Self-supervised Graph Learning for Recommendation

- ABSTRACT
  - 用于推荐的用户-项目图的表示学习已经从使用**单一 ID** 或**交互历史**发展到**利用高阶邻居**。这导致了用于推荐的图卷积网络 (GCN) 的成功，例如 PinSage 和 LightGCN。尽管有效，但我们认为它们存在两个限制：
    - **(1)高度节点对表示学习的影响较大，降低了低度（长尾）项目的推荐；**
    - **(2) 表示容易受到噪声交互的影响，因为邻域聚合方案进一步扩大了观察到的边缘的影响。**
  - 在这项工作中，我们**探索了用户-项目图上的自监督学习，以提高 GCN 推荐的准确性和鲁棒性**。这个想法是**用一个辅助的自我监督任务来补充推荐的经典监督任务，通过自我辨别来加强节点表示学习**。具体来说，我们**生成一个节点的多个视图，与其他节点相比，最大化同一节点的不同视图之间的一致性。**我们设计了三个操作符来生成视图——**节点丢弃**、**边缘丢弃**和**随机游走**——它们以不同的方式改变图结构。我们将这种新的学习范式称为自监督图学习 (SGL)，并在最先进的模型 LightGCN 上实现它。通过理论分析，我们发现SGL具有自动挖掘hard negatives的能力。对三个基准数据集的实证研究证明了 SGL 的有效性，它提高了推荐的准确性，尤其是在长尾项目上，以及对交互噪声的鲁棒性。
- INTRODUCTION
  - 从交互数据中学习高质量的用户和项目表示是协作推荐的主题。**矩阵分解 (MF) 等早期工作将每个用户（或项目）的单个 ID 投影到嵌入向量中。**一些后续研究通过交互历史丰富了单个 ID，以学习更好的表示。最近，表示学习已经发展到利用用户项目图中的高阶连通性。该技术的灵感来自图卷积网络 (GCN)，它提供了一种端到端的方法，将多跳邻居集成到节点表示学习中，并实现最先进的推荐性能。
  - 尽管有效，但当前基于 GCN 的推荐模型仍存在一些局限性：
    - **Sparse Supervision Signal（稀疏监督信号）**
      - 大多数模型在监督学习范式下处理推荐任务，其中监督信号来自观察到的用户-项目交互。 然而，与整个交互空间相比，**观察到的交互非常稀疏，因此不足以学习质量表示。**
    - **Skewed Data Distribution（倾斜的数据分布）**
      - 观察到的交互通常表现出幂律分布，其中长尾由缺乏监督信号的低度项目组成。相比之下，高度项目在邻域聚合和监督损失中出现的频率更高，因此对表示学习的影响更大。 **因此，GCNs 很容易偏向高度项目，牺牲了低度（长尾）项目的性能。**
    - **Noises in Interactions（交互中的噪音）**
      - 用户提供的大多数反馈是隐含的（例如点击、查看），而不是明确的（例如评分、喜欢/不喜欢）。 因此，观察到的交互通常包含噪音，例如，用户被误导点击一个项目并在消费后发现它无趣。 **GCN 中的邻域聚合方案扩大了交互对表示学习的影响，使学习更容易受到交互噪声的影响。**

- 在这项工作中，我们专注于探索推荐中的自我监督学习（SSL），以解决上述限制。 虽然在计算机视觉 (CV) 和自然语言处理 (NLP) 中很流行，但 SSL 在推荐方面的探索相对较少。 这个想法是**设置一个辅助任务，从输入数据本身中提取额外的信号，特别是通过利用未标记的数据空间。**例如，**BERT 随机屏蔽句子中的一些标记，将屏蔽标记的预测设置为可以捕获标记之间依赖关系的辅助任务；RotNet 随机旋转标记图像，在旋转图像上训练模型以获得对象识别或图像分类的掩码任务的改进表示。**与监督学习相比，**SSL 允许我们通过更改输入数据来利用未标记的数据空间，从而在下游任务中取得显着改进。**
- 在这里，我们希望将 SSL 的优势带入推荐表示学习，这与 CV/NLP 任务不同，因为数据是离散的和相互连接的。为了解决基于 GCN 的推荐模型的上述限制，我们将辅助任务构建为区分节点本身的表示。 具体来说，它由两个关键组件组成：
  - **data augmentation（数据增加）**
    - **为每个节点生成多个视图。**
  - **contrastive learning（对比学习）**
    - **与其他节点相比，它最大化了同一节点的不同视图之间的一致性。**
  - 对于用户-项目图上的 GCN，图结构作为输入数据，对表示学习起着至关重要的作用。从这个角度来看，通过改变图的邻接矩阵来构建“未标记”的数据空间是很自然的，为此我们开发了三个算子：**node dropout**、**edge dropout**和**random walk**，其中每个算子的工作合理性不同。此后，我们基于 GCN 对改变的结构进行对比学习。 因此，SGL 通过探索节点之间的内部关系来增强节点表示学习。
- 从概念上讲，我们的 SGL 在以下方面补充了现有的基于 GCN 的推荐模型：
  - **(1) 节点自判别提供了辅助监督信号，这是对仅来自观察到的交互的经典监督的补充;**
  - **(2) 增强算子，尤其是边缘丢失，通过有意减少高度节点的影响来帮助减轻度偏差；**
  - **(3) 节点 w.r.t 的多个视图。 不同的局部结构和邻域增强了模型对交互噪声的鲁棒性。**
  - 最后但同样重要的是，我们对对比学习范式进行了理论分析，发现它具有挖掘难负样本的副作用，这不仅提高了性能，而且加速了训练过程。
- 值得一提的是，**我们的 SGL 与模型无关，可以应用于任何包含用户和/或项目嵌入的基于图的模型。** 在这里，我们在简单但有效的模型 LightGCN 上实现它。 对三个基准数据集的实验研究证明了 SGL 的有效性，它显着提高了推荐的准确性，尤其是在长尾项目上，并增强了对交互噪声的鲁棒性。 我们将这项工作的贡献总结如下：
  - 我们设计了一种新的学习范式 SGL，它将节点自判别作为自监督任务，为表征学习提供辅助信号。
  - 除了减轻程度偏差和增加对交互噪声的鲁棒性外，我们在理论上证明 SGL 本质上鼓励从硬负样本中学习，由 softmax 损失函数中的温度超参数控制。
  - 我们对三个基准数据集进行了广泛的实验，以证明 SGL 的优越性。
- PRELIMINARIES
  - 我们首先总结了基于 GCN 的协同过滤模型的常见范式。 让 U 和 I 分别是用户和项目的集合。 令 O+ = {yui |u ∈ U,i ∈ I} 为观察到的交互，其中 yui 表示用户 u 之前采用过项目 i。 大

多数现有模型都构建了一个二分图 G = (V, E)，其中节点集 V = U ∪ I 涉及所有用户和项目，边集 E = O+ 表示观察到的交互。

- Recap GCN
  - 核心是在 G 上应用邻域聚合方案，通过聚合邻居节点的表示来更新自我节点的表示：（1）其中 Z(l) 表示第 l 层的节点表示，Z(l -1) 是前一层的节点表示，Z(0) 是 ID 嵌入（可训练参数）。 H 表示邻域聚合的函数，从向量级别更容易解释：（2）为了更新自我节点 u 在第 l 层的表示，它首先在第 (l-1) 层聚合其邻居 Nu 的表示，然后与它自己的表示 z(l-1) 组合。faggregate(·) 和 fcombine(·) 有多种设计。 我们可以看到第 l 层的表示对图中的 l 阶邻居进行了编码。 在获得 L 层表示后，可能会有一个读出函数来生成最终的表示进行预测：（3）常见的设计包括仅最后一层、串联和加权和。
- Supervised Learning Loss（监督学习损失）
  - 预测层建立在最终表示的基础上，以预测您采用 i 的可能性。 一个经典的解决方案是内积，它支持快速检索：（4）为了优化模型参数，现有工作通常将任务框架为监督学习之一，其中监督信号来自观察到的交互（也是 G 的边缘）。 例如，鼓励预测值 y^ui 接近真实值 yui，并从缺失数据中选择负例。除了上述逐点学习之外，另一个常见的选择是成对贝叶斯个性化排名（BPR）损失，它强制对观察到的交互的预测得分高于其未观察到的对应物：（5）其中 O = {(u,i,j)|(u,i) ∈ O+,(u,j) ∈ O−} 是训练数据，O− = U × I \ O+ 是未观察到的交互。 在这项工作中，我们选择它作为主要的监督任务。

- METHODOLOGY
  - 我们提出了提出的自监督图学习 (SGL) 范式，该范式通过自监督学习增强了主要的监督任务。 图 1 说明了 SGL 的工作流程。 具体来说，自监督任务（也称为借口任务或辅助任务）是根据输入数据内的相关性构建监督信号。
  - 具体来说，我们介绍了如何执行生成多个表示视图的数据增强，然后基于生成的表示进行对比学习以构建借口任务。 SSL 以多任务学习的方式与经典 GCN 相结合。 此后，我们从梯度层面对 SSL 进行了理论分析，揭示了与硬负挖掘的联系。 最后，我们分析了 SGL 的复杂性。
  - Data Augmentation on Graph Structure（图结构上的数据增强）
    - 直接嫁接在 CV 和 NLP 任务中采用的数据增强对于基于图的推荐是不可行的，因为具体的特点是：
      - (1)用户和项目的特征是离散的，如 one-hot ID 和其他分类变量。 因此，图像上的增强算子（例如随机裁剪、旋转或模糊）不适用。
      - (2) 更重要的是，与将每个数据实例视为孤立的 CV 和 NLP 任务不同，交互图中的用户和项目本质上是相互连接和相互依赖的。 因此，我们需要为基于图的推荐量身定制的新增强算子。
    - 二分图建立在观察到的用户-项目交互的基础上，因此包含协同过滤信号。 具体来说，第一跳邻域直接描述了自我用户和项目节点——即用户的历史项目（或项目的交互用户）可以被视为用户（或项目）的预先存在的特征。 用户（或项目）的第二跳相邻节点展示出相似的用户 w.r.t. 行为（或类似的项目 w.r.t. 观众）。 此外，从用户到项目的高阶路径反映了用户对项目的潜在兴趣。 毫无疑问，挖掘图结构中的固有模式有助于表示学习。 因此，我们在图结构上设计了三个算子，节点丢失、边缘丢失

和随机游走，以创建不同的节点视图。 算子可以统一表示如下：(6)其中两个随机选择 s1 和 s2 独立应用于图 G，并建立节点 Z1(l) 和 Z2(l) 的两个相关视图。 我们详细说明增广算子如下：

- Node Dropout (ND)
- Edge Dropout (ED)
- Random Walk (RW)

- 为简单起见，我们在每个 epoch 的图结构上应用这些增强——也就是说，我们在新的训练 epoch 开始时生成每个节点的两个不同视图（对于 RW，每层生成两个不同的视图）。 请注意，两个独立进程（即 s1 和 s2）的 dropout 和 masking 比率保持不变。 我们将在未来的工作中调整不同的比率。 还值得一提的是，仅涉及 dropout 和 masking 操作，并没有添加任何模型参数。

- Contrastive Learning（对比学习）
  - 建立节点的增强视图后，我们将同一节点的视图视为正对（即 $\{(z', z'')|u \in U\}$），将任何不同节点的视图视为负对 (即$\{(z', z'')|u,v \in U,u \}\}$)。 正对的辅助监督鼓励同一节点的不同视图之间的一致性进行预测，而负对的监督则加强了不同节点之间的分歧。 形式上，我们遵循 SimCLR并采用对比损失 InfoNCE 来最大化正对的一致性并最小化负对的一致性：(10)其中 s (·) 衡量两个向量之间的相似度，设置为余弦相似度函数； τ 是超参数，在 softmax 中称为温度。 类似地，我们获得了项目侧 L sslitem 的对比损失。 结合这两个损失，我们得到自监督任务的目标函数为 Lssl =Luser+Liitem。
- Multi-task Training（多任务训练）
  - 为了改进 SSL 任务的推荐，我们利用多任务训练策略来联合优化经典推荐任务（参见方程（5））和自监督学习任务（参见方程 (10)）(11)其中 Θ 是 L 中的模型参数集，因为 Lssl 没有引入额外的参数； λ1 和 λ2 分别是控制 SSL 和 L2 正则化强度的超参数。我们还考虑了替代优化——Lssl 的预训练和 Lmain 的微调。 请参阅第 4.4.2 节中的更多详细信息。
- Theoretical Analyses of SGL（SGL的理论分析）
  - 在本节中，我们对 SGL 进行深入分析，旨在回答以下问题：推荐模型如何从 SSL 任务中受益？ 为此，我们探讨了等式（10）中的自监督损失并找到了一个原因：它具有执行难负挖掘的内在能力，这为优化提供了大而有意义的梯度，并指导了节点表示学习。 接下来，我们将逐步介绍我们的分析。
  - 形式上，对于节点 u ∈ U，自监督损失 w.r.t 的梯度。 zu 表示如下：(12)其中 Lssluser (u) 是等式 (10) 中单个节点 u 的单独项； v ∈ U \ {u} 是另一个节点，作为节点 u 的负视图； c(u) 和 c(v) 分别表示正节点 u 和负节点 {v} 对梯度 w.r.t 的贡献。 zu' :(13) (14).....
- Complexity Analyses of SGL
  - 在本小节中，我们分析了以 ED 为策略、LightGCN 为推荐模型的 SGL 的复杂性； 其他选择也可以类似分析。 由于 SGL 没有引入可训练的参数，因此空间复杂度与 LightGCN 相同。 模型推理的时间复杂度也是一样的，因为模型结构没有变化。 在接下来的部分中，我们将分析 SGL 训练的时间复杂度。

- 假设用户-项目交互图中的节点数和边数为 |V | 和 |E | 分别。 让 s 表示 epoch 的数量，B 表示每个训练批次的大小，d 表示嵌入大小，L 表示 GCN 层数，ρ^ = 1 - ρ 表示 SGL-ED 的保持概率。 复杂性主要来自两部分：
  - 邻接矩阵的归一化。 由于我们在每个 epoch 生成两个独立的子图，考虑到完整训练图和两个子图的邻接矩阵中非零元素的数量为 2 |E | , 2ρ^ |E | 和 2ρ^ |E | 分别，其总复杂度为 O(4ρ^ |E| s + 2 |E|)。
  - 评估自我监督损失。 我们在分析中只考虑内积。 如等式（10）所定义，我们在计算用户侧的InfoNCE损失时将所有其他用户节点视为负样本。 在一个批次中，分子和分母的复杂度分别为 O(Bd) 和 O(BMd)，其中 M 是用户数。 因此，每个 epoch 的用户端和项目端的总复杂度为 O(|E| d(2 + |V |))。 因此，整个训练阶段的时间复杂度为O(|E| d(2 + |V |)s)。 降低时间复杂度的另一种方法是仅将批次中的用户（或项目）视为负样本 [6, 49]，从而导致总时间复杂度为 O(|E| d(2 + 2B)s)。
- 我们在表 1 中总结了 LightGCN 和 SGL-ED 在训练中的时间复杂度。LightGCN 和 SGL-ED 的分析复杂度实际上是相同的数量级，因为 LightGCN 的增加只会缩放 LightGCN 的复杂度。 在实践中，以 Yelp2018 数据为例，ρ 为 0.8 的 SGL-ED (alternative) 的时间复杂度大约是 LightGCN 的 3.7 倍，考虑到我们将在 4.3 节中展示的收敛速度的加快，这是完全可以接受的。 2. 测试平台为配备Inter i7-9700K CPU（32GB内存）的Nvidia Titan RTX显卡。 在 Yelp2018 上每个 epoch 的时间成本分别为 LightGCN 和 SGL-ED (alternative) 的 15.2s 和 60.6s，这与复杂度分析一致。