基于随机化矩阵分解的网络嵌入方法

谢雨洋[1),2)] 冯栩[1),2)] 喻文健[1),2)] 唐杰[1),2)]

[1)](清华大学计算机科学与技术系 北京 100084)

[2)](北京信息科学与技术国家研究中心 北京 100084)

摘 要 随着互联网的普及，越来越多的问题以社交网络这样的网络形式出现。网络通常用图数据表示，由于图数据处理的挑战性，如何从图中学习到重要的信息是当前被广泛关注的问题。网络嵌入就是通过分析图数据得到反映网络结构的特征向量，利用它们进而实现各种数据挖掘任务，例如边预测、节点分类、网络重构、标签推荐和异常检测。最近，基于矩阵分解的网络嵌入方法 NetMF 被提出，它在理论上统一了多种网络嵌入方法，并且在处理实际数据时表现出很好的效果。然而，在处理大规模网络时，NetMF 需要极大的时间和空间开销。本文使用快速随机化特征值分解和单遍历奇异值分解技术对 NetMF 进行改进，提出一种高效率、且内存用量小的矩阵分解网络嵌入算法 eNetMF。首先，我们提出了适合于对称稀疏矩阵的随机化特征值分解算法 freigs，它在处理实际的归一化网络矩阵时比传统的截断特征值分解算法快近 10 倍，且几乎不损失准确度。其次，我们提出使用单遍历奇异值分解处理 NetMF 方法中高次近似矩阵从而避免稠密矩阵存储的技术，它大大减少了网络嵌入所需的内存用量。最后，我们提出一种简洁的、且保证分解结果对称的随机化单遍历奇异值分解算法，将它与上述技术结合得到 eNetMF 算法。基于 5 个实际的网络数据集，我们评估了 eNetMF 学习到的网络低维表示在多标签节点分类和边预测上的有效性。实验结果表明，使用 eNetMF 替代 NetMF 后在后续得到的多标签分类性能指标上几乎没有损失，但在处理大规模数据时有超过 40 倍的加速与内存用量节省。在一台 32 核的机器上，eNetMF 仅需约 1.3 个小时即可对含一百多万节点的 YouTube 数据学习到网络嵌入，内存用量仅为 120GB，并得到较高质量的分类结果。此外，最近被提出的网络嵌入算法 NetSMF 由于图稀疏化过程的内存需求太大，无法在 256GB 内存的机器上处理两个较大的网络数据，而 ProNE 算法则在多标签分类的结果上表现不稳定，得到的 Macro-F1 值都比较差。因此，eNetMF 算法在结果质量上明显优于 NetSMF 和 ProNE 算法。在边预测任务上，eNetMF 算法也表现出与其他方法差不多甚至更好的性能。

关键词 网络嵌入；网络表示学习；随机化特征值分解；单遍历奇异值分解；多标签节点分类

中图法分类号 TP18

Learning Network Embedding with Randomized Matrix Factorization

XIE Yuyang[1), 2)] FENG Xu[1), 2)] YU Wenjian[1), 2)] TANG Jie[1), 2)]

[1)](Department of Computer Science and Technology, Tsinghua University, Beijing 100084)

[2)](Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084)

Abstract With the popularity of the Internet, more and more problems appear in the form of networks such as social networks. Networks are often organized by graph data and it is widely accepted that graph data is sophisticated and challenging to deal with. Therefore, how to learn important information from the graph is currently a widespread concern. Network embedding, which aims to learn latent representations for networks, is

本课题得到国家科技创新计划(2019AAA0103502)、国家自然科学基金(No. 61872206)资助. 谢雨洋，博士研究生，主要研究领域为数据挖掘、矩阵算法. E-mail:xyy18@mails.tsinghua.edu.cn. 冯栩，博士研究生，主要研究领域为矩阵算法. E-mail: fx17@mails.tsinghua.edu.cn. 喻文健(通信作者)，博士，副教授，计算机学会(CCF)会员（10183S），主要研究领域为集成电路计算机辅助设计、矩阵算法. E-mail: yu-wj@tsinghua.edu.cn. 唐杰，博士，教授，计算机学会(CCF)会员，主要研究领域为数据挖掘与知识工程. E-mail: jietang@tsinghua.edu.cn.

an important field in data mining. The result of network embedding should preserve network structure and can be used as the features for further applications on data mining tasks like link prediction, node classification, network reconstruction, tag recommendation and anomaly detection. Skip-gram based algorithms like DeepWalk, LINE, node2vec have been commonly considered as powerful benchmark solutions for evaluating network embedding research. Recently, network embedding as matrix factorization (NetMF) was proposed. It shows that DeepWalk is actually implicitly factorizing a network's normalized Laplacian matrix, and LINE, in theory, is a special case of DeepWalk. NetMF theoretically unifies several network embedding methods and shows its superiority when processing the real network data. However, the excessive runtime and storage of NetMF limits its usage on large-scale data. In this work, we use fast randomized eigenvalue decomposition (EVD) and single-pass singular value decomposition (SVD) to improve NetMF and present an efficient randomized matrix factorization based network embedding algorithm (eNetMF). Firstly, we propose a fast randomized EVD algorithm (freigs) for sparse matrix. It is about 10X faster than traditional truncated EVD algorithms for the normalized network matrix, while keeping accuracy. Secondly, we propose to use single-pass SVD approach to process high-order proximity matrix in the NetMF, so that we can avoid storing the large dense matrix and largely reduce the memory cost for network embedding. Thirdly, we propose a simplified, randomized single-pass SVD algorithm that guarantees a symmetric decomposition result. Combining the above techniques, we finally obtain the eNetMF algorithm. With five real network datasets, we evaluate the efficiency and effectiveness of the eNetMF algorithm on multi-label node classification and link prediction, commonly used tasks for network embedding evaluation. Experimental results show that in terms of the metrics Macro-F1 and Micro-F1, the embedding generated by eNetMF has the same performance as NetMF in multi-label node classification. For a large-scale graph data, eNetMF exhibits more than 40X acceleration and memory usage saving over NetMF. On a machine with 32 cores, eNetMF takes only 1.3 hours and 120GB memory to learn a good network embedding for the largest test data (YouTube) which has more than one million nodes. In addition, due to the memory cost of spectral sparsification, the recently proposed network embedding algorithm NetSMF cannot handle two large network data on a machine with 256GB memory, while the ProNE algorithm produces unstable results of multi-label node classification. To be precise, Macro-F1 obtained with ProNE is generally poor. Therefore, the eNetMF algorithm is advantageous to the NetSMF and ProNE algorithms in terms of performance. In the link prediction task, the eNetMF algorithm achieves better or comparable performance when compared to other baselines.

Key words network embedding; network representation learning; randomized eigenvalue decomposition; single-pass singular value decomposition; multi-label node classification

1 引言

社交网络等大规模复杂网络的挖掘与智能分析是当前的研究热点。这种网络包括微信、微博用户之间通过好友关系和互相联动形成的在线社交网络、学术论文之间的引用关系形成的网络[1]，等等。进行图数据挖掘主要有图卷积网络和网络嵌入两种方法。网络嵌入一般不考虑节点的属性，侧重于学习节点的拓扑结构，而图卷积网络可以将节点的属性作为特征一起学习，对于属性敏感的实验数据，图卷积网络的效果要更好，但由于很多网络数据缺少节点属性特征，图卷积网络对此类数据节点拓扑结构的学习不如网络嵌入，因此网络嵌入在图数据挖掘中占有很重要的地位。应用网络嵌入的方法对网络进行分析，首先要将网络用图的邻接矩阵表示。接下来，还要将网络中的每个节点映射到一个低维稠密实数向量上，这样可以很好地保存节点的拓扑结构信息，同时有利于使用机器学习等智能处理方法进行后续分析。因此网络嵌入的目标就是将网络映射到潜在的低维空间，得到网络节点的低维表示[2]。研究表明，网络嵌入技术可以促进各种网络分析与应用[3]，例如边预测、节点分类、网络

重构、标签推荐和异常检测。通常，基于传统拓扑的网络表示直接使用对应图的邻接矩阵，但这可能包含噪声或冗余信息。若使用网络嵌入技术，每个节点都由包含潜在信息的向量表示，因此许多网络分析中的迭代或组合的问题可以通过计算映射函数，距离度量或嵌入向量运算的方式解决，从而避免了高复杂度。

主要的网络嵌入方法包括 DeepWalk[3]、LINE[4]、node2vec[5]、metapath2vec[6] 等，它们都是基于 skip-gram 的模型。最近，文献[7]指出 DeepWalk 和 LINE 都可以看成是基于矩阵分解的技术，并提出了一种基于矩阵分解的网络嵌入方法 NetMF，基于它在后续进行网络节点多标签分类时表现出比使用 DeepWalk 和 LINE 更好的性能。然而，NetMF 在处理超大规模网络时需要巨大的内存开销和计算成本，这主要是由于它得到的高次近似矩阵是稠密的。后续，人们基于谱图稀疏化理论对 NetMF 进行了改进，得到了利用谱图稀疏化理论和稀疏矩阵分解的网络嵌入方法 NetSMF[8]和 ProNE[9]，这使得处理超大规模网络成为可能。但是，稀疏化矩阵的过程仍然需要大量的内存，这使得 NetSMF 的应用受到一定的限制，而 ProNE 得到的网络嵌入在应用时的效果也存在不稳定的问题。

另一方面，牺牲少量准确度的随机化低秩矩阵分解方法由于其适用于现代计算机体系结构和某些大数据应用场景正得到越来越多的关注[10][11]。本文将随机化矩阵分解技术与网络嵌入方法 NetMF 相结合，提出了一种快速、节省内存、且保证准确度的网络嵌入算法 eNetMF。具体的创新点如下：

- 提出使用单遍历奇异值分解处理 NetMF 方法中高次近似矩阵从而避免存储此稠密矩阵的思路，显著减少了 NetMF 方法所需的内存开销。
- 提出针对稀疏对称矩阵的随机化特征值分解算法，并调节需进行特征值分解的归一化网络矩阵奇异值衰减速度，从而在保证准确性的同时显著加快了 NetMF 方法中高次近似矩阵的构造。
- 提出一种简洁的、且保证分解结果对称的随机化单遍历奇异值分解算法，将它用于 NetMF 算法中减少了计算时间、又保证了网络嵌入的质量。

实验结果表明，在相对较大的真实数据集上，随机化特征值分解算法比传统截断特征值分解算法快 10 倍，几乎没有误差。而进行网络嵌入后再做节点多标签分类和边预测的实验中，eNetMF 算法在保持 NetMF 结果质量的同时大大减少了其计算时间与内存用量。例如，对 Flickr 数据 eNetMF 的运行时间和内存用量分别为 NetMF 的 1/41 和 1/46，而对于节点数超过 1 百万、NetMF 与 NetSMF 由于内存需求太大无法处理的 YouTube 数据，eNetMF 只需 1.3 个小时和 120GB 内存开销即可得到网络嵌入，并取得很好的多标签分类结果。

2 相关工作

近年来兴起的网络嵌入的研究起源于自然语言处理领域的表示学习[12]，该类问题的研究可以追溯到谱图理论[13]，社交维度学习[14]等研究。随着该领域的发展，很多网络嵌入的方法都旨在建模节点的相似性。谱网络嵌入方法和谱降维方法相关，例如快速近似谱聚类方法[15]和拉普拉斯特征映射方法[16]。受 word2vec 模型[12]的启发，一类基于 skip-gram 模型的方法被提出，并有很好的性能，例如 DeepWalk[3]、LINE[4]、node2vec[5]、metapath2vec[6] 等模型。最近，受词嵌入问题转化为矩阵分解的启发[17]，文献[7]指出基于 skip-gram 这一类的方法可以转化成矩阵分解问题。通过将随机游走的过程建模成矩阵的乘法，然后在近似的高阶矩阵上做奇异值分解，可以得到网络节点的低维表示。并且，其中提出的 NetMF 方法在设置较大窗口大小参数时可以得到高质量的网络嵌入，基于它再做网络节点多标签分类时表现出比使用 DeepWalk 和 LINE 更好的性能。

除了文献[7]的 NetMF 方法外，其他基于矩阵分解的网络嵌入模型还包括 GraRep[18]、HOPE[19]、TADW[20]和 DHPE[21]，其中 GraRep 模型考虑整合高次相似度矩阵来获取节点的相似度信息，但巨大的计算量和内存开销限制了其在大规模网络上的应用，HOPE 模型可以应用于有向图的场景，TADW 模型可以应用于节点带有特征信息的数据，DHPE 模型可以应用于动态网络的场景。最近，也有几个 NetMF 方法的改进算法被提出。NetSMF[8]通过引入谱图稀疏化方法使得模型可以运行得较快，不过仍然需要较大的运行内存。ProNE[9]方法通过交换随机游走和矩阵分解的过程，先对稀疏矩阵分解得到节点的低维表示，然后再对节点的低维表示做信息传播增强，这样大大降低了整体的计算复杂度。然而，在后续使用节点低维表示的应用任务（如多标签分类）中，ProNE 的性能很不稳定，往往比 NetMF 的结果差。

3 技术背景

在本文中，为了方便描述算法，我们采用 Matlab 所使用的方法来指定矩阵的部分元素。

3.1 奇异值分解和特征值分解

矩阵$\boldsymbol{A} \in \mathbb{R}^{m\times n}$的奇异值分解(SVD)可以表示为：

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \ ,\#(1)$$

其中$\boldsymbol{U} = [\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots]$和$\boldsymbol{V} = [\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots]$是分别包含左右奇异向量的正交矩阵。对角矩阵$\boldsymbol{\Sigma}$包含了矩阵$\boldsymbol{A}$的奇异值$(\sigma_1, \sigma_2, \cdots)$。设$\boldsymbol{U}_k$和$\boldsymbol{V}_k$分别是$\boldsymbol{U}$和$\boldsymbol{V}$的前$k$列，对角矩阵$\boldsymbol{\Sigma}_k$包含矩阵$\boldsymbol{A}$的前$k$个最大的奇异值，矩阵$\boldsymbol{A}$的截断 SVD 可以表示为$\boldsymbol{A}_k$，即：

$$\boldsymbol{A} \approx \boldsymbol{A}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T \ .\#(2)$$

$\boldsymbol{A}_k$也是初始矩阵$\boldsymbol{A}$的最佳秩k近似[22]。

若$\boldsymbol{A}$为对称矩阵，它存在如下形式的特征值分解(EVD)：

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \ ,\#(3)$$

其中$\boldsymbol{\Lambda}$为对角阵，其对角元为$\boldsymbol{A}$的特征值，正交矩阵$\boldsymbol{U}$的各列是$\boldsymbol{A}$的特征向量。由于实对称阵的特征值均为实数，只需将公式(3)中$\boldsymbol{\Lambda}$的负对角元取相反数、然后将$\boldsymbol{U}^T$的相应行也取相反数，公式(3)即可变成$\boldsymbol{A}$的奇异值分解，其中$\boldsymbol{U}$的各列是$\boldsymbol{A}$的左奇异向量。

类似于截断奇异值分解，由公式(3)也可以得到截断特征值分解：

$$\boldsymbol{A} \approx \boldsymbol{A}_h = \boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^T \ ,\#(4)$$

其中对角阵$\boldsymbol{\Lambda}_h$为$h \times h$的对角阵，对角元为h个绝对值最大、或代数值最大的特征值，$\boldsymbol{U}_h$为$\boldsymbol{U}$中与这些特征值相应的特征向量组成的矩阵。计算截断特征值分解的传统算法为 ARPACK 算法[23]，其具体实现包括 Matlab 中的 eigs 命令和 Python 的 eigs、eigsh 命令等，它们特别适合处理稀疏矩阵。

3.2 基于矩阵分解的网络表示

本文的主要目的是提出一种 NetMF 算法的改进版本，它能够显著减少 NetMF 算法的内存用量和计算时间。由于 NetMF 算法是针对无向图的网络嵌入方法，我们提出的方法也只考虑无向图、即对称稀疏矩阵。我们用$G = (V, E, \boldsymbol{A})$表示一个无向网络，其中$V$表示顶点集，$E$表示边的集合，$\boldsymbol{A}$表示$G$的邻接矩阵。网络嵌入的目标是学习到一个投影，该投影可以映射每一个顶点到一个k维向量$(k \ll n)$，使得可以保存它的结构属性信息，进而这种网络表示被提供给后续的网络分析使用，例如边预测和节点分类等。

最近，基于矩阵分解的网络嵌入显示出它在网络表示上的优越性。DeepWalk 是一个近年来被广泛承认的网络嵌入模型，随机游走的长度对 DeepWalk 模型的实验结果有影响。该参数越大，随机游走得越远且更多较远的点也会被认为是当前节点相似的节点，但一定程度上也相当于放宽相似性约束，可能会引入一些噪声，因此如何设置这个参数使得实验结果更好并没有普遍适用的结论。文献[7]揭示了当 DeepWalk 里的随机游走的长度达到无穷大时，它相当于是对矩阵

$$trunc\_log^{\circ}\left(\frac{vol(G)}{bq}\sum_{r=1}^{q}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\boldsymbol{D}^{-1}\right)\#(5)$$

作分解，其中$trunc_log^{\circ}$表示矩阵的每个元素和 1 取最大值后再取log，即

$$trunc\_log^{\circ}(x) = log(max(1, x)) \ .\#(6)$$

公式(5)的参数$\boldsymbol{D}$为邻接矩阵$\boldsymbol{A}$生成的对角阵，其对角元$d_{ii} = \sum_j \boldsymbol{A}_{ij}$，$vol(G) = \sum_i\sum_j \boldsymbol{A}_{ij}$，而参数$q$和参数$b$分别表示窗口大小和 skip-gram 模型里负采样的大小。在下面的工作里，我们考虑带有较大的窗口大小的情况，比如 DeepWalk 默认的情况下q=10。文献[7]提出的 NetMF 给基于 skip-gram 的网络嵌入方法奠定了理论基础，使得网络嵌入模型有更强的可解释性。算法 1 描述了处理较大窗口的 NetMF 算法[7]。

算法 1. NetMF.

输入：邻接矩阵$\boldsymbol{A} \in \mathbb{R}^{n\times n}$，特征值截断参数$h$，奇异值截断参数$k$，窗口大小$q$。

输出：网络嵌入$\boldsymbol{E} \in \mathbb{R}^{n\times k}$。

1. 截断特征值分解：$\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \approx \boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^T$
2. 构造矩阵$\boldsymbol{M} = \boldsymbol{D}^{-1/2}\boldsymbol{U}_h(\sum_{r=1}^{q}\boldsymbol{\Lambda}_h^r)\boldsymbol{U}_h^T\boldsymbol{D}^{-1/2}$
3. 计算$\bar{\boldsymbol{M}} = trunc_log^{\circ}(\frac{vol(G)}{bq}\boldsymbol{M})$
4. 截断奇异值分解：$\bar{\boldsymbol{M}} \approx \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$
5. 返回$\boldsymbol{E} = \boldsymbol{U}_k{\boldsymbol{\Sigma}_k}^{1/2}$

然而，当直接应用 NetMF 算法到大规模的数据集上时存在一些问题。首先，在算法 1 第 1 步，对矩阵$\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$做截断特征值分解，但直接计算特征值分解需要很长的时间。算法 1 的第 2、4 步也需要花费较多的时间，且第 2 步存储大的稠密矩阵$\boldsymbol{M}$是很有挑战性的。比如，当网络节点数n为 110 万时，稠密矩阵$\boldsymbol{M}$的存储量超过 8T 字节，这使得该算法很难应用到大型网络分析当中。

3.3 随机化奇异值分解有关技术

基础的随机化奇异值分解算法[10]可以用来计算近似的截断 SVD，该算法如算法 2 所示。

算法 2. 基础随机化奇异值分解.

输入：矩阵$\boldsymbol{A}\in\mathbb{R}^{m\times n}$，奇异值截断参数$k$，幂迭代参数$p$，过采样参数$s$。

输出：$\boldsymbol{U}\in\mathbb{R}^{m\times k},\boldsymbol{\Sigma}\in\mathbb{R}^{k\times k},\boldsymbol{V}\in\mathbb{R}^{n\times k}$。

1. $\boldsymbol{\Omega}=randn(n,k+s)$
2. $\boldsymbol{Q}=orth(\boldsymbol{A\Omega})$
3. $FOR\ i=1,2,\cdots p\ DO$
4. $\quad\boldsymbol{G}=orth(\boldsymbol{A}^T\boldsymbol{Q})$
5. $\quad\boldsymbol{Q}=orth(\boldsymbol{AG})$
6. END FOR
7. $\boldsymbol{B}=\boldsymbol{Q}^T\boldsymbol{A}$
8. $[\overline{\boldsymbol{U}},\boldsymbol{\Sigma},\boldsymbol{V}]=svd(\boldsymbol{B})$
9. $\boldsymbol{U}=\boldsymbol{Q}\overline{\boldsymbol{U}}$
10. $\boldsymbol{U}=\boldsymbol{U}(:,1:k),\boldsymbol{\Sigma}=\boldsymbol{\Sigma}(1:k,1:k),\boldsymbol{V}=\boldsymbol{V}(:,1:k)$

在算法 2 中，$\boldsymbol{\Omega}$是一个高斯随机矩阵，参数s使得采样矩阵$\boldsymbol{\Omega}$超过k列以获得更好的计算结果。矩阵$\boldsymbol{Q}$的列包含了$\boldsymbol{A}$列空间的主要基，因此$\boldsymbol{A}\approx\boldsymbol{QB}=\boldsymbol{QQ}^T\boldsymbol{A}$，然后对一个$(k+s)\times n$大小的矩阵$\boldsymbol{B}$使用 SVD，就可以得到矩阵$\boldsymbol{A}$的近似截断 SVD。算法 1 的第 3～6 步为幂迭代技术[10]，应用它之后相当于计算矩阵$(\boldsymbol{AA}^T)^p\boldsymbol{A}$的奇异向量，由于$(\boldsymbol{AA}^T)^p\boldsymbol{A}$的奇异值衰减比原始矩阵$\boldsymbol{A}$快得多，使用幂迭代后再做随机化奇异值分解可以使结果更接近最佳秩 k 近似。幂迭代过程中的正交化$orth()$可减轻浮点数计算的舍入误差影响，这个正交化操作通过 QR 分解来实现。随机化奇异值分解算法在很高概率下能保证：

$$\|\boldsymbol{A}-\boldsymbol{QQ}^T\boldsymbol{A}\|=\|\boldsymbol{A}-\boldsymbol{U\Sigma V}^T\|\leq(1+\epsilon)\|\boldsymbol{A}-\boldsymbol{A}_k\|\#,(7)$$

其中$\boldsymbol{A}_k$是矩阵$\boldsymbol{A}$的最佳秩k近似，ϵ是一个误差阈值。

对于实对称阵$\boldsymbol{A}\in\mathbb{R}^{n\times n}$，对算法 2 稍作修改就可以得到截断特征值分解[10]。此时算法 2 中的矩阵$\boldsymbol{Q}$表示了$\boldsymbol{A}$的行空间和列空间的主要基，因此，

$$\boldsymbol{A}\approx\boldsymbol{QZQ}^T=\boldsymbol{QQ}^T\boldsymbol{AQQ}^T\ .\#(8)$$

对实对称阵$\boldsymbol{Z}=\boldsymbol{Q}^T\boldsymbol{AQ}$做截断特征值分解，然后将分解结果代入(8)就可以得到矩阵$\boldsymbol{A}$的近似截断特征值分解。

应当指出，由公式(8)得到的算法相比基于$\boldsymbol{A}\approx\boldsymbol{QQ}^T\boldsymbol{A}$的算法其低秩近似效果差一些。对于对称矩阵，Tropp 等人最近提出了基于$\boldsymbol{QQ}^T\boldsymbol{A}$形式的近似、且保证结果仍为对称矩阵的方法[24]，它对原矩阵的近似程度更好。基于算法 2 得到的矩阵$\boldsymbol{Q}$、$\boldsymbol{B}$，即得到原矩阵$\boldsymbol{A}$的近似矩阵$\widehat{\boldsymbol{A}}=\boldsymbol{QB}$，文献[24]指出，

$$\widehat{\boldsymbol{A}}_{sym}=\frac{1}{2}\left(\widehat{\boldsymbol{A}}+\widehat{\boldsymbol{A}}^T\right)=\frac{1}{2}(\boldsymbol{QB}+\boldsymbol{B}^T\boldsymbol{Q}^T)\ \#(9)$$

为对$\boldsymbol{A}$近似程度更好的对称矩阵。根据[25]，$\widehat{\boldsymbol{A}}_{sym}$是$\widehat{\boldsymbol{A}}$到所有 n 阶实对称阵形成的凸集上的投影，由于$\boldsymbol{A}$本身也在这个凸集内，则根据[26]一定有：

$$\left\|\boldsymbol{A}-\widehat{\boldsymbol{A}}_{sym}\right\|_F\leq\left\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\right\|_F.\#(10)$$

并且，对$\widehat{\boldsymbol{A}}_{sym}$再做低秩近似得到的结果也比$\widehat{\boldsymbol{A}}$的结果更准确[24]。

此外，文献[27]提出了一种单遍历奇异值分解算法，它在数学上与幂迭代参数$p=0$的算法 2 等价，但只需要对矩阵$\boldsymbol{A}$访问一遍，特别适合于在内存受限的情况下处理大型的稠密矩阵、或者高效率地处理流数据。针对大型稀疏矩阵，文献[28]提出了高效率的随机化 SVD 算法，它与算法 2 在数学上等价，但使用了多种加速技巧可使得对实际大型稀疏矩阵的处理时间缩小 9 倍。这些加速技巧主要包括：

(1).利用特征值分解来快速计算细长条矩阵的 SVD 和 QR 分解，即使用 eigSVD 算法[28]。

(2).在幂迭代过程中，每隔一次矩阵乘法之后再做正交化操作。

(3).幂迭代中可使用 LU 分解替代 QR 分解。

4 基于高效矩阵分解的网络嵌入方法

在这一节，我们首先给出利用随机化矩阵分解技术提高 NetMF 方法处理大型网络的效率的主要思路，然后依次提出针对稀疏实对称阵的快速随机化特征值分解、矩阵$\boldsymbol{M}$的高效率构造与隐式存储、以及快速单遍历奇异值分解这三项技术，最后给出算法 eNetMF 的完整描述，并做计算量分析。

4.1 基本思想

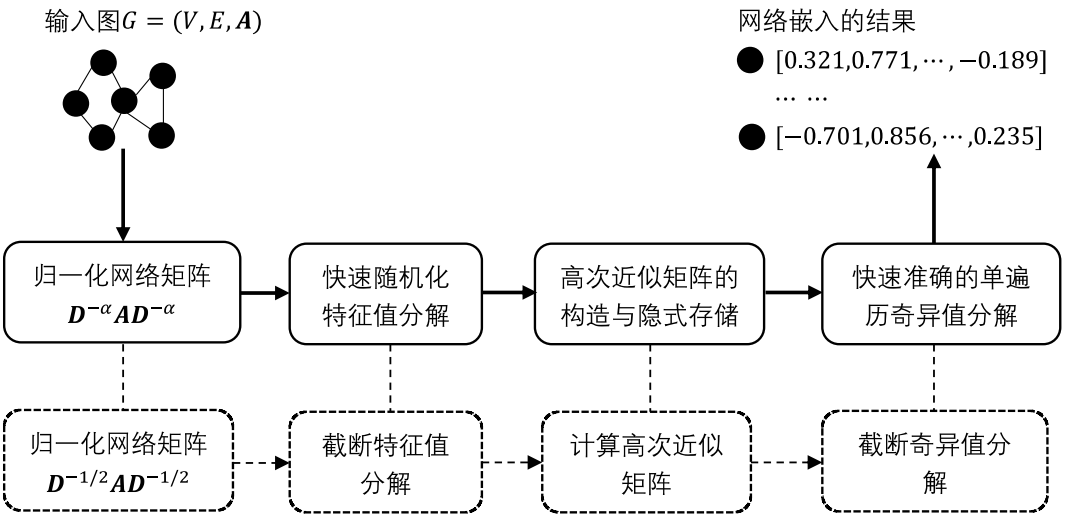

图 1 eNetMF 算法的流程图（下排虚线框为 NetMF 算法中相应的步骤）

可粗略地将 NetMF 方法分为三个主要步骤：用截断特征值分解归一化网络矩阵来近似矩阵、基于特征值分解构造高次近似矩阵$\boldsymbol{M}$、计算$\bar{\boldsymbol{M}}$矩阵的截断奇异值分解。对于大型网络，第一步和第三步都无法使用传统的矩阵分解算法来实现，它们需要巨大的运行时间，而第二步不但有很大的计算量，而且形成的稠密矩阵造成巨大的内存开销，也给第三步的执行带来困难。

首先我们可以使用随机化特征值分解算法来代替 NetMF 方法的第一个主要步骤，在几乎不损失准确度的情况下减少其计算量。其次，为了解决大型稠密矩阵$\boldsymbol{M}$或$\bar{\boldsymbol{M}}$给内存量带来的挑战，我们提出高次近似矩阵的高效率构造方法以及不显式存储$\boldsymbol{M}$和$\bar{\boldsymbol{M}}$，然后采用单遍历随机化 SVD 算法近似计算奇异值分解的思路，这样极大地减少了整个算法的内存用量，同时我们还提出快速、准确的单遍历奇异值分解技术保证最终网络嵌入的结果有足够的准确度。

本文提出的 eNetMF 算法的流程图如图 1 所示，图中上排实线框表示本文提出的 eNetMF 算法的主要步骤，下排虚线框表示 NetMF 算法的主要步骤。

4.2 针对稀疏实对称阵的快速随机化特征值分解

我们将文献[28]提出的适用于稀疏矩阵加速分解的技巧与文献[24]提出的对称矩阵随机化特征值分解算法结合，得到算法 3。其中，lu表示对矩阵做 LU 分解。

算法 3. 快速随机化特征值分解(freigs).

输入：矩阵$\boldsymbol{A}\in\mathbb{R}^{n\times n}$，特征值截断参数$k$，幂迭代参数$p$，过采样参数$s$。

输出：$\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{\Lambda}\in\mathbb{R}^{k\times k}$。

1. $\boldsymbol{\Omega}=randn(n,k+s)$
2. $\boldsymbol{Y}=\boldsymbol{A}\boldsymbol{\Omega}$
3. $[\boldsymbol{Q},\sim,\sim]=eigSVD(\boldsymbol{Y})$
4. FOR $i=1,2,\cdots p$ DO
5. IF $i<p$ THEN
6. $[\boldsymbol{Q},\sim]=lu(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{Q}))$
7. ELSE
8. $[\boldsymbol{Q},\sim,\sim]=eigSVD(\boldsymbol{A}(\boldsymbol{A}\boldsymbol{Q}))$
9. END FOR
10. $\boldsymbol{Z}=[\boldsymbol{Q},\boldsymbol{A}\boldsymbol{Q}]$
11. $[\boldsymbol{P},\boldsymbol{T}]=qr(\boldsymbol{Z})$
12. $\boldsymbol{T}_1,\boldsymbol{T}_2$分别是矩阵$\boldsymbol{T}$的前$k+s$列和后$k+s$列
13. $\boldsymbol{S}=(\boldsymbol{T}_1\boldsymbol{T}_2^T+\boldsymbol{T}_2\boldsymbol{T}_1^T)/2$
14. $[\widehat{\boldsymbol{U}},\widehat{\boldsymbol{\Lambda}}]=eig(\boldsymbol{S})$
15. ind为$\widehat{\boldsymbol{\Lambda}}$中$k$个最大的特征值的位置索引集合
16. $\boldsymbol{U}=\boldsymbol{P}\widehat{\boldsymbol{U}}(:,ind),\boldsymbol{\Lambda}=\widehat{\boldsymbol{\Lambda}}(ind,ind)$

算法 3 的第 10~14 步基于如下推导。由于

$$\widehat{\boldsymbol{A}}_{sym}=\frac{1}{2}(\boldsymbol{Q}\boldsymbol{B}+\boldsymbol{B}^T\boldsymbol{Q}^T)=\frac{1}{2}[\boldsymbol{Q},\boldsymbol{B}^T]\begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{Q}^T\end{bmatrix},\qquad(11)$$

如果对$[\boldsymbol{Q},\boldsymbol{B}^T]=[\boldsymbol{Q},\boldsymbol{A}\boldsymbol{Q}]$做精简的 QR 分解，

$$[\boldsymbol{Q},\boldsymbol{B}^T]=\boldsymbol{P}[\boldsymbol{T}_1,\boldsymbol{T}_2]\ ,\#(12)$$

其中$\boldsymbol{P}$为列正交阵，$[\boldsymbol{T}_1,\boldsymbol{T}_2]$为上三角方阵且$\boldsymbol{T}_1$和$\boldsymbol{T}_2$维度一样，则：

$$\widehat{\boldsymbol{A}}_{sym}=\frac{1}{2}\boldsymbol{P}[\boldsymbol{T}_1,\boldsymbol{T}_2]\begin{bmatrix}(\boldsymbol{P}\boldsymbol{T}_2)^T\\ (\boldsymbol{P}\boldsymbol{T}_1)^T\end{bmatrix}=\frac{1}{2}\boldsymbol{P}(\boldsymbol{T}_1\boldsymbol{T}_2^T+\boldsymbol{T}_2\boldsymbol{T}_1^T)\boldsymbol{P}^T.\qquad(13)$$

记$\boldsymbol{S}=\frac{1}{2}(\boldsymbol{T}_1\boldsymbol{T}_2^T+\boldsymbol{T}_2\boldsymbol{T}_1^T)$，对$\boldsymbol{S}$作特征值分解，再代入

(13)式即得到$\widehat{\boldsymbol{A}}_{sym}$的特征值分解，进而得到原矩阵$\boldsymbol{A}$的近似特征值分解。

下面进行算法计算量和空间开销的分析。设$l=k+s$，算法 3 的第 2 步的浮点运算次数是$O(nnz(\boldsymbol{A})l)$，$nnz(\boldsymbol{A})$为$\boldsymbol{A}$中非零元素的数目。根据文献[28]，第 3~9 步的浮点运算次数为$O(p(nl^2+nnz(\boldsymbol{A})l))$，第 10 步需要$O(nl+nnz(\boldsymbol{A})l)$次浮点运算，第 11~16 步则需要$O(l^3+nl^2)$次浮点运算。通常情况下，$l$远小于网络顶点数 n，算法 3 的浮点运算次数为$O\big(p(nnz(\boldsymbol{A})l+nl^2)\big)$。考虑到幂迭代参数$p$与过采样参数 s 均为不大的常数，而由于大规模网络的稀疏性，$nnz(\boldsymbol{A})$也不超过 n 的某个常数倍，算法 3 的时间复杂度为$O(nnz(\boldsymbol{A})k+nk^2)$，其中 k 为特征值截断参数。这与 Matlab 中 eigs 命令的复杂度一样，但由于随机化算法中的运算更适合于当前的计算机体系结构，算法 3 的运行时间比 eigs 显著缩短，后面的实验结果将验证这一点。此外，上述分析也表明算法 3 的时间复杂度与矩阵规模呈线性关系。

在空间复杂度方面，矩阵$\boldsymbol{A}$采用稀疏矩阵方式存储，而算法涉及的稠密矩阵维数都不超过$n\times l$或$l\times n$，所以算法 3 的空间复杂度为$O(nnz(\boldsymbol{A})+nk)$，也与矩阵规模呈线性关系。

4.3 矩阵$\boldsymbol{M}$的高效率构造与隐式存储

在 NetMF 方法中，$\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$的特征值范围在$[-1,1]$区间内[7]，对于大型矩阵来说这意味着该矩阵的奇异值衰减趋势比较缓慢。随机化算法在处理奇异值衰减缓慢的矩阵时，得到计算结果的近似误差较大。为了缓解它给矩阵近似准确度带来的问题，我们提出在保证算法 1 第 2 步矩阵$\boldsymbol{M}$不变的同时修改计算截断特征值分解的矩阵，通过对一个奇异值衰减较快的矩阵计算截断特征值提高准确度。具体地，

$$\sum_{r=1}^{q}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\boldsymbol{D}^{-1}$$

$$=\boldsymbol{D}^{-1+\alpha}\boldsymbol{D}^{-\alpha}\boldsymbol{A}\left(\sum_{r=1}^{q}(\boldsymbol{D}^{-1}\boldsymbol{A})^{r-1}\right)\boldsymbol{D}^{-\alpha}\boldsymbol{D}^{-1+\alpha},\quad(14)$$

其中$\alpha\in(0,1)$。如果对矩阵$\boldsymbol{D}^{-\alpha}\boldsymbol{A}\boldsymbol{D}^{-\alpha}$执行截断特征值分解，即：

$$\boldsymbol{D}^{-\alpha}\boldsymbol{A}\boldsymbol{D}^{-\alpha}\approx\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T\ .\#(15)$$

将式(15)代入式(14)当中，得

$$\sum_{r=1}^{q}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\boldsymbol{D}^{-1}$$

$$\approx\boldsymbol{D}^{-1+\alpha}(\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T+\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T\boldsymbol{D}^{-1+2\alpha}\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T+\cdots$$
$$+\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T\cdots\boldsymbol{D}^{-1+2\alpha}\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^T)\boldsymbol{D}^{-1+\alpha}$$

$$=\boldsymbol{D}^{-1+\alpha}\boldsymbol{G}_h\boldsymbol{H}_h\left(\sum_{r=1}^{q}\boldsymbol{K}^{r-1}\right)\boldsymbol{G}_h^T\boldsymbol{D}^{-1+\alpha}\ ,\quad(16)$$

其中$\boldsymbol{K}=\boldsymbol{G}_h^T\boldsymbol{D}^{-1+2\alpha}\boldsymbol{G}_h\boldsymbol{H}_h$，为一个$h\times h$矩阵。由于$h\ll n$，计算矩阵$\boldsymbol{K}$、以及(16)中它的各次幂花费的时间很少。因此，可以将算法 1 中的矩阵$\boldsymbol{M}$替换为：

$$\boldsymbol{M}=\boldsymbol{D}^{-1+\alpha}\boldsymbol{G}_h\boldsymbol{H}_h\left(\sum_{r=1}^{q}\boldsymbol{K}^{r-1}\right)\boldsymbol{G}_h^T\boldsymbol{D}^{-1+\alpha}.\quad(17)$$

这样，选择合适的α可使矩阵$\boldsymbol{D}^{-\alpha}\boldsymbol{A}\boldsymbol{D}^{-\alpha}$的奇异值衰减较快，从而使用算法 3 能比较高效、准确地得到其截断特征值分解，也不损失 NetMF 方法中矩阵$\boldsymbol{M}$的准确度。

为了避免存储稠密矩阵$\boldsymbol{M}$或$\bar{\boldsymbol{M}}$，我们提出一种隐式存储方案，即考虑分解式$\boldsymbol{M}=\boldsymbol{F}\boldsymbol{C}\boldsymbol{F}^T$，其中$\boldsymbol{F}=\boldsymbol{D}^{-1+\alpha}\boldsymbol{G}_h$，$\boldsymbol{C}=\boldsymbol{H}_h\big(\sum_{r=1}^{q}\boldsymbol{K}^{r-1}\big)$，然后分批计算出$\bar{\boldsymbol{M}}$的行，利用下面将介绍的单遍历 SVD 算法计算其截断奇异值分解，由于它只访问矩阵$\bar{\boldsymbol{M}}$的元素一遍，对计算出来的$\bar{\boldsymbol{M}}$的若干行处理后即可将其删除，因此可大大节省内存用量。而需要存储的矩阵$\boldsymbol{F}\in\mathbb{R}^{n\times h}$，$\boldsymbol{C}\in\mathbb{R}^{h\times h}$，远小于$\bar{\boldsymbol{M}}\in\mathbb{R}^{n\times n}$的规模。

4.4 快速准确的单遍历SVD算法

我们首先提出一种简洁的单遍历 SVD 算法，它与文献[27]中的算法等价，但构造$\boldsymbol{Q}\boldsymbol{B}$近似矩阵的操作更简洁，运行效率更高，如算法 4 所示。其中，$\boldsymbol{A}_i$表示输入矩阵$\boldsymbol{A}$的第i行，通过步骤 4～7，我们可以得到两个包含原始矩阵$\boldsymbol{A}$信息的小矩阵$\boldsymbol{Y}$和$\boldsymbol{W}$，它们保存了矩阵$\boldsymbol{A}$的行、列空间的主要成分，基于它们可以进一步计算$\boldsymbol{A}$的奇异值分解。

算法 4. 一种更简洁的单遍历 SVD.

输入：矩阵$\boldsymbol{A}\in\mathbb{R}^{m\times n}$，奇异值截断参$k$，过采样参数$s$。

输出：$\boldsymbol{U}\in\mathbb{R}^{m\times k},\boldsymbol{\Sigma}\in\mathbb{R}^{k\times k},\boldsymbol{V}\in\mathbb{R}^{n\times k}$。

1. $\boldsymbol{\Omega}=randn(n,k+s)$
2. $\boldsymbol{Y}=[\,]$
3. $\boldsymbol{W}=zeros(n,k+s)$
4. FOR $i=1,2,\cdots,m$ DO
5. $\quad\boldsymbol{y}=\boldsymbol{A}_i\boldsymbol{\Omega},\quad\boldsymbol{Y}=[\boldsymbol{Y};\boldsymbol{y}]$
6. $\quad\boldsymbol{W}=\boldsymbol{W}+\boldsymbol{A}_i^T\boldsymbol{y}$
7. END FOR

8. $[\boldsymbol{Q},\boldsymbol{R}]=qr(\boldsymbol{Y})$
9. $\boldsymbol{B}=\boldsymbol{R}^{-T}\boldsymbol{W}^{T}$
10. $[\bar{\boldsymbol{U}},\boldsymbol{\Sigma},\boldsymbol{V}]=svd(\boldsymbol{B})$
11. $\boldsymbol{U}=\boldsymbol{Q}\bar{\boldsymbol{U}}$
12. $\boldsymbol{U}=\boldsymbol{U}(:,1:k),\boldsymbol{\Sigma}=\boldsymbol{\Sigma}(1:k,1:k),\boldsymbol{V}=\boldsymbol{V}(:,1:k)$

在算法 4 中，步骤 8～12 是具体计算矩阵$\boldsymbol{A}$的奇异值分解的过程，它不同于现有的单遍历 SVD 算法[27]。定理 1 表明算法 4 与不做幂迭代的算法 2（$p=0$）在数学上是等价的。

定理 1. 在算法 4 中得到的$\boldsymbol{Q}$的各列是$\boldsymbol{A\Omega}$列空间的单位正交基，$\boldsymbol{B}=\boldsymbol{Q}^{T}\boldsymbol{A}$。

证明. 根据算法 4 的第 4～7 步，我们得$\boldsymbol{Y}=\boldsymbol{A\Omega}$，再根据第 8 步，对$\boldsymbol{Y}$执行 QR 分解，因此得到的$\boldsymbol{Q}$的各列是$\boldsymbol{A\Omega}$列空间的单位正交基。

下面再证$\boldsymbol{B}=\boldsymbol{Q}^{T}\boldsymbol{A}$，由算法 4 的第 4～7 步，有$\boldsymbol{W}=\boldsymbol{A}^{T}\boldsymbol{A\Omega}$，再根据第 9 步，

$$\boldsymbol{B}=\boldsymbol{R}^{-T}\boldsymbol{W}^{T}=\boldsymbol{R}^{-T}(\boldsymbol{A}^{T}\boldsymbol{A\Omega})^{T}=\boldsymbol{R}^{-T}\boldsymbol{\Omega}^{T}\boldsymbol{A}^{T}\boldsymbol{A}=(\boldsymbol{A\Omega R^{-1}})^{T}\boldsymbol{A}\ .$$

又因为$\boldsymbol{QR}=\boldsymbol{A\Omega}$，得$\boldsymbol{Q}=\boldsymbol{A\Omega R}^{-1}$，将$\boldsymbol{Q}$代入上式，得$\boldsymbol{B}=\boldsymbol{Q}^{T}\boldsymbol{A}$。

证毕.

文献[27]中的单遍历 SVD 算法与不做幂迭代的基础随机化 SVD 算法（算法 2）在数学上是等价的，故定理 1 也表明算法 4 与文献[27]里的单遍历 SVD 算法也是等价的。

对于实对称矩阵，文献[24]提出的技术可以保证得到低秩矩阵也是对称的，从而提高结果的准确性。将它与算法 4 结合，得到算法 5。

算法 5. 针对实对称阵的单遍历 SVD.

输入：矩阵$\bar{\boldsymbol{M}}\in\mathbb{R}^{n\times n}$，奇异值截断参$k$，过采样参数$s$。

输出：$\boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{\Sigma}\in\mathbb{R}^{k\times k},\boldsymbol{V}\in\mathbb{R}^{n\times k}$。

1. $\boldsymbol{\Omega}=randn(n,k+s)$
2. $\boldsymbol{Y}=[\,]$
3. $\boldsymbol{W}=zeros(n,k+s)$
4. FOR $i=1,2,\cdots,m$ DO
5. $\quad\boldsymbol{y}=\bar{\boldsymbol{M}}_i\boldsymbol{\Omega},\quad\boldsymbol{Y}=[\boldsymbol{Y};\boldsymbol{y}]$
6. $\quad\boldsymbol{W}=\boldsymbol{W}+\bar{\boldsymbol{M}}_i^{T}\boldsymbol{y}$
7. END FOR
8. $[\boldsymbol{Q},\boldsymbol{R}]=qr(\boldsymbol{Y})$
9. $\boldsymbol{Z}=[\boldsymbol{Q},\boldsymbol{WR}^{-1}]$
10. $[\boldsymbol{P},\boldsymbol{T}]=qr(\boldsymbol{Z})$
11. $\boldsymbol{T}_1,\boldsymbol{T}_2$分别是矩阵$\boldsymbol{T}$的前$k+s$列和后$k+s$列
12. $\boldsymbol{S}=(\boldsymbol{T}_1\boldsymbol{T}_2^{T}+\boldsymbol{T}_2\boldsymbol{T}_1^{T})/2$
13. $[\widehat{\boldsymbol{U}},\widehat{\boldsymbol{\Lambda}}]=eig(\boldsymbol{S})$
14. ind为$\widehat{\boldsymbol{\Lambda}}$中$k$个绝对值最大的特征值的位置索引集合
15. $\boldsymbol{U}=\boldsymbol{P}\widehat{\boldsymbol{U}}(:,ind),\boldsymbol{\Sigma}=abs\left(\widehat{\boldsymbol{\Lambda}}(ind,ind)\right)$
16. $\boldsymbol{V}=\boldsymbol{U}sign(\widehat{\boldsymbol{\Lambda}}(ind,ind))$

算法 5 的第 9 步是由于，根据算法 4，我们有$\boldsymbol{B}=\boldsymbol{R}^{-T}\boldsymbol{W}^{T}$，则公式(12)中的$[\boldsymbol{Q},\boldsymbol{B}^{T}]=[\boldsymbol{Q},\boldsymbol{WR}^{-1}]$。第 14~16 步与算法 3 不同，是由于这里要求奇异值分解，所以需保证$\boldsymbol{\Sigma}$的对角元非负。

4.5 eNetMF算法与相关分析

结合算法 1、算法 3 和算法 5，我们得到基于高效矩阵分解的网络嵌入算法 eNetMF，如算法 6 所示。

算法 6 基于高效矩阵分解的网络嵌入方法(eNetMF).

输入：矩阵$\boldsymbol{A}\in\mathbb{R}^{n\times n}$，特征值截断参数$h$，奇异值截断参数$k$，批处理大小参数$v$，过采样参数$s$，归一化参数$\alpha$。

输出：网络嵌入$\boldsymbol{E}\in\mathbb{R}^{n\times k}$。

1. 用算法 3 计算截断特征值分解：$\boldsymbol{D}^{-\alpha}\boldsymbol{A}\boldsymbol{D}^{-\alpha}\approx\boldsymbol{G}_h\boldsymbol{H}_h\boldsymbol{G}_h^{T}$
2. 计算$\boldsymbol{K}=\boldsymbol{G}_h^{T}\boldsymbol{D}^{-1+2\alpha}\boldsymbol{G}_h\boldsymbol{H}_h$
3. 计算$\boldsymbol{F}=\boldsymbol{D}^{-1+\alpha}\boldsymbol{G}_h$以及$\boldsymbol{C}=\boldsymbol{H}_h\left(\sum_{r=1}^{q}\boldsymbol{K}^{r-1}\right)$
4. $\boldsymbol{\Omega}=randn(n,k+s)$
5. $\boldsymbol{Y}=[\,]$
6. $\boldsymbol{W}=zeros(n,k+s)$
7. FOR $i=1,2,\cdots,n/v$ DO
8. $\quad\boldsymbol{F}_i=\boldsymbol{F}[iv-v+1:iv,\ :]$
9. $\quad\bar{\boldsymbol{M}}_i=trunc\_log^{\circ}\left(\frac{vol(G)}{bq}\boldsymbol{F}_i\boldsymbol{C}\boldsymbol{F}^{T}\right)$
10. $\quad\boldsymbol{y}=\bar{\boldsymbol{M}}_i\boldsymbol{\Omega},\quad\boldsymbol{Y}=[\boldsymbol{Y};\boldsymbol{y}]$
11. $\quad\boldsymbol{W}=\boldsymbol{W}+\bar{\boldsymbol{M}}_i^{T}\boldsymbol{y}$
12. END FOR
13. 执行算法 5 的第 8~15 步
14. $\boldsymbol{E}=\boldsymbol{U}\boldsymbol{\Sigma}^{1/2}$

算法 6 的第 1 步使用算法 3 实现，对中小规模网络α可以取 1/2，对于大规模网络可以通过试验得到更合适的取值，从而提高运行效率、减少近似误差。第 2～12 步实现了矩阵$\boldsymbol{M}$和$\bar{\boldsymbol{M}}$的隐式构造，并对$\bar{\boldsymbol{M}}$的行进行批处理的单遍访问，为后续计算截断 SVD 做准备。这里，为了算法描述简洁、且不失一般性，只考虑了n能被v整除的情况。

下面分析算法 6 的计算量和空间开销，设$l=k+s$。第 1 步的时间复杂度为$O(nnz(\boldsymbol{A})h+nh^{2})$，空间复杂度均为$O(nnz(\boldsymbol{A})+nh)$。算法 6 的第 2、第 3 步包含$O(nh^{2})$次浮点运算，第 7～12 步

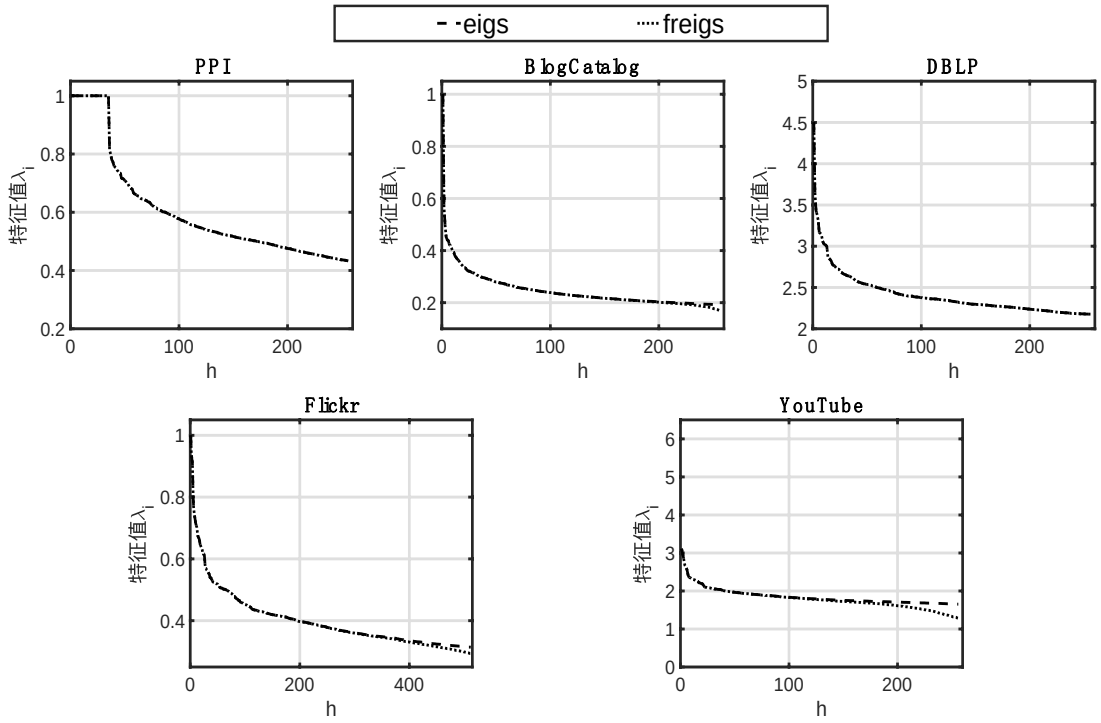

图 2 算法 3(freigs)与 eigsh 函数对归一化网络矩阵进行截断特征值分解的结果.

包含$O(n^2(h+l))$次浮点运算，第 13、14 步的浮点运算次数为$O(nl^2+l^3)$。由于n远大于其他参数的值，算法 6 总的计算复杂度为$O\left(n^2(h+l)\right)$。在空间开销方面，第 1 步需要的存储量是$O(nnz(\boldsymbol{A})+nh)$，在后续计算步骤中，需要存储的矩阵尺寸都不超过$n\times h$, $n\times l$或$n\times v$，考虑到矩阵$\boldsymbol{A}$在后续不再使用，算法 6 的空间复杂度为$O(n(h+l+v))$。

可以看出，eNetMF 算法在空间复杂度方面大大优于 NetMF 算法。NetSMF 算法[8]的空间复杂度为$O\left(cq\cdot nnz(\boldsymbol{A})\right)$，其中$c$一般取值 1000 或者 10000，因此我们的 eNetMF 算法在内存用量方面也显著小于 NetSMF 算法的内存用量。由于 eNetMF 的时间复杂度仍为$O\left(n^2(h+l)\right)$，它处理特别大的网络时会花费过多的计算时间，将来可以考虑采用分布式并行计算等方式来加以改进。

5 实验

本文算法均使用 C 语言编程实现，其中用到的矩阵计算和 QR 分解、LU 分解、SVD 分解等算法通过调用 Intel MKL 的库函数进行实现，同时采用 OpenMP 编程实现了多线程计算。所有实验均运行于装有 32 核 Intel(R) Xeon(R) CPU、256GB 内存的 Linux 工作站上，实验结果中的时间均为 32 线程并行计算的实测运行时间。

下面我们首先介绍测试的网络数据，然后验证 4.2 节提出的特征值分解算法 freigs 的准确度与有效性，最后通过多标签分类的实验验证 eNetMF 算法的优势，并给出更多实验结果。

5.1 数据集

我们选择如下五个网络数据集来进行测试。

- PPI[29]：该数据集是一个蛋白质和蛋白质交互的网络子图，数据集里节点的标签是从标志性基因集获取的，可以表示生物学状态。
- BlogCatalog[30]：数据集是一个线上博客主的社交关系的网络，其中节点标签表示博客主的兴趣。
- DBLP[31]：该数据集是一个学术引用网络，网络中的节点是作者，标签是他们所活跃的会议。
- Flickr[30]：该据集是一个 Flickr 网站上的用户互相联系的网络，标签代表了用户的兴趣组。
- YouTube[30]：数据集是一个视频分享网站上的用户交互网络，用户的标签为他们喜欢的视频的类型。

其中，PPI 和 BlogCatalog 规模相对较小但是在网络嵌入方面的研究工作中被广泛使用，而 DBLP、Flickr 和 YouTube 的规模相对较大。这些数据集的一些具体信息如表 1 所示。

表 1 数据集的具体信息

数据集名称	PPI	BlogCatalog	DBLP	Flickr	YouTube
点个数	3890	10312	51264	80513	1138499
边个数	76584	333983	127968	5899882	2990443
标签个数	50	39	60	195	47

5.2 特征值分解算法freigs的有效性

计算截断奇异值分解的传统算法是 ARPACK 算法[23]，Python 里 scipy 库的 eigs、eigsh 函数是基于它、且使用 Intel 的 MKL 实现的，下面将我们的算法 3（即 freigs）与处理对称稀疏矩阵的 eigsh 函数做比较，后者也就是 NetMF[7]中所使用的。

测试的矩阵是由表 1 中 5 个数据集得到的矩阵$\boldsymbol{D}^{-\alpha}\boldsymbol{A}\boldsymbol{D}^{-\alpha}$，除了对 DBLP 和 YouTube 我们分别设$\alpha = 0.3$和$\alpha = 0.375$外，对其他数据集我们都设α值为缺省的 0.5。为了与后面的网络嵌入实验保持一致，对 PPI、BlogCatalog、DBLP 和 YouTube 这四个数据集，特征值截断参数$k = 256$，对 Flickr 我们取$k = 512$。对于幂迭代参数和过采样参数，除了 DBLP、Flickr 外都取$p = 10,\ s = 50$。Flickr 对应的矩阵较稠密、且特征值截断参数较大，我们取$p = 6,\ s = 100$，DBLP 对应的归一化网络矩阵特征值下降缓慢，我们取$p = 16,\ s = 256$。我们的算法 3（freigs 算法）与 eigsh 的运行时间列于表 2 中。从中可看出，随机化特征值分解算法在大数据集上比传统算法快接近 10 倍。

表 2 算法 3 (freigs)与 eigsh 的运行时间比较（单位:秒）

	PPI	BlogCatalog	DBLP	Flickr	YouTube
eigsh	3.0	11.0	81	415	1320
freigs	0.7	1.9	9.0	79	138
加速比	4.3X	5.8X	9X	5.3X	9.6X

图 2 显示了算法 3 和 eigsh 计算出的最大特征值曲线。从中可以看出，两者的结果几乎是不可区分的，这反映出随机化特征值算法 freigs 有较好的准确度。通过后续多标签分类问题上的实验结果，我们也可以看出 freigs 算法得到的截断特征值分解几乎不会影响网络嵌入的性能。

5.3 eNetMF算法在多标签分类问题上的效果

5.3.1 实验参数设置与评价指标

我们将本文提出的 eNetMF 网络嵌入方法与 DeepWalk、NetMF、NetSMF、ProNE 四种方法进行对比，对比的程序是这四种方法的作者在文献[3][7][8][9]中给出的程序。根据文献[7]，对于所有的方法在不额外说明的情况下都默认设置成窗口大小$q = 10$、负采样参数$b = 1$以及嵌入映射维度$k = 128$。在 NetMF 和 eNetMF 算法中，对归一化网络矩阵作截断特征值分解的参数h基本上都取值 256，只是对 Flickr 令$h = 512$。

对于 DeepWalk 算法，我们按照文献[3]的作者建议的参数进行设置，即随机游走的长度取 40，每一个点的游走数量取 80 以及窗口大小的参数取 10。

对于 NetSMF 算法，需要设置网络稀疏化的非零元参数M。我们和文献[8]保持一致，M的值几乎都是$10^3 \times q \times nnz(\boldsymbol{A})/2$，只有对 BlogCatalog，$M = 10^4 \times q \times nnz(\boldsymbol{A})/2$。

对于 ProNE 算法，我们和文献[9]设置相同的参数，即切比雪夫扩展次数为 10，拉普拉斯算子里的$\mu = 0.2,\ \theta = 0.5$。

对于 eNetMF 方法，除了在 5.2 节已经介绍的α参数和进行第一步快速随机化特征值分解所需要的幂迭代参数、过采样参数外，还需要设置逐行遍历矩阵$\boldsymbol{M}$的批处理参数v和最后单遍历奇异值分解中使用的过采样参数s。批处理参数v影响内存开销以和矩阵乘法的计算时间，较大的v会使得矩阵乘法加快，但也导致峰值内存用量较大。我们对大多侧例使用$v = 3200$，而对于最大的 YouTube 设置$v = 12800$。对最后的过采样参数，统一取$s = 100$。在上述各种方法得到网络嵌入后，与文献[3][7][8][9]一样我们将其应用到多标签分类问题上，通过分类的准确性来评价不同网络嵌入表示的性能。对各个数据集，我们随机采样一部分的带标签节点作为训练数据，剩下的作为测试数据。对于 PPI 和 BlogCatalog，训练节点所占比例从 10%起每次增加 10%递增到 90%，总共进行 9 组节点分类的实验。对于 DBLP、Flickr 和 YouTube，训练节点所占比例从 1%起逐渐增到 10%，共进行 10 组节点分类的实验。在实验中，我们使用由 LIBLINEAR[32]库实现的一对多的逻辑回归模型进行多标签分类。在测试阶段，逻辑回归模型生成排序的标签、而不是确切的标签分配，为了避免阈值效应，我们假定对于测试数据，标签的数量是给定的[3][33]。对于每个实验，我们重复预测 10 次，然后观测不同网络嵌入方法得出的平均 Micro-F1、平均 Macro-F1 指标[34]和准确率指标[35]。F1分数指标的定义为：

$$\mathrm{F1} = \frac{2PR}{P+R}\ ,\#(18)$$

其中P表示精确率，R表示召回率。计算 Micro-F1

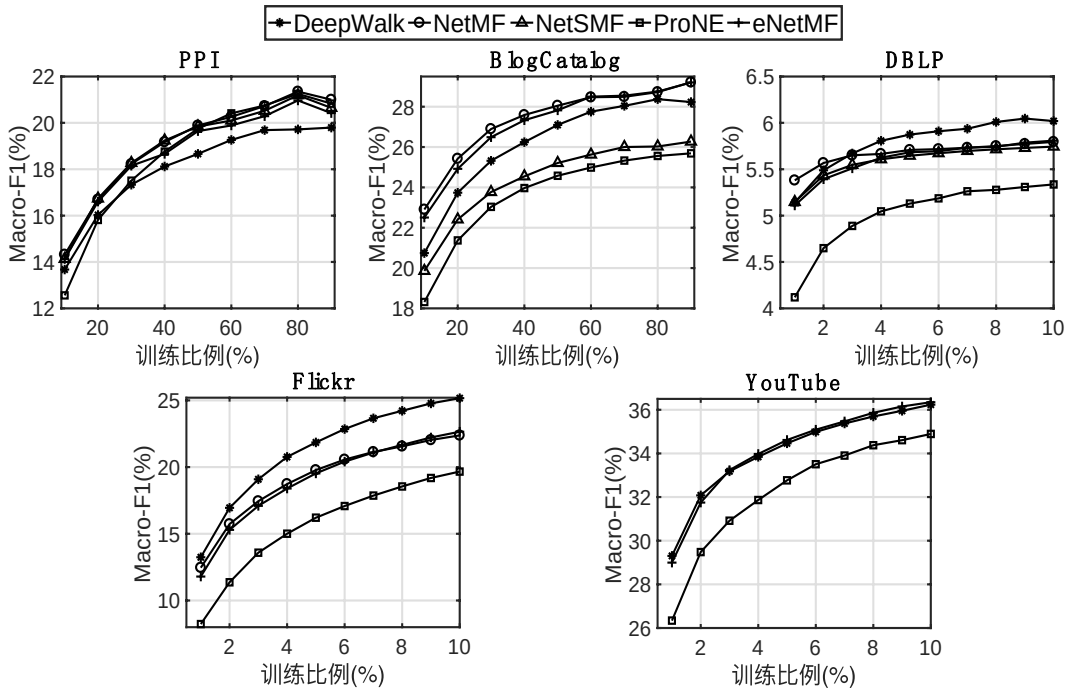

图 3 使用不同网络嵌入算法后进行节点多标签分类得到的 Marco-F1 值 (横轴为训练数据所占的比例).

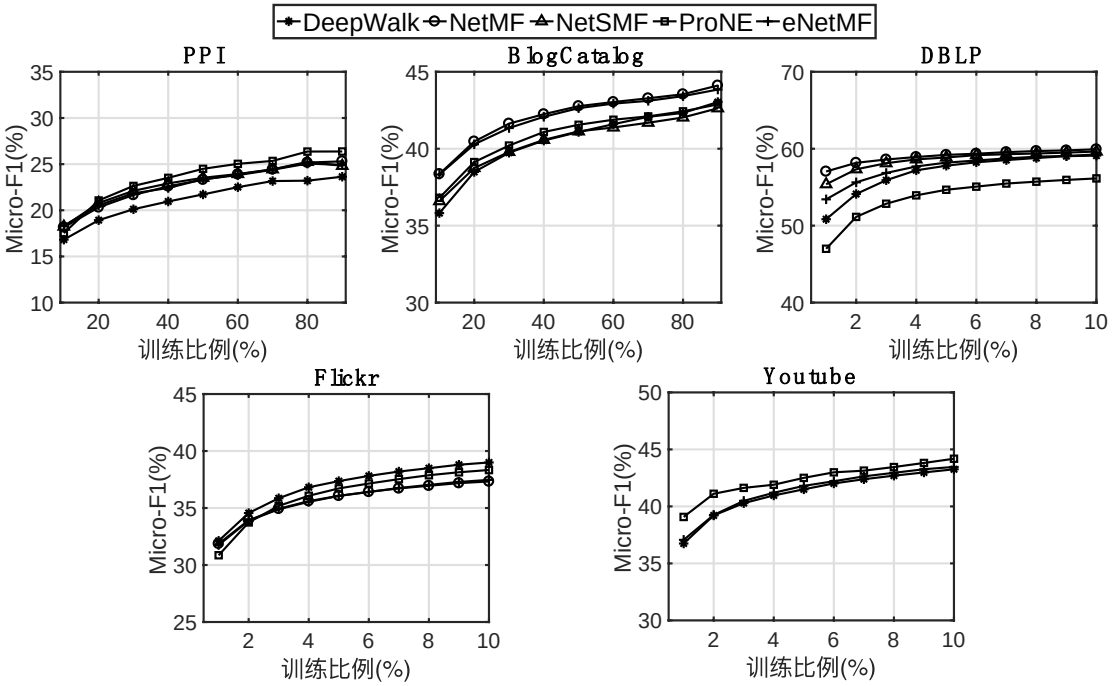

图 4 使用不同网络嵌入算法后进行节点多标签分类得到的 Micro-F1 值 (横轴为训练数据所占的比例).

时，先将所有类别的真正例数量（TP）、假正例数量（FP）和假负例数量（FN）相加得到总体的$\overline{TP}$、$\overline{FN}$、$\overline{FP}$值以及总体精确率$\bar{P}$和召回率$\bar{R}$。

$$\overline{TP} = TP_{class\,1} + TP_{class\,2} + \cdots TP_{classN} \quad ,\#(19)$$

$$\overline{FP} = FP_{class\,1} + FP_{class\,2} + \cdots FP_{classN} \quad ,\#(20)$$

$$\overline{FN} = FN_{class\,1} + FN_{class\,2} + \cdots FN_{classN} \quad ,\#(21)$$

$$\bar{P} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} \quad ,\#(22)$$

$$\bar{R} = \frac{\overline{TP}}{\overline{TP} + \overline{FN}} \quad .\#(23)$$

然后，类似公式(18)计算 Micro-F1：

$$\text{Micro-F1} = \frac{2\bar{P}\bar{R}}{\bar{P} + \bar{R}} \quad .\#(24)$$

计算 Macro-F1 时，则是分别计算每个类别的F1分数，然后求平均：

$$\text{Macro-F1} = \frac{F1_{class\,1} + \cdots + F1_{classN}}{N} \quad .\#(25)$$

在计算多标签分类的准确率时，我们先计算一个节点的多标签准确率为预测正确的标签除以总共的标签数量，然后取整体的分类准确率为所有节点的平均：

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|T_i \cap S_i|}{|T_i \cup S_i|} \quad ,\#(26)$$

其中T_i表示正确的标签集，S_i表示预测的标签集。

5.3.2 实验结果

图 3 和图 4 显示了不同网络嵌入算法在 5 个数据集上进行多标签节点分类的性能。其中，由于所需内存超出了 256GB，NetSMF 对 Flickr 和 YouTube 的实验结果无法获得，而由于同样的原因 NetMF 对 YouTube 的结果也无法获得。从图 3 和图 4 可以看出，无论是 Macro-F1 还是 Micro-F1，使用 eNetMF 算法后的结果都与使用 NetMF 得到的一致（尤其在训练比例较大时），这说明本文提出的改进 NetMF 算法的技术并没有给网络嵌入的质量带来明显的影响。DeepWalk 方法在 PPI、BlogCatalog、DBLP 和 YouTube 上有和 eNetMF 差不多甚至更差的效果，仅在 Flickr 上要好于我们的方法。ProNE 方法虽然在某些数据上得到的 Micro-F1 值最好，但在所有测试数据上其得到的 Macro-F1 值明显比其他方法的差，这说明 ProNE 得到的嵌入表达没有包含图中足够的信息，因此对一些标签较少的节点分类效果不够好。

网络嵌入算法在多标签分类问题上的准确率结果列于表 3 中。我们记录当训练节点所占比例为 60%时的 PPI 和 BlogCatalog 的准确率结果，以及当训练节点所占比例为 6%时的 DBLP、Flickr 和 YouTube 的准确率结果。从表 3 的结果可以看出，我们的方法和 NetMF 的效果差不多，在所有数据集上与其他对比对象比较均有差不多或者更好的准确率。

表 3 各种网络嵌入算法的多标签分类准确率（%）

	PPI	BlogCatalog	DBLP	Flickr	YouTube
DeepWalk	17.082	40.143	58.280	34.351	98.247
NetMF	18.020	41.118	59.919	33.173	N.A.
NetSMF	18.621	39.361	59.287	N.A	N.A.
ProNE	18.985	39.529	55.527	33.485	98.289
eNetMF	18.148	40.958	58.352	33.080	98.256

各种网络嵌入算法的运行时间列于表 4 中，所有算法对应的程序都是适合用 32 线程运行的并行程序，且时间包括了读取数据文件的时间。从表 4 可以看出，ProNE 算法的运行时间最短，eNetMF 算法在矩阵规模不大的时候，速度也比较快。在 Flickr 数据上其相对于 NetMF 的加速比高达 41 倍。而 NetSMF 虽然在处理 DBLP 时也比 NetMF 耗时短，但在处理更大例子是由于内存需求太大而无法运行。DeepWalk 算法是所有对比对象里运行时间最久的，eNetMF 算法相对于 DeepWalk 算法在每个数据集上的加速比都要超过 20 倍。

表 4 各种网络嵌入算法的运行时间（单位:秒）

	PPI	BlogCatalog	DBLP	Flickr	YouTube
DeepWalk	318	816	4540	7920	108015
NetMF	14	84	632	6120	N.A.
NetSMF	19	1140	82	N.A	N.A.
ProNE	1.4	5	11	72.7	194.3
eNetMF	1.5	4.6	39	150	4852
加速比	9.3X	18X	16X	41X	--

对于 NetMF、NetSMF、ProNE 和 eNetMF 这四种算法的峰值内存用量列于表 5。从中可以看出，对 Flickr 数据 eNetMF 的内存用量仅为 NetMF 的 1/46。处理最大的 YouTube 数据，eNetMF 只需 120GB 内存即可算出网络嵌入，并得到非常好的网络节点分类结果（如图 3、4）。虽然在运行时间和内存用量上，本文的方法不如 ProNE，但是如前面所展示的，本文方法在多标签分类结果的效果上更稳定、更好。

表 5 各种网络嵌入算法的峰值内存用量（单位:GB）

	PPI	BlogCatalog	DBLP	Flickr	YouTube
NetMF	0.39	2.2	31	182	>256
NetSMF	6.86	135	18.2	>256	>256
ProNE	0.16	0.46	0.72	5.5	12.0
eNetMF	0.29	0.55	2.0	4.0	120
缩小倍数	1.3X	4.0X	16X	46X	--

对 eNetMF 算法各部分的运行时间进行详细统计后发现，第一步随机化特征值分解和最后的单遍历奇异值分解所花费时间占比例很小，而两者之间的矩阵计算部分（算法 6 的第 7~12 步）消耗时间最多。例如，对 YouTube 数据，矩阵计算部分花费的时间为 4574 秒，占总运行时间 90%以上，而单遍历奇异值分解的时间仅为 35 秒。对于该数据，

我们还比较了算法 6 中的单遍历奇异值分解与文献[27]中的单遍历奇异值分解算法，后者的运行时间为 1434 秒，可见我们提出的简洁单遍历算法与保证分解结果对称的技术显著加速了计算、也有很高的准确度。

5.4 eNetMF算法在边预测问题上的效果

5.4.1 实验设置与评价指标

我们将本文提出的 eNetMF 网络嵌入算法和 DeepWalk、NetMF、NetSMF、ProNE 在网络的边预测问题上进行比较，这些方法的参数设置和 5.3 节所介绍的一致。边预测问题旨在基于现有的网络结构来预测可能存在的边。在我们的实验当中，我们随机地抽取原始图 30%的边作为测试集，剩余 70%的边作为训练集。首先将不同的算法在训练集上进行学习得到训练集的网络嵌入结果，然后根据网络嵌入结果来计算每一个在测试集的点对(u,v)上的相似度分数，最后采用广泛使用的 AUC 指标[5]来评价边预测质量的好坏。在上述过程中，计算点对(u,v)的相似度分数可以按照点对(u,v)对应的嵌入向量的内积、余弦距离或欧式距离来计算，也可以使用边特征方法[5]来计算。对于每个网络嵌入算法，我们选取使其 AUC 指标结果最好的相似度分数计算方法并记录 AUC 的结果。

5.4.2 实验结果

各种网络嵌入算法在边预测问题上的 AUC 指标的结果列于表 6 中。从表 6 可以看出在 PPI 和 Flickr 数据集上 eNetMF 算法比 DeepWalk、ProNE 要差，但在数据集 BlogCatalog、DBLP 和 YouTube 上，eNetMF 算法的结果比其他方法好或差不多。整体来看，eNetMF 的效果和 NetMF 的效果相比差不多甚至更好，特别是在 DBLP 数据集上，eNetMF 算法显著更好，这是因为 DBLP 的归一化网络矩阵的特征值衰减非常缓慢，NetMF 算法里的 eigsh 函数在不设置迭代次数的时候很难收敛，而设置了迭代次数的特征值计算的结果有一定的误差，故该数据集的结果也验证了本文第 4.3 节提出的矩阵高效率构建方法的有效性。边预测实验验证了 eNetMF 算法具有很好的推断预测的能力，这得益于我们的模型和 NetMF 一样，较好地保存了网络的结构信息。

表 6 各种网络嵌入算法在边预测问题上的 AUC 值

	PPI	BlogCatalog	DBLP	Flickr	YouTube
DeepWalk	0.743	0.880	0.848	0.923	0.779
NetMF	0.737	0.878	0.857	0.879	N.A.
NetSMF	0.764	0.883	0.848	N.A.	N.A.
ProNE	0.797	0.886	0.845	0.926	0.805
eNetMF	0.735	0.876	0.916	0.871	0.805

6 结论与展望

本文在网络嵌入问题上，提出了一个基于高效随机化矩阵分解的网络嵌入方法，该方法显著减少了 NetMF[7]模型的内存开销和运行时间。本文首先提出了一种快速随机化特征值分解算法，用以加速了归一化网络矩阵求特征值分解的速度，该算法展现出相比传统的稀疏矩阵截断特征值分解方法近十倍的加速，且几乎不损失精度。然后提出了一种简洁的且适用于对称矩阵的单遍历奇异值分解算法，利用该算法避免了显式存储稠密矩阵，使得算法的内存开销跟矩阵维度呈线性关系，适合于处理大规模网络数据。最后，结合以上技术，提出了基于高效随机化矩阵分解的网络嵌入算法 eNetMF，并将其应用于实际网络数据，基于得到的低维嵌入进行多标签分类和边预测。多标签分类的实验结果表明，eNetMF 在各个数据集上的评测结果均与 NetMF 得到的结果差不多，显著优于 NetSMF 和 ProNE 等其他网络嵌入方法，而且 eNetMF 方法内存用量小、运行速度快。对于节点数超过 1 百万、NetMF 与 NetSMF 由于内存需求太大无法处理的 YouTube 数据，eNetMF 只需 1.3 个小时和 120GB 内存开销即可得到网络嵌入，并进而获得很好的多标签分类结果。边预测的实验结果表明 eNetMF 算法也表现出与其他方法差不多甚至更好的性能。

本文提出的方法，虽然在运行时间上相比 NetMF 有显著的优势，但其时间复杂度仍为$O(n^2(h+l))$，其中 n 为网络节点数目。因此，在它在处理特别大的网络时需要很长、甚至不能忍受的时间。后续我们将考虑用 GPU 并行计算或分布式计算对改算法进行加速，使得它在保证准确的同时能够处理更大规模的网络数据。其次，本文通过随机化矩阵计算方法改进 NetMF 方法，取得了一定的效果，后续还可以研究随机化矩阵方法在其他大数据挖掘和处理问题上的应用，期望在保证结果准确的同时减少算法的时间和内存开销。此外，对于本文实验中体现出的 ProNE 算法得到的网络嵌入结果不稳定的问题，很可能是由于 ProNE 算法做了近似的信息传播增强，导致在一些稀疏图数据上的

表现较差。这种信息传播过多导致的过度平滑问题最近也得到学术界的关注，将来我们也计划针对该问题开展工作，希望研究出既保证网络嵌入质量又具有非常低算法复杂度的网络嵌入方法。

参 考 文 献

[1] Zhang F, Liu X, Tang J, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs//Proceedings of the Twenty-Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). Anchorage, USA, 2019: 2585-2595.

[2] Hamilton, W.L., Ying, R., Leskovec, J. Representation Learning on Graphs: Methods and Applications. IEEE Data(base) Engineering Bulletin, 2017, 40: 52–74.

[3] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations//Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. New York, USA, 2014: 701-710.

[4] Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding//Proceedings of the 24th international conference on World Wide Web. Florence, Italy, 2015: 1067-1077.

[5] Grover A, Leskovec J. node2vec: Scalable feature learning for networks//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. San Francisco, USA, 2016: 855-864.

[6] Dong Y, Chawla N V, Swami A. metapath2vec: Scalable representation learning for heterogeneous networks//Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. Halifax, Canada, 2017: 135-144.

[7] Qiu J, Dong Y, Ma H, et al. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec//Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. Maina Del Rey, USA, 2018: 459-467.

[8] Qiu J, Dong Y, Ma H, et al. Netsmf: Large-scale network embedding as sparse matrix factorization//Proceedings of the 28th international conference on World Wide Web. San Francisco, USA, 2019: 1509-1520.

[9] Zhang J, Dong Y, Wang Y, et al. ProNE: fast and scalable network representation learning//Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China. 2019: 4278-4284.

[10] Halko N, Martinsson P G, Tropp J A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. Society for Industrial and Applied Mathematics review, 2011, 53(2): 217-288.

[11] Drineas P, Mahoney M W. RandNLA: randomized numerical linear algebra. Communications of the ACM, 2016, 59(6): 80-90.

[12] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality//Advances in neural information processing systems. Lake Tahoe, USA, 2013: 3111-3119.

[13] Chung F R K, Graham F C. Spectral graph theory. Providence, Rhode Island:American Mathematical Society. 1997.

[14] Tang L, Liu H. Relational learning via latent social dimensions//Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. Paris, France, 2009: 817-826.

[15] Yan D, Huang L, Jordan M I. Fast approximate spectral clustering//Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. Paris, France, 2009: 907-916.

[16] Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering//Advances in neural information processing systems. Vancouver, Canada, 2002: 585-591.

[17] Levy O, Goldberg Y. Neural word embedding as implicit matrix factorization//Advances in neural information processing systems. Montreal, Quebec, Canada, 2014: 2177-2185.

[18] Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information//Proceedings of the 24th ACM international on conference on information and knowledge management. Melbourne, Australia, 2015: 891-900.

[19] Ou M, Cui P, Pei J, et al. Asymmetric transitivity preserving graph embedding//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. San Francisco, USA, 2016: 1105-1114.

[20] Yang C, Liu Z, Zhao D, et al. Network representation learning with rich text information//Twenty-Fourth International Joint Conference on Artificial Intelligence. Buenos Aires, Argentina, 2015: 2111-2117.

[21] Zhu D, Cui P, Zhang Z, et al. High-order proximity preserved embedding for dynamic networks. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(11): 2134-2144.

[22] Eckart C, Young G. The approximation of one matrix by another of lower rank. Psychometrika, 1936, 1(3): 211-218.

[23] Lehoucq R B, Sorensen D C, Yang C. ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods. Philadelphia:Society for Industrial and Applied Mathematics, 1998.

[24] Tropp J A, Yurtsever A, Udell M, et al. Practical sketching algorithms for low-rank matrix approximation. SIAM Journal on Matrix Analysis and Applications, 2017, 38(4): 1454-1485.

[25] N. J. Higham, Matrix nearness problems and applications//Proceedings of the IMA Conference on Applications of Matrix Theory. New York, USA, 1989: 1–27.

[26] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge, UK:Cambridge University Press, 2004 (Section 4.2.3)

[27] Yu W, Gu Y, Li J. Single-pass PCA of large high-dimensional data//Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, Australia, 2017: 3350-3356.

[28] Feng X, Xie Y, Song M, et al. Fast Randomized PCA for Sparse Data//Proceedings of the 10th Asian Conference on Machine Learning. Beijing, China, 2018: 710-725.

[29] Stark C, Breitkreutz B J, Chatr-Aryamontri A, et al. The BioGRID interaction database: 2011 update. Nucleic acids research, 2010, 39(suppl_1): D698-D704.

[30] Social-Dimension Approach to Classification in Large-Scale Networks, http://leitang.net/social_dimension.html 2010,7,29

[31] Tang J, Zhang J, Yao L, et al. Arnetminer: extraction and mining of academic social networks//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. Las Vegas, USA, 2008: 990-998.

[32] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification. Journal of machine learning research, 2008, 9(Aug): 1871-1874.

[33] Tang L, Rajan S, Narayanan V K. Large scale multi-label classification via metalabeler//Proceedings of the 18th international conference on World Wide Web. Madrid, Spain, 2009: 211-220.

[34] Yang Y. An evaluation of statistical approaches to text categorization. Information retrieval, 1999, 1(1-2): 69-90.

[35] Godbole S, Sarawagi S. Discriminative methods for multi-labeled classification//Pacific-Asia conference on knowledge discovery and data mining. Berlin, Germany: Springer,2004: 22-30.

Xie Yuyang, Ph.D. student. His research interest is data mining and matrix analysis.

Feng Xu, Ph.D. student. His research interest is matrix analysis.

Yu Wenjian, Ph.D., associate professor. His research interests include computer aided design of IC and matrix analysis.

Tang Jie，Ph.D., professor. His research interests include social network mining and academic knowledge graph.

Background

In order to process network data effectively, the challenge is to find effective network data representation. The traditional network data representation has become a bottleneck in large-scale network data nowadays. Network embedding, which aims to learn low-dimensional vector representations for network nodes and effectively preserve the network structure, is a promising way of network representation. During the past few years, there is a surge of research on network embedding. DeepWalk, LINE and node2vec models which are skip-gram based have been considered as powerful benchmark solutions for network embedding research. GraRep, HOPE, NetMF, NetSMF and ProNE models are matrix factorization based network embedding methods. NetMF algorithm theoretically unifies several network embedding methods and show the embedding learned by NetMF is as powerful as those learned from the skip-gram based methods. The major disadvantage of existing methods which is based on matrix factorization is the large cost on computation and memory.

In this work, we use fast randomized eigenvalue decomposition (EVD) and single-pass singular value decomposition (SVD) to improve NetMF and present an efficient randomized matrix factorization based network embedding algorithm (eNetMF). Firstly, we propose a fast randomized EVD algorithm (freigs) for sparse matrix. Secondly, we propose to use single-pass SVD approach to process high-order proximity matrix in the NetMF, so that we can avoid storing the large dense matrix and largely reduce the memory cost. Thirdly, we propose a simplified, randomized single-pass SVD algorithm that guarantees a symmetric decomposition result. Combining the above techniques, we finally obtain the eNetMF algorithm. With five real network datasets, we evaluate the efficiency and effectiveness of the eNetMF algorithm on multi-label node classification. Experimental results show that in terms of the metrics Macro-F1 and Micro-F1, the embedding generated by eNetMF has the same performance as NetMF in multi-label node classification. Benefiting from randomized matrix decomposition algorithms, eNetMF is fast and memory-friendly compared to NetMF. We believe the randomized matrix decomposition algorithms are generic and can be applied to matrix factorization based network embedding models, and hope this paper will inspire more future research in this area.

This work is supported by the National S&T Innovation Program of China (Grant No. 2019AAA0103502) and the National Natural Science Foundation (Grant No. 61872206). This work focuses on the randomized matrix decomposition and its application in network embedding. Our group has been studying randomized matrix decomposition and network embedding for years, and has proposed a series of work such as frPCA, NetMF, NetSMF and ProNE.