推荐他们可能感兴趣的内容起到信息过滤的作用，从而提高用户的粘性和平台的活跃度。因此，推荐系统近年来受到学术界和工业界的广泛关注[1-3]。

不过，虽然推荐系统在各种应用中产生了巨大的影响，但它也面临着许多难以处理的问题，推荐偏差问题便是其中之一。推荐系统往往以用户的行为历史对用户的兴趣偏好加以预测。通常，用户的行为数据可分为显式反馈、隐式反馈两类，前者包括用户对物品的直接打分，后者则包括用户对物品的点击、浏览等记录。隐式反馈以数据量更大、更容易收集的特点，成为了现代多数推荐系统的基础。

但是，处理隐式反馈是更具有挑战性的，因为它们之中存在着内在的曝光偏差[4]. 曝光偏差产生的根因，在于隐式反馈是偏好和曝光的共同产物，而不仅仅是用户偏好的表现。用户行为数据是观察性，而非实验性的特点，其难免会受到系统曝光机制和用户自我选择的干扰。具体地，未点击数据是 (a) 用户喜欢但未曝光 (b) 用户被曝光但不喜欢 (c) 用户不喜欢且未曝光三种情况的综合。因此，它混合了用户对这个物品的积极信息和消极信息，且其中的积极信息往往被忽略。又由于推荐系统的有着反馈循环的特性，其曝光机制与用户行为会互相作用，因此曝光偏差如不能很好的处理，将会反作用于用户行为，最终造成偏差随时间的推移加剧。因此，准确归因观察到的用户-物品交互数据，从而实现高效、无偏的推荐是推荐系统领域的一项重要任务。

最近几年里有许多处理曝光偏差的工作，它们可大致被分为以下几类：

1) 加权矩阵分解模型 (Weighted Matrix Factorization, WMF)[5]。其把所有的未点击数据视作负样本，并进行启发式加权来拟合曝光。显然用户未点击一个物品不一定意味着用户不喜欢这个物品，而可能是用户喜欢这个物品但没有对这个物品被曝光。

2) 基于曝光的矩阵分解模型 (Exposure-based Matrix Factorization, ExpoMF)[6]。其引入了用户曝光于一个物品的概率，并会基于曝光数据训练得到一个推荐模型 (即训练用户是否对一个物品曝光)。但是，由于物品在实际当中并不总是均匀曝光，曝光数据会表现出与用户偏好相关的明显的有偏性，如对热门物品的明显偏好。

3) 基于倾向的矩阵分解模型 (Item Propensity Weighted models, IPW). 其像往常一样将未点击数据视为负样本，但使用了一种精巧的倾向策略去偏移这么做所带来的不利后果。尽管有着理论上的无偏性，找到一个合适的倾向本身也是一个有挑战性的问题，而且这种方法所提出的无偏估计式中存在着倾向性得分的倒数，容易在倾向性得分较小时引起较高的方差。

综上，我们认为现有的方法在利用未点击数据上还有许多优化空间。

1.2 研究内容

为了更好的解决曝光偏差给推荐带来的问题，笔者试对隐式反馈进行归因。考虑对于每个用户-物品对，在使用一个偏好 01 变量 $R = 0/1$ 表示偏好与否的基础上，引入曝光 01 变量 $O = 0/1$ 表示物品是否曝光给用户。由此，隐式反馈被这两个变量的取值分为四种情况：(a) 偏好且曝光; (b) 偏好但未曝光; (c) 不偏好但曝光; (d) 不偏好且不曝光。如果能同时训练出 R 和 O, 就可以明确属于 (a)(b) 两类的用户对商品的真实兴趣，从而优化推荐效果。

在本文中，笔者提出了我们设计的纠偏模型。我们的纠偏模型采用生成模型为基础，同时显式训练代表每个用户-物品对偏好或曝光变量为 1 的概率的偏好矩阵和曝光矩阵，由此来归隐隐式反馈的四种情况，通过期望最大化算法进行估计；同时通过猜想偏好与曝光的不同性质，对训练曝光的模型加以修改来增强矩阵分解的效果。

笔者通过理论证明该方法对比其他方法方差更小，并在基准数据集上进行实验验证模型性能。实验结果显示，曝光模型的加入对偏好模型的训练效果提升是显著的，通过让两个模型以不同的特征值正则项与不同的 LightGCN 层数进行训练，能进一步提升训练效果。

1.3 论文结构

本文的剩余部分组织如下：

第 2 节介绍本课题的相关工作；第 3 节给出问题描述，并介绍生成嵌入向量的基础模

型；第 4 节给出笔者的方法；第 5 节实验测试了纠偏模型和极限方法的有效性、并对关键参数做了消融实验；第 6 节进行总结与讨论，对笔者的工作进行总结，并讨论课题在未来可能的提升方向。

2 相关工作

本节将从协同过滤、推荐系统种的偏差、现有曝光偏差的纠偏方法三个方面介绍与本课题相关的工作。

2.1 协同过滤

协同过滤 (Collaborative Filtering, CF) 是一种在现代推荐系统中广泛使用的技术[7]。协同过滤模型可分为基于邻居和基于模型两类。基于邻居的方法计算用户与用户、商品与商品间的相似度矩阵，为用户推荐与其相似的用户的喜爱商品或与其喜爱商品类似的商品；基于模型的矩阵分解 (Matrix Factorization, MF) 方法为每个用户与物品建立一个 k 维向量，通过对应向量的点击值来反映用户与物品的相关程度。最近的神经推荐模型如 NCF[8], LRML[9], CJMF[10], BISER[11]等大体使用类似的嵌入构造，同时通过各自的神经网络构造增强对数据集的建模。此外，为了丰富对用户特征的学习，最近的一些研究还会把用户的一些历史行为纳入模型中。其中最典型的方法包括 FISM[12]和 SVD++[13]，它们通过汇集交互物品的嵌入来生成用户的嵌入。本文中用到特征值加入模型也受了 SVD++ 文章的启发。最近，物品交互的时序收到了广泛关注，还催生出了序列推荐折一新兴领域。各种技术被用于编码用户的隐式反馈信息，从传统的马尔科夫链[14]、序列概率模型到最近的 transformer[15]与神经微分模型[16]都属此类。

2.2 推荐系统中的偏差

运用于推荐系统中的数据往往是通过调取推荐平台的已有数据，通过观察而非严格实验收集的，偏差经常在这些系统中体现[17]。盲目的在这种有偏的数据上训练模型，不仅可能导致推荐准确度下降，还可能加剧马太效应。[18]

近期的研究中，数据偏差被分为四大类，本文所研究的曝光偏差是其中的一类。另外三类包括选择偏差、位置偏差和从众偏差。选择偏差[19-20]指所观察到的数据不一定能代表实际情况，这主要出现在显式反馈数据中；位置偏差[21-22]指用户反馈可能会受到物品被陈列位置的影响，如处于显眼位置的物品相比需要拖动进度条才能看见的物品可能会更容易受到曝光；从众偏差[23-24]则是指用户可能受到来自平台氛围等诸多因素的影响，倾向于在

一个群体中表现得与其他人类似。

不同类型的偏差，表现出迥异的特征。这些偏差，伴随着推荐系统的反馈循环[4]各个环节而生，并在其中逐步累积。

推荐系统的主体由用户、数据、模型组成，其反馈循环可自然地分为用户-数据阶段、数据-模型阶段和模型-用户三个阶段，而这些偏差大多出现在用户-物品阶段。由于用户更倾向于给自己喜欢的商品评分，或给特别优、劣的商品刷分，导致观察到的数据出现选择偏差；用户可能受到来自平台氛围等因素的影响，违背自己偏好而给出类似评分，导致数据出现从众偏差；同样，用户可能倾向于对推荐列表中较醒目位置的物品进行交互，未观察到的交互可能源于物品未对用户曝光，由此导致数据出现位置偏差和曝光偏差。

2.3 针对曝光偏差的纠偏

曝光偏差是常见于隐式反馈数据中的偏差，前文已分析了其根本特点。曝光偏差很常见、成因亦很多，包括物品本身的流行度、用户背景和平台先前的推荐策略等等。

加权矩阵分解和基于曝光的矩阵分解是前文中具体分析过的方法。最近的一些工作还提出了基于倾向得分[25]（propensity score）来解决曝光偏差。这种方法具有一个显著的理论特性——在倾向正确指定时实现理想损失的无偏估计。由于这些方法的关键在于倾向的准确性，研究人员针对地开发了各种策略，例如使用推荐模型的预测[11]、物品流行度[26]或者利用联合学习策略[27]来同时推断倾向和用户偏好等。

除了以上这些针对曝光偏差的纠偏方法外，还有一些通用的纠偏方法可以用于解决曝光偏差；篇幅有限不作详细讨论。

3 预备部分

3.1 问题描述

本小节将给出带有隐式反馈的推荐问题描述。

假设我们有一个推荐系统，其中用户集合为 U, 物品集合为 I. 让 $u \in U$ 作为一个用户，$i \in I$ 作为一个物品，$D = U \times I$ 作为所有用户-物品对的集合。用户对物品的历史交互信息可以被表示为一个 $U \times I$ 大小的 01 矩阵 Y，其中的元素 y_{ui} 为 01 变量，用于表示用户 u 是否点击过物品 i. 根据已有的研究成果[28-29]，在隐式反馈中，点击会被视作用户偏好和曝光的综合结果。形式化地，记 r_{ui} 表示用户 u 是否实际上对物品 i 感兴趣，o_{ui} 表示用户 u 是否对 i 曝光。现有的学习一般会做出如下假设：

$y_{ui} = o_{ui} r_{ui}$.

即，用户点击物品当且仅当它既喜欢它又对它被曝光。同时，定义

$$P(y_{ui} = 1) = P(o_{ui} = 1)P(r_{ui} = 1) = \theta_{ui}\gamma_{ui}$$

将物品点击物品的概率分解为兴趣程度和曝光概率，其中 θ_{ui} 表示用户-物品之间的曝光概率，γ_{ui} 表示用户对物品的兴趣程度。

在没有曝光偏差时, θ_{ui} 是一个常数，而 γ_{ui} 是一个随机变量。在有曝光偏差时，θ_{ui} 是一个随机变量，而 γ_{ui} 是一个常数。在实际应用中，我们往往只能观测到 y_{ui}，而不能观测到 r_{ui} 和 o_{ui}。因此，纠偏隐式反馈推荐目标即是学习一个模型，能够通过观测到的点击变量 y_{ui} 预测用户对物品的偏好概率 ξ_{ui}，即 $P(r_{ui} = 1)$。

根据[26] 等人提出的观点, 推荐模型的理想损失 $L_{ideal}(\hat{\xi})$ 可被表示如下[28]：

$L_{ideal} = \frac{1}{|D|} \sum_{(u,i) \in D} (\xi_{ui} \delta^{(1)}(\hat{\xi_{ui}}) + (1 - \xi_{ui}) \delta^{(0)}(\hat{\xi_{ui}}))$

其中 ξ_{ui} 代表用户 u 对物品 i 真实的偏好概率（即，$\xi_{ui} = P(r_{ui} = 1)$). $\hat{\xi_{ui}}$ 表示推荐模型对 ξ_{ui} 的预测值，$\delta^{(r)}$ 是衡量预测值与标签之间的函数。其可以是诸如最小二乘、交叉熵之类的损失函数。

自然，理想损失中的 ξ 在训练过程中是不可获取的，模型训练只能基于点击数据 y. 这启发研究者们探索一种对理想损失的无偏估计。

3.2 基线方法

本小节将介绍论文中用于生成嵌入的生成模型，它们在测试时也会作为基线方法之一。

笔者先介绍神经网络协同过滤[30](Neural Graph Collaborative Filtering, NGCF)、矩阵分解 (Matrix Factorization, MF)[31]模型和轻量化图卷积神经网络 (Light Graph Convolutional Network, LightGCN)[32]这两个在方法中作为基准模型以及方法改进基础的模型。协同过滤 (Collaborative Filtering，CF) 是一种通过分析用户或事物之间的相似度来预测用户可能感兴趣的内容，协同大家的反馈、评价和意见一起对海量的信息进行过滤，从中筛选出目标用户可能感兴趣的信息的推荐过程。协同过滤具有简单、可解释性强的优点，但其显著缺点是推荐结果的头部效应明显，处理稀疏向量的能力弱，在数据具有明显长尾效应时劣势明显。神经网络协同过滤以神经网络图为范式进行协同过滤。可以分为三个部分来看：

嵌入 (Embeddings)：也称嵌入向量，对用户和物品的嵌入向量。嵌入传播层 (Embedding Propagation Layers)：挖掘高阶连通性关系来捕捉交互以细化嵌入的多个嵌入传播层预测层 (Prediction Layer)：用更新之后带有交互信息的用户和物品嵌入向量来进行预测

而矩阵分解 (matrix factorization，MF) 正是在协同过滤算法中的共现矩阵的基础上，加入了隐向量的概念，加强了模型处理稀疏矩阵的能力，针对性地解决了协同过滤存在的主要问题。隐向量的本质是将高维稀疏输入向量 x(one-hot 编码) 转换为低维密集的嵌入向量的 embedding 矩阵。

轻量化图卷积神经网络则是结合了矩阵分解的图卷积神经网络。LightGCN 的卷积操作被如下定义：

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

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

其中 N_u 和 N_i 分别是用户 u 和物品 i 的邻居集合。LightGCN 的卷积操作是一个简单的邻居聚合操作，它只是将邻居的嵌入向量相加，而不是像传统的图卷积神经网络那样进行复杂的线性变换。LightGCN 的优势在于它的简单性，它只有一个卷积层，没有任何正则化项，也没有任何激活函数。LightGCN 的简单性使得它非常容易实现，而且非常容易训练。在聚合多层时，LightGCN 使用了多层求和的方式，使用公式 $e^u = \sum_{k=0}^{K} \alpha_k e_u^{(k)}$ 和 $e^i = \sum_{k=0}^{K} \alpha_k e_i^{(k)}$

来聚合多层的信息。这种多层求和的本质是让预测结果更加平滑 (Smoothing)。这种方式使得 LightGCN 的训练过程非常简单，而且非常容易实现。LightGCN 的简单性使得它非常容易实现，而且非常容易训练。

矩阵形式。令用户-物品交互矩阵为 R_{nm}，用户和物品的嵌入矩阵分别为 $U_{n\times d}$ 和 $V_{m\times d}$，则数据集的邻接矩阵 A 为：

$$\begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$

令第 0 层嵌入为 $E^{(0)} \in R^{(M+N)\times T}$，其中 T 为嵌入大小，则 LightGCN 的卷积操作可以写成矩阵形式：

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

其中 D 是一个 $(M+N)\times(M+N)$ 的对角矩阵，D_{ii} 为第 i 行非零元素的个数，即 $\sum_j \hat{A}_{ij}$，$\hat{A} = A + I$，I 为单位矩阵。最后，我们得到了用于模型预测的推荐矩阵：

$$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + \cdots + \alpha_k E^{(K)} = \alpha_0 E^{(0)} + \alpha_1 \hat{A}E^{(0)} + \cdots + \alpha_k \hat{A}^k E^{(0)}$$

其中 K 为卷积层数，α_k 为每一层的权重，$\hat{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ 为 A 的归一化矩阵。

结构对比。图 3.1给出了三者的结构对比。

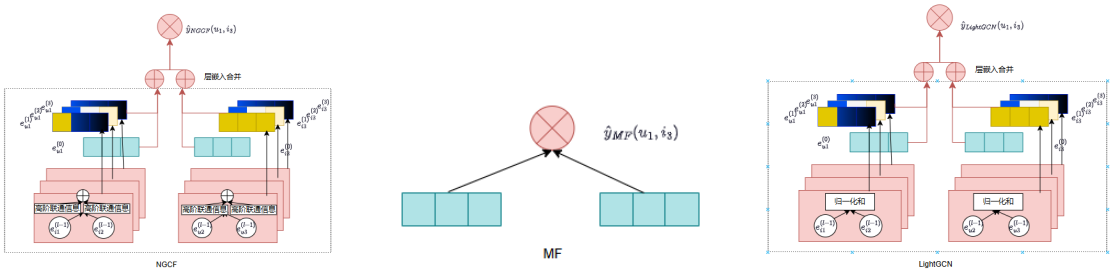


图 3.1 NGCF,MF,LightGCN 模型结构对比。MF 模型本身没有用到卷积层，结构最简单；NGCF 最为复杂，LightGCN 则是在两者之间, 简化了 NGCF 的领域信息获取部分

3.3 现有方法

基于前文对带有隐式反馈的推荐的探索，我们重新考察现有方法并讨论它们的局限性。

3.3.1 加权矩阵分解模型

加权矩阵分解模型（Weighted Matrix Factorization, WMF）[5] 将所有未点击数据都视作负样本，其损失函数中 $\hat{\xi} = y$, 又启发式地加上了置信权重 w_{ui}. 故其 loss 可表示如下：$L_{WMF} = \frac{1}{|D|}\sum_{(u,i)\in D}(y_{ui}\delta^{(1)}(\hat{\xi_{ui}}) + (1-y_{ui})w_{ui}\delta^{(0)}(\hat{\xi_{ui}}))$ 由于它会将有偏好未曝光的情况都置为 1, 这个 loss 无疑是有偏的。

3.3.2 基于曝光的矩阵分解模型

基于曝光的矩阵分解模型 (Exposured-based Matrix Factorization, ExpoMF)[6]提出了 $\hat{P}(o_{ui})$ 表示用户曝光于物品的概率。其 loss 可表示如下：$L_{ExpoMF} = \frac{1}{|D|}\sum_{(u,i)\in D}(y_{ui}\delta_{ui}^{(1)} + (1-y_{ui})o_{ui}\delta_{ui}^{(0)})$ 这个损失是基于物品明确被曝光给用户的情况计算的。它可以纠偏一些标签，因为当 $o_{ui} = 1$ 时，$y_{ui} = r_{ui}$ 是成立的。但是，由于物品往往不是被均匀曝光的，曝光数据往往会显露出一定的偏性，所以模型的预测也是有偏的。

3.3.3 基于倾向的矩阵分解模型

基于倾向的矩阵分解模型 (Item Propensity Weighted models, IPW). 针对无偏理想损失函数,[26]提出了基于倾向的矩阵分解模型。其 loss 可表示如下：$L_{IPW} = \frac{1}{|D|}\sum_{(u,i)\in D}(y_{ui}\delta_{ui}^{(1)} + (1-y_{ui})\frac{1}{\hat{P}(o_{ui})}\delta_{ui}^{(0)})$ 其中，倾向性得分 $\theta_{ui} = p(o_{ui} = 1)$. 该式被证明对于理想损失函数是无偏的。然而，该方法所提出的无偏估计式中存在着 $\frac{1}{\theta}$, 由于 $\theta \in (0,1)$，其可能很小，这会引起较高的方差。

4 方法

本章节将详细介绍笔者所设计的纠偏模型结构，并与前人的模型进行对比。

4.1 模型设计思路

据上一节中对问题的描述，问题的目标是从包含隐式反馈的数据集中推断出用户对物品的偏好。对于一个用户-物品对，隐式反馈只将数据分为观察到和未观察到交互两种情况，而用户与物品的联系受偏好和曝光控制实际存在四种情况。因此，直接将观察到的样本数据作为用户兴趣的负反馈会使预测存在曝光偏差。

基于此，我们令 $r_{ui} \in \{0,1\}$ 表示用户对物品感兴趣与否，又令 $o_{ui} \in \{0,1\}$ 表示用户对物品曝光与否。显然有 $y_{ui} = r_{ui} \cdot o_{ui}$ 表示用户是否会与物品交互。通过引入曝光变量，我们可以更准确的对用户兴趣与观察到的交互信息进行建模，训练时以 y_{ui} 尽量拟合训练集，而以 r_{ui} 作为模型输出。如此，可排除曝光偏差的影响。

4.2 模型结构

本小节将介绍模型结构。

根据前面的分析，由于曝光数据会显露出一定的偏性，笔者选择使用两个 lightGCN 模型构建纠偏模型 (图 4.1)，每个模型包括用户和物品的嵌入向量 U_r, I_r 和 U_o, I_o, 分别训练偏好矩阵 $R = U_r \cdot I_r$ 和曝光矩阵 $O = U_o \cdot I_o$. 迭代过程的本质是期望最大化（Expectation Maximum, EM）算法，重复迭代，并在每次迭代中执行以下两个步骤：

1) 根据当前的模型变量参数计算得到 r_{ui} 和 o_{ui} 变量的先验概率，即提取模型当前的预测得分。

2) 取两个先验概率过 sigmoid 函数 ($sigmoid(x) = \frac{1}{e^{-x}+1}$) 后相乘得到的结果作为 $\xi_{ui} = P(y_{ui} = 1)$，并据此和训练集样本计算损失函数，更新模型参数，然后进行反向传播。

若干轮迭代后，我们取偏好模型的预测结果作为最终的推荐结果，在测试集上进行测试。迭代的终止判断采用领域内比较流行的早停测试，即对测试集划分约 20% 的数据，每若干次迭代后在其上测试模型效果，当此效果连续未超过先前结果的迭代轮数超过上限时停止训练，最后在测试集剩余数据上测试效果，作为模型的最终效果。

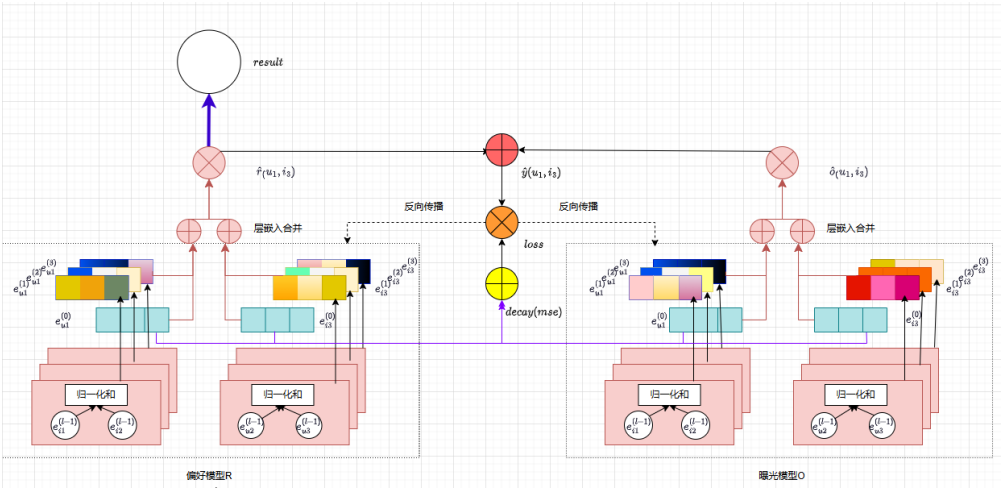


图 4.1 纠偏模型的基本结构

但仅仅如此，偏好和曝光矩阵就是对偶的，因为模型在结构上没有区别，两个矩阵都可以被认为是偏好矩阵，难以进行区分。由此，需要挖掘出偏好和曝光的不同之处，在两个矩阵训练时体现这一点，如此才能优化训练效果。

笔者对二者的性质进行了一些猜想。

a) 维度猜想如图 4.2。猜想曝光矩阵的秩应比偏好小一些，因为广告投放的方式会随着信息技术的发展愈发普遍，而人的偏好则是比较稳定的。前人的研究[26]中，已经明确了偏好向量的维度会影响训练效果，且在一定范围内维度越大训练效果越好。因此，笔者考虑保持偏好矩阵嵌入维度不变，降低曝光矩阵的维度。

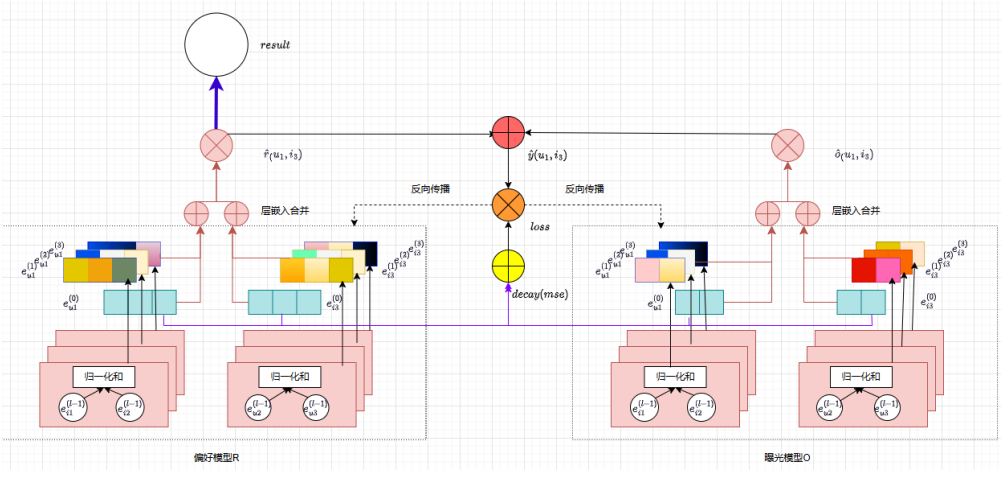


图 4.2 维度猜想对应的模型结构

b) 层数猜想如图 4.3。使用 lightGCN 训练的本质是控制预测向量的平滑程度。猜想让

偏好和曝光所需要的平滑程度应该不同，因此在两个模型上使用不同的 lightGCN 层数可能可以提升训练效果。

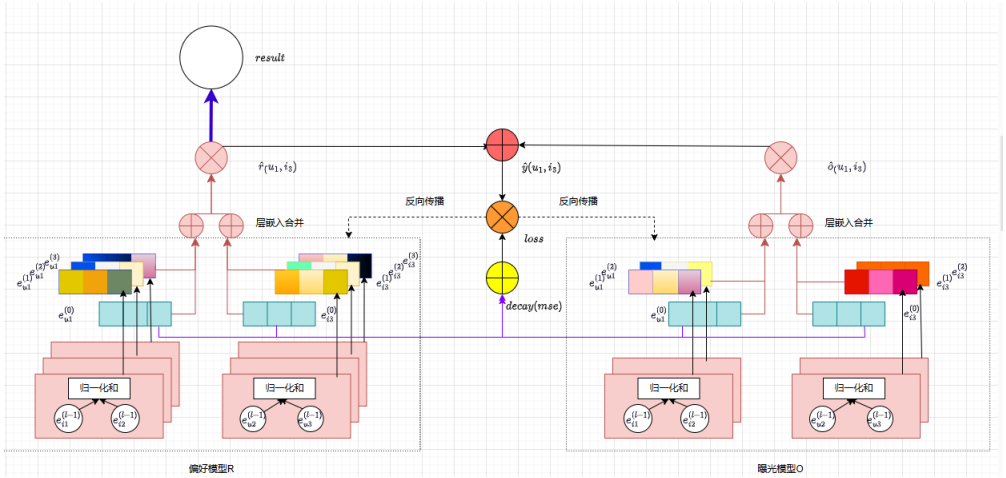


图 4.3 层数猜想对应的模型结构

c) 特征值猜想如图 4.5。我们猜想偏好矩阵应有更小的矩阵的最大特征值，曝光应有更大的矩阵的最大特征值。因为偏好矩阵的长尾效应更严重，所以偏好矩阵的特征值的分布应该更紧凑，而曝光矩阵的特征值则会两极分化。笔者将其引入训练损失函数。

可以通过对基准数据集进行数据统计来验证其长尾效应情况。Yahoo!R3[33]、Kuairand[34]和下文实验中 Coat[11]是三个纠偏推荐训练中的基准数据集。它们相比 Gowalla 等传统推荐场景下数据集的特性是，训练集是用户的主观提供的打分数据，有曝光偏差；而测试集是由平台用户提供了每个物品的详细信息后用户打分得到的，没有曝光偏差。我们可以通过比较这些数据集训练集和测试集的分布来验证曝光偏差的存在。图 4.4的横坐标为将所有物品按被用户点击的次数升序排序后前 k 个物品的结果，纵坐标为这些物品的点击总数占整体的比例。可以看到，无曝光偏差的测试集曲线较为平滑，折线图右侧最受喜好的约前 20% 只占了总喜好数的约 40% 以下；而有曝光偏差的训练集曲线有明显的头部效应，折线图在右侧斜率较高，说明最受喜好的小部分物品占总喜好数的比例很大。这说明了曝光矩阵具有一定的偏性，而偏好矩阵相对均匀，有更强的长尾效应。

对于嵌入矩阵的特征值，有两种可用的求法：

a）考虑幂法求矩阵特征值；近似的以第一步迭代的结果，即一个全 1 向量和矩阵相乘得到的结果作为最大特征值。

b）由于使用矩阵分解模型，注意到我们要训练的矩阵可以直接被分解为用户和物品

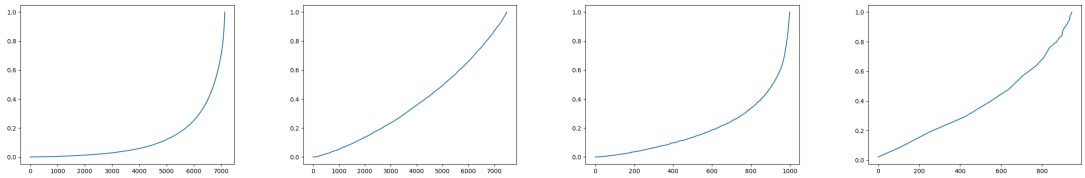


图 4.4 kuairand,yahoo 训练集物品偏好/分布图，从左至右分别为 kuairand 训练集,kuairand 测试集,yahoo 训练集,yahoo 测试集。

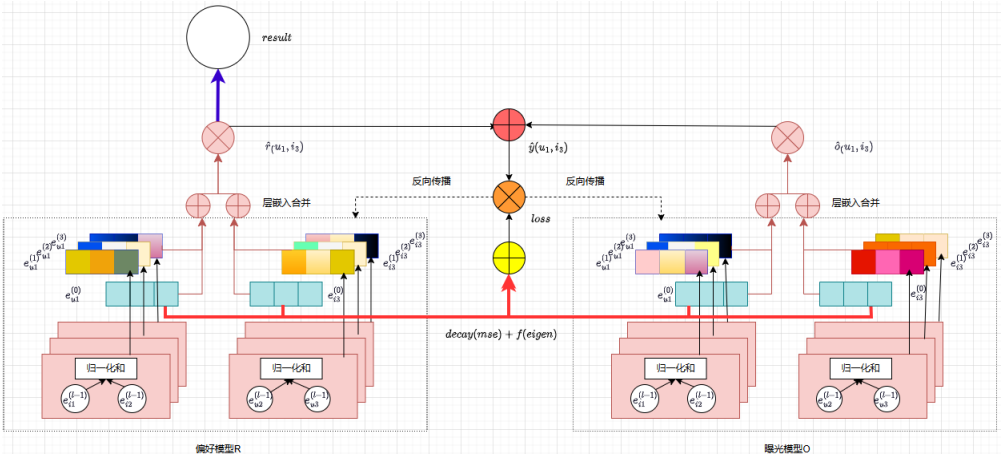


图 4.5 特征值猜想对应的模型结构

嵌入的乘积，而嵌入向量在训练过程中共线的可能性很低，所以嵌入向量每一维的二范数之积就是全部的特征值。起初我们尝试了使用幂法求特征值的近似法来，这样训练速度明显更快，但只能控制最大特征值；后来我们发现直接控制每一维的二范数之积更为有效，还可以控制前 k 大特征值，因此最后采用了第二种方式。

4.3 损失函数

本小节介绍模型使用的损失函数。

对于 MF 模型和 LightGCN 模型，我们使用经典的交叉熵 loss 加上正则项作为损失函数。

$$Logloss(\xi) = \sum_{(u,i)\in D} (\xi_{ui} \log(\hat{\xi_{ui}}) + (1 - \xi_{ui}) \log(1 - \hat{\xi_{ui}})) + \lambda ||E||^2$$

其中 ξ_{ui} 代表用户 u 对物品 i 真实的偏好概率，$\hat{\xi_{ui}}$ 表示推荐模型对 ξ_{ui} 的预测值；λ 是超参数，为正则项的系数；E 是嵌入向量。

如模型结构图所示，我们以两个模型预测值过 sigmoid 函数后的结果拟合训练集，而只用第一个模型的预测值进行测试；即用 $sigmoid(r_{ui})sigmoid(o_{ui})$ 代入 ξ_{ui}，使用上面的 loss 函数进行训练。

而为了验证特征值猜想，我们还尝试了在 loss 中加入了有关特征值的正则项，得到如下的损失函数：

$$Logloss(\xi) = \sum_{(u,i)\in D} (\xi_{ui}\log(\hat{\xi_{ui}}) + (1-\xi_{ui})\log(1-\hat{\xi_{ui}})) + \lambda||E||^2 + \lambda_r f(eigen_r) + \lambda_o f(eigen_o)$$

其中 $f(eigen)$ 是一个函数，用于计算矩阵的特征值，$eigen_r$ 和 $eigen_o$ 分别是偏好和曝光矩阵的特征值，λ_r 和 λ_o 是正则化参数。$f(eigen)$ 可以是幂法求矩阵特征值的结果，也可以是嵌入向量每一维的二范数之积。$f(eigen)$ 的设置笔者尝试了 2 种：

1) $f(eigen) = \frac{\sum_{i=0}^{k-1} eigen}{k}$. 即直接取前 k 大特征值均值作为正则项。加入这一项便是限制最大特征值，调整 λ_r 与 λ_o 便让偏好和曝光的最大特征值有所差别。

2) $f(eigen) = \frac{\sum_{i=0}^{k-1} eigen_i - eigen_{k+i}}{k}$. 即取前 k 大特征值之和与第 $k+1$ 到第 $2k$ 大特征值之和的差的均值作为正则项。

整体算法如 算法 4.1所示。

4.4 模型对比

本小节试将本论文提出的模型结构与已有的纠偏模型进行对比分析.

4.4.1 WMF

WMF 模型会将有偏好未曝光的情况都置为 1，仅仅通过启发式的设置置信权重，训练得到的曝光偏差不能被很好的去除；而我们的方法则能解决这一问题。

4.4.2 ExpoMF

根据前文对此模型的分析, 可以发现 ExpoMF 模型只对曝光数据的兴趣偏好进行估计, 忽略了未曝光的数据; 而曝光数据和未曝光数据中用户兴趣的分布不一致, 造成了其方法存在有偏性. 而我们的方法通过单独设立曝光模型, 通过估计曝光概率得分来对所有的数据

算法 4.1 纠偏模型

输入：训练集 Y

输出：预测的兴趣矩阵 R

$U_r, I_r, U_o, I_o \leftarrow$ 随机初始化状态

while 验证集性能在最近 K 个周期内有所提升 **do**

计算矩阵 $R = P(r_{ui} = 1) = U_r I_r^T$ 和 $O = P(o_{ui} = 1) = U_o I_o^T$，基于嵌入 U_r, I_r, U_o, I_o

使用 $sigmoid(R) \cdot sigmoid(O)$ 计算 $\xi_{ui} = P(y_{ui} = 1)$

使用 Y 和 ξ 计算交叉熵 loss

使用 $\sum_{embedding=U_r,I_r,U_o,I_o} norm(embedding)$ 计算正则项 loss

if 在损失中添加特征值 **then**

使用 $norm(U_r) \cdot norm(I_r), norm(U_o) \cdot norm(I_o)$ 计算 $Eigen_r, Eigen_o$

计算 $f(Eigen_r), f(Eigen_o)$

end if

计算 $Loss$

使用不同的偏好模型与曝光模型结构，同时对 U_r, I_r, U_o, I_o 反向传播 $Loss$

使用验证集和 R 评估性能

end while

进行偏好的估计; 利用了未曝光数据. 在实验中, 笔者发现 ExpoMF 在基准数据集上的效果并不理想, 远不如直接使用单个 MF 模型进行训练的效果.

4.4.3 IPW

前文提到, IPW 模型的无偏估计式为 $L_{IPW} = \frac{1}{|D|}\sum_{(u,i)\in D}(y_{ui}\delta_{ui}^{(1)} + (1-y_{ui})\frac{1}{\hat{P}(o_{ui})}\delta_{ui}^{(0)})$ 其中存在 $\frac{1}{\delta}$ 一项, 由于 $\delta \in (0,1)$，其可能很小，这使得其方差没有理论上界, 在极端情况下可能很大。而笔者模型对兴趣偏好变量的损失函数可记为 $L_{prefer} = (\frac{1}{|D|}\sum_{ui}\phi_{ui}\delta_{ui}^{1} + (1-\phi_{ui})\delta_{ui}^{0}) + f(eigen)$ 其中 $\phi_{ui} = p(r_{ui}=1)$.

以下证明: 以上损失函数的方差 $D(L_{prefer}) \le \frac{1}{4}\frac{1}{|D|}\sum_{ui}(\delta_{ui}^{(1)} - \delta_{ui}^{(0)})^2 + D(f(eigen))$

证明: 记 $l_{ui} = \phi_{ui}\delta_{ui}^{(1)} + (1-\phi_{ui})\delta_{ui}^{(0)}$, $D(l_{ui}) = E[(l_{ui})^2] - (E[l_{ui}])^2$.

对第一项平方和公式展开:

$$l_{ui}^2 = (\phi_{ui}\delta_{ui}^{(1)} + (1-\phi_{ui}\delta_{ui}^{(0)}))^2 = \phi_{ui}^2(\delta_{ui}^{(1)})^2 + (1-\phi_{ui})^2(\delta_{ui}^{(0)})^2 + 2\phi_{ui}(1-\phi_{ui})\delta_{ui}^{(1)}\delta_{ui}^{(0)}$$

$$E[l_{ui}^2] = E[\phi_{ui}^2(\delta_{ui}^{(1)})^2 + (1-\phi_{ui})^2(\delta_{ui}^{(0)})^2 + 2\phi_{ui}(1-\phi_{ui})\delta_{ui}^{(1)}\delta_{ui}^{(0)}]$$

同理，记 $E(\phi_{ui}) = \gamma$，对于第二项也有

$$(E[l_{ui}])^2 = (\gamma\delta_{ui}^{(1)} + (1-\gamma)\delta_{ui}^{(0)})^2 = (\gamma)^2(\delta_{ui}^{(1)})^2 + (1-\gamma)^2(\delta_{ui}^{(0)})^2 + 2\gamma(1-\gamma)\delta_{ui}^{(1)}\delta_{ui}^{(0)}$$

最后得到

$$D(l_{ui}) = E[(l_{ui})^2] - (E[l_{ui}])^2 = (E[\phi]^2 - (\gamma)^2)(\delta_{ui}^{(1)} - \delta_{ui}^{(0)})^2 = D(\phi_{ui})(\delta_{ui}^{(1)} - \delta_{ui}^{(0)})^2$$

由于 ϕ_{ui} 取值在 [0,1] 之间，其方差小于 $\frac{1}{4}$，所以 $D(l_{ui}) \le \frac{1}{4}(\delta_{ui}^{(1)} - \delta_{ui}^{(0)})^2$，由此可得

$D(L_{prefer}) \le \frac{1}{4}\frac{1}{|D|}\sum_{ui}(\delta_{ui}^{(1)} - \delta_{ui}^{(0)})^2 + D(f(eigen))$。

$D(f_{eigen})$ 作为矩阵的特征值可以最终推导为矩阵内元素的表示，并且可以通过设置正则项系数来减小，故纠偏模型的平方和是有上界的。

5 实验

本小节将进行实验回答两个主要的问题：我们的方法与现有方法相比推荐性能如何，以及纠偏模型中偏好模型和曝光模型的学习效果是否符合我们的猜想。

5.1 数据集

笔者引用了数据集 Yahoo!R3(以下简称 Yahoo) 和 Coat 用于实验[11]，它们都是上文中提到的测试集和训练集曝光情况不同的数据集。我们使用有偏的数据作为训练集，然后将无偏的测试集分成用于早停测试 (20%) 和用于测试最后结果 (80%) 的两部分。同时，数据集中用户对物品打出的 1-5 分的分数被转化成了正负样本，4-5 分为正样本，1-3 分为负样本.

表 5.1 数据集信息

数据集	用户 #	物品 #	点击数 #	图密度
Yahoo!R3	15400	1000	125077	0.00812
Coat	290	300	6960	0.08

5.2 测试细节

5.2.1 评价指标

笔者用 Recall@5 和 NDCG@5 指标作为评价指标。Recall@K 是指在前至多 K 个无偏曝光推荐物品中，用户真实感兴趣的物品的比例。NDCG@K 是指在前 K 个推荐物品中，用户真实感兴趣的物品的排名倒数的加权和. 公式化的表达如下：

$$Recall@K = \frac{1}{|U|}\sum_{u\in U}\sum_{i\in R(u)}\frac{r_{ui}I[\hat{Z}_{ui} < K]}{min(K, |T(u)|)} \tag{5-1}$$

其中 U 为用户集合，$R(u)$ 为推荐给用户 u 的物品集合，$T(u)$ 为测试集中用户 u 真实感兴趣的物品集合，Z_{ui} 为推荐模型对用户 u 对物品 i 的预测值在 i 的排序。$I[\hat{Z}_{ui} < K]$ 是一个

指示函数，当 $\hat{Z}_{ui}$ 小于 K 时为 1，否则为 0。

$$NDCG@K = \frac{1}{|U|}\sum_{u\in U}\frac{DCG(u)}{IDCG(u)} \tag{5-2}$$

其中 U 为用户集合，$DCG(u)$ 为推荐给用户 u 的物品集合的 DCG 值，$IDCG(u)$ 为测试集中用户 u 真实感兴趣的物品集合的 IDCG 值。

5.2.2 超参数设置

层数. 没有特别说明使用的 lightGCN 层数的实验均没有使用 LightGCN，以确保实验指标的提升没有来自 LightGCN 本身，更好的对比猜想的效果。在使用 LightGCN 时，我们尝试了 1-4 层的 LightGCN，最后发现 2 层的效果最好；而在层数的调整中，我们就固定了偏好模型使用 2-3 层 lightGCN，改变曝光模型所使用的层数。

维度. 除“改变曝光维度”这组实验中，部分实验组别说明的改变后的曝光维度外，所有模型的维度设置均为 200. 例如，实验结果中的“纠偏模型-基线”一组中，偏好和曝光维度都为 200.

学习率. 除“改变曝光学习率”这组实验中，部分实验组别说明的改变后的学习率外，所有模型的学习率均设置为 0.04. 在实验过程中我们在 $[10^{-4}, 1]$ 的范围内调整了学习率，最后发现 0.04 是一个既大致保证效果又能保证学习速度的参数。

批大小. 由于数据集较小, 我们在实验中没有分批训练, 每次迭代过程均使用整个数据集进行训练.

5.2.3 实现细节

所有实验中的单个矩阵分解模型 (或 LightGCN 模型) 统一使用 Adam 优化器优化, 每次迭代时反向传播.

5.3 实验结果

5.3.1 性能对比

表 5.2展示了每种模型的 Recall@5 和 NDCG@5 的结果。结果显示，不使用 LightGCN 时，使用两个模型联合训练（称为“纠偏模型”）相比使用单个模型训练效果有了明显的

提升；而在使用两个模型的情况下，加入特征值可以让训练效果获得进一步的提升；使用 LightGCN 时，改变层数对于 Yahoo 测试集有一定提升，但对于 Coat 测试集没有提升。

表 5.2 不同模型在 Yahoo 和 Coat 数据集上的性能对比

模型	Yahoo-Recall@5	Yahoo-NDCG@5	Coat-Recall@5	Coat-NDCG@5
MF	0.7955	0.6604	0.5563	0.5214
ExpoMF	0.7333	0.6136	0.5390	0.5207
纠偏模型-基线	0.8092 (+1.72%)	0.6703 (+1.49%)	0.5944 (+6.84%)	0.5245 (+0.60%)
纠偏模型-维度	0.8092 (+1.72%)	0.6703 (+1.49%)	0.5991 (+7.70%)	**0.5337** (+2.36%)
纠偏模型-特征值	**0.8135** (+2.25%)	**0.6716** (+1.69%)	**0.6026** (+8.32%)	0.5290 (+1.46%)
LightGCN-最好层数	0.8142	0.6783	0.5563	0.5214
纠偏模型-层数	0.8214 (+0.88%)	0.6833 (+0.74%)	0.5563 (0%)	0.5214 (0%)

5.4 消融实验

5.4.1 维度猜想

图 5.1, 图 5.2展示了训练时改变曝光模型维度时, 得到的 recall@5 变化折线图. 为了观察曝光模型的训练效果, 笔者分别以偏好和曝光模型进行测试，并将结果绘制在折线图上。

据实验, 笔者遗憾的发现实验结果与维度猜想有不小的出入。根据猜想，两个模型在其他训练参数上没有任何区别，理论上是对偶的，它们单独训练的训练效果应比较接近，尤其在曝光维度与偏好维度相差不多时。但实际上，曝光模型在整个训练的过程中的参数改变相当有限，在偏好维度固定时, 其维度对训练影响甚微。因此，维度猜想基本可以否定。

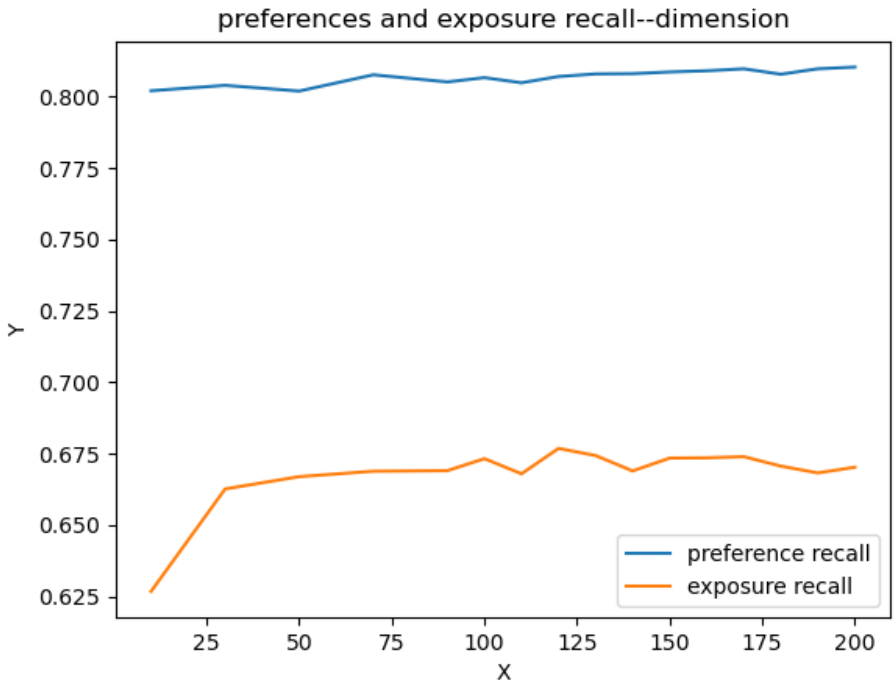


图 5.1 改变曝光模型维度时 YahooR3 数据集上的 recall 变化折线图。横坐标为曝光维度，纵坐标为指标值

在 Yahoo 数据集上，曝光维度为 200 时训练效果最好；更小的曝光维度对效果没有提升；在 Coat 数据集上，曝光维度为 180 时效果略有提升。笔者认为这和 Coat 数据集较小、训练结果容易波动有关系，不能说明使用比 200 低一些的曝光维度就能让训练效果更好。所以, 维度猜想在实验中并没有得到验证。

5.4.2 lightGCN 层数猜想

表 5.3展示了只使用单 LightGCN 模型训练时，层数与效果的关系。可见 Yahoo 和 Coat 需要的 LightGCN 层数是不同的，Yahoo 数据集上最好的层数是 2 层,3 层相对尚可; 而 Coat 数据集上 LightGCN 的作用比较差，甚至不如只使用矩阵分解模型的结果。所以验证层数的猜想笔者主要放在 yahoo 数据机上做。表 5.4展示了固定偏好模型的层数为 2 层或 3 层时，层数与效果的关系。

观察发现，固定偏好层数为 2 的纠偏模型相比只使用一个 2 层 LightGCN 卷积的模型进行训练效果有明显提升，但曝光层数对训练结果的影响有一些奇怪。在 Yahoo 上，固定偏好模型使用 2 层 LightGCN 时，曝光模型的层数从 0 层开始先是逐渐递减，但是加到 6

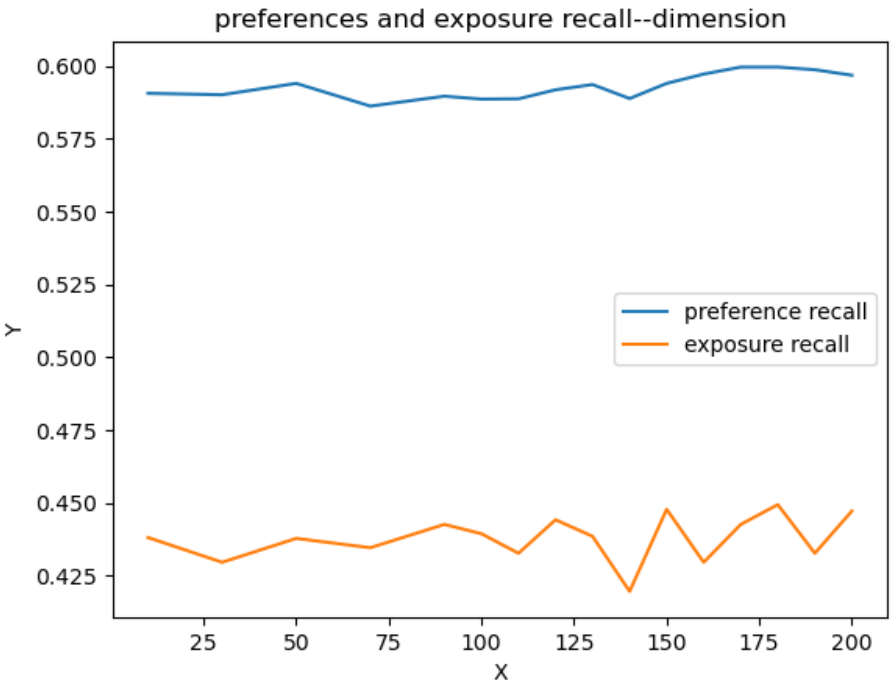


图 5.2 改变曝光模型维度时 Coat 数据集上的 recall 变化折线图。横坐标为曝光维度，纵坐标为指标值

层以上时，效果会突然上涨。笔者认为，这个现象可能与 LightGCN 模型本身存在的过度平滑问题有关；高层的 LightGCN 卷积层上的图接近全连接，从而使得模型结构发生某些质变所致。

这个现象的准确原因，以及在 coat 数据集上 LightGCN 没有表现出较好的效果的原因，可能与数据集本身有关，有待进一步研究；但至少可以说明，在固定偏好模型的层数时，改变曝光模型的层数可以对训练效果有一定的提升。

5.4.3 特征值猜想

根据猜想，偏好矩阵的长尾效应更强，所以我们在训练时让损失函数中偏好矩阵的特征值正则项系数为正，曝光矩阵的特征值正则项系数为负。

图 5.3, 图 5.4对比了加入损失函数中的特征值 topk 数 =6 时，偏好/曝光模型分别使用前两种函数，即直接加入和与后 6 个特征值作差后加入两种方式的情况下，最终训练得到的嵌入矩阵的特征值分布：偏好和曝光使用第一种特征值正则项的模型记为纠偏模型 A，使用第二种特征值正则项的模型记为纠偏模型 B。

表 5.3 使用单 LightGCN 模型进行训练时的性能对比表

LightGCN 层数	Yahoo-Recall@5	Yahoo-NDCG@5	Coat-Recall@5	Coat-NDCG@5
1	0.7781	0.6410	0.4614	0.3992
2	**0.8142**	**0.6783**	0.5563	**0.5214**
3	0.7858	0.6660	**0.5569**	0.4915
4	0.7579	0.6221	0.4487	0.3879

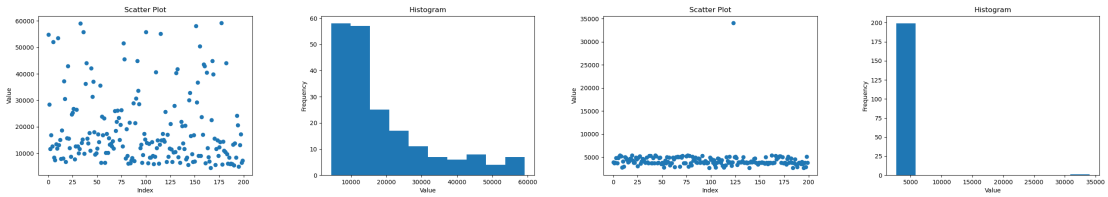


图 5.3 纠偏模型 A 特征值分布图，左二图为偏好模型，右二图为曝光模型

结论与维度猜想得到的结果比较类似。对于偏好矩阵，使用两种方式将特征值加入损失函数的区别不大；而对于曝光矩阵，由于曝光矩阵的学习效果本身有限，不与后 6 个特征值作差得到的矩阵只有一个较大的特征值，与后 6 个特征值做差得到的矩阵则有着一组 6 个较大、剩余较小且组内组外数值差别很小的特征值。而在训练开始时，取初始值的嵌入向量都是标准差接近 0.01 的正态分布，其相乘所得到的矩阵特征值本就是极其接近的，因此特征值差别不大意味着各维嵌入向量的模长小、训练效果有限，难以指望其较好的拟合实际的曝光。实验中，只将最大特征值引入训练，对训练效果没有任何提升，调整参数得到的所有结果均没有显著超过基线数据。只有将它们与随后 k 个较大特征值之差引入训练，才能获得前表中所示的提升。

事实上, 不将特征值引入训练时, 得到最佳效果时, 两个模型所呈现出的特征值分布也有类似的特点图 5.5, 即偏好模型的特征值分布颇为紧凑, 而曝光模型的特征值分布较为分散 (可通过观察纵坐标轴发现)。这也印证了曝光模型的学习效果存在着局限性.

不过，不应追求，也不能指望通过启发式地构造损失函数去“手动”拟合曝光矩阵的特征值。从基线数据相比使用单模型效果的提升来看，即使曝光矩阵的学习效果不佳，其对于偏好矩阵的训练是有帮助的，而使用最大 k 个特征值与紧随其后的几个特征值作差得

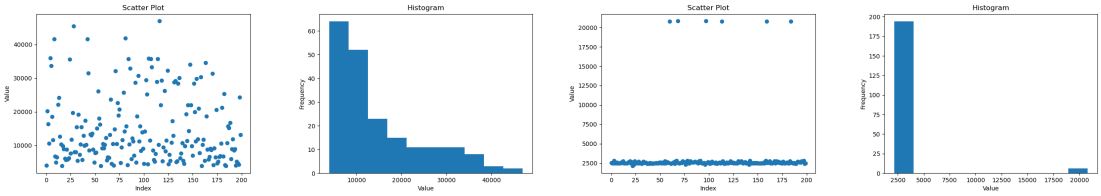


图 5.4 纠偏模型 B 特征值分布散点图，左二图为偏好模型，右二图为曝光模型

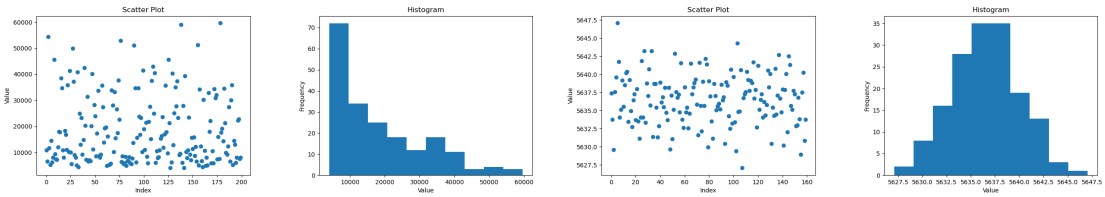


图 5.5 不引入特征值损失的纠偏模型特征值分布散点图，左二图为偏好模型，右二图为曝光模型。偏好模型的散点图纵轴从 0 到 60000，而曝光模型的散点图纵轴仅从 5627.5 到 5647.5.

到的特征值正则项，能相对地让曝光矩阵比较接近于集中在头部的实际情况，从而进一步提升训练效果。

图 5.6展示了改变正则项系数 λ_r, λ_o 与曝光模型所使用的 topk 特征值数，得到的训练结果对比。λ_r 和 λ_o 的值在 $[-10^{-4}, -10^{-8}] \cup [10^{-8}, 10^{-4}]$ 中调整。取用的 topk 特征数的值取了 [1,30] 范围内的整数。每一组均保持另一模型的 $\lambda = 0$，其他参数相同且调整到了最佳状态。

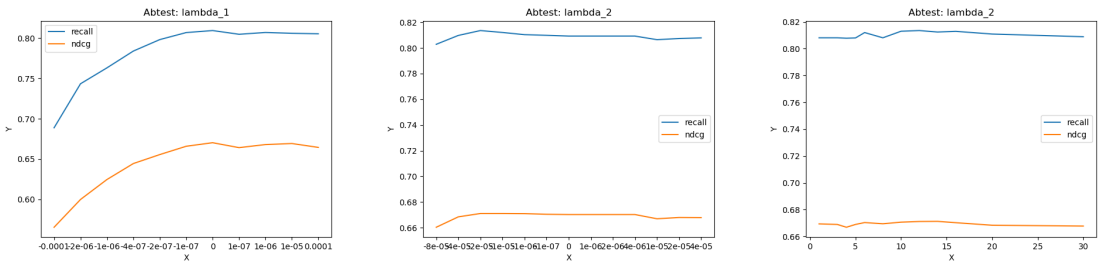


图 5.6 λ_r, λ_o 与 k 数的消融测试结果

可以看到，对于偏好而言，特征值正则项的作用不大，绝对值较大的 λ_r 对训练效果有消极影响，较小的则与基线相差无几。但对于曝光而言，特征值正则项能让效果有一些提升，并且参数不算很敏感，最佳取值的 topk 和 λ_o 均有较广的邻域可以取得相比基线有提升的结果. 由此可以得出结论: 在曝光模型的损失函数上施加特征值正则项, 有助于提升纠偏模型的训练效果。

回到本小节前提出的问题。根据实验，无论是否使用 LightGCN, 我们的纠偏模型相比

单独使用一个模型在性能上都有一定幅度的提升；但对曝光特点的猜想不完全符合预期。曝光模型并不像预想的那样和偏好模型对偶；虽然加入其会提升偏好模型对偏好学习的效果，但其本身对曝光学习的效果有限。如何让曝光模型学习到更充分的，更有助于反馈偏好模型的曝光信息，仍将是本课题未来的前进方向之一。

表 5.4 固定偏好模型 LightGCN 层数时的性能对比表

偏好层数	曝光层数	Yahoo-Recall@5	Yahoo-NDCG@5
2	不使用	0.8142	0.6783
2	0	0.8198	0.6811
2	1	0.8157	0.6813
2	2	0.8116	0.6660
2	3	0.8118	0.6720
2	4	0.8110	0.6410
2	5	0.8112	0.6813
2	6	**0.8232**	0.6833
2	7	0.8214	**0.6834**
3	不使用	0.7858	0.6605
3	0	0.7885	0.6552
3	1	0.7874	0.6540
3	2	0.7869	0.6543
3	3	0.7864	0.6533
3	4	0.7850	0.6507
3	5	0.7744	0.6440
3	6	**0.8008**	**0.6661**
3	7	0.7996	0.6656

6 总结与讨论

本文中，笔者讨论了曝光偏差以及现有模型在这一方面设计上存在的问题。笔者提出了使用两个预测值相乘得到预测结果的纠偏模型，其以矩阵分解模型和轻量化矩阵分解系统为原型，通过发掘偏好与曝光的不同性质修改模型，并进行实验验证猜想的正确与否。实验结果显示，尽管曝光模型本身的训练效果不及预期,它的加入对偏好模型的训练效果提升是显著的。曝光模型本身的训练效果虽然有一定局限性，但引入它相比只使用一个模型对训练效果有一定提升；通过让两个模型以不同的特征值正则项与不同的 LightGCN 层数进行训练，能进一步提升训练效果。

在提出猜想并加以验证的过程中，笔者还发现了许多未来得及完全研究透，有待进一步探索的细节。首当其冲的就是曝光模型难以训练到内容的问题，纠偏模型的基线测试中，几乎没有相异之处的两个模型在训练过程中训练效果却差异悬殊，同时训练效果貌似很有限的曝光模型却仍然能助力偏好模型达到更优的训练效果,这是笔者的工作目前未能解释之处。此外，对于引入特征值正则项所带来的比较稳定但小幅的提升一点，笔者认为，一方面如果进一步添加约束，让曝光模型的训练结果在此层面更贴合真实数据，将会进一步提升训练效果；另一方面想要设计出自然、高泛化性的方法还是比较困难的：笔者在尝试特征值正则项构造的过程中，曾经试过类似”前 $2k$ 大特征值减前 $2k+1$ 大特征值“这样的启发式正则项，发现这样构造最后的结果也往往只会让第一个特征值相比其它的明显大；最后的实验中，LightGCN 层数和特征值系数的使用也仍与数据集本身有较强相关性，在 Yahoo 上有提升的 LightGCN 层数设置并没有在 Coat 上收获效果，有待进一步探索。

其次，现有的模型对图的性质发掘只限于卷积神经网络的嵌入部分；如果改为对偏好模型和曝光模型进行卷积的图进行修改，并根据偏好模型和曝光模型可能具有的特性对两张图构造不同的加减边策略，还有可能收获更有效的纠偏模型。

除此之外,对于模型结构本身的发掘也还有许多能更深入之处。21 年以来的 UltraGCN 等工作[35]提出了 LightGCN 模型的一些问题,比如传播过程递归地将不同类型的关系对（包括用户-物品对、物品对及用户对）组合到模型中，但未能捕捉到不同的重要性、存在限制了 LightGCN 使用过多的消息传递层的过度平滑问题等。较多层数的 LightGCN 卷积出现的过度平滑问题,可能与使用不同层数的 LightGCN 时,训练效果随着曝光层数的增加出

现先减后骤增有关,曝光模型的过度平滑反而让偏好模型提供帮助.如果能利用过度平滑问题,在模型中添加一些模拟过度平滑 (比如在损失函数中引入拉普拉斯算子) 的情况,对类似的工作进行分析，可能可以组合出更能拟合出实际的曝光模型。

7 参考文献

[1] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].

[2] WU L, HE X, WANG X, et al. A Survey on Accuracy-oriented Neural Recommendation: From Collaborative Filtering to Information-rich Recommendation[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2022: 1-1. http://dx.doi.org/10.1109/TKDE.2022.3145690. DOI: 10.1109/tkde.2022.3145690.

[3] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].

[4] CHEN J, DONG H, WANG X, et al. Bias and Debias in Recommender System: A Survey and Future Directions[Z]. 2021. arXiv: 2010.03240 [cs.IR].

[5] HU Y, KOREN Y, VOLINSKY C. Collaborative Filtering for Implicit Feedback Datasets[C]//2008 Eighth IEEE International Conference on Data Mining. 2008: 263-272. DOI: 10.1109/ICDM.2008.22.

[6] LIANG D, CHARLIN L, MCINERNEY J, et al. Modeling User Exposure in Recommendation[Z]. 2016. arXiv: 1510.07025 [stat.ML].

[7] COVINGTON P, ADAMS J, et AL. E S. Deep Neural Networks for YouTube Recommendations[EB]. 2016.

[8] HE X, LIAO L, ZHANG H, et al. Neural Collaborative Filtering[Z]. 2017. arXiv: 1708.05031 [cs.IR].

[9] TAY Y, ANH TUAN L, HUI S C. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking[C/OL]//WWW ’18: Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ’18. ACM Press, 2018. http://dx.doi.org/10.1145/3178876.3186154. DOI: 10.1145/3178876.3186154.

[10] ZHU Z, HE Y, ZHANG Y, et al. Unbiased Implicit Recommendation and Propensity Estimation via Combinational Joint Learning[C/OL]//RecSys ’20: Proceedings of the 14th ACM Conference on Recommender Systems. Virtual Event, Brazil: Association for Computing Machinery, 2020: 551-556. https://doi.org/10.1145/3383313.3412210. DOI: 10.1145/3383313.3412210.

[11] LEE J W, PARK S, LEE J, et al. Bilateral Self-unbiased Learning from Biased Implicit Feedback[Z]. 2022. arXiv: 2207.12660 [cs.IR].

[12] HSIEH C K, YANG L, CUI Y, et al. Collaborative Metric Learning[C/OL]//WWW ’17: Proceedings of the 26th International Conference on World Wide Web. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017: 193-201. https://doi.org/10.1145/3038912.3052639. DOI: 10.1145/3038912.3052639.

[13] JOHNSON C C. Logistic matrix factorization for implicit feedback data: 78[EB]. 2014.

[14] Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition[EB]. 1989.

[15] KANG W C, MCAULEY J. Self-Attentive Sequential Recommendation[Z]. 2018. arXiv: 1808.09781 [cs.IR].

[16] GUO J, ZHANG P, LI C, et al. Evolutionary Preference Learning via Graph Nested GRU ODE for Session-based Recommendation[C/OL]//CIKM ’22: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. ACM, 2022. http://dx.doi.org/10.1145/3511808.3557314. DOI: 10.1145/3511808.3557314.

[17] CHEN J, DONG H, WANG X, et al. Bias and Debias in Recommender System: A Survey and Future Directions[Z]. 2021. arXiv: 2010.03240 [cs.IR].

[18] GAO C, WANG S, LI S, et al. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System[Z]. 2023. arXiv: 2204.01266 [cs.IR].

[19] HERNÁNDEZ-LOBATO J M, HOULSBY N, GHAHRAMANI Z. Probabilistic matrix factorization with non-random missing data[C]//ICML’14: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. Beijing, China: JMLR.org, 2014: II–1512–II-1520.

[20] QIU R, WANG S, CHEN Z, et al. CausalRec: Causal Inference for Visual Debiasing in Visually-Aware Recommendation[C/OL]//MM ’21: Proceedings of the 29th ACM International Conference on Multimedia. ACM, 2021. http://dx.doi.org/10.1145/3474085.3475266. DOI: 10.1145/3474085.3475266.

[21] COLLINS A, TKACZYK D, AIZAWA A, et al. A Study of Position Bias in Digital Library Recommender Systems[Z]. 2018. arXiv: 1802.06565 [cs.DL].
[22] O'BRIEN M, KEANE M T. Modeling result-list searching in the World Wide Web: The role of relevance topologies and trust bias[EB]. 2006.
[23] LIU Y, CAO X, YU Y. Are You Influenced by Others When Rating? Improve Rating Prediction by Conformity Modeling[C/OL]//RecSys '16: Proceedings of the 10th ACM Conference on Recommender Systems. Boston, Massachusetts, USA: Association for Computing Machinery, 2016: 269-272. https://doi.org/10.1145/2959100.2959141. DOI: 10.1145/2959100.2959141.
[24] WANG T, WANG D. Why Amazon's ratings might mislead you: The story of herding effects: 78 [EB/OL]. 2014. https://doi.org/10.1089/big.2014.0063.
[25] QIN Z, CHEN S J, METZLER D, et al. Attribute-based Propensity for Unbiased Learning in Recommender Systems: Algorithm and Case Studies[C/OL]//KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Virtual Event, CA, USA: Association for Computing Machinery, 2020: 2359-2367. https://doi.org/10.1145/3394486.3403285. DOI: 10.1145/3394486.3403285.
[26] SAITO Y. Unbiased pairwise learning from biased implicit feedback[EB]. 2020.
[27] WANG T, WANG D. Why Amazon's ratings might mislead you: The story of herding effects: 78[EB]. 2014.
[28] SAITO Y, YAGINUMA S, NISHINO Y, et al. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback[Z]. 2020. arXiv: 1909.03601 [stat.ML].
[29] SAITO Y. Unbiased Pairwise Learning from Biased Implicit Feedback[C/OL]//ICTIR '20: Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval. Virtual Event, Norway: Association for Computing Machinery, 2020: 5-12. https://doi.org/10.1145/3409256.3409812. DOI: 10.1145/3409256.3409812.
[30] WANG X, HE X, WANG M, et al. Neural Graph Collaborative Filtering[C/OL]//SIGIR '19: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2019. http://dx.doi.org/10.1145/3331184.3331267. DOI: 10.1145/3331184.3331267.
[31] KOREN Y, BELL R, VOLINSKY C. Matrix Factorization Techniques for Recommender Systems[J]. Computer, 2009, 42(8): 30-37. DOI: 10.1109/MC.2009.263.
[32] HE X, DENG K, WANG X, et al. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation[Z]. 2020. arXiv: 2002.02126 [cs.IR].
[33] CHANG S, HARPER F M, TERVEEN L. Using Groups of Items for Preference Elicitation in Recommender Systems[EB]. 2015.
[34] GAO C, LI S, ZHANG Y, et al. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos[C/OL]//CIKM '22: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. ACM, 2022. http://dx.doi.org/10.1145/3511808.3557624. DOI: 10.1145/3511808.3557624.
[35] MAO K, ZHU J, XIAO X, et al. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation[Z]. 2023. arXiv: 2110.15114 [cs.IR].

附录

作者简历

王海纳

手机: (86) 189-5711-3728 | Email: ewallnut@zju.edu.cn

教育背景

浙江大学 **2020年9月至2024年6月**

计算机科学与技术专业，工程学士。将于2024年6月毕业;

GPA: 3.90/4.00 (88.7/100)

研究经历

基于图推荐算法的偏差校正研究 **2023年4月至今**

组长，王灿教授指导，浙江大学ZLST实验室

- 通过研究不同暴露和偏好指数下的长尾效应，优化了现有图推荐用户项训练模型（如矩阵分解（MF）和LightGCN）中使用的损失函数;
- 在分析了用户偏好与物品暴露的关系后，根据暴露矩阵秩较低和长尾效应不太明显的结论，设计了专门的损失函数;
- 建立矩阵分解和LightGCN模型，并调整超参数和优化器以提高模型性能

SIGGRAPH 2016论文复现项目 **2023年4月至2023年6月**

课堂小组组长，周昆教授、任重教授指导

- 使用PyTorch复现了最初用Lua编写的粗略草图清理项目
- 构建了两个可比较的数据集来反映人工噪声和有限噪声的重要性，其中一个来自原始手绘图像，另一个来自直接二值化的彩色图像
- 生成了用于目标模型的训练代码和用于恢复行草稿的其他测试代码

基于机器学习的动画帧插入算法研究 **2022年3月至2023年5月**

组长，王高昂教授指导，浙江大学视觉智能与模式分析实验室；浙江大学学生研究训练计划

- 基于人工智能技术，开发了一个自动程序，能够为两个给定动画手稿的前帧和后帧填充中间帧，以加快动画制作
- 为前端工程设计了一种算法，可以用Python自动绘制具有2000多行代码的中间动画帧
- 针对手绘动漫色彩信息难以区分的问题，建立特征提取模型提取轮廓线
- 提高了计算机图形学的应用能力，特别是在样条曲线和拟合以及机器学习算法（如无监督/有监督聚类）方面

实习经历

华泰证券股份有限公司，远程 **2022年7月至2022年9月**

阿尔法量化策略组，金融工程实习生

- 研究了金融概念和术语，如夏普比率、年化收益率和投资因素
- 分析研究报告，包括基于阻力支撑相对强度的市场时机策略和风险评估模型
- 应用Python再现了市场时机策略报告中关于收入和指标统计的29张图和9张图，以及关于年度资产组合策略收益率的7张图和4张图

追 光动画有限公司，北京 **2022年7月至2022年8月**

技术开发实习生，技术部

- 分析了涉及高级图形的渲染绑定代码，如毛囊节点上的操作和反向运动学
- 参与内部云平台开发研究，改善内部文件格式对齐等合作痛点
- 支持重置场地、操作软件、引导姿势等动画动作捕捉工作

专业课程实验（代码超2000行以上）

机器学习课程 **2023年3月至2023年6月**

- 完成的培训项目包括使用蒙特卡洛树搜索和评估函数的人工智能象棋、垃圾邮件短信识别器和使用深度学习Q学习算法走出迷宫的机器人导航器

编译器原理课程 **2023年3月至2023年5月**

- 编写了一个编译器，可以编程一些简单代码，包括快速排序、矩阵乘法和模拟课程选择系统

人工智能课程 **2022年9月至2022年11月**

- 应用朴素的贝叶斯模型并用Python编码，为给定的疾病描述输出三种最有可能的疾病类型，实现了84%的准确性

迷你SQL数据库设计课程 **2022年4月至2023年6月**

- 实现了一个支持三种基本数据类型的单用户迷你SQL引擎：integer、char（n）和float，以允许简单的CRUD操作，并能够通过索引优化性能
- 实现时钟替换器、浮点精度和内存池以及奇偶校验功能

面向对象设计课程 **2021年9月至2022年1月**

- 使用OpenGL框架开发了一个小型MeshLab程序，包含2000多行代码，支持旋转、放大和缩小、更改表面颜色、表面细分、撤消和保存OBJ格式的文件。

荣誉与奖项

竞赛（ACM/ICPC/CCPC）

2021届中国大学生编程大赛金奖	2022年11月，威海
2021届国际大学生编程大赛东亚赛区金奖	2022年4月，昆明
2021届国际大学生编程大赛东亚赛区金奖	2021年11月，济南
2021届国际大学生编程大赛东亚赛区金奖	2021年4月，昆明
2020届国际大学生编程大赛东亚赛区金奖	2021年5月，银川

奖学金

浙江大学二等学术奖学金，优秀学生（8%）	2023年10月
浙江大学三等学术奖学金（20%）	2022年10月
浙江大学二等学术奖学金（8%）	2021年10月
浙江大学教育基金会南都奖学金二等奖（7/735）	2021年10月

专业技能

编程语言：C/C++（5年），Python（3 年）
应用软件：STATA（1年），EViews（1 年）
开发工具：PyTorch，Vivado，Xilinx，Linux

本科生毕业论文（设计）任务书

一、题目：基于图模型的推荐纠偏方法探究

二、指导教师对毕业论文（设计）的进度安排及任务要求：

根据开题报告中提出的研究计划，学生要在四月中旬前完成论文阅读，并对各算法和数据集进行学习；四月中旬到五月初完成算法设计并在数据集上进行实验；五月份完成毕业论文写作，实现研究计划中的基于图模型的推荐纠偏方法。

起讫日期 2024 年 3 月 1 日　至　2024 年 5 月 20 日

指导教师（签名）_____ 职称 _____

三、系或研究所审核意见：

负责人（签名）_____

年　月　日

本科生毕业论文（设计）考核

一、指导教师对毕业论文（设计）的评语：

王海纳同学在其毕业论文《基于图模型的推荐纠偏方法探究》中，研究了一个推荐系统领域较为重要的问题，提出了一个合理的方法，综合性的实验也验证了提出的方法的有效性。此外，本文写作良好，条理清晰，具有良好的逻辑，达到了本科生毕业论文的水平。

指导教师（签名）_____

年　月　日

二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

成绩比例	文献综述（10%）	开题报告（15%）	外文翻译（5%）	毕业论文质量及答辩（70%）	总评成绩
分值					

负责人（签名）_____

年　月　日

第二部分

毕业论文开题报告

浙 江 大 學

本 科 生 毕 业 论 文

文献综述和开题报告

姓名与学号	王海纳 3200104934
指导教师	陈佳伟
年级与专业	2020级计算机科学与技术
所在学院	计算机科学与技术学院

一、题目：基于图模型的推荐纠偏方法探究

二、指导教师对文献综述、开题报告、外文翻译的具体要求：

个性化推荐旨在从用户的历史行为数据中挖掘出用户的潜在喜好并推荐其满意的物品，被认为是解决“信息过载”的最有效方法之一，已在多个领域得到了广泛地应用。然而当前，主流的推荐算法和研究进展主要集中在数据驱动的有监督学习领域，通过设计更为复杂的推荐模型以更好的拟合用户的历史行为数据，从而对用户的兴趣进行预测。然而在现实场景中，数据的收集是观测性的而非实验的，单纯的数据驱动的方法往往会带来严重的偏差问题。本研究重点研究推荐算法中存在的曝光偏差问题，设计基于图模型的推荐纠偏方法。围绕基于图模型的推荐纠偏方法探究任务，查阅 10-15 篇相关文献，并在此基础上完成 3000 字的文献综述。文献综述要求能够总结当前研究方向的研究现状以及研究问题，思考存在的问题与改进的方向。同时选择相关度较高的文献，完成 3000 字以上的文献翻译。开题报告要求 3500 字以上，阐明本研究课题背景，研究内容以及预期的技术路线、目标以及进度计划等.

指导教师（签名）_____

年　月　日

一、文献综述

基于矩阵分解模型的图卷积神经网络研究进展

1 绪论

推荐系统在当前信息时代具有重要的应用价值，能够为用户提供个性化的信息服务，提高信息检索和推荐的效率和准确性。

推荐问题可概括为，给定分别包含 N 个用户与其对 M 个物品中一些物品的正或负评分的训练集和测试集，设计模型并在训练集上进行训练，最终最大化其预测结果的前 K 名在测试集上的推荐指标，包括精度 (precision), 召回 (recall)，归一化折损累计增益（Normalized Discounted Cumulative Gain, NDCG）, 平均倒数排名 (Mean Reciprocal Rank, MRR)，通常也记作 precision@K, recall@K, NDCG@K, MRR@K 等。

图卷积神经网络 (Graph Convolutional Network, GCN)[1]能够有效地处理具有图结构的数据，如社交网络、用户-商品交互网络等。将 GCNs 应用于推荐系统中，可以充分挖掘用户之间的关系和商品之间的关联，提高推荐的精度和个性化程度。近年来，基于 GCNs 的推荐算法研究取得了许多进展，涵盖了多个方面的创新和改进。

首先，基于 GCN 的推荐算法引入了图结构信息，利用用户-商品交互网络构建图网络结构，通过学习节点之间的关系和特征，实现了更准确的用户行为预测和推荐结果。其次，通过引入注意力机制、多层次特征融合等技术，进一步提升了推荐算法的性能和效果。另外，结合传统的协同过滤算法和深度学习模型，也为推荐系统的发展带来了新的思路和方法。

然而，基于 GCNs 的推荐算法研究仍面临一些挑战和问题。例如，如何处理大规模图数据、如何平衡推荐结果的准确性和多样性等问题，仍需要进一步的探索和改进。随着深度学习技术的不断发展和推广，基于 GCNs 的推荐算法在未来将继续成为研究热点，并在实际应用中发挥重要作用。

基于矩阵分解模型（Matrix Factorization, MF）及在它基础上的一系列优化方法是领域

内流行度最高的方法。本文将对其发展进行综述和分析，探讨其中的关键技术、应用场景和挑战，为推荐系统领域的研究和实践提供参考和启发。

2 国内外研究现状

2.1 研究方向及进展

矩阵分解模型起源很早，Bryan Perozzi 等人论文[2]中提到的“深度游走 (Deep Walk)”概念与之颇为类似。这篇文章提出了一种学习网络中节点潜在表示的新方法，其在连续向量空间中对社会关系进行编码，极容易被统计模型利用。在当时，其代表了语言建模和无监督特征学习从单词序列 (words) 到图 (graph) 的进展，在一些实验中的表现能够在使用的训练数据少 60% 的情况下仍然优于所有的基线方法。而矩阵分解模型则是和其一同提出的，它对用户和物品分别使用了一组类似的潜在表示向量，又称嵌入向量 (Embedding Vector)，取两组向量的乘积作为对目标矩阵的拟合结果。也就是说，其将用户-物品交互矩阵分解为两个低维矩阵的乘积，分别表示用户和物品的隐向量，通过学习这些隐向量，可以预测用户对未知物品的评分。MF 模型的优点是简单、易于实现，但限制也很多，如无法处理稀疏数据、无法利用用户和物品的特征信息等。为了解决这些问题，研究者们提出了多种改进和扩展的方法，如加入正则化项、引入偏置项、结合内容信息等。这些方法在一定程度上提高了推荐的准确性和效果，但仍存在一些局限性。因此，如何进一步提升推荐算法的性能和效果，是当前推荐系统研究的重要课题之一。近年来，对这个模型的优化的主要方向可以概括为加权 (Weighted)、纠偏 (Debias) 等，其中轻量化图卷积神经网络 (Light Graph Convolutional Network) 是最具有代表性的工作之一。在这些方向主要的工作可汇总如下。

2.1.1 加权优化

加权优化即是在训练的各个环节引入加权。具体的策略包括对图的度进行加权 (Degree Weighted)[3]、对样本进行加权 (Sample Weighted)、对待训练向量本身进行加权 (Embedding Weighted)[4]等等。是最简单直观的优化方法。这种方法通过从数据集的本身入手，在发现了其一些特点后就可以通过引入权重来影响训练，简单易行。其中最简单直观的方法是对

度数加权的方法。由于一个数据集中，度数高的点相连的边往往意味着比较活跃的用户，其偏好相对不太活跃的用户可参考价值显然不同，所以基于此进行度数加权的合理性是很自然的，又被称为置信度加权 (Confidence Weighting)。这种策略提出很早，但直到近年来仍有基于这种策略在基准数据集上取得极好结果的例子[5] 而对样本和训练向量本身进行加权的策略也很多，典型例子是 Rong Pan 等人[6]提出的 ExpoMF，其利用了 EM 算法来进行模型的求解，在 M-step 所优化的对数损失是下图所示的 loss。其中，$\theta'_{u,i}$ 是给定真实曝光的后验概率。由于 $E[O_{u,i}|Y_{u,i}=1]=1$, 即用户只要点击就一定曝光了，这个期望值便可以看作是通过曝光后验概率反映的点击所代表的用户偏好信息的置信度。

$$Loss_{ExpoMF}(\hat{R})=\frac{1}{D}\sum_{(u,i)\in D}\theta_{u,i}[Y_{u,i}\delta^{(1)}_{u,i}+(1-Y_{u,i})\delta^{(0)}_{u,i}]$$

2.1.2 纠偏优化

纠偏优化的思想同样在于发掘数据来源的性质，尤其是用户与用户、物品与物品之间潜在而又有诸多特点的相似性进行推荐。通过模型从设计到训练的各个环节进行修改来对之加以优化。相关方法往往将用户对物品的点击数据分为显式和隐式两大类。相关方法往往将用户对物品的点击数据分为显式和隐式两大类，前者包括用户对物品的明确评分，但其完全收集往往困难，因为需要用户主动提供。相比之下，隐式数据，作为用户行为的自然记录（比如是否购买，是否点赞）等更容易被收集，但其中的噪音较大。这也就意味着，大部分基准数据集的结果往往都是不尽准确的。针对显示数据的纠偏需要根据数据及特性补全其中由于用户未主动提供未完全收集的部分，而针对隐式数据的纠偏则需要补全由于某些物品未被曝光给用户所产生的数据缺失，又称“曝光偏差”。基于协同过滤 (Collaborative Filtering, CF) 的模型是其中的典型代表[7]。与之相关的工作中，基于用户倾向 (User Propensity, UP) 的去偏方法[8]、基于对称共同学习倾向 (Symmetric Learning Propensity)[9]和基于来自偏好模型的去偏方法[10]都收获了不错的效果。神经性协同过滤方法 (Neural Graph Collaborative Filter, NGCF)[11]融合了这几种方法的优势，它的传播过程分为 message construction（消息构造）和 message aggregation（消息聚合）两部分，还引入了注意力 (attention) 机制，结构如下图所示。而 LightGCN 作为至今领域内最流行的方法之一，则对它进行了改良。

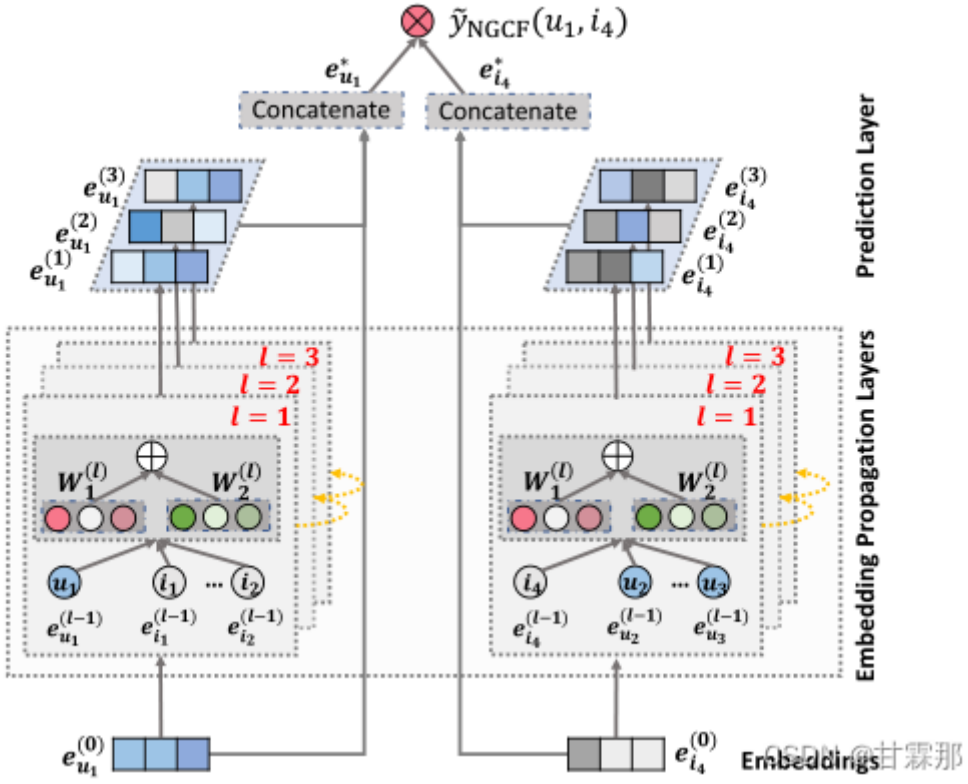


图 2.1 LGCN 模型结构示意图

2.1.3 LightGCN

轻量化卷积神经网络 (Light Graph Convolutional Network, LightGCN) 是对 GCN 基本思想的提炼。GCN 的基本思想是通过平滑图上的特征来学习节点的表示，为此它迭代执行图卷积，即聚合邻居的特征作为目标节点的新嵌入。

在堆叠多层神经网络时，NGCF 使用了注意力机制进行权重学习，但 LightGCN 则提出，事实上使用平均权重的效果就会很好，因此其弃用 GCN 中常见的特征转换、非线性激活函数和自信息链接的设计，并通过消融实验证明了自信息的多余。可以通过公式对比，说明 NGCF 到 LightGCN 的优化。NGCF 的矩阵形式表示如下：

$$E^{(l)} = \sigma(L + I)E^{(l-1)}W_1(l) + LE^{(l-1)} \cdot E^{(l-1)}W_2(l)L = D^{-0.5}AD^{-0.5}.$$

而 LightGCN 的矩阵形式表示如下，其中 α 被实验验证可以取为平均，因此相比上式少了

权重 W：

$$E^{(k+1)} = D^{-0.5}AD^{-0.5}E^{(k)}E = \alpha_0 E_0 + \alpha_1 E_1 + ... + \alpha_k E_k$$

其中 A 为

$$\begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$

2.2 存在问题

矩阵分解模型本身是领域内重要范式，对其性能的逐步优化在短期内不会停步。加权优化和纠偏优化策略本身依赖于策略设计，而启发式地设计策略本身便具有挑战性。即使是集了它们诸多长处的 lightGCN，也在实验中被证明有许多局限，比如其使用的递归传递的消息将不同类型的关系组合到建模中，而传递公式未能捕捉到它们的不同重要性；多层堆叠消息传递可以捕获高阶协作信号，但是 LightGCN 只是堆叠了 2,3 层后性能就由于过度平滑开始下降等。如何应对这些问题，依旧是一个潜在的挑战。

3 研究展望

针对现有工作的问题，我们可以发现，后续的工作开创空间仍然巨大。现有的工作指出了 lightGCN 的消息传递机制存在问题、建模 user-item 内在权重可能是反直觉的；因此跳过无限层的显示消息传递以实现高校推荐的做法是值得尝试的；另一方面，根据显式隐式数据的分析，我们可能可以并行训练偏好和曝光矩阵并以其乘积来数据集的结果，以跳出单个模型外，在更高的高度上进行纠偏。这也成为了笔者毕业设计时的一个努力方向。

4 参考文献

[1] L.Tang, H.Liu. Relational learning via latent social dimensions[EB]. 2009.
[2] Et AL. B P. DeepWalk: Online Learning of Social Representations[EB]. 2014.
[3] HU Y, KOREN Y, VOLINSKY C. Collaborative metric learning[EB]. 2008.
[4] LIANG D, CHARLIN L, MCINERNEY J, et al. Modeling user exposure in recommendation[EB]. 2016.
[5] Et AL. P X. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation[EB]. 2021.
[6] LIANG D, KRISHNAN R G, HOFFMAN M D, et al. Variational Autoencoders for Collaborative Filtering [EB]. 2018.
[7] GOLDBERG D, NICHOLS D, OKI B M, et al. Using Collaborative Filtering to Weave an[EB].
[8] SAITO Y, YAGINUMA S, NISHINO Y, et al. Unbiased recommender learning from missing-not-at-random implicit feedback[EB]. 2020.
[9] ZHU Z, HE Y, ZHANG Y, et al. Unbiased implicit recommendation and propensity estimation via combinational joint learning[EB]. 2020.
[10] AL. J W L E. Bilateral Self-Unbiased Learning from Biased Implicit Feedback[EB]. 2022.
[11] HE X, DENG K, WANG X, et al. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation[EB]. 2020.

二、开题报告

1 问题提出的背景

1.1 背景介绍

1.1.1 项目背景

推荐系统近年来受到学术界和工业界的广泛关注。然而，由于推荐系统中存在的各种偏差问题，当前的许多推荐模型难以在真实场景中直接应用。曝光偏差，即用户的行为数据收到物品的曝光情况所影响从而偏离用户的真实兴趣，是最典型的偏差之一。现有模型主要采用倾向得分、负样本加权等方式来解决偏差问题。然而，这类方法严重依赖于权重等参数的设置，同时也难以适用于 lightgcn 等先进的基于图的推荐模型。因此，如何结合图神经网络来解决曝光问题，是当前推荐领域一个值得研究的关键问题。本项目中，笔者将对推荐曝光偏差问题做全面的调研和分析，并设计基于图卷积神经网络的解决曝光偏差的方法。

在笔者前期的探索中，我们做出了“用户 u 点击物品 I，是用户 u 喜欢物品 I 和物品 I 曝光于用户 u 共同作用的结果”这一猜想，试使用 lightGCN 模型同时训练一个用户-物品偏好矩阵 R_{nm} 和一个用户曝光矩阵 O_{nm}，以两者乘积相对训练集结果的交叉熵作为损失函数，并以偏好矩阵的结果作为推荐结果进行测试；再在此基础上对用户偏好和物品曝光的不同性质做出猜想，并对应的修改损失函数的做法。我们尝试调整了两个模型使用不同层数的 lightGCN 、使用不同维度的 embed 以及将矩阵的特征值乘不同系数后加入损失函数等做法。此外，在单一模型的情境下还可以通过修改 lightGCN 所使用的卷积图进行纠偏。

1.1.2 课题介绍

推荐问题可概括为，给定分别包含 N 个用户与其对 M 个物品中一些物品的正或负评分的训练集和测试集，设计模型并在训练集上进行训练，最终最大化其预测结果的前 K 名在

测试集上的推荐指标，包括精度 (precision), 召回 (recall)，归一化折损累计增益（Normalized Discounted Cumulative Gain, NDCG）, 平均倒数排名 (Mean Reciprocal Rank, MRR)，通常也记作 precision@K, recall@K, NDCG@K, MRR@K 等。基准数据集又可分为无曝光偏差与有曝光偏差两大类。前者的训练集和测试集均属于没有外加曝光时获得的用户点击数据，典型例子包括 Gowalla, Amazon-Book, 等；后者的训练集则是用户在被明确提供了待推荐物品的详细介绍后给出的点击结果，意味着对于给定物品是明确被曝光的；而测试集则是一般的用户点击数据，典型例子包括 YahooR3, Coat 等。在两者上做实验时，指标的计算也有差别，对于有曝光偏差的数据集一般只关心测试集中物品的整体得分排名。以计算 recall@K 为例，如测试集中某个用户 u 有 K 个物品 i 的点击与否信息，其中 K’个为正，则我们只关心用户对于这 K 个物品的训练得分中排名前 K’个的是否为这些物品，用这其中排名前 K’个的物品中正样本的个数而非在所有物品中排名前 K’个物品中正样本的个数计算 recall. 为了在训练期间确定收敛时刻，在训练时还需将训练集划分为训练集和验证集。每次迭代 (epoch) 后让模型在验证集上测试，如结果在连续的 L 个 epoch 中都没有超过历史最大值，则停止训练，让模型在测试集上测试并得到最后结果。L 为超参数，一般需要使得模型能基本收敛。笔者的课题围绕着以上问题，在优化现有方法的基础上展开。

1.2 本研究的意义和目的

随着大数据和人工智能技术的飞速发展，推荐系统已成为现代生活中不可或缺的一部分，广泛应用于电商、社交、娱乐等多个领域。然而，推荐系统中存在的各种偏差问题，尤其是曝光偏差，严重影响了推荐结果的准确性和有效性。因此，本研究旨在通过深入探索曝光偏差问题，提出一种基于图卷积神经网络的解决方法，具有重要的理论意义和实践价值。在理论方面，本研究通过对曝光偏差问题的全面调研和分析，深化了对推荐系统中偏差问题的理解。同时，结合图神经网络的思想，本研究尝试设计新的推荐算法，以解决现有方法在处理曝光偏差问题时的不足。这不仅有助于推动推荐系统领域理论研究的深入发展，也为其他相关领域的研究提供了新的思路和方法。在实践方面，本研究提出的基于图卷积神经网络的解决曝光偏差的方法，旨在优化 lightGCN 作为推荐系统的基准模型进行推荐的效果。通过在基准数据集上进行实验验证，本研究期望能够显著提升推荐结果的准确性和有效性，为实际应用中的推荐系统提供更加优质、个性化的推荐服务。这不仅有助

于提升用户体验和满意度，也能够为企业带来更大的商业价值。综上所述，本研究的意义和目的在于通过深入探索曝光偏差问题，提出一种基于图卷积神经网络的解决方法，以优化推荐系统的性能，推动相关领域的理论研究和实践应用的发展。

2 项目的主要内容和技术路线

2.1 主要研究内容

本项目中，笔者主要研究对象为图卷积神经网络，研究在训练过程中修改用于卷积的图 G，以优化推荐效果的策略。设计策略的过程中，我研究的基本模型包括主流的矩阵分解模型（MF）和其主要变种（Light GCN）。在获取动机的过程中，我学习了所在实验室前任学姐的研究成果 ReCRec；而在研究在图卷积模型中加边来获取思路的过程中学习了 RGCN 模型。在学习这些图卷积神经网络对应的论文时，我还阅读了这些论文引用的论文，来了解对于 MF 和 lightGCN 做优化的前人工作，包括 WMF、ExpoMF,、RelMF,、CJMF、BISER、NGCF、DGCF、SGCN 等。矩阵分解模型 (MF) 是较为基础的模型。其核心思想是为每个用户 u 和物品 i 维护一个 dim 维的嵌入 (embed)，将目标偏好矩阵拟合为两个低维向量的嵌入乘积，以两者的乘积作为整体拟合得分。WMF, ExpoMF, Rel-MF, CJMF, BISER 都是 MF 的一些优化模型[1]。WMF 对未点击数据的贡献使用多种策略进行加权，包括归一化、基于流行度、基于用户活动等策略；ExpoMF 是利用基于暴露的生成模型的代表性去偏方法；Rel-MF 则是基于倾向 (propensity) 的代表性去偏方法，其用项目流行度指定倾向；CJMF 是最先进的基于用户倾向的去偏方法，主要利用对称共同学习倾向和用户偏好的学习策略；而 BISER 则是基于倾向的方法，规定了一系列来自偏好模型的预测。NGCF，DGCF 和 SGCN 是 LightGCN 的前置方法，即在结算拟合得分前使用不同的策略对 embed 做不同的卷积，从而对嵌入起到松弛 (smoothing) 的效果。而 lightGCN 则是集它们之长得到的方法。记 R 为用户-物品的交互矩阵（训练集），E0 为原始嵌入，则 lightGCN 所计算的嵌入 E 可经过如下公式计算：

1）改写计算 UI 的邻接矩阵：

$$\begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$

2）对 A 进行归一化，并结合迭代得到经过 i 层卷积的嵌入：

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

3）对多层卷积的结果进行加权求和，得到最终嵌入 E：

$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + \alpha_2 E^{(2)} + ... + \alpha_k E^{(k)}$

纠偏推荐算法的方法很多。我们可以尝试使用两个模型分别代表兴趣和曝光同步训练，还可以尝试对 lightGCN 使用的卷积图进行优化。相关的对图进行优化的论文并不多，有代表性的三篇是 OpenGSL[2], RGCN[3]和 LDA-GCL[4]。OpenGSL 为笔者导师的参与的论文，提出了一个向图中加边的模型框架；LDA-GCL 使用了 GNNs 通过优化在 GCL 中使用的对抗图增强策略来避免训练过程中获得冗余信息；而 RGCN 则是通过训练两张图，一张为 ui 嵌入乘积去除噪音后的图，另一张则为随机加了一些边的图，然后在 loss 中加入两张图训练得到的 embed 的互信息。通过试跑代码发现，这些论文均没有提出特别行之有效的加边策略，其中 RGCN 论文中的“随机加边图”换为不加边的图，效果并无明显差距。如果能在这些论文的基础之上设计出算法，定能有所创新。

2.2 技术路线

纠偏路线包括使用两个模型同时拟合曝光和偏好进行训练和对卷积图进行优化两条。它们都重在算法设计，技术路线并不复杂。

可分为如下几步：

1) 获取数据集，编写 lightGCN 训练源代码。这一步可以比较快完成。

2) 对模型构建部分构建 LightGCN 图的部分代码进行重写。

3) 在基准数据集上测试策略效果。

4) 选择有效果的策略。

对于使用两个模型同时拟合曝光和偏好的思路，策略设计重在提出曝光和偏好可能具备的不同性质。目前能想到的性质和对应的策略如下，开题后可能会继续探索：

1）曝光矩阵的秩会小于偏好矩阵的秩。对此，可以使用不同维度的 embed 训练。

2）曝光矩阵的特征值会大于偏好矩阵的特征值。对此，可以将特征值引入 loss.

3）曝光矩阵的平滑度更高，可以体现为 laplacian 算子大小的区别。对此，可以将

laplacian 算子引入 loss.

而重构图的策略，可以细分为选边、边权和更新三部分策略。选边策略围绕着边的选择，在 lightGCN 的卷积图中，我们可以加或减用户-物品边来以数据集为基础提高其作为卷积用图的质量，也可以提炼出用户与用户、物品与物品之间的相关性作为卷积用图的补充；对于选出的边，还需确定其以多大的权重加入图中进行卷积；还有在重复加边时，是否要重复使用之前 epoch 加的边，是以一定的权重留下，还是全部扔掉。

开题时笔者想到的策略包括只重构一张图（后与原图对比）和重构两张图的策略，列举如下。每个策略在设计时还需考虑他们的可解释性。如果效果均不佳，后期会继续尝试。重构一张图的策略如下：

a) 求图中 embed，对于用户 embed, 两两求用户 embed 的归一化内积，取每个用户与它人内积的 topk 插入图中。注意在一开始时，图中的这一部分是空的。类似的，还可以取每个物品与其他物品的内积 topk 插入图中。相对应的，还可以对每个物品求 topk. 构造对称策略。

b) 两两求 embed 的归一化内积后，对每个用户 u 构造一个包含与之 embed 内积最高的 K 个用户集合 Su（K 为超参数）；再对每个物品 i，求 Su 中所有用户对其的得分之和，再取 knn 加入图中。这样的构图放大了 u 对 i 偏好的间接效应（来源于其他与自己偏好类似的用户），起到纠偏的目的。

而重构两张图的策略如下：

c) 两张图分别取用不同层数的 lightGCN。如此构建策略试图减少不同层 lightGCN 的差别，和前面的 b）节的的思路类似。

2.3 可行性分析

本研究具备多方面的可行性。

技术可行性：近年来，图卷积神经网络在多个领域取得了显著进展，其在处理复杂关系数据方面的优势为推荐系统提供了新的思路。前文提到的，现有的矩阵分解模型（MF）和 LightGCN 模型及其变种，为推荐系统提供了坚实的基础。结合这些模型，我们可以在实现图卷积神经网络的优化时获取诸多灵感，从而改善推荐效果。数据可行性：随着大数据技术的发展，获取和处理用户与物品之间的交互数据变得越来越容易。这些数据为我们

提供了丰富的实验样本，使得我们可以在基准数据集上进行充分的实验验证。此外，现有的数据集既包括无曝光偏差的，也包括有曝光偏差的，这为研究提供了全面的测试环境。

资源可行性：本研究所在的实验室具备先进的计算设备和充足的存储资源，我可以在配备 8 张 2080 显卡的服务器上进行模型训练，为大规模的图卷积神经网络训练和测试提供了有力的支持。同时，实验室丰富的学术资源和合作网络也为本研究提供了宝贵的学术支持和合作机会。

实施可行性：研究团队具备扎实的图卷积神经网络和推荐系统领域的知识储备，以及丰富的实验经验。通过借鉴前任学者的研究成果和当前研究热点，我们可以设计出有效的实验方案，并在合理的时间内完成实验验证。对推荐系统进行纠偏的意义描述较为丰富，因为数据集有偏是公认事实，对 MF 模型的大部分优化都是围绕着分析数据集性质的。策略成功的关键主要在于实现方式上。如 2.1 节所分析的那样，对图进行优化的论文并不多。而考虑到数据集性质，普通推荐场景下用户的曝光程度是未确定的，所以训练集的本质是一张少了很多边的图。如果能够对其进行有质量的加边，不引入过多噪音，其训练效果理论上必然能有所提升。

3 参考文献

[1] MAO K, ZHU J, XIAO X, et al. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation[Z]. 2023. arXiv: 2110.15114 [cs.IR].
[2] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].
[3] WU L, HE X, WANG X, et al. A Survey on Accuracy-oriented Neural Recommendation: From Collaborative Filtering to Information-rich Recommendation[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2022: 1-1. http://dx.doi.org/10.1109/TKDE.2022.3145690. DOI: 10.1109/tkde.2022.3145690.
[4] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].

4 研究计划进度安排及预期目标

4.1 进度安排

3 月 25 日至 4 月 5 日：

深入研究图卷积神经网络和推荐系统的最新理论，为论文撰写提供坚实的理论基础。

开始初步实验。

4 月 6 日至 4 月 20 日：

进行系列验证实验，收集并分析数据，验证模型的有效性。

撰写论文主体部分，包括引言、相关工作、方法、实验等。

4 月 21 日至 5 月 30 日：

完成论文的撰写和初步修改，确保逻辑清晰、语言流畅。

准备论文所需的附加材料，如引用文献、数据集说明等。

6 月初：

完成毕业论文的终稿，仔细检查格式、排版等细节。

准备答辩所需的 PPT 和讲稿，进行预演。

6 月 3 日至 6 月 5 日：

参加毕业答辩，根据答辩委员会的意见进行必要的修改。

6 月中旬：

根据答辩反馈，对毕业论文进行最后的调整和完善。

提交毕业论文至学校或学院指定的系统。

4.2 预期目标

成功找到可行策略，完成毕业论文设计；争取能产出论文。

5 参考文献

[1] MAO K, ZHU J, XIAO X, et al. UltraGCN: Ultra Simplification of Graph Convolutional Networks for Recommendation[Z]. 2023. arXiv: 2110.15114 [cs.IR].

[2] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].

[3] WU L, HE X, WANG X, et al. A Survey on Accuracy-oriented Neural Recommendation: From Collaborative Filtering to Information-rich Recommendation[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2022: 1-1. http://dx.doi.org/10.1109/TKDE.2022.3145690. DOI: 10.1109/tkde.2022.3145690.

[4] GAO C, ZHENG Y, LI N, et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions[Z]. 2023. arXiv: 2109.12843 [cs.IR].

三、外文翻译

摘要

现有的将 GCN 应用于推荐系统的工作缺乏对 GCN 进行彻底消融分析的研究，GCN 最初是为图分类任务设计的，并配备了许多神经网络操作。然而，我们在实践中发现，在 GCN 中两种最常见的设计——特征转换和非线性激活——对协同过滤的性能贡献不大。更糟糕的是，将它们包含在内增加了训练的难度并降低了推荐的性能。在这项工作中，我们旨在简化 GCN 的设计，使其更加简洁且适用于推荐系统。我们提出了一个新模型，名为 LightGCN，仅包括 GCN 中最基本的组件——邻域聚合——用于协同过滤。具体而言，LightGCN 通过在线性传播用户和物品嵌入到用户-物品交互图上来学习用户和物品嵌入，并将在所有层学习到的嵌入的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，并且在完全相同的实验设置下，相对于现有技术的基于 GCN 的推荐模型 Neural Graph Collaborative Filtering (NGCF)，表现出了显著的改进（平均相对改进约 16.0%）。

我们还从分析和实证两个角度对这种简单 LightGCN 的合理性进行了进一步分析。我们的实现在 TensorFlow 和 PyTorch 中都可以找到。

1 介绍

为了缓解网络上的信息过载，推荐系统被广泛部署以执行个性化信息过滤 [7, 45, 46]。推荐系统的核心是预测用户是否会与物品进行交互，例如点击、评分、购买等形式的交互。因此，协同过滤 (CF) 侧重于利用过去的用户-物品交互来实现预测，仍然是实现有效个性化推荐的基本任务 [10, 19, 28, 39]。CF 的最常见范式是学习潜在特征（也称为嵌入），以表示用户和物品，并基于嵌入向量进行预测 [6, 19]。矩阵分解是早期的一种模型，它直接将用户的单个 ID 投影到她的嵌入中 [26]。后来，一些研究发现，将用户 ID 与她的交互历史作为输入进行增强，可以改善嵌入的质量。例如，SVD++ [25] 表明了使用用户交互历史在预测用户数字评分方面的好处，而神经注意力项相似性 (NAIS) [18] 区分了交互历史中物品的重要性，并在预测物品排名方面展示了改进。从用户-物品交互图的角度来看，这些改进可以视为利用用户的子图结构—更具体地说，她的一跳邻居—来改善嵌入学习。

2 引理

我们首先介绍了 NGCF[39], 这是一个具有代表性并且是推荐领域最新技术的 GCN 模型。然后我们对 NGCF 进行了消融研究，以评估 NGCF 中每个操作的实用性。本节的新颖贡献在于展示了在 GCNs 中两种常见设计，即特征转换和非线性激活，在协同过滤中没有积极的影响。

表 2.1 NGCF 的测试数据和其中的三个主要变量

	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

2.1 NGCF 简介

在初始阶段，每个用户和物品都与一个 ID 相关联，表示物品 i 的 ID 嵌入。然后，NGCF 利用用户-物品交互图来传播嵌入，具体如下：

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

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

在经过 k 层传播后，用户 u 和物品 i 的嵌入，a 是非线性激活函数，N_u 表示用户 u 交互的物品集合，N_i 表示与物品 i 交互的用户集合，W1 和 W2 是每一层进行特征转换的可训练权重矩阵。通过传播 L 层，NGCF 获得了 L + 1 个嵌入来描述一个用户（e_u^1 到 e_u^L）和一个物品（e_i^1 到 e_i^L）。然后，它连接这些 L + 1 个嵌入以获得最终的用户嵌入和物品嵌入，使用内积生成预测分数。NGCF 在很大程度上遵循标准的 GCN [23]，包括使用非线性激活函数 σ(-) 和特征转换矩阵 W1 和 W2。然而，我们认为这两种操作在协同过滤中并不那么有用。

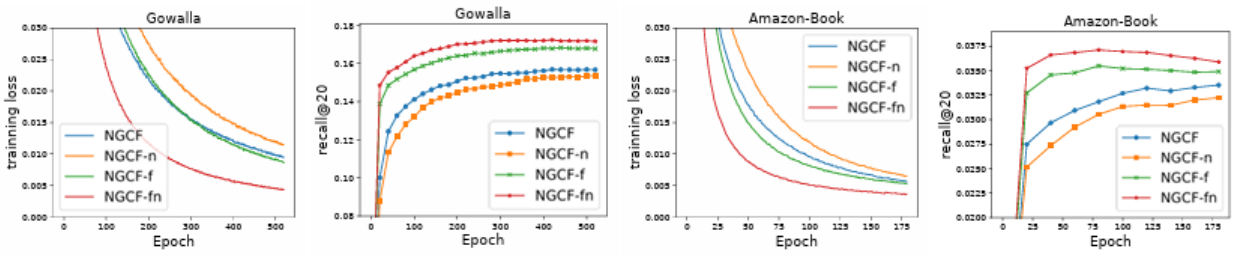
在半监督节点分类中，每个节点都有丰富的语义特征作为输入，例如论文的标题和摘要词语。因此，进行多层非线性转换有助于特征学习。然而，在协同过滤中，用户-物品交互图的每个节点只有一个 ID 作为输入，而这个 ID 没有具体的语义。在这种情况下，进行多次非线性转换不会有助于学习更好的特征；甚至更糟糕的是，它可能增加了训练的难度。在下一小节中，我们将提供这个论点的实证证据。

2.2 NGCF 的经验探索

我们对 NGCF 进行消融研究 (ablation studies)，探讨非线性激活和特征转换的影响。我们使用了 NGCF 的作者发布的代码进行实验，在相同的数据拆分和评估协议下运行，以尽可能保持比较的公平性。由于 GCN 的核心是通过传播来优化嵌入，我们更关注在相同嵌入大小下的嵌入质量。因此，我们改变了获取最终嵌入的方式，从连接（即 $e_u^* = e_u^{(0)}||...||e_u^{(L)}$）改为求和（即 $e_u^* = e_u^{(0)} + ... + e_u^{(L)}$）。请注意，这种改变对 NGCF 的性能影响不大，但使下面的消融研究更具有 GCN 优化的嵌入质量的指示性。

我们实现了三个简化的 NGCF 变体：

NGCF-f，移除了特征转换矩阵 W_1 和 W_2。NGCF-n，移除了非线性激活函数 σ。NGCF-fn，同时移除了特征转换矩阵和非线性激活函数。对于这三个变体，我们保持所有超参数（例如学习率、正则化系数、dropout 比率等）与 NGCF 的最佳设置相同。我们在 Gowalla 和 Amazon-Book 数据集上报告了 2 层设置的结果，见表 1。可以看到，移除特征转换（即 NGCF-f）在所有三个数据集上均导致对 NGCF 的一致改进。相比之下，移除非线性激活并不会对准确性产生太大影响。然而，如果我们在移除特征转换的基础上移除非线性激活（即 NGCF-fn），性能将显著提升。基于这些观察结果，我们得出以下结论：添加特征转换



(a) Training loss on Gowalla (b) Testing recall on Gowalla (c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

图 2.1 NGCF 的训练曲线

对 NGCF 产生了负面影响，因为在 NGCF 和 NGCF-n 模型中移除它显著提高了性能；添加非线性激活在包含特征转换时会稍微影响，但在禁用特征转换时会产生负面影响。总的来说，特征转换和非线性激活对 NGCF 产生了相当负面的影响，因为通过同时移除它们，NGCF-fn 相对于 NGCF 有很大的改进（召回率相对改进了 9.57 为了更深入地了解表 1 中获得的分数，并理解 NGCF 在这两个操作下为什么恶化，我们在图 1 中绘制了记录的训练损失和测试召回率的模型状态曲线。可以看到，NGCF-fn 在整个训练过程中的训练损失要比 NGCF、NGCF-f 和 NGCF-n 低得多。与测试召回率曲线一致，我们发现这种较低的训练损失成功转化为更好的推荐准确性。NGCF 和 NGCF-f 之间的比较显示了类似的趋势，只是改进幅度较小。

根据这些证据，我们得出结论：NGCF 的恶化源于训练的困难，而不是过拟合。从理论上讲，NGCF 的表征能力比 NGCF-f 更高，因为将权重矩阵 W1 和 W2 设为单位矩阵 I 可以完全恢复 NGCF-f 模型。然而，在实践中，NGCF 表现出更高的训练损失和更差的泛化性能。非线性激活的加入进一步加剧了表征能力与泛化性能之间的差异。总之，在为推荐设计模型时，进行严格的消融研究以明确每个操作的影响非常重要。否则，包含不太有用的操作将不必要地复杂化模型，增加训练难度，甚至降低模型的有效性。

3 方法

前一节展示了 NGCF 是一个繁重且繁琐的 GCN 模型，用于协同过滤。在这些发现的驱动下，我们设定了开发一个轻量且有效的模型的目标，通过包含 GCN 用于推荐的最基本的要素。简单模型的优点有很多—更易于解释，实际上更易于训练和维护，技术上更易于分析模型行为并将其调整为更有效的方向等等。

在这一部分中，我们首先介绍我们设计的轻量级图卷积网络（LightGCN）模型，如图 2 所示。然后，我们提供对 LightGCN 进行深入分析，展示其简单设计背后的合理性。最后，我们描述如何为推荐进行模型训练。

3.1 LightGCN

GCN 的基本思想是通过图上的特征平滑学习节点的表示 [23, 40]。为了实现这一点，它通过图卷积的方式进行迭代，即将邻居的特征聚合为目标节点的新表示。这样的邻域聚合可以抽象为：$e_u^{(k+1)} = AGG(e_u^{(k)}, \{e_i^{(k)} : i \in N_u\})$

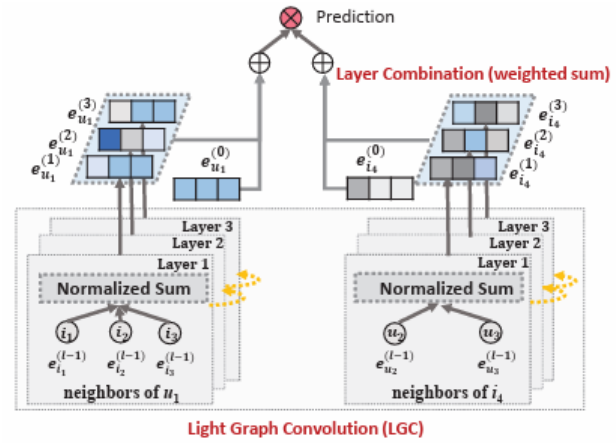


Figure 2: An illustration of LightGCN model architecture. In LGC, only the normalized sum of neighbor embeddings is performed towards next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations.

图 3.1 LGCN 模型结构示意图

AGG 是一个聚合函数——图卷积的核心——它考虑了目标节点和其邻居节点的第 k 层表示。许多工作已经指定了 AGG，例如 GIN 中的加权求和聚合器 [42]，BGNN 中的双线性交互聚合器等。然而，大多数工作都将特征转换或非线性激活与 AGG 函数绑定在一起。尽管它们在具有语义输入特征的节点或图分类任务中表现良好，但对于协同过滤可能会增加负担（见第 2.2 节的初步结果）。

轻量级图卷积（LGC）。

在 LightGCN 中，我们采用简单的加权求和聚合器，并放弃使用特征转换和非线性激活。LightGCN 中的图卷积操作（又名传播规则 [39]）定义为：

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

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

对称归一化项遵循标准 GCN[23] 的设计，它可以避免嵌入随着图卷积操作的增加而增加；也可以在这里应用其他选择，如 L_1 范数，但在实践中，我们发现这种对称归一化表现良好（见第 4.4.2 节的实验结果）。

值得注意的是，在 LGC 中，我们仅聚合连接的邻居，不包括目标节点自身（即自连接）。这与大多数现有的图卷积操作 [14, 23, 36, 39, 48] 不同，后者通常聚合扩展邻居并需要特殊处理自连接。下一小节中将介绍的层组合操作本质上捕获了与自连接相同的效果。因此，在 LGC 中不需要包含自连接。

3.1.1 层组合和模型预测

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 和所有物品的 e_i^0。给定它们后，可以通过 Equation(3) 中定义的 LGC 计算较高层次的嵌入。经过 K 层 LGC 后，我们进一步组合每层获得的嵌入，形成用户（物品）的最终表示：

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

其中 αk 表示第 k 层嵌入在构成最终嵌入中的重要性。它可以被视为手动调整的超参数，或者作为模型参数（例如，注意力网络 [3] 的输出）进行自动优化。在我们的实验中，我们发现将其均匀设置为 1/(K+1) 通常会带来良好的性能。因此，我们不设计专门的组件来优化，以避免不必要地增加 LightGCN 的复杂性并保持其简单性。

我们将层组合进行模型预测定义为：

$$\hat{y}_{ui} = e_u^T e_i$$

它用作生成推荐的排名分数。

3.1.2 矩阵形式

。我们提供 LightGCN 的矩阵形式，以便与现有模型进行实现和讨论。将用户-物品交互矩阵定义为 R∈RjVfXjV，其中 M 和 N 分别表示用户和物品的数量，每个条目 RUi 为 1

表示 u 已与物品 i 进行了互动，否则为 0。然后我们获得用户-物品图的邻接矩阵：

$$\begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix}$$

令第 0 层嵌入为 $E^{(0)} \in R^{(M+N)\times T}$. 我们可以获得与矩阵等价的 LGC 表示：

$$E^{(k+1)} = (D^{-0.5}AD^{-0.5})E^{(k)}$$

其中 D 是 (M+N) (M+N) 的对角矩阵，D_ii 表示第 i 行的非 0 向量。最后我们会获得一个最终嵌入矩阵：

$$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + ... + \alpha_K E^{(k)}$$

3.2 模型分析

我们进行模型分析，以展示 LightGCN 简单设计背后的合理性。首先，我们讨论与简化 GCN（SGCN）[40] 的关系，后者是一种最近的线性 GCN 模型，将自连接整合到图卷积中；这种分析表明，通过进行层组合，LightGCN 包含了自连接的效果，因此 LightGCN 无需在邻接矩阵中添加自连接。然后，我们讨论与近似个性化传播神经预测（APPNP）[24] 的关系，后者是最近的 GCN 变体，从个性化 PageRank [15] 中得到启发以解决过度平滑问题；这种分析显示了 LightGCN 与 APPNP 之间的潜在等价性，因此我们的 LightGCN 在传播长距离和可控过度平滑方面具有相同的优势。最后，我们分析了第二层 LGC，展示了它如何平滑用户与她的二阶邻居，从而更深入地了解了 LightGCN 的工作机制。

3.2.1 与 SGCN 的关系

。在 [40] 中，作者认为 GCN 在节点分类中的复杂性是不必要的，并提出了 SGCN，通过移除非线性并将权重矩阵合并为一个权重矩阵来简化 GCN。SGCN 中的图卷积定义为：

$$E^{(k+1)} = (D+I)^{-0.5}(A+I)(D+I)^{-0.5}E^{(k)}$$

其中 I 为单位矩阵。以下分析中我们为了简洁省略 $(D+I)^{-0.5}$ 这一项。因为只是在放缩嵌入. 在 SGCN 中，最后一层的嵌入可以被如下表示：

$$E^{(K)} = (A+I)E^{(K-1)} = (A+I)^K E^{(0)} = \sum_{k=0}^{K} C_K^k A^k E^{(0)}$$

上述推导表明，在 A 中插入自连接并在其上传播嵌入，本质上等同于在每个 LGC 层传播的嵌入的加权和。

3.2.2 与 APPNP 的关系

在最近的一项工作中 [24]，作者将 GCN 与个性化 PageRank [15] 联系起来，从中得到启发，提出了一种名为 APPNP 的 GCN 变体，可以在不过度平滑的风险下传播长距离。受到个性化 PageRank 中跳转设计的启发，APPNP 通过每个传播层与起始特征（即第 0 层嵌入）进行补充，这可以平衡保留局部性的需求（即保持接近根节点以减轻过度平滑）和利用大邻域信息的需求。APPNP 中的传播层定义为：

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

其中 β 是控制在传播中保留起始特征的跳转概率，$\hat{A}$ 表示归一化的邻接矩阵。在 APPNP 中，最后一层用于最终预测，即

$$E^{(K)} = \beta E(0)+(1-\beta)\hat{A}E^{(k-1)} = \beta E(0)+(1-\beta)\hat{A}E^{(0)}+(1-\beta)^2\hat{A}^2E^{(K-2)} = \beta E(0)+(1-\beta)\hat{A}E^{(0)}+...+(1-$$

我们可以看到，如果另一个用户 v 与目标用户 u 有共同的交互，那么 y 对 u 的平滑强度由系数决定（否则为 0）：

$$c_{v->u} = \frac{1}{\sqrt{|N_u|}\sqrt{|N_v|}} \sum_{i \in N_u \cap N_v} \frac{1}{|N_i|}$$

这个系数是相当可解释的：第二阶邻居 v 对 u 的影响由以下因素决定：1）共同交互的项目数量，数量越多影响越大；2）共同交互项目的流行度，流行度越低（即更能体现用户个性化偏好）影响越大；3）v 的活跃度，活跃度越低影响越大。这种可解释性很好地符合了协同过滤中衡量用户相似性的假设 [2, 37]，也证明了 LightGCN 的合理性。由于 LightGCN 的对称形式，我们可以对项目方面进行类似的分析。

3.3 模型训练

LightGCN 的可训练参数仅为第 0 层的嵌入，即换句话说，模型复杂度与标准矩阵分解（MF）相同。我们采用贝叶斯个性化排名（BPR）损失 [32]，这是一种成对损失，鼓励对观察到的条目的预测高于未观察到的对应条目：

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

其中 λ 控制 L2 正则化强度。我们采用 Adam [22] 优化器，并以小批量方式使用它。我们知道还有其他高级的负采样策略可能会改进 LightGCN 的训练，比如硬负采样 [31] 和对抗采样 [9]。由于这不是本工作的重点，所以我们将这个扩展留给未来。

需要注意的是，我们没有引入 GCN 和 NGCF 中常用的 dropout 机制。原因是在 LightGCN 中我们没有特征转换的权重矩阵，因此对嵌入层进行 L2 正则化就足以防止过拟合。这展示了 LightGCN 简单性的优势: 它比 NGCF 更容易训练和调整。

表 3.1 实验数据的统计值

Dataset	User #	Item #	Interaction #	Density
Gowalla	29858	40981	1027370	0.00084
Yelp2018	31668	38048	1561406	0.00130
Amazon-Book	52643	91599	2984108	0.00062

4 实验

首先，我们描述实验设置，然后在第 4.2 节中与 NGCF [39] 进行详细比较，这是与 LightGCN 最相关但更复杂的方法。接下来，我们在第 4.3 节中与其他最先进的方法进行比较。为了证明 LightGCN 中的设计，并揭示其有效性的原因，我们在第 4.4 节进行消融研究和嵌入分析。最后，在第 4.5 节中展示了超参数研究。

4.1 参数设置

与 NGCF 相同，嵌入维度对所有模型都固定为 64，并且嵌入参数使用 Xavier 方法 [12] 进行初始化。我们使用 Adam [22] 优化 LightGCN，并使用默认的学习率 0.001 和默认的小批量大小 1024（在 Amazon-Book 上，我们将小批量大小增加到 2048 以提高速度）。L2 正则化系数 A 的搜索范围为（le-6，le-5，le-2），在大多数情况下，最佳值为 le-4。层组合系数均匀设定为 K 是层数。我们在 1 到 4 的范围内测试 K，当 K 等于 3 时可以获得令人满意的性能。早期停止和验证策略与 NGCF 相同。通常情况下，LightGCN 收敛所需的 epoch 数为 1000。我们的实现在 TensorFlow6 和 Pytorch 上都有.

Table 3: Performance comparison between NGCF and LightGCN at different layers.

Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)

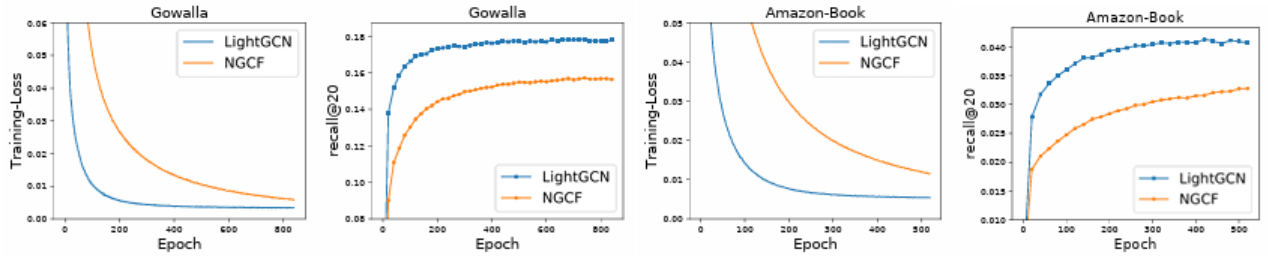


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

图 4.1

4.2 与层数比较

我们与 NGCF 进行了详细的比较，在表 4 中记录了不同层次（1 到 4）的性能，并展示了每个指标的相对改进百分比。我们进一步绘制了训练损失和测试召回率的训练曲线，如图 3 所示，以展示 LightGCN 的优势并清楚地展示训练过程。主要观察结果如下：在所有情况下，LightGCN 的性能都大幅优于 NGCF。例如，在 Gowalla 上，NGCF 论文中报告的最

Table 4: The comparison of overall performance among LightGCN and competing methods.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	**0.0411**	**0.0315**

图 4.2

高召回率为 0.1570，而我们的 LightGCN 在 4 层设置下可以达到 0.1830，高出 16.56%。在三个数据集上，平均召回率提高了 16.52%，ndcg 提高了 16.87%，这些改进相当显著。将表 4 与第 2 节中的表 1 进行对比，我们可以看到 LightGCN 比 NGCF-fn 表现更好，NGCF-fn 是 NGCF 的变体，去除了特征转换和非线性激活。由于 NGCF-fn 仍然包含比 LightGCN 更多的操作（例如，自连接，在图卷积中用户嵌入和物品嵌入之间的交互以及丢弃），这表明这些操作对于 NGCF-fn 也可能是无用的。增加层数可以改善性能，但收益递减。一般观察结果是，将层数从 0（即矩阵分解模型，结果见 [39]）增加到 1 可以获得最大的性能提升，并且在大多数情况下，使用 3 层可以获得令人满意的性能。这个观察结果与 NGCF 的发现一致。在训练过程中，LightGCN 始终获得较低的训练损失，这表明 LightGCN 比 NGCF 更好地适应了训练数据。此外，较低的训练损失成功转化为更好的测试准确性，表明 LightGCN 具有强大的泛化能力。相反，NGCF 的较高训练损失和较低测试准确性反映出训练这样一个繁重模型的实际困难。请注意，在图中我们展示了两种方法在最佳超参数设置下的训练过程。尽管增加 NGCF 的学习率可以降低其训练损失（甚至低于 LightGCN 的训练损失），但测试召回率无法改善，因为以这种方式降低训练损失只会找到 NGCF 的微不足道的解决方案。

4.3 与现有技术的性能比较

表 4 显示了与竞争方法的性能比较。我们展示了每种方法可以获得的最佳得分。我们可以看到，LightGCN 在所有三个数据集上始终优于其他方法，显示了其高效性和简单而合理的设计。请注意，LightGCN 可以通过调整（见图 4 的证据）进一步改进，但在这里我们只使用统一的设置以避免过度调整。在基线方法中，Mult-VAE 展现出最强的性能，优于

GRMF 和 NGCF。GRMF 的性能与 NGCF 持平，优于 MF，这表明了用拉普拉斯正则化器强制嵌入平滑性的效用。通过在拉普拉斯正则化器中添加归一化，GRMF-norm 在 Gowalla 上优于 GRMF，但在 Yelp2018 和 Amazon-Book 上没有带来任何好处。

4.4 消融和有效性分析

我们通过显示层组合和对称 sqrt 归一化如何影响性能来进行 LightGCN 的消融研究，以证明 LightGCN 在第 3.2.3 节中分析的合理性，我们进一步研究了嵌入平滑性的影响，这是 LightGCN 有效性的关键原因。

4.4.1 层组合的影响

图 4 显示了 LightGCN 及其变体 LightGCN-single 的结果，后者不使用层组合（即，对于 K 层的 LightGCN，使用 E(k) 进行最终预测）。由于空间限制，我们省略了 Yelp2018 的结果，它与 Amazon-Book 的趋势相似。我们有三个主要观察结果：关注 LightGCN-single，

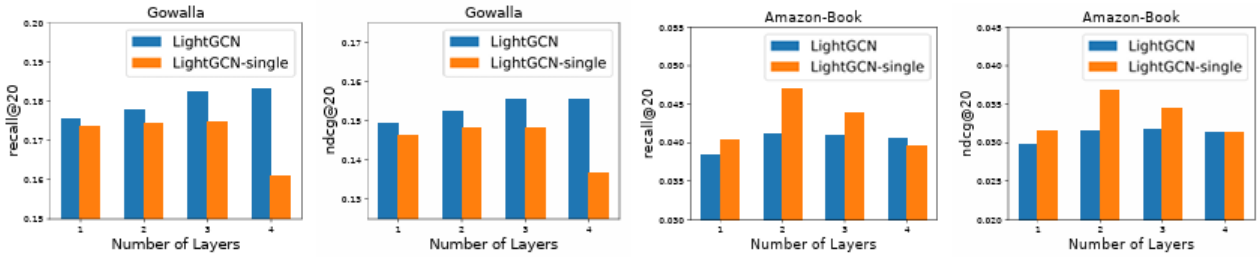


Figure 4: Results of LightGCN and the variant that does not use layer combination (i.e., LightGCN-single) at different layers on Gowalla and Amazon-Book (results on Yelp2018 shows the same trend with Amazon-Book which are omitted for space).

Table 5: Performance of the 3-layer LightGCN with different choices of normalization schemes in graph convolution.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
LightGCN-L_1-L	0.1724	0.1414	0.0630	0.0511	**0.0419**	**0.0320**
LightGCN-L_1-R	0.1578	0.1348	0.0587	0.0477	0.0334	0.0259
LightGCN-L_1	0.159	0.1319	0.0573	0.0465	0.0361	0.0275
LightGCN-L	0.1589	0.1317	0.0619	0.0509	0.0383	0.0299
LightGCN-R	0.1420	0.1156	0.0521	0.0401	0.0252	0.0196
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	0.0411	0.0315

Method notation: -L means only the left-side norm is used, -R means only the right-side norm is used, and -L_1 means the L_1 norm is used.

Table 6: Smoothness loss of the embeddings learned by LightGCN and MF (the lower the smoother).

Dataset	Gowalla	Yelp2018	Amazon-book
	Smoothness of User Embeddings		
MF	15449.3	16258.2	38034.2
LightGCN-single	12872.7	10091.7	32191.1
	Smoothness of Item Embeddings		
MF	12106.7	16632.1	28307.9
LightGCN-single	5829.0	6459.8	16866.0

图 4.3

我们发现其性能在层数从 1 增加到 4 时先提高后下降。在大多数情况下，峰值点在第 2 层，之后很快下降到第 4 层的最差点。这表明使用一阶和二阶邻居平滑节点的嵌入对于 CF 非

常有用，但是使用更高阶邻居时会遇到过度平滑问题。关注 LightGCN，我们发现其性能随着层数增加而逐渐提高。即使使用 4 层，LightGCN 的性能也不会下降。这证明了层组合在解决过度平滑方面的有效性，正如我们在第 3.2.2 节中技术分析的那样（与 APPNP 的关系）。比较这两种方法，我们发现 LightGCN 在 Gowalla 上一直优于 LightGCN-single，但在 Amazon-Book 和 Yelp2018 上不是如此。

4.5 超参数分析

当将 LightGCN 应用于新数据集时，除了标准的超参数学习率之外，最重要的超参数要调整的是 L2 正则化系数 λ。在这里，我们调查了 LightGCN 的性能变化。

如图 5 所示，LightGCN 对 λ 相对不敏感——即使将 X 设置为 0，LightGCN 仍然优于 NGCF，后者另外使用了防止过拟合的 dropout[8]。这表明 LightGCN 不太容易过拟合——因为在 LightGCN 中唯一可训练的参数是 O 层的 ID 嵌入，整个模型易于训练和正则化。对于 Yelp2018、Amazon-Book 和 Gowalla，最佳值分别为 1e-3、1e-4 和 1e-4。当 λ 大于 1e-3 时，性能迅速下降，这表明过强的正则化会对模型的正常训练产生负面影响，不鼓励使用。

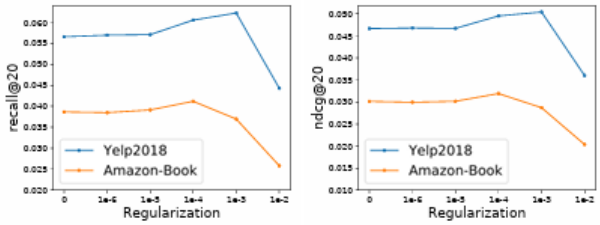


Figure 5: Performance of 2-layer LightGCN w.r.t. different regularization coefficient λ on Yelp and Amazon-Book.

图 4.4

5 相关工作

5.1 协同过滤

协同过滤（CF）是现代推荐系统中普遍采用的技术 [7, 45]。CF 模型的一种常见范式是将用户和物品参数化为嵌入，并通过重构历史用户-物品交互来学习嵌入参数。例如，早

期的 CF 模型如矩阵分解（MF）[26, 32] 将用户（或物品）的 ID 投影到嵌入向量中。最近的神经推荐模型如 NCF [19] 和 LRML [34] 使用相同的嵌入组件，但通过神经网络增强了交互建模。

除了仅使用 ID 信息之外，另一种 CF 方法考虑将历史物品作为用户的现有特征，以更好地表示用户。例如，FISM [21] 和 SVD++ [25] 使用历史物品 ID 嵌入的加权平均作为目标用户的嵌入。最近，研究人员意识到历史物品对塑造个人兴趣有不同的贡献。为此，引入了注意机制来捕获各个历史物品的不同贡献，例如 ACF [3] 和 NAIS [18]，以自动学习每个历史物品的重要性。当将历史交互重新视为用户-物品二部图时，性能提升可以归因于对局部邻域的编码——即一跳邻居——这有助于改进嵌入学习。

5.2 用于推荐的图方法

另一条相关的研究线是利用用户-物品图结构进行推荐。早期的努力如 ItemRank [13]，使用标签传播机制直接在图上传播用户偏好分数，即鼓励连接的节点具有相似的标签。最近出现的图神经网络（GNNs）在建模图结构方面表现出色，尤其是高阶邻居，以指导嵌入学习 [14, 23]。早期的研究在谱域定义图卷积，如拉普拉斯特征分解 [1] 和切比雪夫多项式 [8]，这些方法计算开销较大。后来，GraphSage [14] 和 GCN [23] 在空间域中重新定义了图卷积，即聚合邻居的嵌入以优化目标节点的嵌入。由于其可解释性和高效性，它迅速成为 GNNs 的流行公式，并得到广泛应用 [11, 29, 47]。受益于图卷积的强大，最近的努力如 NGCF [39]、GC-MC [35] 和 PinSage [45] 将 GCN 应用于用户-物品交互图，捕获推荐中高阶邻居的 CF 信号。

值得一提的是，一些最近的努力为 GNNs 提供了深刻的见解 [24, 27, 40]，这些见解启发了我们开发 LightGCN。特别是，Wu 等人 [40] 认为 GCN 的复杂性是不必要的，通过删除非线性并将多个权重矩阵合并为一个简化的 GCN（SGCN）模型。一个主要区别是 LightGCN 和 SGCN 是为不同的任务开发的，因此模型简化的合理性也不同。具体来说，SGCN 是用于节点分类，通过简化来提高模型的可解释性和效率。相比之下，LightGCN 是用于协同过滤（CF），其中每个节点只有一个 ID 特征。因此，我们进行简化是有更强烈的原因的：非线性和权重矩阵对 CF 来说是多余的，甚至会影响模型的训练。对于节点分类准确性，SGCN 与 GCN 相当（有时更弱）。而对于 CF 准确性，LightGCN 比 GCN 要好得多

（相对于 NGCF 提高了超过 15%）。最后，同一时间进行的另一项工作 [4] 也发现了 NGCF 中不必要的非线性，并开发了线性 GCN 模型用于 CF。相比之下，我们的 LightGCN 更进一步——我们删除了所有冗余参数，仅保留了 ID 嵌入，使模型像 MF 一样简单。

6 结论与未来工作

在这项工作中，我们论证了 GCNs 在协同过滤中设计过于复杂，进行了经验研究来证明这一观点。我们提出了 LightGCN，它由两个基本组件组成——轻量级图卷积和层组合。在轻量级图卷积中，我们舍弃了特征转换和非线性激活——这是 GCNs 中的两个标准操作，但不可避免地增加了训练难度。在层组合中，我们将节点的最终嵌入构建为其所有层嵌入的加权和，证明它包含了自连接的效果，并有助于控制过度平滑。我们进行实验来展示 LightGCN 的优势：更容易训练，更好的泛化能力和更有效。

我们相信 LightGCN 的见解对未来推荐模型的发展具有启发作用。随着实际应用中链接图数据的普及，基于图的模型在推荐中变得越来越重要；通过明确利用预测模型中实体之间的关系，它们对于隐式建模关系的传统监督学习方案（如因子分解机 [17, 33]）是有利的。例如，最近的趋势是利用辅助信息，如物品知识图 [38]、社交网络 [41] 和多媒体内容 [44] 用于推荐，其中 GCNs 已经建立了新的技术水平。然而，这些模型也可能面临与 NGCF 相似的问题，因为用户-物品交互图也是通过相同的神经操作建模的，这些操作可能是不必要的。我们计划在这些模型中探索 LightGCN 的思想。另一个未来的方向是个性化层组合权重 a，以便为不同的用户实现自适应顺序平滑（例如，稀疏用户可能需要更多来自高阶邻居的信号，而活跃用户需要更少）。最后，我们将进一步探讨 LightGCN 简单性的优势，研究非抽样回归损失的快速解决方案是否存在，并将其用于在线工业场景。

致谢。感谢 Bing Wu、Jianbai Ye 和 Yingxin Wu 为 LightGCN 的实现和改进所做的贡献。本工作得到了中国国家自然科学基金（61972372、U19A2079、61725203）的支持。

四、外文原文

LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

Xiangnan He
University of Science and Technology of China
xiangnanhe@gmail.com

Kuan Deng
University of Science and Technology of China
dengkuan@mail.ustc.edu.cn

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Yan Li
Beijing Kuaishou Technology Co., Ltd.
liyan@kuaishou.com

Yongdong Zhang
University of Science and Technology of China
zhyd73@ustc.edu.cn

Meng Wang*
Hefei University of Technology
eric.mengwang@gmail.com

ABSTRACT

Graph Convolution Network (GCN) has become new state-of-the-art for collaborative filtering. Nevertheless, the reasons of its effectiveness for recommendation are not well understood. Existing work that adapts GCN to recommendation lacks thorough ablation analyses on GCN, which is originally designed for graph classification tasks and equipped with many neural network operations. However, we empirically find that the two most common designs in GCNs — feature transformation and nonlinear activation — contribute little to the performance of collaborative filtering. Even worse, including them adds to the difficulty of training and degrades recommendation performance.

In this work, we aim to simplify the design of GCN to make it more concise and appropriate for recommendation. We propose a new model named LightGCN, including only the most essential component in GCN — neighborhood aggregation — for collaborative filtering. Specifically, LightGCN learns user and item embeddings by linearly propagating them on the user-item interaction graph, and uses the weighted sum of the embeddings learned at all layers as the final embedding. Such simple, linear, and neat model is much easier to implement and train, exhibiting substantial improvements (about 16.0% relative improvement on average) over Neural Graph Collaborative Filtering (NGCF) — a state-of-the-art GCN-based recommender model — under exactly the same experimental setting. Further analyses are provided towards the rationality of the simple LightGCN from both analytical and empirical perspectives. Our implementations are available in both TensorFlow[1] and PyTorch[2].

*Meng Wang is the corresponding author.
[1]https://github.com/kuandeng/LightGCN
[2]https://github.com/gusye1234/pytorch-light-gcn

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401063

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Collaborative Filtering, Recommendation, Embedding Propagation, Graph Neural Network

ACM Reference Format:
Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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 (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401063

1 INTRODUCTION

To alleviate information overload on the web, recommender system has been widely deployed to perform personalized information filtering [7, 45, 46]. The core of recommender system is to predict whether a user will interact with an item, e.g., click, rate, purchase, among other forms of interactions. As such, collaborative filtering (CF), which focuses on exploiting the past user-item interactions to achieve the prediction, remains to be a fundamental task towards effective personalized recommendation [10, 19, 28, 39].

The most common paradigm for CF is to learn latent features (a.k.a. embedding) to represent a user and an item, and perform prediction based on the embedding vectors [6, 19]. Matrix factorization is an early such model, which directly projects the single ID of a user to her embedding [26]. Later on, several research find that augmenting user ID with the her interaction history as the input can improve the quality of embedding. For example, SVD++ [25] demonstrates the benefits of user interaction history in predicting user numerical ratings, and Neural Attentive Item Similarity (NAIS) [18] differentiates the importance of items in the interaction history and shows improvements in predicting item ranking. In view of user-item interaction graph, these improvements can be seen as coming from using the subgraph structure of a user — more specifically, her one-hop neighbors — to improve the embedding learning.

To deepen the use of subgraph structure with high-hop neighbors, Wang et al. [39] recently proposes NGCF and achieves state-of-the-art performance for CF. It takes inspiration from the Graph Convolution Network (GCN) [14, 23], following the same

propagation rule to refine embeddings: feature transformation, neighborhood aggregation, and nonlinear activation. Although NGCF has shown promising results, we argue that its designs are rather heavy and burdensome — many operations are directly inherited from GCN without justification. As a result, they are not necessarily useful for the CF task. To be specific, GCN is originally proposed for node classification on attributed graph, where each node has rich attributes as input features; whereas in user-item interaction graph for CF, each node (user or item) is only described by a one-hot ID, which has no concrete semantics besides being an identifier. In such a case, given the ID embedding as the input, performing multiple layers of nonlinear feature transformation — which is the key to the success of modern neural networks [16] — will bring no benefits, but negatively increases the difficulty for model training.

To validate our thoughts, we perform extensive ablation studies on NGCF. With rigorous controlled experiments (on the same data splits and evaluation protocol), we draw the conclusion that the two operations inherited from GCN — feature transformation and nonlinear activation — has no contribution on NGCF's effectiveness. Even more surprising, removing them leads to significant accuracy improvements. This reflects the issues of adding operations that are useless for the target task in graph neural network, which not only brings no benefits, but rather degrades model effectiveness. Motivated by these empirical findings, we present a new model named LightGCN, including the most essential component of GCN — neighborhood aggregation — for collaborative filtering. Specifically, after associating each user (item) with an ID embedding, we propagate the embeddings on the user-item interaction graph to refine them. We then combine the embeddings learned at different propagation layers with a weighted sum to obtain the final embedding for prediction. The whole model is simple and elegant, which not only is easier to train, but also achieves better empirical performance than NGCF and other state-of-the-art methods like Mult-VAE [28].

To summarize, this work makes the following main contributions:

- We empirically show that two common designs in GCN, feature transformation and nonlinear activation, have no positive effect on the effectiveness of collaborative filtering.
- We propose LightGCN, which largely simplifies the model design by including only the most essential components in GCN for recommendation.
- We empirically compare LightGCN with NGCF by following the same setting and demonstrate substantial improvements. In-depth analyses are provided towards the rationality of LightGCN from both technical and empirical perspectives.

2 PRELIMINARIES

We first introduce NGCF [39], a representative and state-of-the-art GCN model for recommendation. We then perform ablation studies on NGCF to judge the usefulness of each operation in NGCF. The novel contribution of this section is to show that the two common designs in GCNs, feature transformation and nonlinear activation, have no positive effect on collaborative filtering.

Table 1: Performance of NGCF and its three variants.

	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

2.1 NGCF Brief

In the initial step, each user and item is associated with an ID embedding. Let $\mathbf{e}_u^{(0)}$ denote the ID embedding of user u and $\mathbf{e}_i^{(0)}$ denote the ID embedding of item i. Then NGCF leverages the user-item interaction graph to propagate embeddings as:

$$\mathbf{e}_u^{(k+1)} = \sigma\Big(\mathbf{W}_1\mathbf{e}_u^{(k)} + \sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_i^{(k)} + \mathbf{W}_2(\mathbf{e}_i^{(k)}\odot\mathbf{e}_u^{(k)}))\Big),$$
$$\mathbf{e}_i^{(k+1)} = \sigma\Big(\mathbf{W}_1\mathbf{e}_i^{(k)} + \sum_{u\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1\mathbf{e}_u^{(k)} + \mathbf{W}_2(\mathbf{e}_u^{(k)}\odot\mathbf{e}_i^{(k)}))\Big), \tag{1}$$

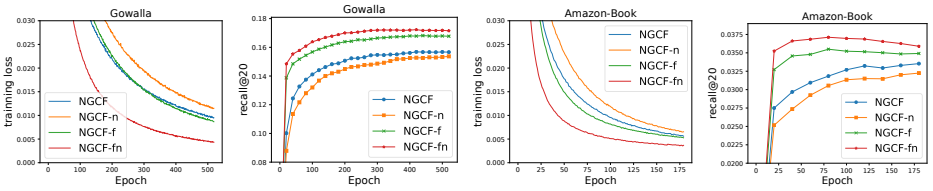
where $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_i^{(k)}$ respectively denote the refined embedding of user u and item i after k layers propagation, σ is the nonlinear activation function, $\mathcal{N}_u$ denotes the set of items that are interacted by user u, $\mathcal{N}_i$ denotes the set of users that interact with item i, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are trainable weight matrix to perform feature transformation in each layer. By propagating L layers, NGCF obtains $L+1$ embeddings to describe a user $(\mathbf{e}_u^{(0)}, \mathbf{e}_u^{(1)}, ..., \mathbf{e}_u^{(L)})$ and an item $(\mathbf{e}_i^{(0)}, \mathbf{e}_i^{(1)}, ..., \mathbf{e}_i^{(L)})$. It then concatenates these $L+1$ embeddings to obtain the final user embedding and item embedding, using inner product to generate the prediction score.

NGCF largely follows the standard GCN [23], including the use of nonlinear activation function $\sigma(\cdot)$ and feature transformation matrices $\mathbf{W}_1$ and $\mathbf{W}_2$. However, we argue that the two operations are not as useful for collaborative filtering. In semi-supervised node classification, each node has rich semantic features as input, such as the title and abstract words of a paper. Thus performing multiple layers of nonlinear transformation is beneficial to feature learning. Nevertheless, in collaborative filtering, each node of user-item interaction graph only has an ID as input which has no concrete semantics. In this case, performing multiple nonlinear transformations will not contribute to learn better features; even worse, it may add the difficulties to train well. In the next subsection, we provide empirical evidence on this argument.

2.2 Empirical Explorations on NGCF

We conduct ablation studies on NGCF to explore the effect of nonlinear activation and feature transformation. We use the codes released by the authors of NGCF[3], running experiments on the same data splits and evaluation protocol to keep the comparison as fair as possible. Since the core of GCN is to refine embeddings by propagation, we are more interested in the embedding quality under the same embedding size. Thus, we change the way of obtaining final embedding from concatenation (i.e., $\mathbf{e}_u^* = \mathbf{e}_u^{(0)}\|\cdots\|\mathbf{e}_u^{(L)}$) to sum (i.e., $\mathbf{e}_u^* = \mathbf{e}_u^{(0)} + \cdots + \mathbf{e}_u^{(L)}$). Note that this change has little effect

[3]https://github.com/xiangwang1223/neural_graph_collaborative_filtering



(a) Training loss on Gowalla (b) Testing recall on Gowalla (c) Training loss on Amazon-Book (d) Testing recall on Amazon-Book

Figure 1: Training curves (training loss and testing recall) of NGCF and its three simplified variants.

on NGCF's performance, but makes the following ablation studies more indicative of the embedding quality refined by GCN.

We implement three simplified variants of NGCF:

- NGCF-f, which removes the feature transformation matrices $\mathbf{W}_1$ and $\mathbf{W}_2$.
- NGCF-n, which removes the non-linear activation function σ.
- NGCF-fn, which removes both the feature transformation matrices and non-linear activation function.

For the three variants, we keep all hyper-parameters (e.g., learning rate, regularization coefficient, dropout ratio, etc.) same as the optimal settings of NGCF. We report the results of the 2-layer setting on the Gowalla and Amazon-Book datasets in Table 1. As can be seen, removing feature transformation (i.e., NGCF-f) leads to consistent improvements over NGCF on all three datasets. In contrast, removing nonlinear activation does not affect the accuracy that much. However, if we remove nonlinear activation on the basis of removing feature transformation (i.e., NGCF-fn), the performance is improved significantly. From these observations, we conclude the findings that:

(1) Adding feature transformation imposes negative effect on NGCF, since removing it in both models of NGCF and NGCF-n improves the performance significantly;

(2) Adding nonlinear activation affects slightly when feature transformation is included, but it imposes negative effect when feature transformation is disabled.

(3) As a whole, feature transformation and nonlinear activation impose rather negative effect on NGCF, since by removing them simultaneously, NGCF-fn demonstrates large improvements over NGCF (9.57% relative improvement on recall).

To gain more insights into the scores obtained in Table 1 and understand why NGCF deteriorates with the two operations, we plot the curves of model status recorded by training loss and testing recall in Figure 1. As can be seen, NGCF-fn achieves a much lower training loss than NGCF, NGCF-f, and NGCF-n along the whole training process. Aligning with the curves of testing recall, we find that such lower training loss successfully transfers to better recommendation accuracy. The comparison between NGCF and NGCF-f shows the similar trend, except that the improvement margin is smaller.

From these evidences, we can draw the conclusion that the deterioration of NGCF stems from the training difficulty, rather than overfitting. Theoretically speaking, NGCF has higher representation power than NGCF-f, since setting the weight matrix $\mathbf{W}_1$ and $\mathbf{W}_2$ to identity matrix $\mathbf{I}$ can fully recover the NGCF-f model. However, in practice, NGCF demonstrates higher training loss and worse generalization performance than NGCF-f. And the incorporation of nonlinear activation further aggravates the discrepancy between representation power and generalization performance. To round out this section, we claim that when designing model for recommendation, it is important to perform rigorous ablation studies to be clear about the impact of each operation. Otherwise, including less useful operations will complicate the model unnecessarily, increase the training difficulty, and even degrade model effectiveness.

3 METHOD

The former section demonstrates that NGCF is a heavy and burdensome GCN model for collaborative filtering. Driven by these findings, we set the goal of developing a light yet effective model by including the most essential ingredients of GCN for recommendation. The advantages of being simple are several-fold — more interpretable, practically easy to train and maintain, technically easy to analyze the model behavior and revise it towards more effective directions, and so on.

In this section, we first present our designed *Light Graph Convolution Network* (LightGCN) model, as illustrated in Figure 2. We then provide an in-depth analysis of LightGCN to show the rationality behind its simple design. Lastly, we describe how to do model training for recommendation.

3.1 LightGCN

The basic idea of GCN is to learning representation for nodes by smoothing features over the graph [23, 40]. To achieve this, it performs graph convolution iteratively, i.e., aggregating the features of neighbors as the new representation of a target node. Such neighborhood aggregation can be abstracted as:

$$\mathbf{e}_u^{(k+1)} = \mathrm{AGG}(\mathbf{e}_u^{(k)}, \{\mathbf{e}_i^{(k)} : i \in \mathcal{N}_u\}). \tag{2}$$

The AGG is an aggregation function — the core of graph convolution — that considers the k-th layer's representation of the target node and its neighbor nodes. Many work have specified the AGG, such as the weighted sum aggregator in GIN [42], LSTM aggregator in GraphSAGE [14], and bilinear interaction aggregator in BGNN [48] etc. However, most of the work ties feature transformation or nonlinear activation with the AGG function. Although they perform well on node or graph classification tasks that have semantic input features, they could be burdensome for collaborative filtering (see preliminary results in Section 2.2).

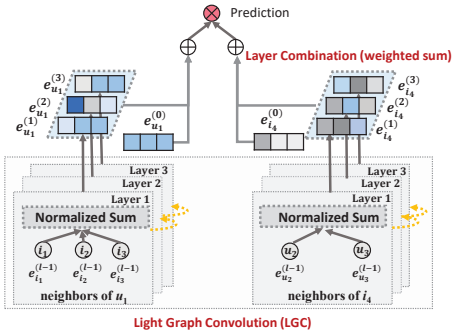


Figure 2: An illustration of LightGCN model architecture. In LGC, only the normalized sum of neighbor embeddings is performed towards next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, we sum over the embeddings at each layer to obtain the final representations.

3.1.1 Light Graph Convolution (LGC). In LightGCN, we adopt the simple weighted sum aggregator and abandon the use of feature transformation and nonlinear activation. The graph convolution operation (a.k.a., propagation rule [39]) in LightGCN is defined as:

$$\begin{aligned} \mathbf{e}_u^{(k+1)} &= \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}, \\ \mathbf{e}_i^{(k+1)} &= \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}. \end{aligned} \tag{3}$$

The symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ follows the design of standard GCN [23], which can avoid the scale of embeddings increasing with graph convolution operations; other choices can also be applied here, such as the L_1 norm, while empirically we find this symmetric normalization has good performance (see experiment results in Section 4.4.2).

It is worth noting that in LGC, we aggregate only the connected neighbors and do not integrate the target node itself (i.e., self-connection). This is different from most existing graph convolution operations [14, 23, 36, 39, 48] that typically aggregate extended neighbors and need to handle the self-connection specially. The layer combination operation, to be introduced in the next subsection, essentially captures the same effect as self-connections. Thus, there is no need in LGC to include self-connections.

3.1.2 Layer Combination and Model Prediction. In LightGCN, the only trainable model parameters are the embeddings at the 0-th layer, i.e., $\mathbf{e}_u^{(0)}$ for all users and $\mathbf{e}_i^{(0)}$ for all items. When they are given, the embeddings at higher layers can be computed via LGC defined in Equation (3). After K layers LGC, we further combine the embeddings obtained at each layer to form the final representation of a user (an item):

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}, \tag{4}$$

where $\alpha_k \geq 0$ denotes the importance of the k-th layer embedding in constituting the final embedding. It can be treated as a hyper-parameter to be tuned manually, or as a model parameter (e.g., output of an attention network [3]) to be optimized automatically. In our experiments, we find that setting α_k uniformly as $1/(K+1)$ leads to good performance in general. Thus we do not design special component to optimize α_k, to avoid complicating LightGCN unnecessarily and to keep its simplicity. The reasons that we perform layer combination to get final representations are three-fold. (1) With the increasing of the number of layers, the embeddings will be over-smoothed [27]. Thus simply using the last layer is problematic. (2) The embeddings at different layers capture different semantics. E.g., the first layer enforces smoothness on users and items that have interactions, the second layer smooths users (items) that have overlap on interacted items (users), and higher-layers capture higher-order proximity [39]. Thus combining them will make the representation more comprehensive. (3) Combining embeddings at different layers with weighted sum captures the effect of graph convolution with self-connections, an important trick in GCNs (proof sees Section 3.2.1).

The model prediction is defined as the inner product of user and item final representations:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i, \tag{5}$$

which is used as the ranking score for recommendation generation.

3.1.3 Matrix Form. We provide the matrix form of LightGCN to facilitate implementation and discussion with existing models. Let the user-item interaction matrix be $\mathbf{R} \in \mathbb{R}^{M \times N}$ where M and N denote the number of users and items, respectively, and each entry R_{ui} is 1 if u has interacted with item i otherwise 0. We then obtain the adjacency matrix of the user-item graph as

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix}, \tag{6}$$

Let the 0-th layer embedding matrix be $\mathbf{E}^{(0)} \in \mathbb{R}^{(M+N) \times T}$, where T is the embedding size. Then we can obtain the matrix equivalent form of LGC as:

$$\mathbf{E}^{(k+1)} = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)}, \tag{7}$$

where $\mathbf{D}$ is a $(M+N) \times (M+N)$ diagonal matrix, in which each entry D_{ii} denotes the number of nonzero entries in the i-th row vector of the adjacency matrix $\mathbf{A}$ (also named as degree matrix). Lastly, we get the final embedding matrix used for model prediction as:

$$\begin{aligned} \mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)} \\ &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \alpha_2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \alpha_K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}, \end{aligned} \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized matrix.

3.2 Model Analysis

We conduct model analysis to demonstrate the rationality behind the simple design of LightGCN. First we discuss the connection with the Simplified GCN (SGCN) [40], which is a recent linear

GCN model that integrates self-connection into graph convolution; this analysis shows that by doing layer combination, LightGCN subsumes the effect of self-connection thus there is no need for LightGCN to add self-connection in adjacency matrix. Then we discuss the relation with the Approximate Personalized Propagation of Neural Predictions (APPNP) [24], which is recent GCN variant that addresses oversmoothing by inspiring from Personalized PageRank [15]; this analysis shows the underlying equivalence between LightGCN and APPNP, thus our LightGCN enjoys the sames benefits in propagating long-range with controllable oversmoothing. Lastly we analyze the second-layer LGC to show how it smooths a user with her second-order neighbors, providing more insights into the working mechanism of LightGCN.

3.2.1 Relation with SGCN. In [40], the authors argue the unnecessary complexity of GCN for node classfication and propose SGCN, which simplifies GCN by removing nonlinearities and collapsing the weight matrices to one weight matrix. The graph convolution in SGCN is defined as[4]:

$$\mathbf{E}^{(k+1)} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}\mathbf{E}^{(k)}, \tag{9}$$

where $\mathbf{I} \in \mathbb{R}^{(M+N)\times(M+N)}$ is an identity matrix, which is added on $\mathbf{A}$ to include self-connections. In the following analysis, we omit the $(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$ terms for simplicity, since they only re-scale embeddings. In SGCN, the embeddings obtained at the last layer are used for downstream prediction task, which can be expressed as:

$$\begin{aligned}\mathbf{E}^{(K)} &= (\mathbf{A}+\mathbf{I})\mathbf{E}^{(K-1)} = (\mathbf{A}+\mathbf{I})^K\mathbf{E}^{(0)} \\ &= \binom{K}{0}\mathbf{E}^{(0)} + \binom{K}{1}\mathbf{A}\mathbf{E}^{(0)} + \binom{K}{2}\mathbf{A}^2\mathbf{E}^{(0)} + \ldots + \binom{K}{K}\mathbf{A}^K\mathbf{E}^{(0)}.\end{aligned} \tag{10}$$

The above derivation shows that, inserting self-connection into $\mathbf{A}$ and propagating embeddings on it, is essentially equivalent to a weighted sum of the embeddings propagated at each LGC layer.

3.2.2 Relation with APPNP. In a recent work [24], the authors connect GCN with Personalized PageRank [15], inspiring from which they propose a GCN variant named APPNP that can propagate long range without the risk of oversmoothing. Inspired by the teleport design in Personalized PageRank, APPNP complements each propagation layer with the starting features (i.e., the 0-th layer embeddings), which can balance the need of preserving locality (i.e., staying close to the root node to alleviate oversmoothing) and leveraging the information from a large neighborhood. The propagation layer in APPNP is defined as:

$$\mathbf{E}^{(k+1)} = \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(k)}, \tag{11}$$

where β is the teleport probability to control the retaining of starting features in the propagation, and $\tilde{\mathbf{A}}$ denotes the normalized adjacency matrix. In APPNP, the last layer is used for final prediction, i.e.,

$$\begin{aligned}\mathbf{E}^{(K)} &= \beta\mathbf{E}^{(0)} + (1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(K-1)}, \\ &= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + (1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(K-2)} \\ &= \beta\mathbf{E}^{(0)} + \beta(1-\beta)\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \beta(1-\beta)^2\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + (1-\beta)^K\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}.\end{aligned} \tag{12}$$

[4]The weight matrix in SGCN can be absorbed into the 0-th layer embedding parameters, thus it is omitted in the analysis.

Aligning with Equation (8), we can see that by setting α_k accordingly, LightGCN can fully recover the prediction embedding used by APPNP. As such, LightGCN shares the strength of APPNP in combating oversmoothing — by setting the α properly, we allow using a large K for long-range modeling with controllable oversmoothing.

Another minor difference is that APPNP adds self-connection into the adjacency matrix. However, as we have shown before, this is redundant due to the weighted sum of different layers.

3.2.3 Second-Order Embedding Smoothness. Owing to the linearity and simplicity of LightGCN, we can draw more insights into how does it smooth embeddings. Here we analyze a 2-layer LightGCN to demonstrate its rationality. Taking the user side as an example, intuitively, the second layer smooths users that have overlap on the interacted items. More concretely, we have:

$$\mathbf{e}_u^{(2)} = \sum_{i\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\mathbf{e}_i^{(1)} = \sum_{i\in\mathcal{N}_u}\frac{1}{|\mathcal{N}_i|}\sum_{v\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}}\mathbf{e}_v^{(0)}. \tag{13}$$

We can see that, if another user v has co-interacted with the target user u, the smoothness strength of v on u is measured by the coefficient (otherwise 0):

$$c_{v->u} = \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}}\sum_{i\in\mathcal{N}_u\cap\mathcal{N}_v}\frac{1}{|\mathcal{N}_i|}. \tag{14}$$

This coefficient is rather interpretable: the influence of a second-order neighbor v on u is determined by 1) the number of co-interacted items, the more the larger; 2) the popularity of the co-interacted items, the less popularity (i.e., more indicative of user personalized preference) the larger; and 3) the activity of v, the less active the larger. Such interpretability well caters for the assumption of CF in measuring user similarity [2, 37] and evidences the reasonability of LightGCN. Due to the symmetric formulation of LightGCN, we can get similar analysis on the item side.

3.3 Model Training

The trainable parameters of LightGCN are only the embeddings of the 0-th layer, i.e., $\Theta = \{\mathbf{E}^{(0)}\}$; in other words, the model complexity is same as the standard matrix factorization (MF). We employ the *Bayesian Personalized Ranking* (BPR) loss [32], which is a pairwise loss that encourages the prediction of an observed entry to be higher than its unobserved counterparts:

$$L_{BPR} = -\sum_{u=1}^{M}\sum_{i\in\mathcal{N}_u}\sum_{j\notin\mathcal{N}_u}\ln\sigma(\hat{y}_{ui}-\hat{y}_{uj}) + \lambda||\mathbf{E}^{(0)}||^2 \tag{15}$$

where λ controls the L_2 regularization strength. We employ the Adam [22] optimizer and use it in a mini-batch manner. We are aware of other advanced negative sampling strategies which might improve the LightGCN training, such as the hard negative sampling [31] and adversarial sampling [9]. We leave this extension in the future since it is not the focus of this work.

Note that we do not introduce dropout mechanisms, which are commonly used in GCNs and NGCF. The reason is that we do not have feature transformation weight matrices in LightGCN, thus enforcing L_2 regularization on the embedding layer is sufficient to prevent overfitting. This showcases LightGCN's advantages of being simple — it is easier to train and tune than NGCF which

Table 2: Statistics of the experimented data.

Dataset	User #	Item #	Interaction #	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062

additionally requires to tune two dropout ratios (node dropout and message dropout) and normalize the embedding of each layer to unit length.

Moreover, it is technically viable to also learn the layer combination coefficients $\{\alpha_k\}_{k=0}^{K}$, or parameterize them with an attention network. However, we find that learning α on training data does not lead improvement. This is probably because the training data does not contain sufficient signal to learn good α that can generalize to unknown data. We have also tried to learn α from validation data, as inspired by [5] that learns hyper-parameters on validation data. The performance is slightly improved (less than 1%). We leave the exploration of optimal settings of α (e.g., personalizing it for different users and items) as future work.

4 EXPERIMENTS

We first describe experimental settings, and then conduct detailed comparison with NGCF [39], the method that is most relevant with LightGCN but more complicated (Section 4.2). We next compare with other state-of-the-art methods in Section 4.3. To justify the designs in LightGCN and reveal the reasons of its effectiveness, we perform ablation studies and embedding analyses in Section 4.4. The hyper-parameter study is finally presented in Section 4.5.

4.1 Experimental Settings

To reduce the experiment workload and keep the comparison fair, we closely follow the settings of the NGCF work [39]. We request the experimented datasets (including train/test splits) from the authors, for which the statistics are shown in Table 2. The Gowalla and Amazon-Book are exactly the same as the NGCF paper used, so we directly use the results in the NGCF paper. The only exception is the Yelp2018 data, which is a revised version. According to the authors, the previous version did not filter out cold-start items in the testing set, and they shared us the revised version only. Thus we re-run NGCF on the Yelp2018 data. The evaluation metrics are recall@20 and ndcg@20 computed by the all-ranking protocol — all items that are not interacted by a user are the candidates.

4.1.1 Compared Methods. The main competing method is NGCF, which has shown to outperform several methods including GCN-based models GC-MC [35] and PinSage [45], neural network-based models NeuMF [19] and CMN [10], and factorization-based models MF [32] and HOP-Rec [43]. As the comparison is done on the same datasets under the same evaluation protocol, we do not further compare with these methods. In addition to NGCF, we further compare with two relevant and competitive CF methods:

- Mult-VAE [28]. This is an item-based CF method based on the variational autoencoder (VAE). It assumes the data is generated from a multinomial distribution and using variational inference for parameter estimation. We run the codes released by the authors[5], tuning the dropout ratio in $[0, 0.2, 0.5]$, and the β in $[0.2, 0.4, 0.6, 0.8]$. The model architecture is the suggested one in the paper: $600 \rightarrow 200 \rightarrow 600$.
- GRMF [30]. This method smooths matrix factorization by adding the graph Laplacian regularizer. For fair comparison on item recommendation, we change the rating prediction loss to BPR loss. The objective function of GRMF is:

$$L = -\sum_{u=1}^{M}\sum_{i\in N_u}\Big(\sum_{j\notin N_u}\ln\sigma(\mathbf{e}_u^T\mathbf{e}_i - \mathbf{e}_u^T\mathbf{e}_j) + \lambda_g||\mathbf{e}_u - \mathbf{e}_i||^2\Big) + \lambda||\mathbf{E}||^2, \tag{16}$$

where λ_g is searched in the range of $[1e^{-5}, 1e^{-4}, ..., 1e^{-1}]$. Moreover, we compare with a variant that adds normalization to graph Laplacian: $\lambda_g||\frac{\mathbf{e}_u}{\sqrt{|N_u|}} - \frac{\mathbf{e}_i}{\sqrt{|N_i|}}||^2$, which is termed as GRMF-norm. Other hyper-parameter settings are same as LightGCN. The two GRMF methods benchmark the performance of smoothing embeddings via Laplacian regularizer, while our LightGCN achieves embedding smoothing in the predictive model.

4.1.2 Hyper-parameter Settings. Same as NGCF, the embedding size is fixed to 64 for all models and the embedding parameters are initialized with the Xavier method [12]. We optimize LightGCN with Adam [22] and use the default learning rate of 0.001 and default mini-batch size of 1024 (on Amazon-Book, we increase the mini-batch size to 2048 for speed). The L_2 regularization coefficient λ is searched in the range of $\{1e^{-6}, 1e^{-5}, ..., 1e^{-2}\}$, and in most cases the optimal value is $1e^{-4}$. The layer combination coefficient α_k is uniformly set to $\frac{1}{1+K}$ where K is the number of layers. We test K in the range of 1 to 4, and satisfactory performance can be achieved when K equals to 3. The early stopping and validation strategies are the same as NGCF. Typically, 1000 epochs are sufficient for LightGCN to converge. Our implementations are available in both TensorFlow[6] and PyTorch[7].

4.2 Performance Comparison with NGCF

We perform detailed comparison with NGCF, recording the performance at different layers (1 to 4) in Table 4, which also shows the percentage of relative improvement on each metric. We further plot the training curves of training loss and testing recall in Figure 3 to reveal the advantages of LightGCN and to be clear of the training process. The main observations are as follows:

- In all cases, LightGCN outperforms NGCF by a large margin. For example, on Gowalla the highest recall reported in the NGCF paper is 0.1570, while our LightGCN can reach 0.1830 under the 4-layer setting, which is 16.56% higher. On average, the recall improvement on the three datasets is 16.52% and the ndcg improvement is 16.87%, which are rather significant.
- Aligning Table 4 with Table 1 in Section 2, we can see that LightGCN performs better than NGCF-fn, the variant of NGCF that removes feature transformation and nonlinear activation. As NGCF-fn still contains more operations than LightGCN (e.g., self-connection, the interaction between user embedding and item

[5] https://github.com/dawenl/vae_cf
[6] https://github.com/kuandeng/LightGCN
[7] https://github.com/gusye1234/pytorch-light-gcn

Table 3: Performance comparison between NGCF and LightGCN at different layers.

Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.0530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

*The scores of NGCF on Gowalla and Amazon-Book are directly copied from Table 3 of the NGCF paper (https://arxiv.org/abs/1905.08108)

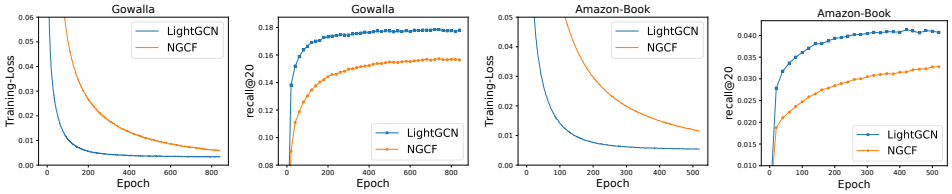


Figure 3: Training curves of LightGCN and NGCF, which are evaluated by training loss and testing recall per 20 epochs on Gowalla and Amazon-Book (results on Yelp2018 show exactly the same trend which are omitted for space).

embedding in graph convolution, and dropout), this suggests that these operations might also be useless for NGCF-fn.

- Increasing the number of layers can improve the performance, but the benefits diminish. The general observation is that increasing the layer number from 0 (i.e., the matrix factorization model, results see [39]) to 1 leads to the largest performance gain, and using a layer number of 3 leads to satisfactory performance in most cases. This observation is consistent with NGCF's finding.
- Along the training process, LightGCN consistently obtains lower training loss, which indicates that LightGCN fits the training data better than NGCF. Moreover, the lower training loss successfully transfers to better testing accuracy, indicating the strong generalization power of LightGCN. In contrast, the higher training loss and lower testing accuracy of NGCF reflect the practical difficulty to train such a heavy model it well. Note that in the figures we show the training process under the optimal hyper-parameter setting for both methods. Although increasing the learning rate of NGCF can decrease its training loss (even lower than that of LightGCN), the testing recall could not be improved, as lowering training loss in this way only finds trivial solution for NGCF.

4.3 Performance Comparison with State-of-the-Arts

Table 4 shows the performance comparison with competing methods. We show the best score we can obtain for each method. We can see that LightGCN consistently outperforms other methods on all three datasets, demonstrating its high effectiveness with simple yet reasonable designs. Note that LightGCN can be further improved by tuning the α_k (see Figure 4 for an evidence), while here we only use a uniform setting of $\frac{1}{K+1}$ to avoid over-tuning it. Among the baselines, Mult-VAE exhibits the strongest performance, which is better than GRMF and NGCF. The performance of GRMF is on a par with NGCF, being better than MF, which admits the utility of enforcing embedding smoothness with Laplacian regularizer. By adding normalization into the Laplacian regularizer, GRMF-norm betters than GRMF on Gowalla, while brings no benefits on Yelp2018 and Amazon-Book.

Table 4: The comparison of overall performance among LightGCN and competing methods.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	recall	ndcg	recall	ndcg	recall	ndcg
NGCF	0.1570	0.1327	0.0579	0.0477	0.0344	0.0263
Mult-VAE	0.1641	0.1335	0.0584	0.0450	0.0407	0.0315
GRMF	0.1477	0.1205	0.0571	0.0462	0.0354	0.0270
GRMF-norm	0.1557	0.1261	0.0561	0.0454	0.0352	0.0269
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	**0.0411**	**0.0315**

4.4 Ablation and Effectiveness Analyses

We perform ablation studies on LightGCN by showing how layer combination and symmetric sqrt normalization affect its performance. To justify the rationality of LightGCN as analyzed in Section 3.2.3, we further investigate the effect of embedding smoothness — the key reason of LightGCN's effectiveness.

4.4.1 Impact of Layer Combination. Figure 4 shows the results of LightGCN and its variant LightGCN-single that does not use layer combination (i.e., $\mathbf{E}^{(K)}$ is used for final prediction for a K-layer LightGCN). We omit the results on Yelp2018 due to space limitation, which show similar trend with Amazon-Book. We have three main observations:

- Focusing on LightGCN-single, we find that its performance first improves and then drops when the layer number increases from

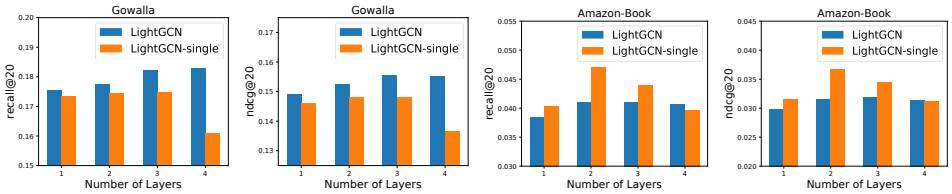


Figure 4: Results of LightGCN and the variant that does not use layer combination (i.e., LightGCN-single) at different layers on Gowalla and Amazon-Book (results on Yelp2018 shows the same trend with Amazon-Book which are omitted for space).

Table 5: Performance of the 3-layer LightGCN with different choices of normalization schemes in graph convolution.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	**recall**	**ndcg**	**recall**	**ndcg**	**recall**	**ndcg**
LightGCN-L_1-L	0.1724	0.1414	0.0630	0.0511	**0.0419**	**0.0320**
LightGCN-L_1-R	0.1578	0.1348	0.0587	0.0477	0.0334	0.0259
LightGCN-L_1	0.159	0.1319	0.0573	0.0465	0.0361	0.0275
LightGCN-L	0.1589	0.1317	0.0619	0.0509	0.0383	0.0299
LightGCN-R	0.1420	0.1156	0.0521	0.0401	0.0252	0.0196
LightGCN	**0.1830**	**0.1554**	**0.0649**	**0.0530**	0.0411	0.0315

Method notation: -L means only the left-side norm is used, -R means only the right-side norm is used, and -L_1 means the L_1 norm is used.

1 to 4. The peak point is on layer 2 in most cases, while after that it drops quickly to the worst point of layer 4. This indicates that smoothing a node's embedding with its first-order and second-order neighbors is very useful for CF, but will suffer from over-smoothing issues when higher-order neighbors are used.

- Focusing on LightGCN, we find that its performance gradually improves with the increasing of layers. Even using 4 layers, LightGCN's performance is not degraded. This justifies the effectiveness of layer combination for addressing over-smoothing, as we have technically analyzed in Section 3.2.2 (relation with APPNP).
- Comparing the two methods, we find that LightGCN consistently outperforms LightGCN-single on Gowalla, but not on Amazon-Book and Yelp2018 (where the 2-layer LightGCN-single performs the best). Regarding this phenomenon, two points need to be noted before we draw conclusion: 1) LightGCN-single is special case of LightGCN that sets α_K to 1 and other α_k to 0; 2) we do not tune the α_k and simply set it as $\frac{1}{K+1}$ uniformly for LightGCN. As such, we can see the potential of further enhancing the performance of LightGCN by tuning α_k.

4.4.2 Impact of Symmetric Sqrt Normalization. In LightGCN, we employ symmetric sqrt normalization $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ on each neighbor embedding when performing neighborhood aggregation (cf. Equation (3)). To study its rationality, we explore different choices here. We test the use of normalization only at the left side (i.e., the target node's coefficient) and the right side (i.e., the neighbor node's coefficient). We also test L_1 normalization, i.e., removing the square root. Note that if removing normalization, the training becomes numerically unstable and suffers from not-a-value (NAN) issues, so we do not show this setting. Table 5 shows the results of the 3-layer LightGCN. We have the following observations:

- The best setting in general is using sqrt normalization at both sides (i.e., the current design of LightGCN). Removing either side will drop the performance largely.
- The second best setting is using L_1 normalization at the left side only (i.e., LightGCN-L_1-L). This is equivalent to normalize the adjacency matrix as a stochastic matrix by the in-degree.
- Normalizing symmetrically on two sides is helpful for the sqrt normalization, but will degrade the performance of L_1 normalization.

Table 6: Smoothness loss of the embeddings learned by LightGCN and MF (the lower the smoother).

Dataset	Gowalla	Yelp2018	Amazon-book
	Smoothness of User Embeddings		
MF	15449.3	16258.2	38034.2
LightGCN-single	12872.7	10091.7	32191.1
	Smoothness of Item Embeddings		
MF	12106.7	16632.1	28307.9
LightGCN-single	5829.0	6459.8	16866.0

4.4.3 Analysis of Embedding Smoothness. As we have analyzed in Section 3.2.3, a 2-layer LightGCN smooths a user's embedding based on the users that have overlap on her interacted items, and the smoothing strength between two users $c_{v\to u}$ is measured in Equation (14). We speculate that such smoothing of embeddings is the key reason of LightGCN's effectiveness. To verify this, we first define the smoothness of user embeddings as:

$$S_U = \sum_{u=1}^{M}\sum_{v=1}^{M} c_{v\to u}\left(\frac{\mathbf{e}_u}{||\mathbf{e}_u||^2} - \frac{\mathbf{e}_v}{||\mathbf{e}_v||^2}\right)^2, \tag{17}$$

where the L_2 norm on embeddings is used to eliminate the impact of the embedding's scale. Similarly we can obtained the definition for item embeddings. Table 6 shows the smoothness of two models, matrix factorization (i.e., using the $\mathbf{E}^{(0)}$ for model prediction) and the 2-layer LightGCN-single (i.e., using the $\mathbf{E}^{(2)}$ for prediction). Note that the 2-layer LightGCN-single outperforms MF in recommendation accuracy by a large margin. As can be seen, the smoothness loss of LightGCN-single is much lower than that of MF. This indicates that by conducting light graph convolution, the embeddings become smoother and more suitable for recommendation.

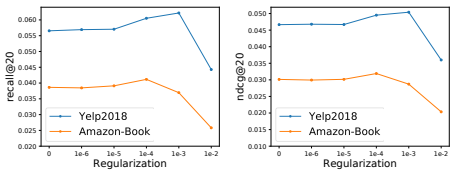


Figure 5: Performance of 2-layer LightGCN w.r.t. different regularization coefficient λ on Yelp and Amazon-Book.

4.5 Hyper-parameter Studies

When applying LightGCN to a new dataset, besides the standard hyper-parameter learning rate, the most important hyper-parameter to tune is the L_2 regularization coefficient λ. Here we investigate the performance change of LightGCN w.r.t. λ.

As shown in Figure 5, LightGCN is relatively insensitive to λ — even when λ sets to 0, LightGCN is better than NGCF, which additionally uses dropout to prevent overfitting[8]. This shows that LightGCN is less prone to overfitting — since the only trainable parameters in LightGCN are ID embeddings of the 0-th layer, the whole model is easy to train and to regularize. The optimal value for Yelp2018, Amazon-Book, and Gowalla is $1e^{-3}$, $1e^{-4}$, and $1e^{-4}$, respectively. When λ is larger than $1e^{-3}$, the performance drops quickly, which indicates that too strong regularization will negatively affect model normal training and is not encouraged.

5 RELATED WORK

5.1 Collaborative Filtering

Collaborative Filtering (CF) is a prevalent technique in modern recommender systems [7, 45]. One common paradigm of CF model is to parameterize users and items as embeddings, and learn the embedding parameters by reconstructing historical user-item interactions. For example, earlier CF models like matrix factorization (MF) [26, 32] project the ID of a user (or an item) into an embedding vector. The recent neural recommender models like NCF [19] and LRML [34] use the same embedding component, while enhance the interaction modeling with neural networks.

Beyond merely using ID information, another type of CF methods considers historical items as the pre-existing features of a user, towards better user representations. For example, FISM [21] and SVD++ [25] use the weighted average of the ID embeddings of historical items as the target user's embedding. Recently, researchers realize that historical items have different contributions to shape personal interest. Towards this end, attention mechanisms are introduced to capture the varying contributions, such as ACF [3] and NAIS [18], to automatically learn the importance of each historical item. When revisiting historical interactions as a user-item bipartite graph, the performance improvements can be attributed to the encoding of local neighborhood — one-hop neighbors — that improves the embedding learning.

[8]Note that Gowalla shows the same trend with Amazon-Book, so its curves are not shown to better highlight the trend of Yelp2018 and Amazon-Book.

5.2 Graph Methods for Recommendation

Another relevant research line is exploiting the user-item graph structure for recommendation. Prior efforts like ItemRank [13], use the label propagation mechanism to directly propagate user preference scores over the graph, *i.e.,* encouraging connected nodes to have similar labels. Recently emerged graph neural networks (GNNs) shine a light on modeling graph structure, especially high-hop neighbors, to guide the embedding learning [14, 23]. Early studies define graph convolution on the spectral domain, such as Laplacian eigen-decomposition [1] and Chebyshev polynomials [8], which are computationally expensive. Later on, GraphSage [14] and GCN [23] re-define graph convolution in the spatial domain, i.e., aggregating the embeddings of neighbors to refine the target node's embedding. Owing to its interpretability and efficiency, it quickly becomes a prevalent formulation of GNNs and is being widely used [11, 29, 47]. Motivated by the strength of graph convolution, recent efforts like NGCF [39], GC-MC [35], and PinSage [45] adapt GCN to the user-item interaction graph, capturing CF signals in high-hop neighbors for recommendation.

It is worth mentioning that several recent efforts provide deep insights into GNNs [24, 27, 40], which inspire us developing LightGCN. Particularly, Wu et al. [40] argues the unnecessary complexity of GCN, developing a simplified GCN (SGCN) model by removing nonlinearities and collapsing multiple weight matrices into one. One main difference is that LightGCN and SGCN are developed for different tasks, thus the rationality of model simplification is different. Specifically, SGCN is for node classification, performing simplification for model interpretability and efficiency. In contrast, LightGCN is on collaborative filtering (CF), where each node has an ID feature only. Thus, we do simplification for a stronger reason: nonlinearity and weight matrices are useless for CF, and even hurt model training. For node classification accuracy, SGCN is on par with (sometimes weaker than) GCN. While for CF accuracy, LightGCN outperforms GCN by a large margin (over 15% improvement over NGCF). Lastly, another work conducted in the same time [4] also finds that the nonlinearity is unnecessary in NGCF and develops linear GCN model for CF. In contrast, our LightGCN makes one step further — we remove all redundant parameters and retain only the ID embeddings, making the model as simple as MF.

6 CONCLUSION AND FUTURE WORK

In this work, we argued the unnecessarily complicated design of GCNs for collaborative filtering, and performed empirical studies to justify this argument. We proposed LightGCN which consists of two essential components — light graph convolution and layer combination. In light graph convolution, we discard feature transformation and nonlinear activation — two standard operations in GCNs but inevitably increase the training difficulty. In layer combination, we construct a node's final embedding as the weighted sum of its embeddings on all layers, which is proved to subsume the effect of self-connections and is helpful to control oversmoothing. We conduct experiments to demonstrate the strengths of LightGCN in being simple: easier to be trained, better generalization ability, and more effective.

We believe the insights of LightGCN are inspirational to future developments of recommender models. With the prevalence of linked graph data in real applications, graph-based models are becoming increasingly important in recommendation; by explicitly exploiting the relations among entities in the predictive model, they are advantageous to traditional supervised learning scheme like factorization machines [17, 33] that model the relations implicitly. For example, a recent trend is to exploit auxiliary information such as item knowledge graph [38], social network [41] and multimedia content [44] for recommendation, where GCNs have set up the new state-of-the-art. However, these models may also suffer from the similar issues of NGCF since the user-item interaction graph is also modeled by same neural operations that may be unnecessary. We plan to explore the idea of LightGCN in these models. Another future direction is to personalize the layer combination weights α_k, so as to enable adaptive-order smoothing for different users (e.g., sparse users may require more signal from higher-order neighbors while active users require less). Lastly, we will explore further the strengths of LightGCN's simplicity, studying whether fast solution exists for non-sampling regression loss [20] and streaming it for online industrial scenarios.

Acknowledgement. The authors thank Bin Wu, Jianbai Ye, and Yingxin Wu for contributing to the implementation and improvement of LightGCN. This work is supported by the National Natural Science Foundation of China (61972372, U19A2079, 61725203).

REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[2] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *WWW*. 2637–2643.
[3] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *SIGIR*. 335–344.
[4] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *AAAI*.
[5] Yihong Chen, Bei Chen, Xiangnan He, Chen Gao, Yong Li, Jian-Guang Lou, and Yue Wang. 2019. λOpt: Learn to Regularize Recommender Models in Finer Levels. In *KDD*. 978–986.
[6] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan S. Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *WWW*. 639–648.
[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.
[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*. 3837–3845.
[9] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. 2230–2236.
[10] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR*. 515–524.
[11] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal Relational Ranking for Stock Prediction. *TOIS* 37, 2 (2019), 27:1–27:30.
[12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[13] Marco Gori and Augusto Pucci. 2007. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *IJCAI*. 2766–2771.
[14] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.
[15] Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *WWW*. 517–526.
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
[17] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.
[18] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *TKDE* 30, 12 (2018), 2354–2366.
[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[20] Xiangnan He, Jinhui Tang, Xiaoyu Du, Richang Hong, Tongwei Ren, and Tat-Seng Chua. 2019. Fast Matrix Factorization with Nonuniform Weights on Missing Data. *TNNLS* (2019).
[21] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *KDD*. 659–667.
[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[24] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.
[25] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.
[26] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.
[27] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.
[28] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. 689–698.
[29] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning. In *KDD*. 2110–2119.
[30] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *NIPS*. 2107–2115.
[31] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*. 273–282.
[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.
[33] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.
[34] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.
[35] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *KDD Workshop on Deep Learning Day*.
[36] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[37] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. 2006. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *SIGIR*. 501–508.
[38] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*. 950–958.
[39] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.
[40] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*. 6861–6871.
[41] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *SIGIR*. 235–244.
[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks?. In *ICLR*.
[43] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*. 140–144.
[44] Yinwei Yin, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multimodal Graph Convolution Network for Personalized Recommendation of Micro-video. In *MM*.
[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD (Data Science track)*. 974–983.
[46] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. In *SIGIR*.
[47] Cheng Zhao, Chenliang Li, and Cong Fu. 2019. Cross-Domain Recommendation via Preference Propagation GraphNet. In *CIKM*. 2165–2168.
[48] Hongmin Zhu, Fuli Feng, Xiangnan He, Xiang Wang, Yan Li, Kai Zheng, and Yongdong Zhang. 2020. Bilinear Graph Neural Network with Neighbor Interactions. In *IJCAI*.

毕业论文（设计）文献综述和开题报告考核

导师对开题报告、外文翻译和文献综述的评语及成绩评定：

该学生对当前研究现状做了较好的总结和归纳，提出了可行的技术路线，同时外文翻译完整、准确，因此给与通过。

成绩比例	文献综述占（10%）	开题报告占（15%）	外文翻译占（5%）
分值	9	12	4

导师签名 _____

年　月　日

学院盲审专家对开题报告、外文翻译和文献综述的评语及成绩评定：

文献综述部分阐述不够连贯，很多概念的提出没有铺垫，需要完善。很多用词要保持统一，如 GCNs。开题报告整体结构可以，同意通过。

成绩比例	文献综述占（10%）	开题报告占（15%）	外文翻译占（5%）
分值	9	12	4

开题报告审核负责人（签名/签章） _____

年　月　日