第 4 章　ProNE: 快速的图嵌入表示

在这个章节中, 我们展示了 ProNE——一个在大规模图数据上非常高效快速同时可扩展性强的图嵌入算法。ProNE 主要由展示在图 4.1中的两个步骤组成。首先，它将图嵌入表示的问题化为稀疏矩阵分解问题，以高效地实现初始的结点的嵌入表示。接着，它借助高阶的 Cheeger 不等式来调制图的谱空间，然后把第一步学到的嵌入表示在调整后的图上传播，从而达到将局域的平滑信息（localized smoothing information）和全局的聚类信息（global clustering information）融合进图嵌入表示中。

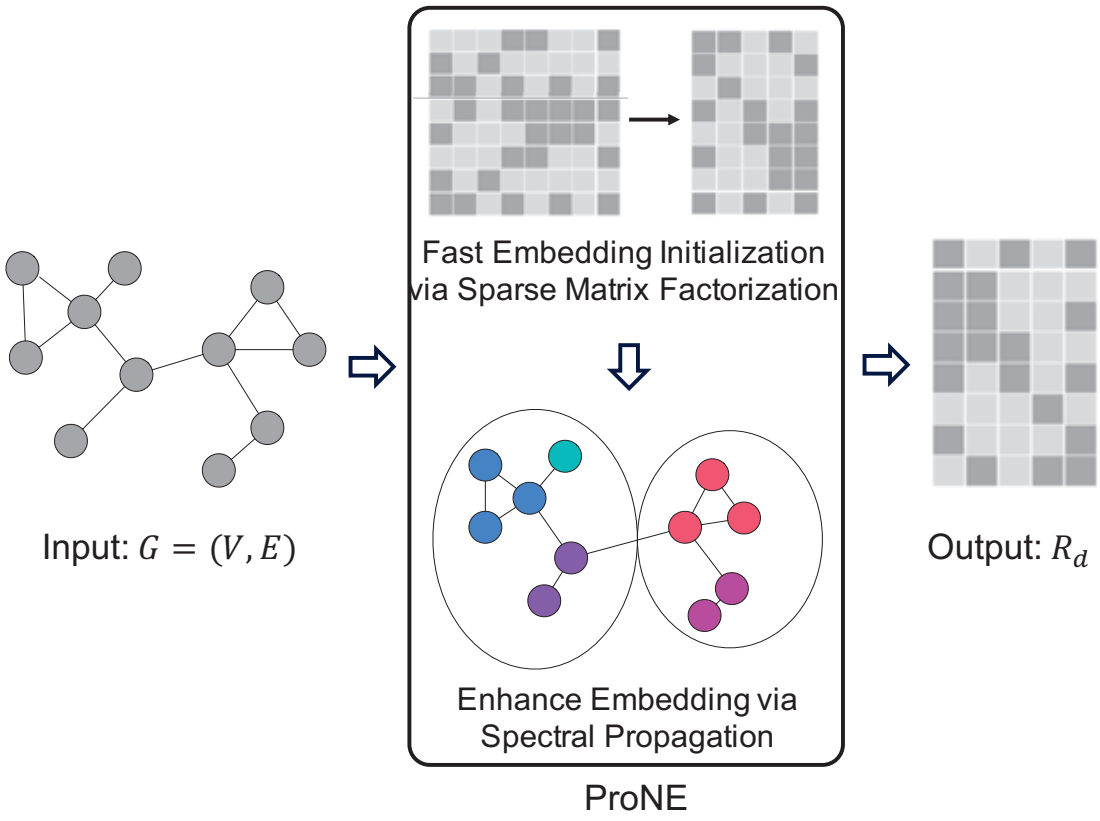


图 4.1　ProNE 模型: 1) 稀疏矩阵分解产生快速的图嵌入作为模型的初始图嵌入表示；2) 在谱空间对图做调制，并将初始的图嵌入在新图上传播，得到最终的图嵌入表示。

4.1　稀疏矩阵分解生成快速图嵌入表示

自然语言处理中的分布假设[38] 在最近启发了词嵌入表示和图嵌入表示的蓬勃发展。在这一小节中我们将展示基于建模分布假设中的相似度的图嵌入问题是如

何转化为矩阵分解的问题，更重要的，我们是如何实现高效的图嵌入。

4.1.1　图嵌入问题转化为稀疏矩阵分解问题

基于分布假设，图中的结点的相似度通常是由结点所在的结构背景 (context) 的差异来度量。我们在这里提出利用最简单的结构——边——来表征结点-背景点对。因此，边集就形成了结点-背景点对集 $\mathcal{D} = E$。形式上，给定了结点 v_i，对于背景点 v_j 的共现概率我们有以下的表示：

$$\hat{p}_{i,j} = \sigma(r_i^T c_j) \tag{4-1}$$

在这里 $\sigma(x) = 1/(1 + e^{-x})$ 是 sigmoid 函数，符号 $r_i, c_i \in R^d$ 分别表示结点 v_i 的嵌入表示向量和背景表示向量。因此，目标函数可以表示为所有边上的对数损失的加权和，如下所示：

$$l = -\sum_{(i,j)\in\mathcal{D}} p_{i,j} \ln \hat{p}_{i,j} \tag{4-2}$$

在这里 $p_{ij} = A_{ij}/D_{ii}$ 表示 (v_i, v_j) 在结点-背景点对集合 $\mathcal{D}$ 中的权重。

为了避免平凡解 (r_i=c_j & $\hat{p}_{i,j}$=1), 对于每一个观测到的点对 (边) (v_i, v_j) 而言, 背景点 v_j 的出现同时伴随着负来自背景分布 $P_{\mathcal{D},j}$ 的负样本, 因此目标函数可以更新为：

$$l = -\sum_{(i,j)\in\mathcal{D}} [p_{i,j} \ln \sigma(r_i^T c_j) + \lambda P_{\mathcal{D},j} \ln \sigma(-r_i^T c_j)] \tag{4-3}$$

在这里 λ 是负样本的比率，结点-背景点对集合上和背景结点 v_j 相关的背景采样分布 $P_{\mathcal{D},j}$ 可以定义为:

$$P_{\mathcal{D},j} \propto \sum_{i:(i,j)\in\mathcal{D}} p_{i,j} \tag{4-4}$$

最小化式子5-21 中的目标损失函数的一个充分条件是让目标损失函数对 $r_i^T c_j$ 的偏导数为 0。由此我们可以得到：

$$r_i^T c_j = \ln p_{i,j} - \ln(\lambda P_{D,j}), \quad (v_i, v_j) \in \mathcal{D} \tag{4-5}$$

注意到 $r_i^T c_j$ 表征的是结点 v_i 的嵌入表示向量和对应的背景结点 v_j 的背景表示之间的点积相似度，我们定义了相似度矩阵 M，其中 M 的元素便是 $r_i^T c_j$, 所以，有

$$M_{i,j} = \begin{cases} \ln p_{i,j} - \ln(\lambda P_{D,j}) & ,(v_i, v_j) \in \mathcal{D} \\ 0 & ,(v_i, v_j) \notin \mathcal{D} \end{cases} \tag{4-6}$$

自然地，基于分布假设的相似性的图嵌入问题就被转换为了矩阵分解。在这里，我们使用截断奇异值分解（tSVD）的方法进行矩阵分解，将目标矩阵 M 近似分解为秩最多为 d 的矩阵的乘积，也就是 $M \approx U_d \Sigma_d V_d^T$，其中 Σ_d 是最大的 d 个奇异值按降序排列的所构成的对角矩阵，$U_d \in R^{n \times d}$ 和 $V_d \in R^{n \times d}$ 是以对应的正交特征向量作为行向量而组成的矩阵。最后，我们初始的图嵌入矩阵可以表示为：

$$R_d \leftarrow U_d \Sigma_d^{1/2} \tag{4-7}$$

其中，R_d 的每一行代表相应结点的嵌入表示向量。对 Σ_d 做开方处理主要是为了考虑对称性，但是由 $M \approx U_d \Sigma_d V_d^T$ 得到图嵌入矩阵的其他方式和背后的意义，我们会在未来工作进一步展开。

4.1.2　稀疏的 Randomized tSVD 实现快速的图嵌入训练

目前为止, 我们展示了如何将基于建模分布假设中的相似性的一大类图嵌入问题理解为矩阵分解问题。然而，用于大规模矩阵（图/网络）的 tSVD 在时间开销和空间负担上仍然是昂贵的。为了实现快速的图嵌入表示学习，我们建议在稀疏的矩阵上使用 randomized tSVD。以陶哲轩等人的随机矩阵理论（random matrix theory）[49] 为基础, randomized tSVD 相对于一般的 tSVD 可以提供显著的速度提升，同时精度的误差也具有强而紧的理论界保证[50]。

在这里，我们展示如何应用 randomized tSVD 来实现快速图嵌入训练。首先，我们可以找一个列正交的矩阵 $Q \in R^{|V| \times d}$, 这样便近似地有 $M \approx QQ^T M$。假设这个 Q 已经找到了, 我们定义 $H = Q^T M$, 现在 $H \in R^{d \times |V|}$ 相对 M 而言是一个小矩阵, 故可以通过正常的 SVD(或者重复这一小节的随机矩阵降维的方法，化为 $R^{d \times d}$ 后再进行正常的 SVD)。对 H 进行 SVD 后，我们得到 $H = S_d \Sigma_d V_d^T$，其中 $S_d \in R^{|V| \times d}$ 和 $V_d \in R^{|V| \times d}$ 是列正交的矩阵，$\Sigma_d \in R^{d \times d}$ 是奇异值按降序排列形成的对角矩阵。最

后 M 就可以表示表示为：

$$M \approx QQ^T M = QH = Q(S_d \Sigma_d V_d^T) = (QS_d)\Sigma_d V_d^T \tag{4-8}$$

我们令 $U_d = QS_d$，便可以结合式子4-7得到图嵌入表示矩阵 R_d:

$$R_d \leftarrow U_d \Sigma_d^{1/2} = QS_d \Sigma_d^{1/2} \tag{4-9}$$

随机矩阵理论可以让我们高效地找到符合要求的 Q。首先是生成高斯随机矩阵 $\Omega \in R^{|V| \times d}$，它的元素服从 $\Omega_{ij} \sim \mathcal{N}(0, \frac{1}{d})$。令 Y=$M\Omega$，然后对 Y 做 QR 分解 (可以通过 Householder reflections)，也就有 Y=QR, 这里 Q 就是符合要求的列正交的矩阵。实际上，还可以通过提高 M 的阶数来让奇异值更快地收敛，从而进一步提高 SVD 的质量。

注意到，受到自然语言处理中词嵌入表示和矩阵分解方法的关系[39] 的启发，最近的一项研究表明，基于 skip-gram 的图嵌入模型可以被视为隐式矩阵分解[6]。然而，先前的工作中隐式分解的矩阵基本都是稠密的，事实上，先前的工作也一般暗示稠密的矩阵才可以捕获图上更全局的结构信息[6]。这边造成了相应的矩阵分解模型 $O(|V|^3)$ 的时间复杂度。然而我们的模型将更全局的结构信息置于提升部分处理，目前为止只涉及到稀疏矩阵的分解，故复杂度是和图的规模大小呈线性的，即 $O(|E|)$ (参见 Eq. 5-13)。

4.2　谱传播作为图嵌入的提升方法

与其他流行的方法类似，例如 DeepWalk 和 LINE，通过稀疏矩阵分解学习的图嵌入表示一般仅能捕获局域的结构信息。为了进一步整合全局网络属性，例如社区结构，三角形结构，等等，我们提出可以通过根据高阶 Cheeger 不等式对图结构进行谱空间的调制，使得调整后的图更加强调局域的平滑信息和全局的聚类信息，然后将上一节中训练得到的初始图嵌入在在新网络中传播，就可以很自然地整合了这些高阶的结构（通过传播，结点的图嵌入会共享到和它同属于一个高阶结构的点的嵌入信息）。因为整个传播过程的关键是先对图做谱空间的调制，所以我们将整个步骤命名为“谱传播”。

注意到谱传播策略是一种通用的将高阶的图信息进一步整合进图嵌入的方法，它一般不依赖所传播的初始图嵌入的具体训练方式。所以，谱传播可以用来提升现有的图嵌入算法（具体的讨论参见图 4.4）。

4.2.1　连接图嵌入表示、图的谱空间和图的划分

高阶 Cheeger 不等式[51,52] 显示了图对应的拉普拉斯矩阵 (Laplacian) 谱空间上的特征值和图空间中的图划分（聚类）之间存在着对应关系。suggests that eigenvalues in graph spectral are closely associated with a network' spatial locality smoothing and global clustering. 我们首先先介绍符号，然后再说明谱空间和图空间的对应关系，并引出利用这种关系来提升图嵌入表示的动机和具体执行细节。

在图的谱分析理论中, 图的拉普拉斯矩阵定义为 $L = D - A$，基于随机游走的归一化拉普拉斯矩阵定义为 $L{=}E_n{-}D^{-1}A$, 其中 E_n 是 n 阶的单位矩阵。我们的方法用到的拉普拉斯矩阵一般特指后者。拉普拉斯矩阵可以分解为 $L{=}U\Lambda U^T$，其中 $\Lambda{=}diag([\lambda_1, ..., \lambda_n])$ 是特征值 $0{=}\lambda_1 \leq \cdots \leq \lambda_n$ 按升序排列形成的对角矩阵，$U \in R^{n\times n}$ 是由和特征值一一对应的特征向量为列向量所组成的矩阵, 它的 i^{th} 列是特征向量 u_i。信号 x 在图上傅里叶变换定义为 $\hat{x} = U^T x$，对应的傅里叶逆变换为 $x = U\hat{x}$。信号 x 在图上的传播可以表示为 $D^{-1}Ax$，从傅里叶变换的角度解释就是信号 x 首先进行傅里叶变换进入了谱空间，然后再谱空间根据特征值进行线性的伸展收缩变化，最后再通过傅里叶逆变换回到了图空间。

我们使用 Cheeger 常数（Cheeger constant），也被称为图的传导 (graph conductance) 来反映图的划分。我们从图的结点集合中抽取一个子集，作为一个图划分，记作 $S \subseteq V$, Cheeger 常数定义为：

$$\phi(S) = \frac{|E(S)|}{min\{vol(S), vol(V - S)\}} \tag{4-10}$$

在这里 $E(S)$ 是有一个结点在划分的结点集 S 中的边的集合，$vol(S)$ 是结点集 S 中的结点的度的和。

Cheeger 常数反映的图的二划分的划分效果，$\phi(S)$ 越小，则图被二划分得越开，划分效果越明显。将 Cheeger 常数推广到将图划分成更多个子图的情形，就是 k 阶 Cheeger 常数（k-way Cheeger constant），它定义为：

$$\rho_G(k) = min\{max\{\phi(S_i) : S_1, S_2, ..., S_k \subseteq V \; disjoint\}\} \tag{4-11}$$

k 阶 Cheeger 常数中，使用 k 划分得到的 k 个划分结点集的 Cheeger 常数的最大值来衡量该 k 划分的效果，并从所有可能的 k 划分中的最好效果 (最小值) 来衡量图能被 k 划分的程度，所以它表示的是图内在的划分（聚类）属性，$\rho_G(k)$ 越小，则图能被 k 划分的效果越好。

k 阶 Cheeger 常数 $\rho_G(k)$ 和图的谱存在着以下的关系：

$$\frac{\lambda_k}{2} \leq \rho_G(k) \leq O(k^2)\sqrt{\lambda_k} \tag{4-12}$$

该不等式叫做高阶 Cheeger 不等式。高阶 Cheeger 不等式中，第 k 小的拉普拉斯矩阵的特征值设置了上下界来控制图的 k 划分指标 $\rho_G(k)$，所以，它架起了图空间和谱空间对应关系的桥梁。

举一个直观简单的例子，在图的谱分析理论中，图的拉普拉斯矩阵中 0 特征值的重数等于图的连通分支的个数[53]。这个结论可以由高阶 Cheeger 不等式直接推导得到。当 $\lambda_k = 0$ 时，显然有 $\rho_G(k) = 0$，由 $\rho_G(k)$ 的定义可以直接推出图存在 k 个连通分支。

从高阶 Cheeger 不等式 4-12中，我们可以推断出，小的拉普拉斯矩阵的特征值控制着图被划分成几个大的子图的划分效果，也就是图的相对全局的聚类效果；大的特征值控制着图被划分成许多个小的子图的划分效果，也就是图的相对局域的聚类效果（局域的聚类效果，信号传播时可以理解为起到一种平滑的效果，所以我们在这篇论文中直接叫它平滑效果）。这一结论可以启发我们通过控制谱空间的大特征值或者小特征值，来控制图的高阶全局或者局域的划分聚类效果，然后我们就可以在新图上传播图嵌入表示，从而将全局的聚类信息或者局域的平滑整合进图嵌入表示，从而提高图嵌入的表达能力，具体的操作为：

$$R_d \leftarrow D^{-1}A(E_n - \widetilde{L})R_d \tag{4-13}$$

在式子 4-13中，$D^{-1}A$ 是归一化的邻接矩阵，也就是调制前的传播转移矩阵，$\widetilde{L} = Ug(\Lambda)U^T$ 是谱空间上拉普拉斯矩阵的调制器，g 就是定义的调制器的形式（数学的意义上也就是函数变换），对应的，$E_n - \widetilde{L}$ 是谱空间上 $D^{-1}A$ 的调制器，它控制了 $D^{-1}A$ 的谱的放缩因子，因此 $D^{-1}A(E_n - \widetilde{L})$ 是调制后的图的归一化“邻接矩阵”，也就是调制后的传播转移矩阵。

为了同时考虑全局和局域的图结构信息，我们设计的谱调制器为 $g(\lambda) = e^{-\frac{1}{2}[(\lambda-\mu)^2-1]\theta}$，在这里 $\mu \in [0, 2]$。因此，我们可以得到拉普拉斯矩阵的调制器：

$$\widetilde{L} = Udiag([g(\lambda_1), ..., g(\lambda_n)])U^T \tag{4-14}$$

带入具体形式可以得到：

$$\widetilde{L} = Udiag([e^{-\frac{1}{2}[(\lambda_1-\mu)^2-1]\theta}, ..., e^{-\frac{1}{2}[(\lambda_n-\mu)^2-1]\theta}])U^T \tag{4-15}$$

$g(\lambda)$ 可以看作是一个带通滤波器[54,55]，函数图线大概呈钟型，它可以让特定值范围内的特征值“通过”（变化不明显）而让值范围外的特征值“不通过”（衰减明显）。一方面，由高阶 Cheeger 不等式中特征值对 $\rho_G(k)$ 的控制效果的推断，对于小特征值的衰减效果，是对图的全局聚类效果的增强；对于大特征值的衰减效果，是对图的局域平滑效果的增强。另一方面，从谱分析本身的角度出发，抑制特征值的极端值，可以有效地减小矩阵的噪声，减小图中连边存在的“随机噪声”。

4.2.2　带通、高通、低通滤波器

事实上，考虑到不同的图对于全局聚类和局部平滑的不同需求，我们也可以用高通滤波器或者低通滤波器来替代带通滤波器，比如低通滤波器可以设计为 $g(\lambda) = e^{-\lambda\theta}$。我们工作中的很多推导，也很容易通过简单的变化就能推广到高通或者低通滤波器。不过鉴于带通滤波器比较通用，可以调整位置参数 μ 来粗略地得到高通或者低通的效果，为了讨论方便，我们在工作中暂时只讨论带通滤波器。对于高通或者低通滤波器的研究，也许同样有惊喜的实验效果。

4.2.3　Chebyshev 多项式展开提升计算效率

我们通过截断 Chebyshev 多项式展开来避免公式 4-15中的显式特征值分解、傅里叶变换和逆变换。有两类 Chebyshev 多项式，我们采用的是第一类 Chebyshev 多项式，它有如下的递归定义：

$$\begin{cases} T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x) \\ T_0(x) = 1 \\ T_1(x) = x \end{cases} \tag{4-16}$$

所以我们对拉普拉斯矩阵的调制器做截断 Chebyshev 多项式展开，有如下形式：

$$\widetilde{L} \approx U\sum_{i=0}^{k-1} c_i(\theta)T_i(\bar{\Lambda})U^T = \sum_{i=0}^{k-1} c_i(\theta)T_i(\bar{L}) \tag{4-17}$$

在这里 $\bar{\Lambda} = -\frac{1}{2}[(\Lambda - \mu E_n)^2 - E_n], \bar{L} = -\frac{1}{2}[(L - \mu E_n)^2 - E_n]$. $\bar{\lambda} = \frac{1}{2}[(\lambda - \mu)^2 - 1] \in [-1, 1]$, 换元后新的滤波器的形式为 $f(\bar{\lambda}) = g(\lambda) = e^{-\bar{\lambda}\theta}$.

根据 ChebyShev 多项式的性质，T_i 在 $[-1, 1]$ 的区间上正交，且正交积分的权重为 $1/\sqrt{1-x^2}$, 也就是说

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0 & , \quad i \neq j \\ \pi & , \quad i = j = 0 \\ \frac{\pi}{2} & , \quad i = j \neq 0 \end{cases} \tag{4-18}$$

所以，$e^{-x\theta}$ 的 Chebyshev 多项式展开可以通过如下的积分得到：

$$c_i(\theta) = \begin{cases} \frac{1}{\pi} \int_{-1}^{1} \frac{T_i(x)e^{-x\theta}}{\sqrt{1-x^2}} dx = (-)^i I_i(\theta) & , i = 0 \\ \frac{2}{\pi} \int_{-1}^{1} \frac{T_i(x)e^{-x\theta}}{\sqrt{1-x^2}} dx = 2(-)^i I_i(\theta) & , i \neq 0 \end{cases} \tag{4-19}$$

在上式中，$I_i(\theta)$ 是著名的特殊函数，第一类修正贝塞尔函数[56]，i 是 $I_i(\theta)$ 的阶数。给定阶数 i 和自变量 θ，$I_i(\theta)$ 的值可以直接通过查表得到（例如，Python 语言的 Scipy 包支持直接查表）。综上，拉普拉斯矩阵的调制器的 Chebyshev 多项式展开如下：

$$\widetilde{L} \approx I_0(\theta)T_0(\bar{L}) + 2\sum_{i=1}^{k-1}(-)^i I_i(\theta)T_i(\bar{L}) \tag{4-20}$$

截断 Chebyshev 多项式展开一般比 Taylor 级数展开更快收敛，故广泛地应用于信号处理等领域[57]。因为其快速的收敛速度，$k = 10$ 的截断项数基本可以提供 $e^{-x\theta}$ 足够精度的近似。

将式子 4-20带入式子 4-13，有

$$\begin{aligned} R_d &\leftarrow D^{-1}A(E_n - \widetilde{L})R_d \\ &= D^{-1}A\{E_n - [I_0(\theta)T_0(\bar{L}) + 2\sum_{i=1}^{k-1}(-)^i I_i(\theta)T_i(\bar{L})]\}R_d \end{aligned} \tag{4-21}$$

式子 4-21的计算可以结合 Chebyshev 多项式的递归形式定义 4-16，递归地高

Algorithm 1 ProNE

输入: $G = (V, E)$, 维数 d, 位置参数 μ, 尺度参数 θ

输出: 图嵌入表示矩阵 R_d

1: 依据式子 (5-13) 构建相似度矩阵 M
2: 生成高斯随机矩阵 Ω, 计算得到 $Y = M\Omega$
3: 对 Y 进行 QR 分解, 分解结果为 $QR = Y$，即找到了算法需要的 Q
4: 对 $H = Q^T M$ 进行 SVD 分解，$S_d\Sigma_d V_d^T \leftarrow \text{SVD}(H, d)$
5: $R_d \leftarrow QS_d\Sigma_d^{1/2}$
6: 通过式子 (4-21) 以 Chebyshev 的递归方式 (4-22) 对 R_d 进行谱传播
7: 通过 SVD 对 R_d 进行正交化
8: **返回**图嵌入表示矩阵 R_d

效计算。我们引入符号标记 $\bar{R}_d^{(i)} = T_i(\bar{L})R_d$, 那么

$$\begin{cases} \bar{R}_d^{(i)} = 2\bar{L}\bar{R}_d^{(i-1)} - \bar{R}_d^{(i-2)} \\ \bar{R}_d^{(0)} = R_d \\ \bar{R}_d^{(1)} = \bar{L}R_d \end{cases} \tag{4-22}$$

注意到 $\bar{L} = -\frac{1}{2}[(L - \mu I_n)^2 - I_n]$ 和 L 都是稀疏的，整个计算过程都只涉及到稀疏矩阵的乘法，所以，整个计算过程的算法是线性的，即 $O(k|E|)$。

另外，由于初始的 R_d 是通过 SVD 得到的，故 R_d 具有较好的列正交的性质，也就是说图嵌入的每个维度之间的解耦合性质较好，为了避免正交性的损失，我们可以对 R_d 进行正交处理。因为这个处理不属于算法的核心，我们仅仅在新的 R_d 上又一次使用了 SVD，并没有对其他常规的正交处理方法展开讨论和实验，比如施密特正交化（Schmidt orthogonalization）等方法。

4.3　复杂度分析

算法的全部流程总结在算法 1中。对于第一个步骤，快速的稀疏矩阵分解生成初始的图嵌入表示，主要的时间开销在小矩阵 H 的 SVD 和 Y 的 QR 分解上，故时间复杂度为 $O(|V|d^2)$，另外，因为 $|\mathcal{D}| \ll |V| \times |V|$, 所以 M 是稀疏矩阵，第一个步骤涉及到的是稀疏矩阵的乘法的复杂度是 $O(|E|)$。综上，第一个步骤的时间复杂度是 $O(|V|d^2 + |E|)$。

而第二个步骤，谱传播进一步提升图嵌入的质量，主要涉及到的仍然是稀疏

矩阵的乘法和最后的小矩阵的 SVD 正交化（可选）, 故第二个步骤的算法复杂度是 $O(|V|d^2 + k|E|)$。

总结而言，算法 ProNE 的复杂度为 $O(|V|d^2 + k|E|)$, 所以算法非常高效。

关于空间复杂度，由于算法 ProNE 在计算过程中只涉及到稀疏矩阵和图嵌入矩阵的存储，所以空间复杂度是 $O(|V|d + |E|)$。

4.4　并行性

注意到 ProNE 的计算主要涉及到稀疏矩阵的乘法，randomized SVD 的内部的实现也涉及到稀疏矩阵的乘法，而稀疏矩阵的乘法目前我们使用的是单线程的，实验效果表明 ProNE 在主要单线程的情况下已经有能力非常高效地处理大规模的图（对于稠密的小矩阵 (属于 $R^{|V|\times d}$ 甚至可以进一步优化至 $R^{d\times d}$) 的 SVD, 我们因为是调用 Python 包，默认使用了多线程，但是是实验上它们占用的时间相比单线程的时间，占比很小，而且这些步骤其实可以进一步优化，有些 SVD 甚至并不一定是必须的，故我们仍然可以粗略地认为我们是单线程的）。

如果我们能使用现有的稀疏矩阵乘法的多线程并行化处理的技术[58,59]，就可以实现算法的真正意义的多线程，理论上，它可能可以为我们当前主要是单线程的版本提供 $\frac{p}{\log(p)}\times$ 的加速 (p 是线程数)[60]。

4.5　实验

我们通过多标签的点分类问题——一个被广泛地应用于图嵌入的评价的任务[2,7,8]，来评估 ProNE 的精度有效性，并通过运行时间和运行图数据的规模来评估模型的运行速度和可扩展性[7]。.

4.5.1　数据集

在精度有效性的实验中，我们使用了 5 个被广泛使用的数据集。数据集的统计量列在表 4.1中。另外，我们也使用了一系列的仿真随机网络来进一步定量地辅助评估方法的训练效率和可扩展性。

- BlogCatalog[61] 是一个社交博客网站。在 BlogCatalog 中，点和边分别表示博主和他们之间的社交关注关系，而博主提交的兴趣被视为分类任务中的标签。
- Wiki[62] 是维基百科的词共现网络中的前一百万字节的数据。图中的点表征词，边表征词的共现关系，点的标签通过斯坦福的词性标注器推理得到。
- PPI[63] 是智人的蛋白质相互作用网络的子图。图中的点表征蛋白质，边表征蛋白质之间的相互作用，点的标签表征生物状态。

- DBLP[64] 是学术引用网络。图中的结点表征作者，边表征作者之间的引用关系，点的标签表征作者主要参与的学术会议和活动。
- Youtube[61] 是 Youtube 的用户之间的社交网络。图中的结点表征用户，边表征用户之间的社交关系，结点的标签表征喜欢共同的视频题材的用户的群。

4.5.2　基线算法

我们将 ProNE 和目前流行的基线算法进行比较，这些基线算法包括基于 skip-gram 的图嵌入算法 (DeepWalk[2], LINE[7] 和 node2vec[8]) 和基于矩阵分解的图嵌入算法 (GraRep[4] and HOPE[5])。这些基线算法的简单介绍如下：

- DeepWalk 是将自然语言处理中的 skip-gram 模型应用于图结构数据的开山之作。通过从每个图结点开始随机游走生成一系列的结点序列来模拟自然语言处理中的语料库，每个结点序列对应句子，而每个结点对应单词。DeepWalk 将得到的“语料库”输入进 skip-gram 模型，正如自然语言处理中会得到词嵌入向量表示一样，DeepWalk 便得到了每个图结点的嵌入向量表示。
- LINE 定义了结点一阶相似性和二阶相似性，总体而已，LINE 相当于窗口大小设置为 1 的 DeepWalk。相比于 DeepWalk, 由于 LINE 更小的“语料库”和使用 C++ 实现，它可以认为是目前为止最快的图嵌入表示模型之一。
- node2vec 是 DeepWalk 的另一个变体版本。和 DeepWalk 主要的不同来源于“语料库”的生成步骤。node2vec 额外使用了两个参数来控制随机游走的去向，使得随机游走更类似于深度优先搜索来探索图结构中偏向于全局结构的相似性，或者更类似于广度优先搜索来探索分布假设为基础的局域的相似性。一般而言，运行 node2vec 需要对两个参数做探索，而且随机游走时涉及的随机采样比均匀分布稍复杂，等等原因，node2vec 比 DeepWalk 慢得多。
- GraRep 是一种矩阵分解为基础的图嵌入方法，它要分解的相似矩阵的推导基本源于 skip-gram 的矩阵理解。但是 GraRep 强调不同阶的相似性，最终的版本也是将各阶求得的图嵌入表示向量拼接起来。为公平比较，特别是减少集成算法对增益影响，它和其他模型的比较一般倾向于控制维数相同。
- HOPE 是一种矩阵分解为基础的图嵌入方法，它要分解的相似矩阵是 Katz 矩阵，隐式地考虑了无穷阶的相似性。分解的方法是广义 SVD，这依赖于相似矩阵必须具有某种特殊形式，故一般不作为一种适用性广泛的方法。

注意到基于图卷积的方法并没有包括在基线算法中，因为这些算法主要是半监督的设定，而且算法的输入需要结点的额外属性特征。

Dataset	*BlogCatalog*	*Wiki*	*PPI*	*DBLP*	*Youtube*
#nodes	10,312	4,777	3,890	51,264	1,138,499
#edges	333,983	184,812	76,584	127,968	2,990,443
#labels	39	40	50	60	47

表 4.1　数据集的统计量

4.5.3　参数设置

为了公平比较，所有的方法的图嵌入向量的维度都统一设为 $d = 128$。至于其他参数，我们遵从了原作者在他们的工作中倾向使用的设置。对于 DeepWalk 和 node2vec，窗口大小设为 m=10, 每个结点的随机游走数设为 r=80, 随机游走的长度设为 t=40。在 node2vec 中的 p, q 按照原文的建议在 {0.25, 0.50, 1, 2, 4} 中寻找。对于 LINE, 负采样的数量设为 $k = 5$，总的采样数设为 T=r×t×|V|。对于 GraRep, 阶数 (等价于 DeepWalk 和 node2vec 中的窗口大小) 同样设为 $m = 5$，并且为了比较的公平性，最后所有阶的图嵌入拼接起来的维数设为 d=128。对于 HOPE, β 的值由作者提供的源代码自动计算得到，我们也在 (0, 1) 上搜索来尝试提高效果。对于我们的模型 ProNE, Chebyshev 多项式展开的项数 k 设为 10, 其他默认参数包括 μ=0.1, θ=0.5。

4.5.4　运行环境

实验是在 Red Hat 发行版的 Linux 服务器上运行，CPU 信息为 Intel Xeon(R) CPU E5-4650 (2.70GHz)，内存为 1T。ProNE 使用了 Python 3.6.1 实现。

4.5.5　评估方法

我们严格遵从了以下评估方法，它们在基线算法在原始文章中被采用[2,4,7,8]。我们随机采样了不同百分比的有标签结点来训练 LR 分类器，并把剩下的结点作为测试集。我们重复了以上的随机划分、训练、预测 *10* 次，然后汇报了平均的 Micro-F1。使用 Macro-F1 等精度指标也会得到类似的比较结果，由于空间受限我们只展示了 Micro-F1 的结果。

和 LINE[7] 一样，我们在运行速度的评价中使用了程序的运行时间来进行衡量。对于可扩展性的评估，我们让算法在不同规模大小的图上以可以接受的时间运行。

Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

表 4.2　运行时间对比 (单位：秒)。

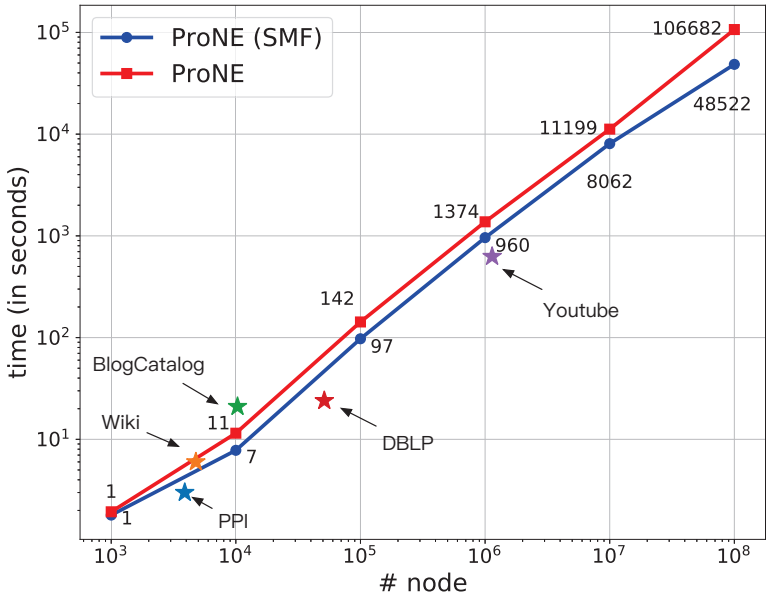


图 4.2　ProNE 在图规模变化下的可扩展性。其中蓝色表示 ProNE 在第一步稀疏矩阵分解 (SMF) 产生快速图嵌入表示的运行时间。图中的结点的平均度固定为 10, 结点的数量由 1,000 增长到 100,000,000。

4.5.6　运行效率和可扩展性

4.5.6.1　运行效率

我们比较了不同方法的运行效率，同时也展示了我们的算法 ProNE 的可扩展性。注意到所有比较的基线算法的效率都已经使用 *20* 个线程/进程加速了，而我们的模型 *ProNE* 基本只使用了单个线程。

表 4.2 汇报了 ProNE 和其他 3 个比较快且扩展性强的基线算法—DeepWalk, LINE 和 node2vec 的运行时间 (包括 IO 时间和计算时间)。基于矩阵分解的方法比这些参与比较的方法要慢得多。比如，GrapRep 的时间复杂度是 $O(|V|^3)$, 所以它在稍微大一些的图上的应用也受限，比如在拥有一百万结点的 Youtube 图上。

运行时间的结果表明，在 PPI 和 Wiki 这些相对小的图上 (1,000 个结点量级

Dataset	training ratio	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PPI	DeepWalk	16.4	18.5	19.4	20.3	21.1	21.8	22.3	22.6	22.7
	LINE	16.3	18.9	20.1	21.0	21.5	22.1	22.7	22.9	23.1
	node2vec	16.2	18.4	19.7	20.4	21.6	22.2	23.1	23.1	24.1
	GraRep	15.4	17.6	18.9	19.9	20.2	20.5	20.4	20.7	20.9
	HOPE	16.4	18.6	19.8	20.6	21.0	21.5	21.7	22.2	22.5
	ProNE (SMF)	15.8	18.7	20.6	21.7	22.7	23.5	23.7	23.9	24.2
	ProNE	**18.2**	**21.2**	**22.7**	**23.7**	**24.6**	**24.9**	**25.4**	**25.8**	**25.9**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.4)	($\pm$0.3)	($\pm$0.6)	($\pm$0.7)	($\pm$0.9)	($\pm$1.0)	($\pm$1.0)	($\pm$1.1)
Wiki	DeepWalk	40.4	43.1	45.9	47.7	48.5	48.7	49.1	49.2	49.4
	LINE	**47.8**	49.6	50.4	51.0	51.2	51.6	51.6	51.7	52.4
	node2vec	45.6	46.3	47.0	47.2	48.2	48.7	49.6	49.8	50.0
	GraRep	47.2	48.8	49.7	50.4	50.6	50.7	50.9	51.0	51.8
	HOPE	38.5	39.5	39.8	40.2	40.1	40.2	40.1	39.8	40.1
	ProNE (SMF)	47.6	50.4	51.6	52.4	53.2	53.3	53.5	53.6	53.9
	ProNE	47.3	**51.3**	**53.1**	**53.8**	**54.7**	**55.0**	**55.2**	**55.7**	**57.2**
	($\pm\sigma$)	($\pm$0.7)	($\pm$0.5)	($\pm$0.4)	($\pm$0.4)	($\pm$0.8)	($\pm$0.4)	($\pm$0.8)	($\pm$1.2)	($\pm$1.3)
BlogCatalog	DeepWalk	36.2	38.4	39.6	40.5	40.9	41.1	41.4	41.7	42.2
	LINE	28.2	29.9	30.6	31.0	33.2	34.5	35.5	36.0	36.8
	node2vec	**36.3**	38.5	39.7	**40.8**	41.1	41.7	42.0	42.2	42.1
	GraRep	34.0	32.0	32.5	33.0	33.3	33.6	33.7	33.8	34.1
	HOPE	30.7	32.6	33.4	33.9	34.3	34.6	35.0	35.3	35.3
	ProNE (SMF)	34.6	36.6	37.6	38.2	38.6	39.0	39.3	39.7	39.0
	ProNE	36.2	**38.8**	**40.0**	40.7	**41.2**	**41.8**	**42.1**	**42.6**	**42.7**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.4)	($\pm$0.3)	($\pm$0.4)	($\pm$0.6)	($\pm$0.7)	($\pm$0.7)	($\pm$0.9)	($\pm$1.2)
Dataset	training ratio	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
DBLP	DeepWalk	49.3	53.2	55.0	56.2	57.1	57.5	57.9	58.2	58.4
	LINE	48.7	51.4	52.6	53.2	53.5	53.8	54.1	54.4	54.5
	node2vec	48.9	53.2	55.1	56.3	57.0	57.6	58.0	58.2	58.4
	GraRep	50.5	52.1	52.6	52.9	53.2	53.4	53.5	53.7	53.8
	HOPE	**52.2**	**54.2**	55.0	55.5	55.9	56.1	56.3	56.5	56.6
	ProNE (SMF)	50.8	53.8	54.9	55.7	56.1	56.5	56.7	56.9	57.0
	ProNE	48.8	**54.2**	**56.2**	**57.3**	**58.0**	**58.4**	**58.8**	**59.0**	**59.2**
	($\pm\sigma$)	($\pm$1.0)	($\pm$0.6)	($\pm$0.5)	($\pm$0.4)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)	($\pm$0.1)
Youtube	DeepWalk	38.0	39.3	40.1	40.8	41.3	41.7	42.1	42.5	42.8
	LINE	33.2	34.7	35.5	36.2	37.0	37.6	38.2	38.8	39.3
	ProNE (SMF)	36.5	39.1	40.2	40.8	41.2	41.4	41.7	41.9	42.1
	ProNE	**38.2**	**40.6**	**41.4**	**41.3**	**42.3**	**42.7**	**42.9**	**43.2**	**43.3**
	($\pm\sigma$)	($\pm$0.8)	($\pm$0.5)	($\pm$0.3)	($\pm$0.2)	($\pm$0.2)	($\pm$0.1)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)

表 4.3　多标签结点分类的 Micro-F1 (%).

的图)，ProNE 只需要 10 秒以内便完成了训练，而最快的基线算法 LINE 则比我们的算法慢至少 14 倍，而 DeepWalk 和 node2vec 则慢了 100 倍。我们算法的效率优势在 BlogCatalog 和 DBLP 等中等规模的图上 (10,000 个结点量级的图) 和在 Youtube 等相对更大规模的图 (1,000,000 个结点量级的图) 上, 也有类似的表现。特

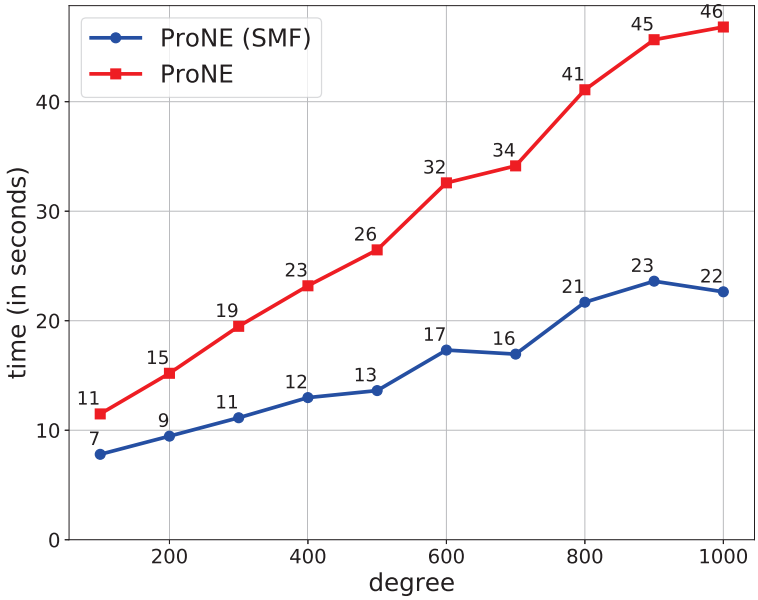
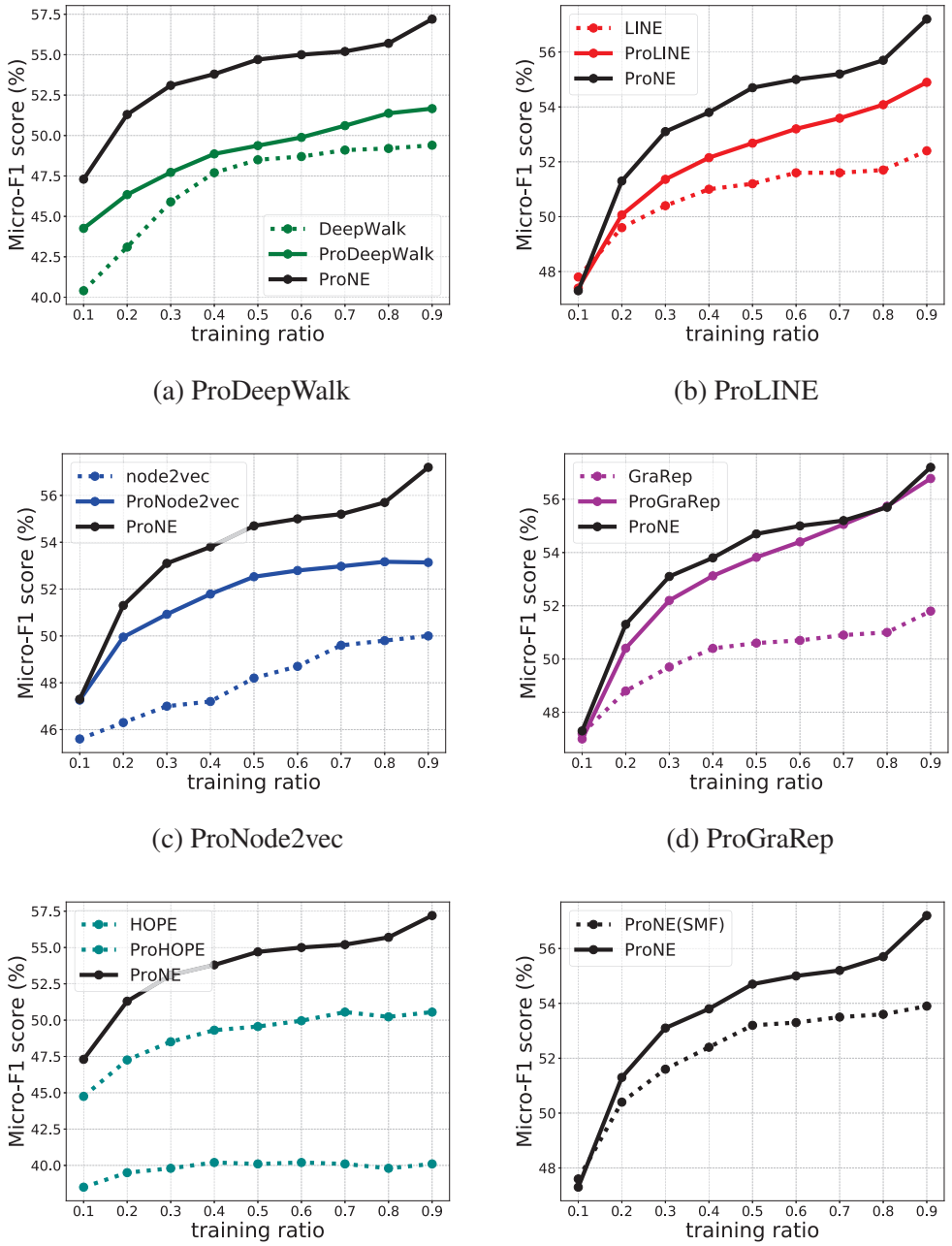


图 4.3　ProNE 在图的密度变化下的可扩展性。其中蓝色表示 ProNE 在第一步稀疏矩阵分解 (SMF) 产生快速图嵌入表示的运行时间。图中的结点的数目固定为 10000, 结点的平均度由 100 增长到 1, 000。

别地，ProNE 可以只使用 1 个 CPU 内核就能在 11 分钟内完成对 Youtube 等百万量级的图的嵌入表示的学习，然而，在使用 20 个线程的情况下，LINE 需要耗费 100 分钟，DeepWalk 需要耗费 19 个小时，而 node2vec 则需要耗费数天。 总结的说，基本单线程的 *ProNE* 模型大概比 *20* 线程的 *LINE, DeepWalk* 和 *node2vec* 等模型快上 *10–400* 倍，而由于复杂度的优势，和基于矩阵分解的图嵌入方法相比，这种时间效率和空间效率的优势会更加明显。

4.5.6.2　可扩展性

为了更定量地探究算法的不同规模的图上的可扩展性，我们使用仿真图来进行可扩展性的实验，实验结果见图 4.2和图 4.3。实验结果表明我们的算法具有处理 10 亿结点量级的图的潜力。实验中，我们首先生成结点度固定为 10，结点数从 1, 000 到 100, 000, 000 变化的随机正规图。图 4.2 展示了 ProNE 在不同规模大小的图上的运行时间。结果表明 ProNE 的运行时间随着图的规模大小而大致线性增长。另外，在 5 个不同规模的真实数据集上的算法运行时间也嵌入在图中，它们和仿真图的扩展性的实验结果的增长趋势大体一致。在实验中，我们看到，在单线程条件下只需要花费 29 小时，ProNE 就可以完成对 1 亿个结点 5 亿条边的图的嵌入表示。而对于 LINE, 20 个线程下完成对相同规模的图的嵌入表示需要一周以上，而



(a) ProDeepWalk　(b) ProLINE

(c) ProNode2vec　(d) ProGraRep

(e) ProHOPE　(f) ProNE

图 4.4　谱传播在 Wiki 数据集上提升各个算法—ProDeepWalk, ProLINE, ProNode2vec, ProGrarep, ProHOPE 和 ProNE。

对于更慢的 Deepwalk 和 node2vec，则可能需要数月。

另一方面，图 4.3显示了 ProNE 在固定 10, 000 个结点，度在 100 到 1, 000 变化的随机正规图上的运行时间。图上的结果很清楚地表明了 ProNE 和图的密度也是呈线性相关的。综上，我们可以总结出，ProNE 是一个在大规模的图结构数据上快速的可扩展的图嵌入表示方法，而且它也具有处理更稠密的图的潜力。

4.5.7　精度效果

4.5.7.1　精度效果对比

我们在表 4.3中总结了模型在多标签的图结点的分类任务上的预测精度效果。因为空间的限制，我们在表中只展示了 Micro-F1 作为精度指标的结果，只汇报了我们模型的标准差 (σ)，但是我们的以下的结论对 Macro-F1 等其他精度指标也同样适用。另外，对于我们的模型 ProNE, 我们也汇报了第一步稀疏矩阵分解 (sparse matrix factorization, SMF) 产生的初始图嵌入的效果, 由此来分别体现模型的两个步骤对精度的贡献。最后，在 Youtube 数据集上，node2vec 无法在 5 天内完成训练，GraRep 等其他基于矩阵分解的算法也由于内存溢出的问题（内存需求超过 1T）无法进行计算。

我们观察到，ProNE 在 5 个数据集上一致性地表现了比基线算法更好的效果, 这显示了模型在嵌入表示的质量（精度效果）上的优势。值得注意的是，我们模型的第一步通过简单的稀疏矩阵分解产生的初始的快速图嵌入 (ProNE(SMF)) 和现有的主流的图嵌入方法的精度效果相当，有时甚至更好。在模型的谱传播步骤的进一步加持下, ProNE 因为同时刻画了局域的平滑性质 (local structure smoothing) 和全局的聚类信息 (global clustering information)，产生了最好的精度效果。

4.5.7.2　作为普适的图嵌入提升方法的谱传播

注意到 ProNE 由两个步骤组成：1) 快速的稀疏矩阵分解生成初始的图嵌入表示；2) 用于融入更有局域和全局的图信息的谱传播。一个很自然的问题是：谱传播可以作为一种普适的图嵌入提升方法来受益其他模型中的图嵌入表示么？

我们将 DeepWalk, LINE, node2vec, GraRep 和 HOPE 学到的图嵌入作为 ProNE 的谱传播步骤的输入。图 4.4 显示了各个算法在 Wiki 数据集上的原始结果和提升结果 (提升结果记作 “ProBaseline’)。实验结果显示了 Pro 版本的所有 5 个基线算法的效果和原来相比都得到了显著的提升。平均而言，我们的谱传播策略为所有的方法提供了 +10% 的相对增益, 特别的，对于 HOPE 算法有 +25% 的相对提升。值得一提的是，这些基线算法在 Wiki 上的提升实验在 1 秒内就完成了。*这些实验结果表明 ProNE 的谱传播是一个普适通用而且高效快速的图嵌入提升方法。*

4.6　工作总结

在这个工作中，我们提出了 ProNE——一个在大规模图结构数据上快速可扩展的图嵌入算法。首先，ProNE 将图嵌入表示的问题化为稀疏矩阵分解问题，以高

效地实现初始的结点的嵌入表示，从而刻画结点的分布相似度。然后，它借助高阶的 Cheeger 不等式来调制图的谱空间，把第一步学到的嵌入表示在调整后的图上传播，从而达到将局域的平滑信息和全局的聚类信息融合进图嵌入表示中。

ProNE 在运行效率和精度效果上都比现在流行的基线算法，比如 DeepWalk, LINE, node2vec, GraRep 和 HOPE，都要更优秀。值得注意的是，主体单线程的 ProNE 模型比 20 个线程加速的基线算法们快上大约 10—400 倍。

正如章节4.4所讨论的，在未来的工作中，我们将稀疏矩阵乘法的多线程技术应用于 ProNE 中，来进一步提升我们算法的运行效率，用以处理更大规模的图数据。另外，我们也会进一步尝试将 ProNE 中的谱传播与图卷积神经网络结合，从而将算法推广到（半）监督学习的设定上。