

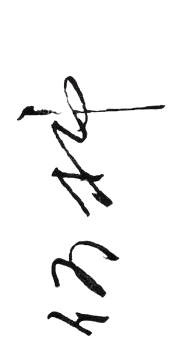
毕业论文

|  |  |
| --- | --- |
| 题 目 | 基于超图正则深度非负矩阵分解的社区发现研究 |
| 学生姓名 | 何郴 |
| 学 号 | 2103070062 |
| 学 院 | 前沿交叉学院 |
| 专业班级 | 数据科学与大数据技术2102班 |
| 指导教师 | 罗鹏 |
| 职 称 | 副教授 |

2025 年 04 月

湖南工商大学本科毕业论文原创性声明

本人郑重声明：所呈交的本科毕业论文是本人在指导老师的指导下，独立进行研究工作所取得的成果，成果不存在知识产权争议，除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。



作者签名：

日期： 2025 年 04 月 23 日

湖南工商大学本科毕业论文知识产权及使用授权声明书

本毕业论文(设计)《基于超图的非负矩阵分解的社区发现研究》是本人在校期间所完成学业的组成部分，是在学校教师的指导下完成的。因此，本人特授权学校可将本毕业论文(设计)的全部或部分内容编入有关书籍、数据库保存，可采用复制、印刷、网页制作等方式将论文(设计)文本和经过编辑、批注等处理的论文(设计)文本提供给读者查阅、参考，可向有关学术部门和国家有关教育主管部门呈送复印件和电子文档。本毕业论文(设计) 无论做何种处理，必须尊重本人的著作权，署明本人姓名。



论文(设计) 作者 时间 2025 年 04 月 23 日

指导教师已阅 时间 2025 年 05 月 04 日

# 内容摘要

本论文主要关注复杂网络中的社区发现问题，构建了一种基于超图正则化的深度非负矩阵分解模型，简称为HDNMF模型，该模型借助引入超图邻接矩阵以及超图拉普拉斯矩阵，实现了对节点间高阶关系的有效建模，弥补了传统非负矩阵分解方法只能处理一阶关系的不足。此模型融合了深度自编码器结构，提升了特征学习的非线性表达能力，同时引入稀疏约束和正交约束，提高了社区划分的区分性与鲁棒性，在实验部分，选用了Email、Wiki、Cora、Citeseer以及Pubmed等多个真实网络数据集，全面比较了HDNMF模型与多种经典社区发现方法的性能。结果说明，所提出的模型在聚类精度、稳定性以及高阶关系捕捉方面有优势，还验证了其在大规模数据场景下的可扩展性，论文最后总结了研究的主要贡献，包括超图与深度非负矩阵分解的有机结合、优化策略的改进以及对高阶复杂网络结构的精准建模，同时指出未来可在模型泛化能力提升以及跨领域应用方面拓展研究方向。

**关键词**

社区发现；超图；非负矩阵分解；深度自编码器；高阶关系建模

# ABSTRACT

This thesis focuses on the problem of community detection in complex networks and proposes a deep non-negative matrix factorization model based on hypergraph regularization, referred to as the HDNMF model. By introducing the hypergraph adjacency matrix and the hypergraph Laplacian matrix, the model effectively captures high-order relationships between nodes, addressing the limitation of traditional non-negative matrix factorization methods, which can only handle first-order relationships. The model integrates a deep autoencoder structure to enhance the non-linear representation capability of feature learning and incorporates sparsity and orthogonality constraints to improve the distinctiveness and robustness of community detection. In the experimental section, multiple real-world network datasets, including Email, Wiki, Cora, Citeseer, and Pubmed, were used to comprehensively compare the performance of the HDNMF model with various classical community detection methods. The results demonstrate that the proposed model outperforms others in clustering accuracy, stability, and high-order relationship capture, and also confirm its scalability in large-scale data scenarios. Finally, the thesis summarizes the main contributions, including the seamless integration of hypergraphs with deep non-negative matrix factorization, improvements in optimization strategies, and precise modeling of high-order complex network structures. It also highlights future research directions, such as enhancing the model’s generalization ability and exploring cross-domain applications.

**KEY WORDS**

Community Detection; Hypergraph; Non-negative Matrix Factorization; Deep Autoencoder; High-order Relationship Modeling

目 录

[1. 引言 1](#_Toc197793075)

[1.1研究背景与意义 1](#_Toc197793076)

[1.1.1社区发现的概念与重要性 1](#_Toc197793077)

[1.2研究现状与发展趋势 3](#_Toc197793078)

[1.2.1传统社区发现方法 3](#_Toc197793079)

[1.2.2基于矩阵分解的社区发现方法 3](#_Toc197793080)

[1.2.3深度学习与超图方法的发展 5](#_Toc197793081)

[1.3研究目标与创新点 5](#_Toc197793082)

[1.3.1研究目标 5](#_Toc197793083)

[1.3.2研究创新点 6](#_Toc197793084)

[2. 相关理论与技术基础 6](#_Toc197793085)

[2.1图论与超图基础 6](#_Toc197793086)

[2.1.1图的基本概念与社区结构 6](#_Toc197793087)

[2.1.2超图的基本概念与其优势 7](#_Toc197793088)

[2.2非负矩阵分解（NMF） 8](#_Toc197793089)

[2.2.1NMF基本原理 8](#_Toc197793090)

[2.2.2NMF在社区发现中的应用与局限性 9](#_Toc197793091)

[2.3深度非负矩阵分解（DNMF） 11](#_Toc197793092)

[2.3.1DNMF的基本结构与优势 11](#_Toc197793093)

[2.3.2深度自编码器在DNMF中的应用 11](#_Toc197793094)

[3. 基于超图的深度非负矩阵 13](#_Toc197793095)

[3.1超图社区网络构建 13](#_Toc197793096)

[3.1.1传统邻接矩阵的不足 13](#_Toc197793097)

[3.1.2超图邻接矩阵的构建方法 13](#_Toc197793098)

[3.2DNMF模型改进设计 15](#_Toc197793099)

[3.2.1超图正则化的引入 15](#_Toc197793100)

[3.2.2深度自编码器与DNMF的结合 16](#_Toc197793101)

[3.3模型算法的提出与优化求解 17](#_Toc197793102)

[3.3.1模型的提出 17](#_Toc197793103)

[3.3.2模型优化求解 18](#_Toc197793104)

[4. 实验与结果分析 22](#_Toc197793105)

[4.1实验准备 22](#_Toc197793106)

[4.1.1数据集选择与预处理 22](#_Toc197793107)

[4.1.2实验设置与评价指标 23](#_Toc197793108)

[4.2模型实现与参数调优 24](#_Toc197793109)

[4.2.1超图深度非负矩阵分解模型训练过程 25](#_Toc197793110)

[4.2.2参数选择与调优 27](#_Toc197793111)

[4.3实验结果与讨论 27](#_Toc197793112)

[4.3.1模型性能对比分析 27](#_Toc197793113)

[4.3.2约束机制的影响分析 30](#_Toc197793114)

[5. 结论与展望 32](#_Toc197793115)

[5.1研究总结 32](#_Toc197793116)

[5.1.1本文主要贡献与创新点 32](#_Toc197793117)

[5.1.2研究过程中的挑战与解决方案 32](#_Toc197793118)

[5.2未来研究方向 33](#_Toc197793119)

[5.2.1模型优化与扩展 33](#_Toc197793120)

[5.2.2超图与其他技术的结合 33](#_Toc197793121)

[5.2.3社区发现的实际应用前景 34](#_Toc197793122)

[参考文献 35](#_Toc197793123)

[附录 38](#_Toc197793125)

[实验代码 38](#_Toc197793126)

[1.数据预处理 38](#_Toc197793127)

[2.模型设计 41](#_Toc197793128)

[3.参数设置 47](#_Toc197793129)

[4.主函数设计 48](#_Toc197793130)

[5.辅助函数 49](#_Toc197793131)

[致谢 52](#_Toc197793124)

# 1. 引言

## 1.1研究背景与意义

### 1.1.1社区发现的概念与重要性

（1）社区发现的概念

复杂网络广泛存在于社会生活的各个领域[1]。对于复杂网络模型的研究已成为众多交叉学科和行业的热点课题。社区结构是复杂网络的核心特征之一，社区发现算法有利于探索复杂网络的内部潜在规律。社区发现本质上是一类Np-hard问题，是复杂网络分析中的一个核心问题，旨在识别网络中节点的自然划分结构。复杂网络是由大量节点和边组成的图结构，其中节点表示系统的个体，而边表示节点之间的关系。在这些网络中，节点往往不是孤立存在的，而是通过各种连接形成不同程度的聚集。社区发现的任务就是将网络中的节点划分成若干个子集，这些子集称为“社区”，社区内的节点之间具有较强的关联或联系，而不同社区之间的联系则相对较弱[2]。在数学上，社区发现可以被看作是对图结构进行聚类的过程。给定一个无向加权图G=(V,E,W)，其中 V 表示节点集合，E 表示边集合，W为边权重矩阵，社区发现的任务就是将节点集合 V 划分为多个子集 C1,C2,...,Ck，使得同一子集内的节点之间的连接紧密，而不同子集的节点之间的连接较为稀疏。

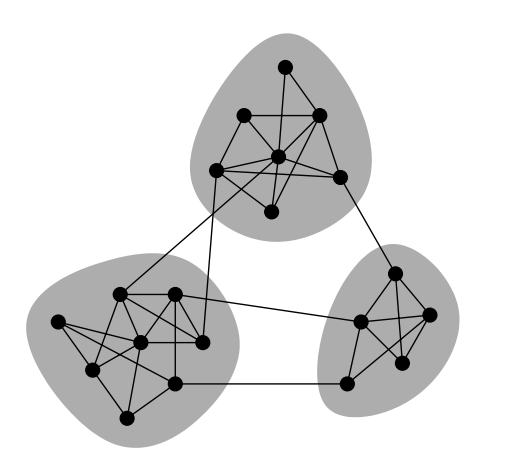


图 1 社区结构

社区发现属于网络科学里的基础问题，同时也是图论、机器学习以及数据挖掘等领域里比较关键的研究方向，它可揭示并理解复杂网络的内在结构，给后续的网络分析、预测以及优化奠定基础，一个网络的社区结构体现了节点之间的协作关系、信息传播路径以及功能模块等关键特征，挖掘网络中的社区信息有着关键的学术价值和应用价值。（2）社区发现的重要性

社区发现用于揭示社交网络中用户之间潜在的社交群体和兴趣圈层，例如在Facebook、Twitter等平台上，用户会基于兴趣、活动以及互动模式自然形成社交团体，通过社区发现能够更深入理解用户的社交行为、影响力传播和信息扩散过程[3]，并广泛应用于社交推荐系统、朋友推荐、舆情分析及广告定向等任务，生物信息学和基因网络研究中，复杂的基因调控网络和蛋白质相互作用网络通常被建模为图结构，基因和蛋白质会组成不同的功能模块或社区，社区发现技术能够识别基因与疾病之间的潜在关系、挖掘生物分子间的协作关系、发现新的药物靶点和生物标志物，因此为生物医学研究、疾病预测和精准治疗提供了新的理论支持和实践途径，推荐系统与个性化服务领域中，社区发现有助于识别用户的兴趣群体，电商平台可以基于用户的购买行为模式识别出相似兴趣的用户群体，从而实现个性化商品推荐，视频推荐平台可以结合用户的观看历史、评分和评论数据发现兴趣相近的用户群体，并向其推荐更契合的内容，这不仅提升了用户体验，还增强了平台的用户粘性和商业价值，金融欺诈检测领域中，银行、证券公司、保险公司等机构的交易行为共同构成一个复杂的网络结构，社区发现可以帮助识别潜在的金融欺诈行为，如洗钱活动或恶意交易，通过挖掘交易网络中的异常社区，监管机构能够提前发现潜在风险并及时采取防范措施，有效降低金融系统的整体风险，在信息传播与舆情分析中，社区发现有助于揭示信息或谣言传播的路径、模式及扩散机制，识别出信息传播的中心群体以及其传播速度和范围，这对于社会事件的舆情管理、公共危机的应对以及选民行为的动态分析具有十分重要的意义，在城市交通与基础设施优化方面，社区发现可以识别交通流量密集区或交通“热点”区域，通过对城市交通网络的社区划分，可优化交通设施建设、调整公共交通线路和制定更合理的交通管理政策，从而实现资源配置的最大化效率，同时社区发现还可应用于基础设施建设规划、能源网络优化等相关领域，在互联网和网络安全领域，社区发现技术被用于识别恶意活动团体、追踪网络攻击源、发现潜在的系统漏洞和风险，通过分析网络流量、攻击模式和用户行为等数据，社区发现能够揭示隐藏的异常群体，帮助网络管理者识别潜在威胁、预防网络攻击，综上所述，社区发现不仅在揭示网络结构和理解网络行为方面具有重要的理论价值，同时在促进跨学科应用、提升各行业的运营效率和安全性等方面也展现出广阔的应用前景和实际意义[4][5][6]。

## 1.2研究现状与发展趋势

社区发现是复杂网络分析的重要研究方向之一，广泛应用于社交网络分析、生物信息学、推荐系统等领域。近年来，随着深度学习和矩阵分解方法的发展，研究者们提出了多种改进的社区发现方法，以提高算法的准确性和可解释性。本文主要围绕传统方法、基于矩阵分解的方法、深度学习方法及未来发展趋势进行综述。

### 1.2.1传统社区发现方法

传统的社区发现方法主要依赖于图划分、模块度优化、随机游走、谱聚类等技术。这些方法主要基于网络的拓扑结构，发现内部连接紧密、外部连接稀疏的社区结构。[7]

图划分方法基于最小割、最大流理论，将网络划分为多个子图。这类方法适用于小规模网络，但在大规模复杂网络上计算成本较高，且划分结果依赖于预设的社区数量。Girvan-Newman算法通过不断移除网络中的高介数边来划分社区，进而优化网络的模块度。[8][9]尽管模块度优化方法在无监督社区发现任务中表现较好，但在大规模网络中易受到分辨率限制影响，导致较小的社区难以识别。随机游走方法 以随机游走为基础的社区发现方法借助节点之间的随机游走概率去衡量社区结构，这类方法可以自行适应并发现社区数量，而且可较好地捕捉局部结构，不过其计算复杂度比较高，在处理超大规模网络时存在险阻，谱聚类方法谱聚类是依据图的拉普拉斯矩阵来开展特征分解的，借助聚类方法实现社区划分。虽然谱聚类在诸多应用里有着不错的表现，然而因为涉及矩阵特征分解，计算成本较高，在扩展到大规模数据集方面存在难度。综上，传统社区发现方法在小规模网络上较为有效，但难以处理异构信息、高维属性以及动态网络结构，因此近年来研究者开始引入基于矩阵分解的方法，以提高算法的可扩展性和鲁棒性。

### 1.2.2基于矩阵分解的社区发现方法

矩阵分解技术（Matrix Factorization, MF）是一种有效的数据降维和特征提取方法，常用于社区发现任务。矩阵分解能够通过对网络邻接矩阵的低秩表示，提取社区结构的潜在模式。其中，非负矩阵分解是最常见的方法之一。

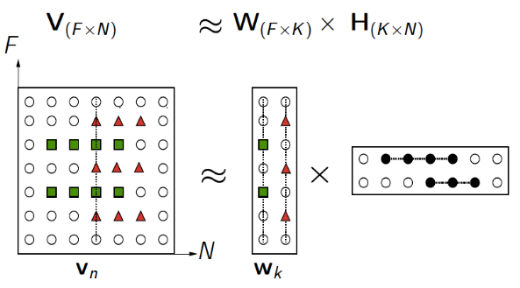


图 2 非负矩阵分解概念图

（1）非负矩阵分解

NMF通过将网络邻接矩阵𝐴分解为两个非负矩阵的乘积𝑊和𝐻即A=WH。其中𝑊代表节点的社区分布，H代表社区的特征表达。NMF的非负性约束使得分解结果更加稀疏，能够直接用于社区划分。然而，传统NMF方法依赖于简单的欧几里得距离度量，难以有效捕捉复杂网络的高阶结构信息。[9]

（2）图正则化GNMF

GNMF 在 NMF 的基础上引入了图正则化项，它是一种**利用图结构先验知识来约束学习过程**的方法，通过约束社区成员在邻域中的相似性来优化分解结果，即如果两个用户是好友，那他们在嵌入空间中学到的表示向量也应该“接近”。其目标函数如下：

其中，𝐿为拉普拉斯矩阵，𝜆为正则化强度。GNMF能够有效提升社区划分的鲁棒性，但仍存在对超大规模网络计算成本较高的问题。

（3）深度非负矩阵分解（DNMF）

近年来，研究者提出了深度非负矩阵分解（DNMF），通过构建多层非负矩阵分解结构，以更好地提取社区网络的深层特征。DNMF类似于深度自编码器（Autoencoder），其核心思想是：

其中，Wi表示第 i 层的特征变换矩阵，HL为最终的社区嵌入表示。DNMF能够在不同层级上学习更复杂的社区结构，提升了算法的表示能力。然而，传统DNMF方法仍然存在高阶关系建模能力不足、收敛性较慢等问题。[8]

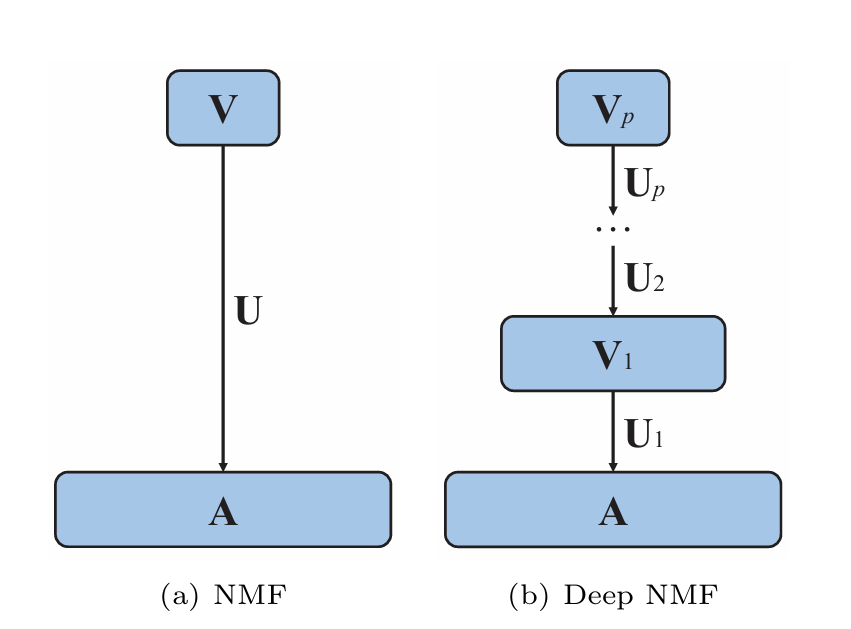


图 3 非负矩阵分解与深度非负矩阵分解概念对比图

### 1.2.3深度学习与超图方法的发展

随着深度学习的发展，基于图神经网络（GNNs） 和 超图方法（Hypergraph Methods） 的社区发现研究逐渐成为主流。

（1）图神经网络（GNNs）在社区发现中的应用

GNNs 能够通过消息传递机制（Message Passing）学习节点的高阶特征表示，如 Graph Convolutional Networks (GCN)、Graph Attention Networks (GAT) 等方法。近些年来，研究人员将GNNs与矩阵分解方法相结合，提出了有更强表达能力的社区发现算法，像是GCN-NMF以及GAT-NMF。[10][11]

（2）超图在社区发现中的应用

传统邻接矩阵表示形式仅能描述一阶或二阶邻居关系，没办法捕捉多个节点间的高阶交互，超图借助超边连接多个节点，可有效呈现复杂关系网络。以社交网络为例，超图可用来构建群聊、共同兴趣群体等结构，让社区发现方法能更精确地识别紧密关联的子群体。

## 1.3研究目标与创新点

### 1.3.1研究目标

本研究聚焦于运用超图正则化的深度非负矩阵分解方法，也就是HDNMF方法，希望能够构建一个更为精准且鲁棒的社区发现模型，其目的在于可有效地挖掘复杂网络里潜在的社区结构，基于此，本文的研究目标囊括以下几个要点：

（1）改进DNMF方法以更好地适应社区发现任务

传统的DNMF方法一般只是单纯考虑节点的局部关系，却忽视了社区内部存在的高阶关联性，在本文中创新性地引入了超图正则化，借助超边对多个节点之间复杂的交互关系进行建模，如此一来社区发现便可更为准确地描绘复杂网络里的拓扑结构。

（2）引入超图结构，提升高阶信息表达能力

传统的图正则化方式主要依靠图拉普拉斯矩阵，不过它仅仅可描述一阶邻接方面的信息，在本文中创新性地运用超图拉普拉斯矩阵来构建正则化项，让DNMF可维持社区内部的连贯性，同时提高对于复杂网络结构的适应能力。

（3）结合自动编码器，提升特征表示能力

传统的非负矩阵分解方法主要是依靠线性分解，在提取复杂网络里的非线性结构信息方面存在险阻，本文结合深度自动编码器，进行了创新性尝试，借助非线性映射来学习社区的潜在表示，以此提升社区发现算法的表达能力。

### 1.3.2研究创新点

（1）提出超图正则化的深度非负矩阵分解方法

传统DNMF方法通常仅考虑节点的局部关系，而忽略了社区内部的高阶关联性。本文创新性地引入超图正则化，通过超边来建模多个节点之间的复杂交互关系，使得社区发现能够更准确地刻画复杂网络中的拓扑结构。

（2）结合超图拉普拉斯进行正则化约束

传统的图正则化方法主要依赖于图拉普拉斯矩阵，但其仅能描述一阶邻接信息。本文创新性地采用超图拉普拉斯矩阵来构造正则化项，使得DNMF能够更好地保持社区内部的连贯性，并增强对复杂网络结构的适应能力。

（3）引入深度自动编码器提升特征学习能力

传统的NMF方法主要依赖线性分解，难以有效提取复杂网络中的非线性结构信息。本文创新性地结合深度自动编码器，通过非线性映射学习社区的潜在表示，提高社区发现算法的表达能力。

本研究的创新点主要集中在**超图正则化、深度非负矩阵分解、超图优化策略、计算效率提升以及多领域实验验证**五个方面。研究目标旨在构建一个能够兼顾**高阶信息建模、计算效率和社区划分准确性**的社区发现方法，为复杂网络分析提供新的思路和方法。

# 2. 相关理论与技术基础

## 2.1图论与超图基础

### 2.1.1图的基本概念与社区结构

（1）图的基本概念

图（Graph）是网络结构的数学抽象，通常表示为，其中：表示节点（Vertex）集合；表示边集合，反映节点之间的关系。

（2）社区结构的定义与特征

社区一般是指这样一种子结构，其中一组节点之间的连接较为密集，而与其他节点的连接则相对稀疏，社区结构在社交网络、生物网络、引文网络等诸多复杂系统中广泛存在，对其进行挖掘，对于深入理解系统的组织形式有着关键意义。[12]

1. 社区的数学定义

设为一个无向图，其中V 是节点集合，E 是边集合。对于任意两个节点u,v ∈V ，定义其连接关系为：

内部连接：若 和 属于同一社区，则它们之间的边数较多；

外部连接：若 和 分属不同社区，则它们之间的边数较少。

基于此，社区 的一个直观判别准则是：

1. 社区结构的典型特征

社区结构一般具有以下特征：高内部连接密度，社区内部的节点彼此之间紧密连接，形成局部的高关联性子图。低外部连接密度，社区与其他社区之间的边较少，表明它们相对独立。重叠性，在某些复杂网络（如社交网络）中，一个节点可能同时属于多个社区；层次结构，部分网络具有多层次的社区组织，即较大的社区可被进一步划分为子社区。

1. 社区结构在不同领域的表现

社区结构在现实世界的复杂系统里广泛存在，于不同领域的网络中呈现出各异的社区特性，在社交网络即人际关系网络中，社区一般代表着兴趣相近且互动频繁的社交群体，像Facebook、Twitter这类社交平台上的好友圈以及兴趣小组，均可被视作社区，在生物网络比如蛋白质相互作用网络中，社区或许对应着功能相似的蛋白质组或者代谢途径。而在学术论文的引文网络里，社区一般对应特定研究领域的学术团体，例如机器学习、计算机视觉等领域。[13]

### 2.1.2超图的基本概念与其优势

（1） 超图的基本概念

在传统的图模型里面，边仅仅可连接两个节点，没办法直接去表示多个实体之间的高阶关系，超图则是一种更为通用的图结构，它准许一条边连接多个节点，可描述现实世界里的复杂关系，从形式上来说，一个超图H=(V,E)是由节点集V和超边集E构成的，其中超边e∈E是V的一个非空子集，也就是说：

和传统图所有的二元关系不一样，超图可允许多元关系存在，这意味着一条超边可连接两个、三个或者更多数量的节点，比如说，在学术合作网络这个环境当中，一篇论文有可能是由多个作者一起撰写完成的，当运用超图进行建模操作的时候，每一篇论文会对应一个超边，而所有的合作者共同构成了这个超边的节点集合。

（2） 超图与传统图的对比

超图与传统的图结构存在许多差异，如下表总结所示：

表 1 传统图结构与超图结构对比

|  |  |  |
| --- | --- | --- |
| 特性 | 传统图 | 超图 |
| 边的定义 | 仅连接两个节点 | 可连接多个节点 |
| 适用关系 | 二元关系 | 多元关系 |
| 灵活性 | 结构较简单 | 结构更复杂，适应更广泛 |
| 表达能力 | 适用于成对的关系，如好友关系 | 适用于群体关系，如科研团队、基因互作等 |

相比于传统图模型，超图在社区发现、数据挖掘、社交网络分析等任务中具有以下优势：

以更自然的方式呈现高阶关系：在诸多实际应用里，数据对象间的关系大多时候是高阶的，并非仅仅局限于简单的二元关系，比如在社交网络中，一个社群或许由多个成员共同参与构建，超图可借助超边把社群的所有成员连接起来，然而传统图只能凭借二元关系进行近似表示，在推荐系统中，一个用户可能会同时对多个商品产生兴趣，超图可更为真实地呈现这些复杂的交互情况。强化社区发现的能力：传统图的社区发现方法一般是基于二元连接的，不过在许多情形下，社区的形成机制是由高阶关系推动的，超图可捕捉到更贴近实际的社交群体结构，像学术合作网络，传统图只能呈现研究人员之间的合作关系，而超图可以把一篇论文的所有作者当作一个整体来开展分析，基因相互作用网络，多个基因可能凭借复杂的生物过程产生相互作用，超图能更精准地描绘这些高阶依赖关系。赋予更精确的数学建模能力：在数学领域，超图可以凭借超邻接矩阵、超拉普拉斯矩阵等形式来进行建模，让机器学习和优化算法可更高效地处理复杂关系，例如超图拉普拉斯算子在谱聚类和图神经网络中有着广泛的应用，提升数据聚合与信息传播能力：超图模型可在多种任务当中提升信息传播的效果，例如在知识图谱里，超图可以将多个实体的相互关系统一整合到一个超边内，更有效地开展推理，在病毒传播建模方面，超图可更准确地描述病毒在高阶社交关系网络里是如何传播的。超图在社区发现中的应用：超图分割，就是把超图划分成若干个子集，使同一子集内的节点关联紧密，而不同子集之间的关联较为薄弱，超图嵌入，运用深度学习方法，将超图映射到低维向量空间，以便开展聚类和分类任务。

## 2.2非负矩阵分解（NMF）

### 2.2.1NMF基本原理

（1） 概念简介

非负矩阵分解，它是一种借助线性代数的降维以及特征提取手段，其核心的思想是把一个非负的数据矩阵，分解成为两个或者多个非负矩阵相乘的形式，因为分解之后得到的结果里所有的元素都是非负的，这种方法可产生直观且容易解释的特征表达，在图像处理、文本挖掘以及生物信息学等多个领域都有较为广泛的应用。

（2） 数学模型

非负矩阵分解的目标在于找寻两个低秩且非负的矩阵，也就是矩阵W以及矩阵H，其中k表示降维之后的维度，一般其对应着数据里的潜在特征数量，矩阵W被称作基矩阵，它的列向量可被视作数据里的基本模式，而矩阵H则被叫做系数矩阵，其描述的是原始数据在这些基本模式之下的表示权重。

（3） 优化目标与求解方法

NMF的分解过程一般是借助构建一个目标函数，借助迭代优化算法来进行求解，常见的目标函数有以下这些：

1. 欧几里得距离损失:

其中 表示 Frobenius 范数，该目标函数旨在最小化原始数据矩阵与重构矩阵之间的均方误差。

1. Kullback-Leibler（KL）散度

KL 散度更适合用于概率建模场景，例如文本主题提取。

求解这些目标函数的方法主要包括乘法更新规则（Multiplicative Update Rules）和交替最小二乘法（Alternating Least Squares, ALS）。乘法更新规则通过保持非负性的同时不断迭代更新 𝑊 和 𝐻 的值，直至收敛。

（4） 可解释性与应用优势

NMF分解结果里所有元素都是非负的，它能生成加性组合模型，也就是additive model，这样一来每个数据样本都可看作是若干基本模式的线性组合，提升了模型的可解释性，NMF在降维和特征抽取方面有不错的稀疏性特性，这让在高维数据里提取局部特征成为了可能。

### 2.2.2NMF在社区发现中的应用与局限性

（1） NMF在社区发现中的应用

社区发现会将网络中的节点划分成若干子集，这些子集内部的连接较为紧密，而外部连接则相对稀疏，在这一任务中，网络一般借助邻接矩阵来表示，矩阵的每个元素都体现出节点之间是否存在连接或者连接的权重情况，NMF在社区发现中的基本思路是进行邻接矩阵分解。

假设 为网络的邻接矩阵，其中 表示节点 和 之间的联系强度。利用NMF对 进行分解:

其中, 可以看作节点的社区归属特征矩阵，而则反映了社区之间的关系或表达强度。通过对 的分析，可以将每个节点归类到相应的社区。

NMF的分解结果可让一个节点对多个社区有一定的归属权重，如此这般的“软聚类”能力刚好与现实网络里节点存在可能同时归属于多个社区的状况相契合，比如说，在社交网络里，一个人或许归属于工作圈，又归属于兴趣圈，而这种多重身份依靠 NMF 的系数矩阵 H 可得到不错的表达。

（2） NMF在社区发现中的局限性

尽管NMF在社区发现领域有一定优势，但其仍存在一些局限性，具体表现为以下几点，线性建模存在局限，NMF本质上是一种线性分解方法，它将网络数据看作是若干基本模式的线性组合，然而在许多实际网络中，存在着复杂的非线性关系，仅依靠线性组合难以充分捕捉这些复杂特征。如此一来，在面对非线性结构时，社区划分结果的精度会降低，高阶关系表达不足，传统NMF主要依赖邻接矩阵，这种矩阵只能反映节点之间的直接即一阶连接关系，忽略了节点之间依靠中间节点间接形成的高阶关联，而在现实网络中，社区结构大多由高阶关系驱动，传统NMF在这方面存在欠缺。对噪声敏感，网络数据大多时候存在噪声和异常值，NMF在优化过程中容易受到噪声干扰，导致分解结果不稳定，在稀疏网络中，噪声可能会对重构误差产生较大影响，影响社区划分效果，对初始值有依赖性，NMF的求解过程一般采用迭代更新算法，算法的最终结果在一定程度上依赖于初始值的选择。不同的初始值可能会导致收敛到不同的局部最优解，这在社区发现任务中可能会引入不确定性，影响结果的一致性和可重复性。

（3） 应对策略与改进方向

面对上述局限性，研究者们提出了诸多改进措施，比如引入非线性核函数，运用深度非负矩阵分解即DNMF，结合图正则化以及开展超图建模等，凭借这些方法提升模型对高阶关系和非线性结构的捕捉能力，提高模型的鲁棒性，这些改进措施为优化社区发现算法提供了有力支持，推动了社区发现理论与应用的发展。

## 2.3深度非负矩阵分解（DNMF）

### 2.3.1DNMF的基本结构与优势

深度非负矩阵分解，即DNMF，是传统非负矩阵分解NMF的扩展，它凭借引入多层结构，逐步提取数据中的高层次抽象特征，DNMF能实现有效降维，提取潜在特征，捕捉数据中的非线性和多尺度信息，在社区发现等复杂任务中表现出色。

（1） 基本结构

DNMF借助堆叠多层非负矩阵分解的办法，将原始数据矩阵X分解为一系列低秩非负矩阵的乘积，其常见表示形式为：

在此之中，L表示分解的层数，每一层都负责提取不同层次的特征，是第L层的权重矩阵，其作用是把上一层的输出映射到更低维度。是最终得到的低维表示矩阵，一般用于后续的聚类或者社区划分任务，这种分层结构使DNMF可逐步学习数据的多层次表示，把低级特征逐步组合、抽象成高级语义信息。

（2） DAMNF优势

在多层特征抽象这一领域，DNMF运用多层分解的方式，促使每一层都可针对数据逐步开展抽象工作，提取出更具判别力的特征，这种分层学习机制与深度神经网络的表征学习颇为相似，可更有效地捕捉数据中的潜在非线性关系以及高阶关联，对于揭示社区内部复杂结构有一定的帮助作用。在提升表达能力方面，传统NMF仅仅进行单层线性分解，一般很难表示数据中的复杂模式，而DNMF的多层结构使得模型有更强的表达能力，可将复杂数据表示为多个基本模式的组合，在处理复杂网络数据时呈现出更高的准确性，在鲁棒性提高方面，借助分层提取特征，DNMF可抑制噪声影响，每一层的分解过程都可以被看作是一种降噪操作，使最终得到的低维表示更加稳定且鲁棒，有利于提升社区发现任务中的聚类效果。在自适应学习能力上，DNMF模型可依靠层次结构自动发现数据中的内在模式，无需预先设定复杂的非线性映射函数，这种自适应特性使得模型在面对不同规模和类型的网络数据时，可灵活调整自身结构，获得较好的泛化能力。

### 2.3.2深度自编码器在DNMF中的应用

深度自编码器也就是 Deep Autoencoder，简称为 DAE，它是一种无监督学习模型，主要用于数据降维和特征提取，将深度自编码器与 DNMF 相结合，可提高特征抽象和重构能力，为社区发现提供更精确的低维表示。

（1）自编码器结构简介

深度自编码器主要有两个部分，编码器会把高维的输入数据映射到低维潜在空间，从中提取数据的关键信息，解码器则是将低维表示重新构建回到原始数据空间，保证所提取的特征可完整描述输入数据，自编码器的训练目标一般是使输入数据与重构数据之间的差异最小化，这样的过程可以看作是对数据进行的压缩和去噪操作。

（2）与DNMF的融合方式

在DNMF框架中引入深度自编码器主要有以下几种方式：在初始化与预训练方面，可以借助自编码器对原始数据进行预训练，获得较好的低维表示作为DNMF的初始输入，这种方式可DNMF在起始分解时拥有更合理的初始值，避免随机初始化可能导致的局部最优问题，在联合训练框架层面，将自编码器的编码器和解码器与DNMF模型融合，构建统一的损失函数。在深度分解过程中，编码器部分负责生成低维表示，解码器部分负责将低维表示重新构建回原始数据，联合训练能使编码器和解码器相互起到指导作用，保证低维表示有较高的重构精度，又提高了特征的区分能力，这种联合训练框架可提升模型对数据非线性以及高阶结构的捕捉能力，在去噪与鲁棒性提升方面，自编码器在降噪方面有天然优势，将其引入DNMF可以有效过滤噪声数据，得到更清晰的特征表达，在面对实际网络数据中常见的噪声和异常值时，联合自编码器的DNMF模型可提高社区发现任务的鲁棒性和准确性。

（3）应用效果与优势

在更精细的特征提取层面：深度自编码器有自动学习数据非线性特征的本领，依靠解码器实现高质量重构，当它与DNMF相结合时，该模型可捕捉线性组合模式，还可揭示复杂的非线性关系，可为社区划分提供更为丰富的特征信息，在提高模型稳定性方面：联合训练框架使模型在训练过程中可自适应地调整编码器与解码器之间的平衡，降低因随机初始化而引发的波动，保证分解结果在不同数据集上可稳定发挥作用。在扩展应用领域方面：借助引入深度自编码器，DNMF模型适用于传统的网络邻接矩阵分解，还可以拓展到处理多模态数据、动态网络数据等复杂场景，为社区发现在如社交网络、生物信息学、金融网络等各类应用领域的推广提供了更为广阔的空间。

# 3. 基于超图的深度非负矩阵

## 3.1超图社区网络构建

这一部分重点对传统邻接矩阵在社区网络建模中存在的不足之处展开剖析，探讨如何凭借构建超图邻接矩阵来获取更为丰富的网络结构信息，超图可利用超边将多个节点连接起来，有效呈现高阶关系，为复杂社区结构的分析提供一个更为全面的视角。

### 3.1.1传统邻接矩阵的不足

在社区发现中，传统邻接矩阵 通常用于表示网络中节点之间的直接关系。对于一个无向图 ，邻接矩阵的定义为：

这种表示方式虽然直观，但在实际复杂网络中存在诸多局限性：一阶关系有其局限性：邻接矩阵记录的只是节点之间的直接连接信息，也就是一阶邻居关系，在实际网络中，许多关键的结构特性依赖于多阶或高阶关联，比如节点间的间接关系、群组活动或者共同行为模式等，然而这些信息在传统邻接矩阵里无法直接体现，这使得在进行社区划分时，无法充分利用网络中隐藏的高阶关系。存在信息丢失问题：在不少真实网络中，同一组节点可能凭借多个中介节点形成紧密联系，但是传统邻接矩阵仅反映节点间是否存在直接边，这种二元化表示方式容易丢失丰富的关联信息，例如在社交网络里，两个用户可能没有直接好友关系，但他们共同参与了多个社交活动或兴趣群体，这种高阶关联在邻接矩阵里大多会被忽略。异构关系难以表达：传统邻接矩阵一般只考虑单一类型的关系，无法有效区分不同性质的边，在很多实际应用中，节点间存在多种异构关系，而单一的邻接矩阵难以捕捉这种多维度的交互模式，有噪声敏感性：在稀疏或噪声较多的网络中，传统邻接矩阵可能无法有效区分真正的社群结构和偶然性连接，影响社区发现的精度与鲁棒性。传统邻接矩阵在表示复杂网络中的高阶、多样化信息方面存在不足，需引入更灵活且信息丰富的模型。[15]

### 3.1.2超图邻接矩阵的构建方法

超图模型依靠引入超边将多个节点连接起来，能更全面地捕捉节点间多元交互关系，构造超图邻接矩阵的最关键的是如何定义超边以及如何将其转化为适用于社区分析的矩阵表示形式，下面介绍几种常见的超图邻接矩阵构建方法， 超图模型借助引入超边把多个节点连接起来的这种方式，使得它可更加全面地去捕捉节点之间的多元交互关系。 而构造超图邻接矩阵时，关键要点在于怎样去定义超边以及怎样把它转化为适合社区分析的矩阵表示形式， 随后会介绍几种常见的超图邻接矩阵构建方法。

（1） 基于群组信息构建超边

定义超边：如果网络中存在明显的群组或聚集现象，例如社交网络中的聊天群组、共同参与某项活动的用户集，则可以直接将每个群组视为一个超边。构造方法：对于每个群组 ，定义超图邻接矩阵 的元素为：

其中 表示超边 的权重，可以基于群组规模、活跃度或其他先验信息进行设定。这样一来，两个节点如果同时属于多个群组，其对应的超图邻接矩阵元素将更大，反映了更强的高阶关联。

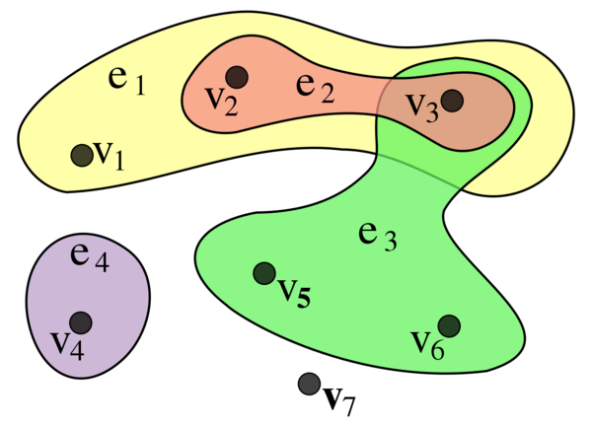


图 4 超图概念图

（2） 基于相似性或共现关系构建超边

超边的定义：在没有事先设定群组信息时，可依据节点之间的相似程度以及共现的频次来构建超边，比如在文本领域或者社交网络环境中，如果多个用户频繁地同时出现在同一篇文章里，或者是同一事件当中，那么就可以把这些用户组合成一个超边，超边的构造方法是：凭借共现矩阵或者相似性度量手段，将相似度较高的节点聚合在一起，形成超边。常见的方法有基于阈值的聚类方式，或者运用K近邻方法来构建超边，假设S是节点相似性矩阵，当超过所设定的阈值时，就把i和j归为同一个超边，之后凭借合并操作来构建完整的超边集合。[16]

（3） 基于属性信息融合构建超边

超边的定义是这样的：针对有多模态或者多属性的网络来讲，节点除了有结构方面的信息之外，还会包含如文本、图像这类的属性信息，超边的构造方式为：借助节点的属性信息开展相似性聚类，将属性相似的节点归到同一个超边之中，在具体实现时，可以先利用深度学习模型提取节点的低维属性表示，随后运用聚类算法把节点划分成若干组，每一组都作为一个超边。

（4） 权重设计与矩阵表示

在超边权重这一环节，对于每一个构建好的超边来说，一般要设计一个权重函数 w(e)，借助这个权重函数，可呈现出该超边在网络中的关键程度，权重的设定可以依据超边中的节点数量、节点之间的平均相似度，或者参考其他领域的相关知识来完成，而在邻接矩阵构建部分，凭借汇总所有超边的信息，构造出超图邻接矩阵 。常见的构造公式为：

其中，归一化因子 考虑了超边中节点数目的影响，使得较大超边不会因为节点数多而导致权重失衡。

## 3.2DNMF模型改进设计

为提升深度非负矩阵分解在社区发现任务中的性能，此部分提出一系列改进措施，具体包括引入超图正则化、融合深度自编码器，同时设计优化与约束机制，这些改进可充分利用网络中的高阶结构信息，提高特征提取的非线性表达能力，依靠合理约束提升模型稳定性与聚类效果。

### 3.2.1超图正则化的引入

（1） 动机与目标

传统的DNMF主要依赖邻接矩阵，一般情况下它只能捕捉一阶或二阶关系，难以全面呈现节点间高阶以及多元的交互关系，针对这一问题，引入超图正则化，借助超边可构建更为丰富的邻接表示，捕获多个节点间的联合关系，提高社区发现的精度与鲁棒性。

（2） 方法描述

在模型中，通过构造超图邻接矩阵 来反映高阶关系，然后将超图正则化项加入到DNMF的目标函数中。常见的正则化项可以基于超图拉普拉斯矩阵 构造，其形式为：

其中， 为深度分解得到的社区指示矩阵， 是正则化参数，调控超图信息对整体优化目标的影响。该项旨在使得在同一超边内的节点具有相似的低维表示，从而更好地保留群体间的高阶结构信息。

（3） 优势分析

超图正则化有捕捉高阶关系的本事，它可直接呈现多个节点之间的共现关系，切实弥补传统邻接矩阵所呈现出的一阶局限性，超图正则化还可提升聚类质量，它凭借约束低维表示在超图结构下的平滑性，使得同一超边内的节点拥有更为一致的特征，获取更加紧凑且分离明确的社区，超图正则化也可提高鲁棒性，在噪声数据或稀疏网络环境中，超图结构可提供更为稳定的多节点关联信息，借此提高模型对异常值的抗干扰能力。

### 3.2.2深度自编码器与DNMF的结合

（1） 动机与目标

传统非负矩阵因子分解方法依靠多层分解能捕捉数据多尺度特征，不过在数据重构和非线性关系表达上存在一定局限，深度自编码器是一种功能强大的非线性降维工具，它可利用编码 - 解码结构提取表达能力更强的低维表示，将深度自编码器与非负矩阵因子分解相结合，可提升特征抽象能力以及数据重构精度。

（2） 融合方式

在预训练与初始化环节，最初会借助深度自编码器对原始数据实施预训练操作，得到低维特征表示，将此低维特征表示作为DNMF分解的初始值，可降低随机初始化对收敛性造成的影响，在联合训练框架方面，会把自编码器的编码器部分与DNMF的分解过程相融合，构建出统一的损失函数。该损失函数囊括DNMF的重构误差，又包含自编码器的重构损失。

其中， 为自编码器重构输出， 表示自编码器参数， 是平衡两部分损失的权重。特征对齐与协同优化：在联合训练过程中，编码器生成的低维表示与DNMF得到的特征保持一致，使得网络结构与数据属性信息得到共同优化，进一步提高社区划分的精度。

（3） 优势分析

深度自编码器可捕捉数据当中的非线性特征，达成非线性映射的提升，这与DNMF的线性分解形成互补关系，提升整体的特征表达能力，自编码器有良好的去噪效果以及去噪能力，在联合训练时可降低噪声对分解结果所产生的干扰，预训练和联合训练机制可模型取得更好的初始化与收敛效果，提高稳定性，使得最终的低维表示变得更加稳定且鲁棒。

## 3.3模型算法的提出与优化求解

### 3.3.1模型的提出

HDNMF算法的核心想法是将网络的邻接矩阵分解成两个非负矩阵的乘积，其中一个是{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>m</mi><mo>&#xD7;</mo><mi>k</mi></mstyle></math>"}的非负矩阵，用来表示节点的特征向量，另一个矩阵是{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>k</mi><mo>&#xD7;</mo><mi>n</mi></mstyle></math>"}的非负矩阵，用于表示社区结构，NMF是要让原始数据矩阵V和分解后的矩阵W、H之间的误差达到最小，为提升算法精确程度，HDNMF算法采用多层非负矩阵分解形式，类似深度自编码器架构。假设有L个隐藏层，第个{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}隐藏层的输出是{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>H</mi><mrow><mo>(</mo><mi>l</mi><mo>)</mo></mrow></msup></mstyle></math>"}，那么第{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi><mo>+</mo><mn>1</mn></mstyle></math>"}个隐藏层的输入就是{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>H</mi><mrow><mo>(</mo><mi>l</mi><mo>)</mo></mrow></msup></mstyle></math>"}，输出为{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>H</mi><mrow><mo>(</mo><mi>l</mi><mo>+</mo><mn>1</mn><mo>)</mo></mrow></msup></mstyle></math>"}，最终输出是{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi>H</mi><mrow><mo>(</mo><mi>L</mi><mo>+</mo><mn>1</mn><mo>)</mo></mrow></msup></mstyle></math>"}，也就是最终的特征表示，HDNMF是要使原始数据矩阵V和最终的特征表示HL之间的误差最小化，即：

相应的权重矩阵以及特征矩阵会分别被应用于每一个隐藏层，上述所呈现的内容就是传统深度非负矩阵分解的计算流程，这个流程采用非负矩阵分解的办法，把原始网络信息一层一层地分解成特征矩阵，不过在这个过程当中，缺少了把隐含层信息重新构建回原始社区网络的环节，需要对隐含层的相关信息给予重构，实现对原始邻接矩阵的重构目标。假设已经得到了第{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}层的节点特征矩阵{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi mathvariant=\"bold\">H</mi><mrow><mo>(</mo><mi>l</mi><mo>)</mo></mrow></msup></mstyle></math>"}和社区结构矩阵{"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msup><mi mathvariant=\"bold\">W</mi><mrow><mo>(</mo><mi>l</mi><mo>)</mo></mrow></msup></math>"}，借助于{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>H</mi><mo>=</mo><msup><mi>W</mi><mi>T</mi></msup><mi>V</mi></mstyle></math>"}，那么可以得到：

其中，{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi mathvariant=\"bold\">H</mi><mrow><mo>(</mo><mi>L</mi><mo>)</mo></mrow></msup></mstyle></math>"}和{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi mathvariant=\"bold\">W</mi><mrow><mo>(</mo><mi>L</mi><mo>-</mo><mn>1</mn><mo>)</mo></mrow></msup><mo>,</mo><mo>&#x2026;</mo><mo>,</mo><msup><mi mathvariant=\"bold\">W</mi><mrow><mo>(</mo><mi>l</mi><mo>)</mo></mrow></msup></mstyle></math>"}分别表示第{"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><mi>L</mi></math>"}层到第{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}层的节点特征矩阵和社区结构矩阵。因此，可以将隐含层信息重构回去的公式表示为：

其中，{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>A</mi></mstyle></math>"}表示重构后的邻接矩阵，{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><msup><mi mathvariant=\"bold\">H</mi><mrow><mo>(</mo><mi>L</mi><mo>)</mo></mrow></msup></mstyle></math>"}和{"mathml":"<math xmlns=\"http://www.w3.org/1998/Math/MathML\" style=\"font-family:stix;font-size:16px;\"><msup><mi mathvariant=\"bold\">W</mi><mrow><mo>(</mo><mi>L</mi><mo>-</mo><mn>1</mn><mo>)</mo></mrow></msup><mo>,</mo><mo>&#x2026;</mo><mo>,</mo><msup><mi mathvariant=\"bold\">W</mi><mrow><mo>(</mo><mn>1</mn><mo>)</mo></mrow></msup></math>"}分别表示第{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>L</mi></mstyle></math>"}层到第{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>l</mi></mstyle></math>"}层的节点特征矩阵和社区结构矩阵。

将矩阵分解和矩阵重构过程组合起来，可以得到HDNMF算法的完整公式：

同时结合第一点中引入的超图带入公式得：

传统NMF算法由于添加了非负约束，在分解重构过程中，由分解得到的矩阵只能借助线性相加来恢复原矩阵，这使得其解不可避免地存在一定稀疏性，不过稀疏性并非该算法最初的设计目标，且无法对稀疏程度进行控制，借助增添一些额外约束，可有效提升结果的稀疏度，甚至可以实现稀疏可控。聚类指示矩阵有正交性，在对应解的矩阵上施加正交性约束，可产生更为理想的聚类指示矩阵，一般正交约束分为两种。

在对{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>H</mi></mstyle></math>"}矩阵实施正交性约束时有：

最终得到的目标方程为：

其中{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>I</mi></mstyle></math>"}为单位阵，其中，{"mathml":"<math style=\"font-family:stix;font-size:16px;\" xmlns=\"http://www.w3.org/1998/Math/MathML\"><mstyle mathsize=\"16px\"><mi>&#x3BB;</mi></mstyle></math>"}是正则化强度。

### 3.3.2模型优化求解

本节优化求解模型的各个矩阵所对应的更新公式

迭代求解:聚焦在 W(i) 的更新上。

定义的中间变量设：

那么：

同样，在自编码项里：

于是：

对求梯度有：

对 的梯度：

展开：

对 求梯度：

对 的梯度：

展开：

对 求导

因此，乘法更新可以写为：

有，则：

优化求解更新公式：

第一项：

设：

对求偏导：

第二项：

求导：

稀疏项：

度方向是 α，在乘法更新里处理为 α项。

正交项：

这个项对 的导数是（设）：

超图正则项：

根据 Laplacian 正则的推导：

总梯度：

乘法更新式

使用 Lee & Seung 的 NMF 经典乘法更新法，构造形式：

同理更新公式为

其中：，。

# 4. 实验与结果分析

本章实验通过多组对比验证所提出的基于超图正则化的深度非负矩阵分解（HDNMF）模型在社区发现任务中的有效性，涵盖数据集选择、预处理、实验设置、评价指标、模型训练、参数调优，并对实验结果进行讨论与对比分析，具体内容如下所述。

## 4.1实验准备

### 4.1.1数据集选择与预处理

（1） 数据集选择

为充分评估模型的性能和泛化能力，实验中选用了多个具有代表性的真实网络数据集，这些数据集涵盖了不同领域和规模：

社交网络数据集：Email-Eu-core，反映用户之间的社交关系和交互模式，具有明显的社区结构特征。边数据:每一行表示两个用户之间的一次邮件交流，格式为：node1 node2为无向图（边是互相的邮件联系）共包含 1005 个节点 和 25571 条边。标签数据：表示该节点所属的部门，反映真实的社区结构，部门数量约为 42 类（即社区）。引文网络数据集：Cora，收录自学术论文引用网络（Cora 学术论文分类子集）2708 个节点，每个节点代表一篇机器学习相关的论文，5429 条边，每条边表示一篇论文引用另一篇论文，每个节点有一个分类标签，共 7 类。数据集选择主要考虑规模覆盖小型与大型复杂网络，以测试模型扩展性和鲁棒性；结构特性涵盖不同社区密度、网络稀疏度及高阶关系，验证模型适用性；涉及多领域数据，检验模型在跨领域社区发现任务中的泛化能力。具体数据集与数据特征信息见表1。

表 2 网络数据集与信息特征

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 节点 | 边 | 社区数 | 社区结构 |
| Email | 1005 | 25571 | 42 | 每个用户属于一个具体的部门 |
| Cora | 2708 | 5429 | 7 | 每个节点有一个分类标签 |
| Wiki | 2405 | 17981 | 19 | 每个页面有一个主题标签 |
| Citeseer | 3312 | 4732 | 6 | 每个页面有一个主题标签 |
| Pubmed | 19717 | 44338 | 3 | 每个页面有一个主题标签 |

（2） 数据预处理

本研究选用经典图结构数据集 Cora 作为实验数据。Cora 为计算机科学领域的引文网络数据集，节点表示论文，边表示引用关系，节点具有稀疏词袋特征并带有七类主题标签，如神经网络、强化学习等。Cora 数据集由 cora.content 和 cora.cites 两部分构成，前者包含论文编号、1433维词向量和类别标签，后者记录论文间的引用关系。为适配图挖掘任务，将原始数据处理为结构化输入：利用 NetworkX 读取 cora.cites 构建无向图，节点对应论文，边对应引用；从 cora.content 提取节点特征和类别，形成 node\_features（N × 1433）和 node\_labels（N × 1）；将论文编号映射为连续整数索引，适配 Numpy 与 PyTorch；最终保存为边列表和 .npy 格式，便于模型训练与评估，其他数据集采用类似处理方式。

### 4.1.2实验设置与评价指标

（1） 实验设置

为全面评估模型在社区发现任务中的表现，实验设置囊括以下内容：对比方法选择多种社区发现算法作为基准，借助对比验证所提模型在精度上的优势，模型参数设定有超图正则化系数λ、自编码器权重α、稀疏性约束系数β以及正交约束系数γ，利用网格搜索和交叉验证确定最优参数，并分析参数敏感性，训练策略采用预训练、联合训练以及交替最小化，设置合适的迭代次数与收敛阈值，保证在复杂网络数据上模型能稳定收敛。

基于NMF的浅层模型包括：

NMF：NMF是所提出DANMF模型的基础组件，已在文献[21]中应用于社区发现。

ONMF：ONMF是在映射矩阵U上施加正交约束（即UTU=I）的NMF变体[27]。

PNMF：PNMF通过最小化‖A−UUTA‖²\_F，直接将原始网络投影到子空间[30]。

BNMF：BNMF是一种贝叶斯NMF模型，已在文献[28]中应用于社区发现。

BigClam：BigClam是一种簇隶属模型，将图拟合问题放宽为连续优化问题[26]。

HNMF：HNMF是一种概率方法，建模边与社区之间的同质关系以实现社区发现[25]。

NSED：NSED是一种非负对称编码器-解码器方法，用于社区发现。虽然包含编码器组件，但其从映射矩阵U而非特征矩阵V中提取社区成员信息[24]。

网络嵌入方法包括：

LINE：LINE保持节点之间的一阶和二阶相似性，以学习节点的低维表示[23]。

Node2Vec：Node2Vec通过截断随机游走学习节点间的高阶相似性[22]，其中in-out超参数固定为2，以更好捕捉网络的社区结构。

（2） 评价指标

在社区发现任务中，评价指标主要用于衡量模型划分社区的准确性、稳定性与可解释性。常用评价指标包括：

**归一化互信息（Normalized Mutual Information，NMI）：**衡量算法划分结果与真实社区标签之间的一致程度，取值范围为0到1，数值越接近1表示划分与真实标签越一致。其定义公式如下：

其中：：表示聚类结果 与真实标签 的互信息；：表示各自的熵。

优点：对类别数量不一致有一定的鲁棒性。

**调整后的兰德指数（Adjusted Rand Index (ARI)）：**考虑了所有样本对之间的一致性，调整了随机情况下的聚类结果偏差。其值在 [-1, 1] 之间，越接近 1 越好。

其中 RI 是原始的 Rand Index，衡量的是样本对的一致性。

优点：考虑随机期望，使得对随机聚类的惩罚更加合理。

**聚类准确率（Clustering Accuracy, ACC）:** 表示通过最佳映射后，聚类结果与真实标签之间的匹配程度，其值在 [0, 1] 之间。由于聚类标签可能是任意顺序，ACC 需要通过匈牙利算法对聚类标签与真实标签做最优匹配，计算匹配后的准确率：

表 3 模型效果评价指标含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指标 | 范围 | 是否考虑标签顺序 | 是否对随机聚类有惩罚 | 说明 |
| NMI | [0, 1] | 否 | 否 | 信息论角度的衡量 |
| ARI | [-1, 1] | 否 | 是 | 样本对一致性衡量 |
| ACC | [0, 1] | 是 | 否 | 匹配后准确率 |

## 4.2模型实现与参数调优

本节详细描述了超图深度非负矩阵分解（HDNMF）模型的实现过程及其参数调优策略。通过对模型训练过程的细化描述以及对关键超参数的选择，旨在提高模型在社区发现任务中的准确性和鲁棒性。

### 4.2.1超图深度非负矩阵分解模型训练过程

（1）模型实现框架

**代码实现中模块化设计**：包括数据预处理、超图构建、DNMF层堆叠、损失函数计算及梯度更新等。数据预处理部分将网络数据集合转化为便于建模训练的边集数据格式，构造传统邻接矩阵及基于预定义策略构建的超图邻接矩阵 。模型架构：模型由多个非负矩阵分解层组成，每层对应一个权重矩阵 和中间表示，最终输出低维表示 。同时，引入超图正则化、稀疏性与正交约束，通过联合训练自编码器进一步优化低维表示。

（2） 训练过程

**预训练阶段**：使用浅层的矩阵分解算法对输入数据进行预训练，获得初步低维特征表示，并作为HDNMF各层矩阵 与 的初始值。预训练可以降低随机初始化带来的不稳定性，改善模型收敛速度。**交替最小化与联合优化**：训练过程采用交替最小化策略，将模型参数分为三个部分：**HDNMF分解参数** 、**自编码器参数** 、**正则化与约束项参数、**在每个训练周期内，先固定 更新 及自编码器参数，再固定 更新 ，同时计算超图正则化、稀疏和正交约束项的梯度，通过梯度下降更新各参数。

表 4 模型训练参数设置

|  |  |  |
| --- | --- | --- |
| Dataset | 节点数 | 网络层配置 |
| Email | 1,005 | 1005 → 256 → 128 → 42 |
| Wiki | 2,405 | 2405 → 256 → 128 → 19 |
| Cora | 2,708 | 2708 → 256 → 64 → 7 |
| Citeseer | 3,312 | 3312 → 256 → 64 → 6 |
| Pubmed | 19,717 | 19717 → 512 → 64 → 3 |

**损失函数的计算与迭代更新：**综合目标函数包括重构误差、自编码器重构损失、超图正则化项及约束项：

每次迭代中，更新后的损失值将用于监控模型收敛性，当损失下降至预设阈值或达到最大迭代次数时训练结束。

模型训练过程如下：

### 4.2.2参数选择与调优

（1） 超参数种类

模型中涉及的主要超参数包括：

超图正则化系数 :控制超图正则化项对整体损失函数的贡献，影响高阶关系在低维表示中的保留效果。稀疏约束系数 :调控低维表示 的稀疏性，促进特征选择，减少噪声干扰。正交约束系数 :保证聚类指示矩阵 具有较好的正交性，从而提高不同社区之间的区分度。

（2） 参数调优策略

网格搜索与交叉验证：通过网格搜索调试各超参数，选取使模型性能最优的参数组合。敏感性分析：评估不同参数对模型训练中损失值变化及社区划分结果的影响，识别对性能影响显著的关键参数，进行细粒度优化。动态调整策略：对学习率、正则化系数等采用动态调整方法，训练初期使用较大学习率加快收敛，后期逐步降低以精细优化，确保训练过程稳定。实验对比与消融实验：在有无各正则项（超图正则、稀疏性、正交约束）条件下开展实验，评估各约束对模型性能的提升，明确参数设置的合理区间。

（3） 参数调优的实验评估

评价指标：通过性能指标评估参数调优效果，结合运行时间、迭代次数等计算复杂度指标，综合判断模型在不同参数设置下的表现。实验结果记录：记录并对比不同参数设置下的实验结果，绘制参数影响曲线，直观展示参数调优对模型性能的影响趋势，为最终参数确定提供依据。

## 4.3实验结果与讨论

### 4.3.1模型性能对比分析

表 5 效果评估ARI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Email | Wiki | Cora | Citeseer | Pubmed |
| NMF | 0.4989 | 0.1195 | 0.2145 | 0.0590 | 0.0978 |
| ONMF | 0.4832 | 0.1233 | 0.1964 | 0.0825 | 0.1589 |
| PNMF | 0.4641 | 0.1151 | 0.1863 | 0.0801 | 0.0967 |
| BNMF | 0.3545 | **0.1705** | 0.1812 | 0.0838 | 0.0872 |
| BigClam | 0.2478 | 0.0217 | 0.0306 | 0.0283 | 0.0258 |
| HNMF | 0.2079 | 0.1448 | 0.1113 | 0.0262 | 0.0360 |
| NSED | 0.5215 | 0.1253 | 0.1782 | 0.0866 | 0.1258 |
| LINE | 0.3325 | 0.1344 | 0.1271 | 0.0278 | 0.1017 |
| Node2Vec | 0.4195 | 0.1621 | 0.1063 | 0.0182 | 0.0170 |
| MNMF | 0.0041 | 0.0016 | 0.0002 | 0.0007 | 0.0001 |
| DNMF | 0.5256 | 0.1341 | 0.2452 | 0.0990 | 0.1185 |
| **HDANMF** | **0.5625** | 0.1628 | **0.3194** | **0.1343** | **0.2563** |

表 6 效果评估NMI

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Email | Wiki | Cora | Citeseer | Pubmed |
| NMF | 0.6751 | 0.2673 | 0.2851 | 0.1319 | 0.1606 |
| ONMF | 0.6734 | 0.2607 | 0.2416 | 0.1423 | 0.1582 |
| PNMF | 0.6770 | 0.2684 | 0.2893 | 0.1355 | 0.1511 |
| BNMF | 0.5960 | 0.2903 | 0.2521 | 0.0835 | 0.0714 |
| BigClam | 0.5796 | 0.2722 | 0.1864 | 0.0735 | 0.0291 |
| HNMF | 0.5146 | 0.2959 | 0.1425 | 0.0312 | 0.0311 |
| NSED | 0.6845 | 0.2659 | 0.2928 | 0.1492 | 0.1729 |
| LINE | 0.6393 | 0.2772 | 0.2376 | 0.0573 | 0.1357 |
| Node2Vec | 0.6784 | 0.3331 | 0.1978 | 0.0486 | 0.0635 |
| MNMF | 0.2138 | 0.0274 | 0.0035 | 0.0031 | 0.0002 |
| DNMF | 0.6850 | 0.2798 | 0.3572 | 0.1582 | 0.1709 |
| **HDANMF** | **0.7834** | **0.3406** | **0.4114** | **0.1831** | **0.2221** |

表 7 效果评估ACC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Email | Wiki | Cora | Citeseer | Pubmed |
| NMF | 0.5851 | 0.3027 | 0.4103 | 0.3074 | 0.5133 |
| ONMF | 0.5761 | 0.3069 | 0.3811 | 0.3330 | 0.5575 |
| PNMF | 0.5791 | 0.3052 | 0.4029 | 0.3451 | 0.5073 |
| BNMF | 0.4299 | 0.3751 | 0.4191 | 0.3324 | 0.5110 |
| BigClam | 0.4768 | 0.2545 | 0.3781 | 0.3046 | 0.3978 |
| HNMF | 0.3463 | 0.3518 | 0.3903 | 0.2569 | 0.4128 |
| NSED | 0.6179 | 0.2981 | 0.4234 | 0.3448 | 0.5201 |
| LINE | 0.4657 | 0.3289 | 0.4044 | 0.3019 | 0.4990 |
| Node2Vec | 0.5244 | 0.3568 | 0.3674 | 0.2521 | 0.4067 |
| MNMF | 0.1075 | 0.0886 | 0.1647 | 0.1890 | 0.3397 |
| DNMF | 0.6199 | 0.3543 | 0.4849 | 0.3635 | 0.5389 |
| **HDANMF** | **0.6368** | **0.4112** | **0.5499** | **0.4242** | **0.6393** |

由此说明本模型在不同类型网络数据中具有稳定且显著的性能优势，能够有效提升社区发现的精度与一致性。具体来看，HDANMF在Cora、Citeseer和Pubmed等典型学术引文网络上以及社区结构网络信息Email，相较其他方法在ARI、NMI和ACC三项指标中均实现了明显提升，尤其在Pubmed数据集上各项指标均领先，展现了良好的扩展性和鲁棒性。虽然在Wiki数据集上排名略逊，但仍保持与最优方法的接近水平，验证了模型在异构网络环境下的强大适应能力。这些结果充分证明了所提模型在不同规模、结构及领域数据中的泛化潜力与实用价值。

以Email数据集为例，介绍模型训练过程。训练的损失函数如图6所示，Recon Loss1表示重构误差项1，对应矩阵V分解为W与H的重构误差；Recon Loss2表示重构误差项2，对应矩阵W与H反向重构回到矩阵的误差；Reg Loss表示超图正则化误差。结合实验表现，随着迭代次数增加，三项损失均趋于收敛。整体目标函数持续减小，Recon Loss1与Recon Loss2逐步降低，说明矩阵重构效果不断提升。同时，图正则约束在训练中被逐步突破，原因在于模型仅设置了整体目标函数的损失下降，对图正则未做强制约束。训练过程强调重构误差的最小化，图正则作为辅助项，允许其损失上升，以换取整体重构性能的优化。为了便于观察模型效果，使用 t-SNE 将模型训练后得到的最终嵌入向量降至二维空间，并利用真实标签数据为点着色，生成了图7（a）所示的效果图。从图中可以看到，经典社区结构数据集 Email 在社区聚类后的表现呈现团状结构，整体类似于烟花绽放的分布形态；而经典论文引用数据集 Cora 则表现为每个类别呈条带状分布，如图（b），整体看起来像水母的形态。有趣的是，不同类型的网络结构在聚类表现中呈现出不同的类形状与整体分布。例如，Email 数据集代表的是人的人际关系，其聚类结果多为团状分布，反映了团体之间紧密的联系；而 Cora 数据集则是基于论文引用的网络结构，论文之间的引用关系按照研究主题的逐步推进展开，因此聚类结果呈现条带状结构，可能揭示了论文研究之间的逻辑关系。

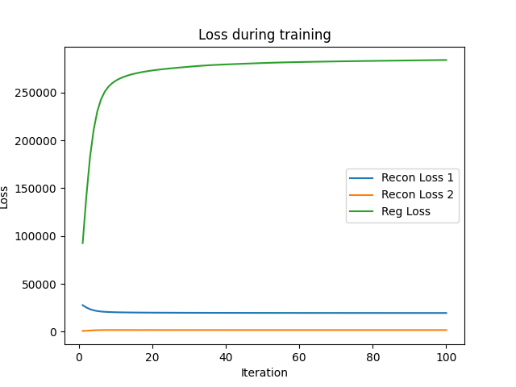


图 5 损失函数曲线

|  |  |
| --- | --- |
| （a）Email | （b）cora |

图 6 聚类效果图

### 4.3.2约束机制的影响分析

模型参数设置以 NMI 指标为纵坐标，超参数 在 1 至 0.001 的三个数量级中进行调节，观察超参数变化对模型性能的影响。绘制的结果如图 8 所示。由实验可见，各超参数对模型的影响趋势不同。整体来看， 在 0.01 附近时模型表现最优， 在 0.01 时也能取得较好的聚类效果，而 在 0.1 附近模型性能最为突出。这表明 和 对模型重构精度与超图结构的影响较为敏感，合理的正则化强度有助于平衡重构误差与结构保持，而 的变化主要影响模型稀疏性和信息压缩程度，适度的 值能够提升模型的泛化能力。值得注意的是，当超参数过大或过小（例如达到 1 或 0.001）时，模型效果普遍下降，表明过强或过弱的正则化都会削弱模型对数据内在结构的刻画能力。因此，在实际应用中应优先在中等强度的参数范围内进行细粒度调优，以确保模型在不同数据集上均具备良好的泛化性和稳定性。

|  |  |  |
| --- | --- | --- |
| （a）Lambda | （b）sigma | （c）alpha |

图 7 超参数设置对模型结果的影响

为了评估模型里各个正则项对性能产生的具体影响，设计了八组有不同设置的模型来开展对比实验，这些模型包含了所有组合的正则项配置，所提出的模型是由三项正则项组成的，分别是稀疏性约束项，其中包含超参数、正交约束项以及超图正则项，在消融实验当中，借助依次把其中一个、两个或者全部正则项的超参数关闭，将其设定为零，构建出对应的模型变体，以此来观察每一项正则项对模型性能的独立作用以及组合贡献。

具体设置如下：

表 8 消融实验设置

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **设置名称** | **α** | **σ** | **λ** |
| ① | 完整模型 | √ | √ | √ |
| ② | α = 0 | ✕ | √ | √ |
| ③ | σ = 0 | √ | ✕ | √ |
| ④ | λ = 0 | √ | √ | ✕ |
| ⑤ | α = σ = 0 | ✕ | ✕ | √ |
| ⑥ | α = λ = 0 | ✕ | √ | ✕ |
| ⑦ | σ = λ = 0 | √ | ✕ | ✕ |
| ⑧ | 全部为0（基础模型） | ✕ | ✕ | ✕ |

实验结果如下所示：

表 9 消融实验结果（NMI）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **数据集** | **完整模型** | **α=0** | **σ=0** | **λ=0** | **α=σ=0** | **α=λ=0** | **σ=λ=0** | **全部为0** |
| **Email** | 0.7834 | 0.5631 | 0.5456 | 0.5302 | 0.4208 | 0.4059 | 0.4934 | 0.4001 |
| **Wiki** | 0.3406 | 0.2679 | 0.2567 | 0.2681 | 0.2585 | 0.2364 | 0.2403 | 0.2250 |
| **Cora** | 0.4114 | 0.3480 | 0.3575 | 0.3486 | 0.3122 | 0.3211 | 0.3225 | 0.2584 |
| **Citeseer** | 0.1830 | 0.1451 | 0.1463 | 0.1389 | 0.1210 | 0.1346 | 0.1292 | 0.1187 |
| **Pubmed** | 0.2221 | 0.1938 | 0.1945 | 0.1960 | 0.1569 | 0.1492 | 0.1514 | 0.1201 |

表 9 呈现出不同正则项配置情形下模型于五个数据集上的 NMI 表现，完整模型在各个数据集里都获取到了最优的结果，这意味着稀疏性、正交性以及超图正则这三项约束共同促使模型的聚类性能得到了提升，当去除任意一项正则项之后，性能均出现了下降，其中去除正交约束以及超图正则所产生的影响相对较为明显，它们在维持嵌入结构的判别性与一致性方面发挥着更为关键的作用。在进行双项正则移除的时候，模型性能下滑，而当三项全部去除时效果是最差的，这验证了正则项对于模型有效性以及鲁棒性所起到的关键作用。

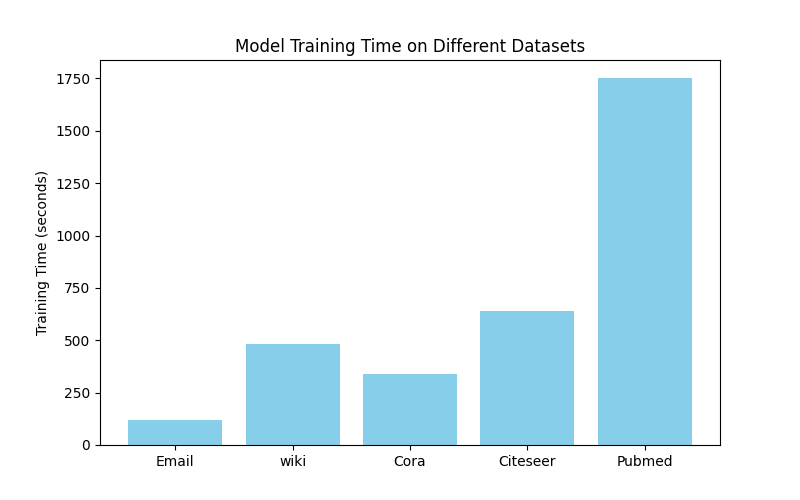


图 8 模型训练时间分析

可以看到，随着数据集规模的增大，模型训练时间呈明显上升趋势。Email数据集规模较小，训练时间最短，仅约 2 分钟；Wiki和Cora数据集的训练时间在 5 到 8 分钟之间，属于中等规模；而Citeseer的训练时间达到约 10 分钟，说明其数据复杂度或节点数更多。Pubmed作为大规模数据集，训练时间超过 29 分钟，明显高于其他数据集。这表明模型的计算复杂度随数据集规模的扩大而线性甚至超线性增长，提示在大规模图数据上模型的可扩展性和训练效率是需要关注的重要问题。整体来看，模型在中小规模数据集上的训练效率尚可，面对大规模数据时建议结合批处理或分布式训练策略进一步优化性能。

# 5. 结论与展望

## 5.1研究总结

本研究围绕社区发现问题，提出了一种基于超图正则化的深度非负矩阵分解（HDNMF）模型。通过充分利用超图在捕捉高阶关系方面的优势，以及深度非负矩阵分解和深度自编码器的融合，本研究在社区检测任务中取得了较好的性能和解释性。下面对本文的主要贡献、创新点以及研究过程中遇到的挑战与解决方案进行总结。

### 5.1.1本文主要贡献与创新点

超图正则化的引入:传统邻接矩阵只能捕捉节点之间的一阶直接关系，而本研究通过构建超图邻接矩阵，有效挖掘节点之间的高阶、多元交互关系。超图正则化项被引入到模型的目标函数中，使得在低维表示中，同一超边内的节点能够保持更一致的特征，从而提高社区划分的精度和鲁棒性。深度非负矩阵分解与自编码器的融合本文提出的HDNMF模型通过多层分解逐步抽象数据中的潜在特征，较传统NMF具有更高的表达能力。

将深度自编码器相互结合，针对输入数据实施预训练以及联合优化，如此一来提升了非线性特征提取能力，依靠编码器和解码器协同训练，数据重构效果得以改善，为社区发现提供了更为精细的低维表示，在优化与约束机制设计方面，为提高模型稳定性和泛化能力，研究中引入了L1稀疏约束以及正交约束，有效减少了冗余信息以及社区间重叠问题，使模型在社区划分上更具辨识力。结合自适应学习率、动量策略以及交替最小化等优化策略，保证了模型在大规模网络数据上能高效收敛，且有良好的训练稳定性，在实验验证与多领域应用方面，借助在社交网络、引文网络以及生物网络等多个领域数据集上开展实验，验证了所提模型在捕捉高阶关系、提高社区检测精度以及处理噪声数据方面有优势。详细的消融实验以及参数敏感性分析说明超图正则化、深度结构以及联合训练机制在提升模型性能中有关键作用。

### 5.1.2研究过程中的挑战与解决方案

高阶关系的建模面临的挑战是，传统邻接矩阵仅能捕捉直接边信息，对于揭示复杂网络里节点间的间接以及多元交互关系存在险阻，解决办法是采用超图模型构建超图邻接矩阵，借助引入超图正则化项，让模型可充分运用高阶关系信息，以此提升社区划分效果，非线性特征提取与数据重构方面的挑战是，传统NMF以及浅层分解方法难以捕捉数据中的非线性特征，致使社区检测精度受限。解决办法是将深度非负矩阵分解与深度自编码器相结合，在模型里实现多层非负分解以及非线性映射，这样提高了特征抽象能力，又借助联合训练改进了数据重构效果，提高了模型在复杂网络上的表现，参数调优与模型收敛的挑战是，模型中涉及多项超参数，像超图正则化、稀疏性与正交约束系数等，这些参数的选择对模型的稳定性和社区划分质量有着直接影响。解决办法是运用网格搜索和交叉验证等方法对超参数进行细致调优，同时引入自适应学习率、动量策略以及交替最小化更新机制，保证模型在训练过程中可快速稳定地收敛，避免出现局部最优问题。

## 5.2未来研究方向

根据当前已有的研究成果，后续工作可围绕模型自身优化拓展、超图与其他先进技术相互融合以及社区发现在实际场景中的推广应用展开，接下来将详细分析各方向的研究内容及可能存在的发展前景。

### 5.2.1模型优化与扩展

在模型结构改进方面，多尺度分解的工作是去设计有多尺度以及多层次特性的分解架构，这样能让模型同时捕捉局部和全局的社区结构，未来可以探索基于分层结构的动态调整机制，实现自适应层数以及参数的选择，非线性激活机制方面，引入非线性激活函数或者核方法，提升模型对于复杂非线性关系的表达能力，呈现现实网络里的隐含特征。在优化算法研究中，自适应优化器的开发要更加高效且鲁棒，比如结合Adam、RMSprop以及动量机制，针对大规模网络数据进行参数更新，提高模型的收敛速度以及全局最优概率，对于分布式与并行计算，针对大规模网络环境，探索基于分布式计算框架的模型训练策略，降低计算复杂度以及内存消耗。在扩展模型应用范围时，异构网络处理要扩展模型使其可处理有多种类型节点和边的异构网络，借助设计多模态特征融合模块，实现对不同信息源的统一建模，动态网络建模方面，针对实时变化的网络环境，设计动态社区发现模型，捕捉社区演化过程中的增减、合并或者分裂现象，为时间序列数据分析提供支持。

### 5.2.2超图与其他技术的结合

超图神经网络和图神经网络相互融合，目的在于探寻超图结构与图神经网络的结合途径，构建如超图卷积、超图注意力等相关机制，以此更高效地学习高阶关系以及多节点交互信息，跨领域图学习是把超图与GNN在表示学习方面的优势结合起来，提升在大规模复杂网络里节点嵌入和社区划分的精确程度。超图神经网络还与强化学习、自监督学习相结合，其中自监督特征学习借助自监督学习机制生成伪标签或者构造预训练任务，自动提取网络中的潜在特征，为超图建模提供更精细的特征表达，强化学习优化策略是在社区发现任务中运用强化学习框架，依靠智能体动态调整超图构造以及模型参数，实现更精准且自适应的社区划分。超图神经网络也和大数据处理技术相融合，云计算与边缘计算借助云计算平台和边缘计算技术，将超图社区发现算法部署在大规模分布式系统中，为实时数据处理和决策制定提供支撑，流式数据处理结合流式处理框架，针对动态网络数据进行实时更新以及在线社区检测，契合物联网、社交媒体等领域的实时需求。

### 5.2.3社区发现的实际应用前景

社交网络与舆情分析领域覆盖用户行为分析以及舆情监测这两个方面，用户行为分析会借助高精度社区发现模型，以此帮助企业和平台去深入认识用户兴趣、行为模式以及群体特征，优化广告投放与内容推荐，而舆情监测是在社会事件、热点新闻等场景之下，及时辨别意见领袖与关键群体，为政府和企业给予舆情监控及风险预警方面的支持。生物信息与医疗健康领域包含疾病关联研究和精准医疗，疾病关联研究在基因调控网络和蛋白质相互作用网络当中，依靠社区发现来识别功能模块，可揭示疾病潜在病理机制并找寻新的药物靶点，精准医疗依据患者数据构建医疗网络，借助社区发现技术识别相似病症群体，实现个性化治疗方案制定以及健康管理。金融风险与欺诈检测领域包括金融网络分析和欺诈行为检测，金融网络分析在金融交易网络里，凭借社区发现识别异常交易与风险集群，辅助金融机构开展风险监控和预警，欺诈行为检测利用超图建模捕捉多节点间的高阶交互模式，可有效识别金融欺诈、洗钱等非法活动，为监管部门提供决策支持。智慧城市与交通管理领域涉及城市基础设施规划和交通流量预测，城市基础设施规划依靠社区发现技术分析城市交通、能源和通信网络中的结构特性，优化资源分配与基础设施建设，交通流量预测结合动态社区发现方法，对城市交通网络进行实时监控和预测，提高交通管理和应急响应能力。

# 参考文献

1. Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.
2. Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. nature, 2005, 435(7043): 814-818.
3. Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical review E, 2004, 69(2): 026113.
4. Radicchi F, Castellano C, Cecconi F, et al. Defining and identifying communities in networks[J]. Proceedings of the national academy of sciences, 2004, 101(9): 2658-2663.
5. Danon L, Diaz-Guilera A, Duch J, et al. Comparing community structure identification[J]. Journal of statistical mechanics: Theory and experiment, 2005, 2005(09): P09008.
6. Richardson M, Domingos P. Mining knowledge-sharing sites for viral marketing[C]//Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002: 61-70.
7. McCollough M A, Berry L L, Yadav M S. An empirical investigation of customer satisfaction after service failure and recovery[J]. Journal of service research, 2000, 3(2): 121-137.
8. Ye F, Chen C, Zheng Z. Deep autoencoder-like nonnegative matrix factorization for community detection[C]//Proceedings of the 27th ACM international conference on information and knowledge management. 2018: 1393-1402.
9. Li J, Zhou G, Qiu Y, et al. Deep graph regularized non-negative matrix factorization for multi-view clustering[J]. Neurocomputing, 2020, 390: 108-116.
10. Berge C. Les problemes de coloration en théorie des graphes[C]//Annales de l'ISUP. 1960, 9(2): 123-160.
11. Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.
12. 胡一飞. 稀疏非负矩阵分解算法研究[D].广东工业大学,2019.DOI:10.27029/d.
13. Lee D D, Seung H S. Learning the parts of objects by non-negative matrix factorization[J]. Nature, 1999, 401(6755): 788-791.
14. He C, Fei X, Cheng Q, et al. A survey of community detection in complex networks using nonnegative matrix factorization[J]. IEEE Transactions on Computational Social Systems, 2021, 9(2): 440-457.
15. Su X, Xue S, Liu F, et al. A comprehensive survey on community detection with deep learning[J]. IEEE Transactions on Neural Networks and Learning Systems, 2022.
16. Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. IEEE T-PAMI 35, 8 (2013), 1798–1828.
17. Chien I, Lin C Y, Wang I H. Community detection in hypergraphs: Optimal statistical limit and efficient algorithms[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2018: 871-879.
18. Ding C, Li T, Peng W. Orthogonal nonnegative matrix t-factorizations for clustering[C]. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006: 126-135.
19. Lancichinetti A, Fortunato S, Kertész J. Detecting the overlapping and hierarchical community structure in complex networks[J]. New journal of physics, 2009, 11(3): 033015
20. Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society[J]. nature, 2005, 435(7043): 814-818.
21. Gregory S. Finding overlapping communities in networks by label propagation[J]. New journal of Physics, 2010, 12(10): 103018.
22. Xie J, Szymanski B K, Liu X. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process[C]//2011 ieee 11th international conference on data mining workshops. IEEE, 2011: 344-349.
23. 杨海陆,赵鑫,陈晨等.基于节点影响力扩张的社交网络社区发现算法[J/OL].哈尔滨理工大学学报:1-10[2023-06-07].
24. 杨煜,段威威.基于谱聚类的社交网络动态社区发现算法[J/OL].计算机应用:1-10[2023-06-07].
25. Liu F, Xue S, Wu J, et al. Deep learning for community detection: progress, challenges and opportunities[J]. arXiv preprint arXiv:2005.08225, 2020.
26. Cai B, Wang Y, Zeng L, et al. Edge classification based on Convolutional Neural Networks for community detection in complex network[J]. Physica A: Statistical Mechanics and its Applications, 2020, 556: 124826.
27. Jia Y, Zhang Q, Zhang W, et al. Communitygan: Community detection with generative adversarial nets[C]//The World Wide Web Conference. 2019: 784-794.
28. Shchur O, Günnemann S. Overlapping Community Detection with Graph Neural Networks[J]. Computer Science, 2019, 50(2.0): 49.2-2.0.
29. Park C, Kim D, Han J, et al. Unsupervised attributed multiplex network
30. embedding[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(04): 5371-5378.

# 附录

## 实验代码

### 1.数据预处理

1. import csv
2. *# 1. 收集所有节点ID*
3. node\_set = set()
4. *# 从边文件收集*
5. with open('cora.cites', 'r') as f:
6. for line in f:
7. src, dst = line.strip().split()
8. node\_set.add(src)
9. node\_set.add(dst)
10. *# 从内容文件收集*
11. node\_label\_pairs = []
12. label\_set = set()
13. with open('cora.content', 'r') as f:
14. for line in f:
15. parts = line.strip().split()
16. node\_id = parts[0]
17. label = parts[-1]
18. node\_label\_pairs.append((node\_id, label))
19. label\_set.add(label)
20. node\_set.add(node\_id)
21. *# 2. 创建节点ID映射*
22. sorted\_nodes = sorted(node\_set)
23. node\_to\_new\_id = {node\_id: idx for idx, node\_id in enumerate(sorted\_nodes)}
24. *# 3. 导出边集合（映射后的ID）*
25. with open('cora.cites', 'r') as f\_in, open('cora.csv', 'w', newline='') as f\_out:
26. writer = csv.writer(f\_out)
27. writer.writerow(['from', 'to'])
28. for line in f\_in:
29. src, dst = line.strip().split()
30. writer.writerow([node\_to\_new\_id[src], node\_to\_new\_id[dst]])
31. *# 4. 创建类别到数字的映射*
32. label\_to\_id = {label: idx for idx, label in enumerate(sorted(label\_set))}
33. *# 5. 导出标签文件（映射后的ID，按编号排序）*
34. *# 先转成 (新编号, label\_id)*
35. mapped\_labels = [
36. (node\_to\_new\_id[node\_id], label\_to\_id[label])
37. for node\_id, label in node\_label\_pairs
38. ]
39. *# 排序*
40. mapped\_labels.sort()
41. with open('cora-labels.csv', 'w', newline='') as f\_out:
42. writer = csv.writer(f\_out)
43. writer.writerow(['node', 'label'])
44. for new\_node\_id, label\_id in mapped\_labels:
45. writer.writerow([new\_node\_id, label\_id])
46. import pandas as pd
47. from sklearn.preprocessing import LabelEncoder
48. *# 1. 读取 content 文件（包含节点 ID 和标签）*
49. content = pd.read\_csv('citeseer.content', sep='\t', header=None)
50. content.columns = ['paper\_id'] + [f'feat\_{i}' for i in range(content.shape[1] - 2)] + ['label']
51. content.drop\_duplicates(subset='paper\_id', inplace=True)  *# 去重*
52. content\_paper\_ids = content['paper\_id'].tolist()
53. labels = content['label'].tolist()
54. *# 2. 创建 paper\_id 到 node\_id 编码（仅对 content 中的 paper）*
55. paper\_id\_to\_node = {pid: idx for idx, pid in enumerate(content\_paper\_ids)}
56. valid\_nodes\_set = set(content\_paper\_ids)
57. *# 3. 标签编码*
58. label\_encoder = LabelEncoder()
59. label\_encoded = label\_encoder.fit\_transform(labels)
60. label\_df = pd.DataFrame({
61. 'node': list(range(len(content\_paper\_ids))),
62. 'label': label\_encoded
63. })
64. label\_df.to\_csv("citeseer-labels.csv", index=False, header=False)
65. *# 4. 读取 cites 边文件*
66. cites = pd.read\_csv('citeseer.cites', sep='\t', header=None)
67. cites.columns = ['source', 'target']
68. *# 5. 过滤出 source 和 target 都在 content 中的边（双向都在 content 中）*
69. cites\_filtered = cites[cites['source'].isin(valid\_nodes\_set) & cites['target'].isin(valid\_nodes\_set)]
70. *# 6. 将边映射为连续编号（0 ~ N-1）*
71. cites\_mapped = cites\_filtered.applymap(lambda x: paper\_id\_to\_node[x])
72. cites\_mapped.to\_csv("citeseer.csv", sep=',', header=False, index=False)
73. *# 验证信息输出*
74. print(f"原始边数: {len(cites)}")
75. print(f"保留有效边数: {len(cites\_mapped)}")
76. print(f"节点数: {len(content\_paper\_ids)}")
77. import csv
78. import re
79. from sklearn.preprocessing import LabelEncoder
80. *# 文件路径*
81. node\_file = 'Pubmed-Diabetes.NODE.paper.tab'
82. edge\_file = 'Pubmed-Diabetes.DIRECTED.cites.tab'
83. *# 读取节点和标签*
84. paper\_ids = []
85. labels = []
86. with open(node\_file, 'r', encoding='utf-8') as f:
87. lines = f.readlines()[2:]  *# 跳过前两行说明*
88. for line in lines:
89. parts = line.strip().split('\t')
90. paper\_id = parts[0].strip()
91. label\_match = re.search(r'cat=(\d)', parts[1])
92. if label\_match:
93. label = int(label\_match.group(1))
94. paper\_ids.append(paper\_id)
95. labels.append(label)
96. *# 创建映射和编码*
97. paper\_id\_to\_idx = {pid: idx for idx, pid in enumerate(paper\_ids)}
98. encoded\_labels = LabelEncoder().fit\_transform(labels)
99. *# 保存 label 到 CSV*
100. with open('pubmed-labels.csv', 'w', newline='', encoding='utf-8') as f:
101. writer = csv.writer(f)
102. writer.writerow(['node\_id', 'label'])
103. for idx, label in enumerate(encoded\_labels):
104. writer.writerow([idx, label])
105. *# 读取边并保存为 CSV*
106. with open('pubmed.csv', 'w', newline='', encoding='utf-8') as f\_out:
107. writer = csv.writer(f\_out)
108. writer.writerow(['source', 'target'])
109. with open(edge\_file, 'r', encoding='utf-8') as f\_in:
110. lines = f\_in.readlines()[2:]  *# 跳过前两行*
111. for line in lines:
112. parts = line.strip().split('\t')
113. if len(parts) != 2:
114. continue
115. src\_match = re.search(r'paper:(\d+)', parts[0])
116. dst\_match = re.search(r'paper:(\d+)', parts[1])
117. if src\_match and dst\_match:
118. src\_id, dst\_id = src\_match.group(1), dst\_match.group(1)
119. if src\_id in paper\_id\_to\_idx and dst\_id in paper\_id\_to\_idx:
120. writer.writerow([paper\_id\_to\_idx[src\_id], paper\_id\_to\_idx[dst\_id]])
121. print("CSV 文件保存成功：pubmed\_labels.csv 和 pubmed\_edges.csv")

### 2.模型设计

1. """DANMF class."""
2. import json
3. import os
4. import numpy as np
5. import pandas as pd
6. from tqdm import tqdm
7. import networkx as nx
8. from sklearn.decomposition import NMF
9. from scipy.optimize import linear\_sum\_assignment
10. from sklearn.metrics import normalized\_mutual\_info\_score, adjusted\_rand\_score
11. from sklearn.manifold import TSNE
12. from sklearn.decomposition import PCA
13. import matplotlib.pyplot as plt
14. class HDANMF(object):
15. """
16. Deep autoencoder-like non-negative matrix factorization class.
17. 深度类自动编码器非负矩阵分解类
18. """
19. def \_\_init\_\_(self, graph, args):
20. """
21. Initializing a DANMF object.
22. :param graph: Networkx graph.
23. :param args: Arguments object.
24. 初始化 DANMF 对象。:param 图：Networkx 图。:param args： Arguments 对象。
25. """
26. self.graph = graph  *# 传入 NetworkX 图*
27. self.A = nx.adjacency\_matrix(self.graph)  *# 计算邻接矩阵 A*
28. self.L = nx.laplacian\_matrix(self.graph)  *# 计算拉普拉斯矩阵 L*
29. self.D = self.L + self.A  *# 计算度矩阵 D = L + A*
30. self.args = args  *# 解析命令行参数*
31. self.p = len(self.args.layers)  *# 计算 NMF 分解层数*
32. *# self.args.lamb = self.args.lamb+self.args.cigma+self.args.alpha*
33. *# 读取 true\_labels 并保存为属性*
34. self.true\_labels = None
35. if args.true\_labels\_path:
36. try:
37. df = pd.read\_csv(args.true\_labels\_path)
38. *# 假设你的标签在第一列*
39. self.true\_labels = df.iloc[:, 1].values
40. except Exception as e:
41. print(f"读取 true labels 失败：{e}")
42. def setup\_z(self, i):
43. """
44. Setup target matrix for pre-training process.
45. 为预训练过程设置目标矩阵。
46. """
47. if i == 0:
48. self.Z = self.A
49. else:
50. self.Z = self.V\_s[i-1]
51. def sklearn\_pretrain(self, i):
52. """
53. Pretraining a single layer of the model with sklearn.
54. :param i: Layer index.
55. 使用 sklearn 预训练模型的单个层。:param i：图层索引。
56. """
57. nmf\_model = NMF(n\_components=self.args.layers[i],
58. init="random",
59. random\_state=self.args.seed,
60. max\_iter=self.args.pre\_iterations)
61. U = nmf\_model.fit\_transform(self.Z)
62. V = nmf\_model.components\_
63. return U, V
64. def pre\_training(self):
65. """
66. Pre-training each NMF layer.
67. 预训练每个 NMF 层。
68. """
69. print("\nLayer pre-training started. \n")
70. self.U\_s = []
71. self.V\_s = []
72. for i in tqdm(range(self.p), desc="Layers trained: ", leave=True):
73. self.setup\_z(i)
74. U, V = self.sklearn\_pretrain(i)
75. self.U\_s.append(U)
76. self.V\_s.append(V)
77. def setup\_Q(self):
78. """
79. Setting up Q matrices.
80. 设置 Q 矩阵。
81. """
82. self.Q\_s = [None for \_ in range(self.p+1)]
83. self.Q\_s[self.p] = np.eye(self.args.layers[self.p-1])
84. for i in range(self.p-1, -1, -1):
85. self.Q\_s[i] = np.dot(self.U\_s[i], self.Q\_s[i+1])
86. def update\_U(self, i):
87. """
88. Updating left hand factors.
89. :param i: Layer index.
90. 更新左手系数。:param i：图层索引。
91. """
92. if i == 0:
93. R = self.U\_s[0].dot(self.Q\_s[1].dot(self.VpVpT).dot(self.Q\_s[1].T))
94. R = R+self.A\_sq.dot(self.U\_s[0].dot(self.Q\_s[1].dot(self.Q\_s[1].T)))
95. Ru = 2\*self.A.dot(self.V\_s[self.p-1].T.dot(self.Q\_s[1].T))
96. self.U\_s[0] = (self.U\_s[0]\*Ru)/np.maximum(R, 10\*\*-10)
97. else:
98. R = self.P.T.dot(self.P).dot(self.U\_s[i]).dot(self.Q\_s[i+1]).dot(self.VpVpT).dot(self.Q\_s[i+1].T)
99. R = R+self.A\_sq.dot(self.P).T.dot(self.P).dot(self.U\_s[i]).dot(self.Q\_s[i+1]).dot(self.Q\_s[i+1].T)
100. Ru = 2\*self.A.dot(self.P).T.dot(self.V\_s[self.p-1].T).dot(self.Q\_s[i+1].T)
101. self.U\_s[i] = (self.U\_s[i]\*Ru)/np.maximum(R, 10\*\*-10)
102. def update\_P(self, i):
103. """
104. Setting up P matrices.
105. :param i: Layer index.
106. 设置 P 矩阵。:param i：图层索引。
107. """
108. if i == 0:
109. self.P = self.U\_s[0]
110. else:
111. self.P = self.P.dot(self.U\_s[i])
112. def update\_V(self, i):
113. """
114. Updating right hand factors.
115. :param i: Layer index.
116. 更新右手系数。:param i：图层索引
117. """
118. if i < self.p-1:
119. Vu = 2\*self.A.dot(self.P).T
120. Vd = self.P.T.dot(self.P).dot(self.V\_s[i])+self.V\_s[i]
121. self.V\_s[i] = self.V\_s[i] \* Vu/np.maximum(Vd, 10\*\*-10)
122. else:
123. Vu = 2\*self.A.dot(self.P).T+(self.args.lamb\*self.A.dot(self.V\_s[i].T)).T
124. Vd = self.P.T.dot(self.P).dot(self.V\_s[i])
125. Vd = Vd + self.V\_s[i]+(self.args.lamb\*self.D.dot(self.V\_s[i].T)).T
126. self.V\_s[i] = self.V\_s[i] \* Vu/np.maximum(Vd, 10\*\*-10)
127. def calculate\_cost(self, i):
128. """
129. Calculate loss.
130. :param i: Global iteration.
131. 计算损失。:param i： 全局迭代.
132. """
133. reconstruction\_loss\_1 = np.linalg.norm(self.A-self.P.dot(self.V\_s[-1]), ord="fro")\*\*2
134. reconstruction\_loss\_2 = np.linalg.norm(self.V\_s[-1]-self.A.dot(self.P).T, ord="fro")\*\*2
135. regularization\_loss = np.trace(self.V\_s[-1].dot(self.L.dot(self.V\_s[-1].T)))
136. self.loss.append([i+1, reconstruction\_loss\_1, reconstruction\_loss\_2, regularization\_loss])
137. def save\_embedding(self):
138. """
139. Save embedding matrix.
140. 保存嵌入矩阵。
141. """
142. embedding = [np.array(range(self.P.shape[0])).reshape(-1, 1), self.P, self.V\_s[-1].T]
143. embedding = np.concatenate(embedding, axis=1)
144. columns = ["id"] + ["x\_" + str(x) for x in range(self.args.layers[-1]\*2)]
145. embedding = pd.DataFrame(embedding, columns=columns)
146. embedding.to\_csv(self.args.output\_path, index=None)
147. def save\_membership(self):
148. """
149. Save cluster membership predictions to JSON.
150. 保存聚类成员信息到 JSON 文件。
151. """
152. membership\_path = self.args.membership\_path
153. *# 自动创建目录（如果不存在）*
154. os.makedirs(os.path.dirname(membership\_path), exist\_ok=True)
155. membership = np.argmax(self.P, axis=1).tolist()
156. with open(membership\_path, "w") as f:
157. json.dump(membership, f)
158. print(f"Cluster membership saved to: {membership\_path}")
159. def training(self):
160. """
161. Training process after pre-training.
162. 预训练后的训练过程。
163. """
164. print("\n\nTraining started. \n")
165. self.loss = []
166. self.A\_sq = self.A.dot(self.A.T)
167. for iteration in tqdm(range(self.args.iterations), desc="Training pass: ", leave=True):
168. self.setup\_Q()
169. self.VpVpT = self.V\_s[self.p-1].dot(self.V\_s[self.p-1].T)
170. for i in range(self.p):
171. self.update\_U(i)
172. self.update\_P(i)
173. self.update\_V(i)
174. if self.args.calculate\_loss:
175. self.calculate\_cost(iteration)
176. self.save\_membership()
177. self.save\_embedding()
178. def evaluate\_clustering(self):
179. """
180. Evaluate clustering performance using NMI and ARI.
181. Optionally compares with true labels if provided.
182. 使用 NMI 和 ARI 评估集群性能。（可选）与 true 标签（如果提供）进行比较。
183. """
184. true\_labels = self.true\_labels*# 从模型内部取，不用手动传*
185. if true\_labels is None:
186. print("未提供 True 标签。无法计算外部集群指标。")
187. return
188. pred\_labels = np.argmax(self.P, axis=1)
189. *# ===== NMI & ARI =====*
190. nmi = normalized\_mutual\_info\_score(true\_labels, pred\_labels)
191. ari = adjusted\_rand\_score(true\_labels, pred\_labels)
192. *# ===== ACC =====*
193. acc = self.clustering\_accuracy(true\_labels, pred\_labels)
194. print(f"NMI: {nmi:.4f}, ARI: {ari:.4f}, ACC: {acc:.4f}")
195. return nmi, ari, acc
196. def clustering\_accuracy(sel,true\_labels, pred\_labels):
197. """
198. 计算聚类准确率（ACC），基于匈牙利算法对标签进行最佳匹配。
199. """
200. true\_labels = np.array(true\_labels)
201. pred\_labels = np.array(pred\_labels)
202. D = max(pred\_labels.max(), true\_labels.max()) + 1
203. confusion\_matrix = np.zeros((D, D), dtype=np.int64)
204. for i in range(len(true\_labels)):
205. confusion\_matrix[pred\_labels[i], true\_labels[i]] += 1
206. row\_ind, col\_ind = linear\_sum\_assignment(-confusion\_matrix)  *# 最大化匹配*
207. acc = confusion\_matrix[row\_ind, col\_ind].sum() / len(true\_labels)
208. return acc
209. def visualize\_embedding(self, method='tsne'):
210. """
211. Visualize the embedding using t-SNE or PCA.
212. :param method: 'tsne' or 'pca'
213. :param labels: Optional true labels for coloring.
214. 将嵌入向量（即降维后的特征表示）通过 t-SNE 或 PCA 方法降维到二维空间，并以散点图的形式进行可视化，用于展示各个节点/样本在嵌入空间中的分布情况。
215. 使用 t-SNE 或 PCA 可视化嵌入。:param method： 'tsne' 或 'pca' :param labels： 用于着色的可选 true 标签。
216. """
217. labels = self.true\_labels
218. embedding = self.P
219. if method == 'tsne':
220. reducer = TSNE(n\_components=2, random\_state=42)
221. elif method == 'pca':
222. reducer = PCA(n\_components=2)
223. else:
224. raise ValueError("Method must be 'tsne' or 'pca'.")
225. reduced = reducer.fit\_transform(embedding)
226. plt.figure(figsize=(8, 6))
227. if labels is not None:
228. scatter = plt.scatter(reduced[:, 0], reduced[:, 1], c=labels, cmap='tab10', alpha=0.7)
229. plt.legend(\*scatter.legend\_elements(), title="Classes")
230. else:
231. plt.scatter(reduced[:, 0], reduced[:, 1], alpha=0.7)
232. plt.title(f'{method.upper()} visualization of DANMF embedding')
233. plt.xlabel('Dim 1')
234. plt.ylabel('Dim 2')
235. plt.grid(False)
236. plt.show()
237. def plot\_loss(self):
238. """
239. Plot loss curves recorded during training.
240. 绘制训练期间记录的损失曲线
241. """
242. if not hasattr(self, 'loss') or len(self.loss) == 0:
243. print("No loss recorded.")
244. return
245. loss\_array = np.array(self.loss)
246. plt.plot(loss\_array[:, 0], loss\_array[:, 1], label='Recon Loss 1')
247. plt.plot(loss\_array[:, 0], loss\_array[:, 2], label='Recon Loss 2')
248. plt.plot(loss\_array[:, 0], loss\_array[:, 3], label='Reg Loss')
249. plt.xlabel("Iteration")
250. plt.ylabel("Loss")
251. plt.title("Loss during training")
252. plt.legend()
253. plt.grid(False)
254. plt.show()

### 3.参数设置

1. """Parsing the model parameters."""
2. import argparse
3. def parameter\_parser(data\_type):
4. """
5. data\_type =  'Email/email-Eu-core','cora/cora', 'Citeseer/citeseer', 'Pubmed/pubmed'
6. A method to parse up command line parameters.
7. By default it gives an embedding of the Twitch Brasilians dataset.
8. The default hyperparameters give a good quality representation without grid search.
9. Representations are sorted by node identifiers.
10. 一种解析命令行参数的方法。默认情况下，它提供 Twitch Brasilians 数据集的嵌入。
11. 默认超参数在没有网格搜索的情况下提供高质量的表示。表示按节点标识符排序。
12. """
14. *# 创建 ArgumentParser 实例*
15. *# 为这个程序添加了描述 "Run DANMF."，用于在 --help 选项时显示*
16. parser = argparse.ArgumentParser(description="Run DANMF.")
17. *# 数据路径参数*
18. *# --edge-path：输入数据的路径；nargs="?" 表示该参数可选‘help 用于 --help 选项时的提示信息。*
19. parser.add\_argument("--edge-path",
20. nargs="?",
21. default=f"./input/{data\_type}.csv",
22. help="Edge list csv.")
23. *# --true\_labels:指定真实的标签数据*
24. parser.add\_argument("--true-labels-path",
25. nargs="?",
26. default=f"./input/{data\_type}-labels.csv",
27. help="True labels csv.")
28. *# --output-path：指定嵌入后的数据存储路径*
29. parser.add\_argument("--output-path",
30. nargs="?",
31. default=f"./output/{data\_type}\_danmf.csv",
32. help="Target embedding csv.")
33. *# --membership-path：指定聚类成员关系的 JSON 文件路径*
34. parser.add\_argument("--membership-path",
35. nargs="?",
36. default=f"./output/{data\_type}\_membership.json",
37. help="Cluster membership json.")
38. *# 模型训练参数*
39. *# --iterations：DANMF 训练的总迭代次数*
40. parser.add\_argument("--iterations",
41. type=int,
42. default=100,
43. help="Number of training iterations. Default is 100.")
44. *# --pre-iterations：逐层预训练*
45. parser.add\_argument("--pre-iterations",
46. type=int,
47. default=100,
48. help="Number of layerwsie pre-training iterations. Default is 100.")
49. *# --seed：用于 sklearn 预训练的随机种子*
50. parser.add\_argument("--seed",
51. type=int,
52. default=42,
53. help="Random seed for sklearn pre-training. Default is 42.")
54. *# --lamb：正则化参数（lambda）*
55. parser.add\_argument("--lamb",
56. type=float,
57. default=0.1, *# email 0.01*
58. help="Regularization parameter. Default is 0.01.")
59. *# --layers：指定神经网络的隐藏层维度*
60. *# nargs="+" 表示这个参数可以传入 多个数值，用空格分隔*
61. parser.add\_argument("--layers",
62. nargs="+",
63. type=int,
64. help="Layer dimensions separated by space. E.g. 128 64 32.")
65. *# 损失计算选项*
66. *# 这两个参数是互斥的，只有一个生效，默认情况是 False（不计算损失*
67. *# --calculate-loss：如果在命令行中添加这个参数*
68. parser.add\_argument("--calculate-loss",
69. dest="calculate\_loss",
70. action="store\_true")
71. *# --not-calculate-loss：如果添加这个参数*
72. parser.add\_argument("--not-calculate-loss",
73. dest="calculate\_loss",
74. action="store\_false")
75. *# 设定默认值 calculate\_loss=False，即如果用户没有显式传递 --calculate-loss*
76. parser.set\_defaults(calculate\_loss=False)
77. parser.set\_defaults(layers=[256, 64, 7]) *# email 256, 128, 42*
78. *# 解析参数*
79. return parser.parse\_args()

### 4.主函数设计

1. """Fitting a DANMF model."""  
   “”“拟合 DANMF 模型。”“”
2. from danmf import HDANMF
3. from parser1 import parameter\_parser
4. from utils import read\_graph, tab\_printer, loss\_printer
5. def main():
6. """
7. Parsing command lines, creating target matrix, fitting DANMF and saving the embedding.
8. 解析命令行，创建目标矩阵，拟合 DANMF 并保存嵌入。
9. """
10. args = parameter\_parser('cora/cora')
11. tab\_printer(args)
12. graph = read\_graph(args)
13. model = HDANMF(graph, args)
14. model.pre\_training()
15. model.training()
16. if args.calculate\_loss:
17. loss\_printer(model.loss)
18. model.evaluate\_clustering()
19. model.visualize\_embedding(method='tsne')
20. *# model.visualize\_embedding(method='umap')*
21. model.plot\_loss()
22. *# \_\_name\_\_ 是一个特殊的内置变量，当一个 Python 脚本被直接执行时，\_\_name\_\_ 的值会被设置为 "\_\_main\_\_"。*
23. *# 当脚本作为模块被导入到其他脚本时，\_\_name\_\_ 的值将会是模块的名字，而不是 "\_\_main\_\_"。*
24. if \_\_name\_\_ == "\_\_main\_\_":
25. main()

### 5.辅助函数

1. """Data reading utilities."""
2. '''数据读取实用程序'''
3. import pandas as pd
4. import networkx as nx
5. from texttable import Texttable
6. def read\_graph(args):
7. """
8. Method to read graph and create a target matrix with matrix powers.
9. :param args: Arguments object.
10. 读取图形并创建具有矩阵幂的目标矩阵的方法。:p aram args： Arguments 对象。
11. """
12. print("\nTarget matrix creation started.\n")
13. graph = nx.from\_edgelist(pd.read\_csv(args.edge\_path).values.tolist())
14. return graph
15. def tab\_printer(args):
16. """
17. Function to print the logs in a nice tabular format.
18. :param args: Parameters used for the model.
19. 以漂亮的表格格式打印日志的函数。:param args：用于模型的参数。
20. """
21. args = vars(args)
22. keys = sorted(args.keys())
23. t = Texttable()
24. t.add\_rows([["Parameter", "Value"]])
25. t.add\_rows([[k.replace("\_", " ").capitalize(), args[k]] for k in keys])
26. print(t.draw())
27. def loss\_printer(losses):
28. """
29. Printing the losses for each iteration.
30. :param losses: List of losses in each iteration.
31. 打印每次迭代的损失。:param losses：每次迭代中的损失列表。
32. """
33. t = Texttable()
34. t.add\_rows([["Iteration",
35. "Reconstrcution Loss I.",
36. "Reconstruction Loss II.",
37. "Regularization Loss"]])
38. t.add\_rows(True)
39. print(t.draw())

表 10 符号说明

|  |  |
| --- | --- |
| **符号** | **含义** |
|  | 图结构， 为节点集合， 为边集合 |
|  | 图的邻接矩阵，描述节点之间的连接关系 |
|  | 超图， 为超边集合，每条超边可连接多个节点 |
|  | 超图邻接矩阵，用于刻画高阶关系 |
|  | 节点度矩阵 |
|  | 超边度矩阵 |
|  | 超图拉普拉斯矩阵，用于正则化项 |
|  | 输入数据矩阵 |
|  | 基矩阵，表示节点的低维特征 |
|  | 系数矩阵，描述社区结构 |
|  | 超图正则化项的权重系数 |
|  | 稀疏性约束的权重系数 |
|  | 正交性约束的权重系数 |
|  | 深度自编码器重构误差的权重系数 |
|  | Frobenius 范数，衡量矩阵之间的重构误差 |
|  | 迹运算符，用于正则化表达式 |
|  | 非线性激活函数，用于深度自编码器的特征映射 |
|  | 低维嵌入表示，供社区划分使用 |

# 致谢

春去秋来，岁月如梭，属于我的大学四年生活伴随着这篇学士论文结尾已经抵达尾声，试着放慢脚步再挽回青春的余晖，才发现岁月长河川流不息不曾为谁停留。遗憾、失落、执念交织，情绪错综复杂，一年离别季，离多最是，东西流水，何时两相逢？回首望，往事一一现，旧时把酒言欢，许未来璀璨芳华，少年意气风发，试与天比高；今日泪满长衫，盼鲜花沿路盛开，踟蹰望断天涯路漫漫，谁解心头冷暖；未来披荆斩棘，亦步履不停，终见前路坦荡，愿他朝回首相望时，不言不语少年依旧共笑岁月沉浮。

衷心感谢导师罗鹏先生，这一路走来，承蒙先生悉心教诲与关怀，无论在生活还是科研中，皆给予我莫大帮助。尤为感激的是，罗老师在我多次迟疑徘徊之际，始终未曾放弃对我的引导与信任。若无他的支持与鼓励，我在科研上难有今日之收获。同时，罗老师严谨的治学态度与渊博的学识，潜移默化中促使我养成了扎实的科研素养。这篇论文的顺利完成，同样凝聚着他的智慧与心血。在此，谨致深深的感谢，并衷心祝愿导师未来科研之路硕果累累、再攀高峰。

同时，深深感谢我的家人，始终在我背后默默支持与鼓励。无论顺境或逆境，他们都以无私的爱与包容，给予我坚定的力量，是我不断前行的坚强后盾。正是有了家人的理解与付出，我才能心无旁骛地专注于学业与科研。所有的成长与收获，都离不开他们的陪伴与守护，在此向我的家人致以最深的感激和敬意。

回首来时路，风雨兼程，心怀万千感慨。那时懵懂少年，跌跌撞撞间渐得成长，于磨砺中锤炼出独当一面的力量。他始终怀揣一颗真诚之心，温润待人，犹如一束微光，愿为他人照亮前行之途。所幸一路有良师相伴，谆谆教诲，悉心指引；有挚友同行，携手共进，给予无尽勇气与温暖。那微弱的烛火，从未熄灭，亦在无数温暖的回应中愈发明亮。愿这烛火终有一日化作烈日骄阳，普照四方，温暖人间。我始终相信，唯有真诚，方能叩开未来的大门。愿我们不负韶华，各自奔赴那辽阔无垠的未来。

故事的最后，我想对自己说：我明白你的迷茫、无奈与痛苦，也理解你的遗憾、失落与悲伤，能体会你的软弱、迟疑与恐惧。我们总在幻想，另一条路上鲜花盛开，那里有更好的故事。或许你还不够勇敢，还不够坚强。但，没有人能够一夜长大，一切都需要一个过程。幼稚的你，是你；坚强的你，亦是你。过去的你，是你；未来的你，依然是你。最重要的是，你始终拥有选择的权利，去决定，你想成为怎样的自己。接下来慢慢的一起探索了解自己，找到自己。你一直很好—纵有痛苦，你未曾放弃；纵有遗憾，你未曾怨恨；纵然失去许多，你从未丢失真正的自己，感谢你一直在路上。