大型图上的归纳表征学习

威廉-汉密尔顿* (William L. Hamilton)
wleif@stanford.edu

Rex Ying*
rexying@stanford.edu

Jure Leskovec
jure@cs.stanford.edu

斯坦福大学计算机科学系 加利福尼亚州，斯坦福，94305

摘要

大型图中节点的低维嵌入已被证明在从内容推荐到识别蛋白质功能等各种预测任务中极为有用。然而，现有的大多数方法都要求在训练嵌入时，图中的所有节点都要存在；以前的这些方法本质上都是*转导*式的，不能自然地泛化到未见过的节点上。在此，我们介绍一种通用*归纳*框架 GraphSAGE，该框架利用节点特征信息（如文本属性）为之前未见的数据高效生成节点嵌入。我们不为每个节点训练单独的嵌入，而是学习一个函数，通过对节点本地邻域的特征进行采样和聚合来生成嵌入。我们的算法在三个归纳节点分类基准上的表现优于强基准：我们基于引用和 Reddit 帖子数据对演化信息图中的未见节点类别进行了分类，我们还利用蛋白质-蛋白质相互作用的多图数据集展示了我们的算法可泛化到完全未见的图。

1 导言

大型图中节点的低维向量嵌入[1]已被证明非常有用，可作为各种预测和图分析任务的特征输入[5, 11, 28, 35, 36]。节点嵌入方法背后的基本思想是利用降维技术将节点图邻域的高维信息提炼成密集的向量嵌入。然后，这些节点内嵌信息就可以输入到下游机器学习系统中，帮助完成节点分类、聚类和链接预测等任务 [11, 28, 35]。

然而，以前的工作都集中在嵌入单个固定图中的节点，而现实世界中的许多应用都需要快速生成未见节点或全新（子）图的嵌入。这种归纳能力对于高吞吐量的生产型机器学习系统来说至关重要，因为这些系统在不断变化的图上运行，会不断遇到未见过的节点（如 Reddit 上的帖子、Youtube 上的用户和视频）。生成节点内嵌的归纳方法还有利于在具有相同特征形式的图中进行泛化：例如，我们可以在来自模型生物的蛋白质-蛋白质相互作用图上训练内嵌生成器，然后使用训练好的模型为新生物收集的数据轻松生成节点内嵌。

归纳式节点嵌入问题与转导式设置相比尤其困难，因为对未知节点进行归纳需要将新观察

到的子图与算法已经优化过的节点嵌入 "对齐"。归纳框架必须学会

---

*两位第一作者的贡献相同。

[1]虽然通常将这些数据结构称为社会或生物*网络*，但我们使用的术语是*图*以避免与神经网络术语产生歧义。

第 31 届神经信息处理系统大会（NIPS 2017），美国加利福尼亚州长滩。

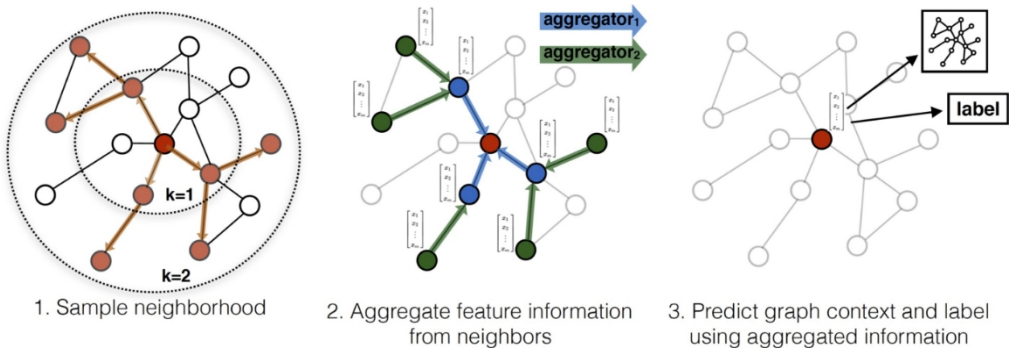

图 1：GraphSAGE 样本和汇总方法的直观图示。

识别节点邻域的结构属性，揭示节点在图中的局部作用及其全局位置。

现有的大多数生成节点嵌入的方法本质上都是转导式的。这些方法大多使用基于矩阵因式分解的目标直接优化每个节点的嵌入，并不能自然地推广到未见过的数据中，因为它们是对单个固定图中的节点进行预测[5, 11, 23, 28, 35, 36, 37, 39]。这些方法可以修改为在归纳环境中运行（例如 [28]），但这些修改往往计算成本高昂，在做出新预测之前需要额外的梯度下降。最近还有一些利用卷积算子对图结构进行学习的方法，有望成为一种嵌入方法[17]。迄今为止，图卷积网络（GCN）只应用于固定图的转导设置[17, 18]。在这项工作中，我们将 GCNs 扩展到归纳式无监督学习任务中，并提出了一个框架，将 GCN 方法推广到使用可训练的聚合函数（超越简单的卷积）。

目前的工作我们提出了一个用于归纳节点嵌入的通用框架，称为 GraphSAGE（SAmple and aggreGatE）。与基于矩阵因式分解的嵌入方法不同，我们利用节点特征（如文本属性、节点轮廓信息、节点度）来学习一个可泛化到未见节点的嵌入函数。通过将节点特征纳入学习算法，我们可以同时学习每个节点邻域的拓扑结构以及邻域中节点特征的分布。虽然我们关注的是特征丰富的图（如带有文本属性的引文数据、带有功能/分子标记的生物数据），但我们的方法也可以利用存在于所有图中的结构特征（如节点度）。因此，我们的算法也可以应用于没有节点特征的图。

我们不为每个节点训练一个不同的嵌入向量，而是训练一组*聚合函数*，学习从节点的本地邻域聚合特征信息（图 1）。每个聚合函数都会聚合来自特定节点的不同跳数或搜索深度的信息。在测试或推理时，我们使用训练有素的系统，通过应用学习到的聚合函数，为完全未见过的节点生成嵌入。根据以往生成节点嵌入的工作，我们设计了一种无监督损失函数，使 GraphSAGE 无需特定任务监督即可进行训练。我们还证明，GraphSAGE 可以用完全监督的方式进行训练。

我们在三个节点分类基准上评估了我们的算法，这些基准测试了 GraphSAGE 在未见数据上生成有用嵌入的能力。我们使用了基于引文数据和 Reddit 帖子数据的两个演化文档图（分别预测论文和帖子类别），以及基于蛋白质-蛋白质相互作用数据集的多图泛化实验（预测蛋白质功能）。利用这些基准，我们证明了我们的方法能够有效地为未见节点生成表征，并在很大程度上优于相关基线：在各个领域，与单独使用节点特征相比，我们的监督方法

将分类 F1 分数平均提高了 51%，GraphSAGE 始终优于强大的转导基线 [28]，尽管该基线在未见节点上的运行时间要长 100 倍。我们还表明，与受图卷积网络[17]启发的聚合器相比，我们提出的新聚合器架构有显著提升（平均提升 7.4%）。最后，我们探究了我们方法的表达能力，并通过理论分析表明，尽管 GraphSAGE 本身是基于特征的，但它能够学习节点在图中作用的结构信息（第 5 节）。

2 相关工作

我们的算法在概念上与以前的节点嵌入方法、对图进行学习的一般监督方法以及最近将卷积神经网络应用于图结构数据的进展相关。[2]

基于因式分解的嵌入方法。最近有一些节点嵌入方法，利用随机漫步统计和基于矩阵因式分解的学习目标来学习低维嵌入[5, 11, 28, 35, 36]。这些方法与光谱聚类[23]、多维缩放[19]以及 PageRank 算法[25]等经典方法也有密切联系。由于这些嵌入算法直接为单个节点训练节点嵌入，因此它们本质上都是传导性的，至少需要昂贵的额外训练（如通过随机梯度下降）才能对新节点进行预测。此外，对于许多此类方法（例如 [11, 28, 35, 36]）来说，目标函数对嵌入的正交变换是不变的，这意味着嵌入空间不能自然地在图之间通用，而且在重新训练过程中可能会漂移。这一趋势的一个显著例外是 Yang 等人提出的 Planetoid-I 算法[40]，这是一种基于嵌入的归纳式半监督学习方法。不过，Planetoid-I 在推理过程中并不使用任何图结构信息，而是在训练过程中将图结构作为一种正则化形式。与之前的方法不同，我们利用特征信息来训练模型，为未见节点生成嵌入。

图的监督学习。除了节点嵌入方法之外，还有大量关于图结构数据监督学习的文献。这包括各种基于核的方法，其中图的特征向量来自各种图核（见 [32] 及其中的参考文献）。最近还出现了一些对图结构进行监督学习的神经网络方法[7, 10, 21, 31]。我们的方法在概念上受到了其中一些算法的启发。不过，以前的这些方法试图对整个图（或子图）进行分类，而本研究的重点则是为单个节点生成有用的表征。

图卷积网络。近年来，人们提出了几种用于图学习的卷积神经网络架构（例如 [4, 9, 8, 17, 24]）。这些方法大多不能扩展到大型图，或设计用于全图分类（或两者兼而有之）[4, 9, 8, 24]。不过，我们的方法与 Kipf 等人提出的图卷积网络（GCN）密切相关[17, 18]。最初的 GCN 算法[17]是为转导环境下的半监督学习而设计的，其精确算法要求在训练过程中已知完整的图拉普拉奇。我们算法的一个简单变体可以看作是 GCN 框架在归纳环境中的扩展，我们将在第 3.3 节中再次讨论这一点。

3 建议的方法：GraphSAGE

我们的方法背后的关键理念是，我们要学习如何从节点的本地邻域（例如，附近节点的度或文本属性）中聚合特征信息。我们首先介绍了 GraphSAGE 嵌入生成（即前向传播）算法，该算法在假设 GraphSAGE 模型参数已被学习到的情况下为节点生成嵌入（第 3.1 节）。然后，我们将介绍如何使用标准随机梯度下降和反向传播技术学习 GraphSAGE 模型参数（第 3.2 节）。

3.1 嵌入生成（即前向传播）算法

在本节中，我们将介绍嵌入生成或前向传播算法（算法 1），该算法假定模型已经过训练

且参数固定。特别是，我们假设已经学习了 K 个聚合器函数（表示为 $AGGREGATE_k$，$\forall k \in \{1, \ldots, K\}$）的参数，这些函数用于聚合来自节点邻居的信息，以及一组权重矩阵 $\mathbf{W}^k$，$\forall k \in \{1, \ldots, K\}$，这些权重矩阵用于在模型的不同层或 "搜索深度 "之间传播信息。第 3.2 节将介绍我们如何训练这些参数。

[2]从这篇论文最初提交给 2017 年 NIPS 到提交最终接受（即 "可上镜"）版本的这段时间里，预印本服务器上已经发表了一些密切相关（如后续）的作品。为了时间上的清晰，我们不对这些论文进行详细审查或比较。

算法 1： GraphSAGE 嵌入生成（即前向传播）算法

输入： 图 G（V，E）；输入特征 $\{\mathbf{x}_v, \forall v \in V\}$；深度 K；权重矩阵 $\mathbf{W}^k$，$\forall k \in \{1, \ldots, K\}$；非线性 σ；可微聚合函数 $AGGREGATE_k$，$\forall k \in \{1, \ldots, K\}$；邻域函数 $N: v \rightarrow 2^V$

输出： 所有 $v \in V$ 的矢量表示 $\mathbf{z}_v$

1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v$，$\forall v \in V$；
2 **对于** $k = 1 \ldots K$ **做**
3 **对于** $v \in V$ **do**
4 $\mathbf{h}_{N(v)}^k \leftarrow AGGREGATE_k(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$；
5 $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{N(v)}^k))$
6 **结束**
7 $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2$，$\forall v \in V$
8 **结束**
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K$，$\forall v \in V$

算法 1 背后的直觉是，在每次迭代或搜索深度时，节点都会从本地邻居那里收集信息，随着这一过程的迭代，节点会从图的更远处获得越来越多的信息。

算法 1 描述了在以下情况下的嵌入生成过程：将整个图 $G = (V, E)$ 和所有节点 $\mathbf{x}_v$，$\forall v \in V$ 的特征作为输入。我们将在下文介绍如何将其推广到迷你批处理设置中。算法 1 外循环的每个步骤如下，其中 k 表示外循环的当前步骤（或搜索深度），$\mathbf{h}^k$ 表示节点在该步骤中的表示：首先，每个节点 $v \in V$ 将其节点中的节点表示聚合在一起。近邻，$\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}$，变成一个单一的向量 $\mathbf{h}_{N(v)}^{k-1}$。请注意，这种聚合这一步取决于外循环前一次迭代（即 k - 1）所生成的表示，而 $k = 0$（"基本情况"）表示被定义为输入节点特征。在聚合相邻的特征向量后，GraphSAGE 会连接节点的当前表示、$\mathbf{h}_v^{k-1}$，与邻域汇总向量 $\mathbf{h}_{N(v)}^{k-1}$，并将该汇总向量通过一个全连接层具有非线性激活函数 σ，可转换算法下一步使用的表征（即 $\mathbf{h}^k$，$\forall v \in V$）。为方便记述，我们将深度 K 的最终输出表示法表示为 $\mathbf{z}_v \equiv \mathbf{h}^K$，$\forall v \in V$。相邻表示法的聚合可由多种聚合器架构完成（算法 1 中用 AGGREGATE 占位符表示），我们将在下文第 3.3 节讨论不同的架构选择。

要将算法 1 扩展到微型批处理设置，给定一组输入节点后，我们首先前向采样所需的邻域集（最大深度为 K），然后运行内循环（算法 1 中的第 3 行），但不是遍历所有节点，而是只计算满足每个深度递归所需的表示（附录 A 包含完整的微型批处理伪代码）。

与 Weisfeiler-Lehman Isomorphism Test 的关系。GraphSAGE 算法在概念上受到经典图同构性测试算法的启发。如果在算法 1 中，我们 (i) 设置 $K = |V|$、(ii)将权重矩阵设置为标识，(iii)使用适当的哈希函数作为聚合器（无非线性），那么算法 1 就是 Weisfeiler-Lehman（WL）同构检验的一个实例，也称为 "天真顶点细化"[32]。如果算

法 1 输出的两个子图的表示集 $\{\mathbf{z}_v, \forall v \in V\}$ 完全相同，那么 WL 检验就会宣布这两个子图是同构的。众所周知，该测试在某些情况下会失败，但对一大类图是有效的 [32]。GraphSAGE 是 WL 检验的一种连续近似方法，我们用可训练的神经网络聚合器取代了哈希函数。当然，我们使用 GraphSAGE 生成有用的节点表示，而不是测试图的同构性。不过，GraphSAGE 与经典 WL 测试之间的联系为我们设计算法学习节点邻域拓扑结构提供了理论背景。

邻域定义。在这项工作中，我们对固定大小的邻域集进行均匀采样，而不是在算法 1 中使用完整的邻域集，以保持每批邻域集的计算量。

固定。[3]也就是说，使用重载符号，我们将 $\mathcal{N}(v)$ 定义为从集合中抽取的一个固定大小的均匀抽样
$\{u \in V : (u, v) \in E\}$，我们在算法 1 的每次迭代 k 中抽取不同的均匀样本。如果没有这种采样，单个批次的内存和预期运行时间是不可预测的，在最坏情况下为 $O(|V|)$。相比之下，GraphSAGE 每批次的空间和时间复杂度是固定的
在 $O(\prod_{i=1}^{K} S_i)$ 时，其中 S_i，$i \in \{1, ..., K\}$ 和 K 是用户指定的常数。实际上
我们发现，在 $K = 2$ 和 $S_1 \cdot S_2 \leq 500$ 的条件下，我们的方法可以达到很高的性能（见详见第 4.4 节）。

3.2 学习 GraphSAGE 的参数

为了在完全无监督的环境下学习有用的预测性表征，我们对输出表征 $\mathbf{z}_u$，$\forall u \in V$ 应用了基于图的损失函数，并通过随机梯度下降调整权重矩阵 $\mathbf{W}^k$，$\forall k \in \{1, ..., K\}$ 和聚合函数参数。基于图的损失函数鼓励邻近节点具有 相似的表示，同时强制要求不同节点的表示高度不同：

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^{\mathsf{T}} \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^{\mathsf{T}} \mathbf{z}_{v_n})), \quad (1)$$

其中，v 是在固定长度随机漫步中共同出现在 u 附近的节点，σ 是 sigmoid 函数，P_n 是负采样分布，Q 定义了负采样的数量。重要的是，与以往的嵌入方法不同，我们输入此损失函数的表示 $\mathbf{z}_u$ 是由节点本地邻域内包含的特征生成的，而不是为每个节点训练一个唯一的嵌入（通过嵌入查找）。

这种无监督设置模拟了节点特征作为服务或静态存储库提供给下游机器学习应用的情况。在表征仅用于特定下游任务的情况下，无监督损失（等式 1）可以简单地用特定任务目标（如交叉熵损失）来替代或增强。

3.3 聚合器架构

与 N-D 网格（如句子、图像或三维体积）上的机器学习不同，节点的邻居没有自然排序；因此，算法 1 中的聚合函数必须在无序的向量集上运行。理想情况下，聚合器函数应该是对称的（即对其输入的排列不变），同时仍可训练并保持较高的表征能力。聚合函数的对称属性确保了我们的神经网络模型可以训练并应用于任意排序的节点邻域特征集。我们研究了三种候选聚合函数：

均值聚合器。我们的第一个候选聚合函数是均值算子，我们只需取 $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$ 中向量的元素均值。均值聚合器几乎等同于转导 GCN 框架中使用的卷积传播规则 [17]。特别是，我们可以用下面的方法取代算法 1 中的第 4 行和第 5 行，从而推导出 GCN 方法的归纳变体：[4]

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}), \quad (2)$$

我们称这种基于均值的修正聚合器为*卷积聚合器*，因为它是局部频谱卷积的粗略线性近似 [17]。这种卷积聚合器与我们提出的其他聚合器的一个重要区别是，它不执行算法 1 第 5 行中的连接操作，也就是说，卷积聚合器会将节点的上一层表示 $\mathbf{h}_v^{k-1}$ 与聚合邻域向量 $\mathbf{h}_{\mathcal{N}(v)}^k$ 连接起来。这种连接可以是
它被视为 GraphSAGE 算法不同 "搜索深度 "或 "层 "之间 "跳接"[13] 的一种简单形式，可显著

提高性能（第 4 节）。

LSTM 聚合器。我们还研究了一种基于 LSTM 架构的更复杂的聚合器 [14]。与平均聚合器相比，LSTM 具有表达能力更强的优势。不过，需要注意的是，LSTM 本身并不对称（即不具有排列不变性），因为它们是按顺序处理输入的。我们只需将 LSTM 应用于节点邻居的随机排列，就能使 LSTM 对无序集进行操作。

---

[3]探索非均匀采样器是未来工作的一个重要方向。

[4]请注意，这与 Kipf 等人的精确方程相差一个小的归一化常数[17]。

池化聚合器。我们研究的最后一种聚合器既对称又可训练。在这种*汇集*方法中，每个邻居的向量都会通过一个全连接的神经网络独立输入；经过 转换后，一个元素最大汇集操作就会应用于汇集整个邻居集的信息：

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\ \mathbf{W}\ \mathbf{h}_{\text{pool}}{}^{k} + \mathbf{b}_i,\ \forall u_i \in \mathsf{N}\ (v)\}),\tag{3}$$

其中，max 表示元素最大值算子，σ 是非线性激活函数。原则上，在最大池化之前应用的函数可以是任意深度的多层感知器，但我们在本研究中重点关注简单的单层架构。这种方法的灵感来自于最近在应用神经网络架构对一般点集进行学习方面取得的进展 [29]。直观地说，多层感知器可以看作是一组函数，用于计算邻居集中每个节点表征的特征。通过对计算出的每个特征应用最大池化算子，模型就能有效捕捉邻域集的不同方面。还需注意的是，原则上可以使用任何对称向量函数来代替最大值运算符（例如，元素平均值）。在开发测试中，我们发现最大值池化和均值池化没有明显区别，因此在其余实验中，我们主要使用最大值池化。

4 实验

我们在三个基准任务上测试了 GraphSAGE 的性能：(i) 利用 Web of Science 引用数据集将学术论文分类到不同学科；(ii) 将 Reddit 帖子分类为属于不同社区；(iii) 在各种生物蛋白质-蛋白质相互作用（PPI）图中对蛋白质功能进行分类。第 4.1 节和第 4.2 节总结了这些数据集，补充材料中包含了更多信息。在所有这些实验中，我们对训练过程中未见的节点进行预测，而在 PPI 数据集中，我们对完全未见的图进行测试。

实验设置。为了说明归纳基准的实证结果，我们与四种基准进行了比较：随机分类器、基于特征的逻辑回归分类器（忽略图结构）、作为基于因子化的代表性方法的 DeepWalk 算法 [28]，以及原始特征和 DeepWalk 嵌入的合并。我们还比较了使用不同聚合函数的 GraphSAGE 的四种变体（第 3.3 节）。由于 GraphSAGE 的 "convo-lutional "变体是 Kipf 等人的半监督 GCN [17]的扩展归纳版本，因此我们将该变体称为 GraphSAGE-GCN。我们测试了根据等式（1）中的损失训练的 GraphSAGE 的无监督变体，以及直接根据分类交叉熵损失训练的有监督变体。对于所有的 GraphSAGE 变体，我们都使用了整流线性单元作为非线性，并设置 $K = 2$，邻域样本量 $S_1 = 25$ 和 $S_2 = 10$（敏感性分析见第 4.4 节）。

对于 Reddit 和引文数据集，我们使用 Perozzi 等人[28]中所述的 DeepWalk "在线 "训练，即在预测之前运行新一轮 SGD 优化来嵌入新的测试节点（详见附录）。在多图环境中，我们无法应用 DeepWalk，因为在不同的不相交图上运行 DeepWalk 算法所生成的嵌入空间可以相对于彼此任意旋转（附录 D）。

所有模型都是用 TensorFlow [1] 和 Adam 优化器 [16] 实现的（DeepWalk 除外，它在使用 vanilla 梯度下降优化器时表现更好）。我们设计实验的目的是：(i) 验证 GraphSAGE 相对于基准方法（即原始特征和 DeepWalk）的改进；(ii) 对不同的 GraphSAGE 聚合器架构进行严

格比较。为了进行公平比较，所有模型都采用了相同的迷你批迭代器、损失函数和邻域采样器（如适用）。此外，为了防止在 GraphSAGE 聚合器之间的比较中出现无意的 "超参数黑客 "行为，我们对所有 GraphSAGE 变体都使用了相同的超参数集（根据验证集上的性能为每个变体选择最佳设置）。可能的超参数值集是在早期验证测试中使用引用和 Reddit 数据子集确定的，我们在分析中舍弃了这些数据子集。附录中包含更多实施细节。[5]

[5]代码和数据集链接：http://snap.stanford.edu/graphsage/

表 1：三个数据集的预测结果（微平均 F1 分数）。表中显示了无监督和完全监督 GraphSAGE 的结果。宏观平均分数也有类似趋势。

	引用		Reddit		PPI	
名称	故障排除F1	超级F1	故障排除F1	超级F1	故障排除F1	超级F1
随机	0.206	0.206	0.043	0.042	0.396	0.396
原始功能	0.575	0.575	0.585	0.585	0.422	0.422
深度行走	0.565	0.565	0.324	0.324	-	-
DeepWalk + 功能	0.701	0.701	0.691	0.691	-	-
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
比特技增益 %。	39%	46%	55%	63%	19%	45%

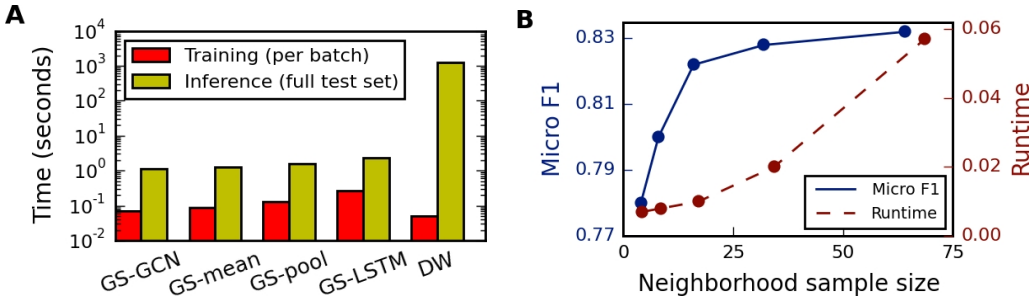

图 2：**A**：在 Reddit 数据上进行的计时实验，训练批次大小为 512，并在完整测试集（79,534 个节点）上进行推理。**B**：模型性能与邻域采样大小的关系，其中 "邻域采样大小 "指的是在 $K = 2$，$S_1 = S_2$ 的情况下，每个深度的邻域采样数量（在引用数据上使用 GraphSAGE-mean）。

4.1 演化图上的归纳学习引用和 Reddit 数据

我们的前两项实验是对不断演化的信息图中的节点进行分类，这项任务与高通量生产系统尤为相关，因为这些系统会不断遇到未知数据。

引用数据。我们的第一项任务是在大型引文数据集上预测论文主题类别。我们使用的无向引文图数据集来自汤姆森路透社科学网核心数据库，对应 2000-2005 年六个生物相关领域的所有论文。该数据集的节点标签与六个不同领域的标签相对应。该数据集总共包含 302424 个节点，平均度数为 9.15。我们在 2000-2004 年的数据上训练所有算法，并使用 2005 年的数据进行测试（其中 30% 用于验证）。对于特征，我们使用了节点度，并根据 Arora 等人[2]的句子嵌入方法处理了论文摘要，使用 GenSim word2vec 实现[30]训练了 300 维词向量。

Reddit 数据。在第二项任务中，我们要预测不同的 Reddit 帖子属于哪个社区。Reddit 是一个大型在线讨论论坛，用户在论坛上发布和评论不同主题社区的内容。我们从 2014 年 9 月

的 Reddit 帖子中构建了一个图数据集。这里的节点标签是帖子所属的社区或 "subreddit"。我们抽取了 50 个大型社区，建立了帖子到帖子图谱，如果同一用户对两个帖子都发表了评论，则将帖子连接起来。该数据集总共包含 232,965 个帖子，平均度数为 492。我们将前 20 天用于训练，其余时间用于测试（30% 用于验证）。对于特征，我们使用现成的 300 维 GloVe CommonCrawl 词向量[27]；对于每篇帖子，我们将（i）帖子标题的平均嵌入度；（ii）帖子所有评论的平均嵌入度；（iii）帖子的得分；以及（iv）对帖子的评论数连接起来。

表 1 的前四列总结了 GraphSAGE 和基线方法在这两个数据集上的性能。我们发现，GraphSAGE 的性能明显优于所有基线方法，而可训练的神经网络聚合器与基线方法相比也有显著提高。

方法相比。例如，在引用数据和 Reddit 数据上，无监督变体 GraphSAGE-pool 的性能分别比 DeepWalk 嵌入和原始特征的聚合高出 13.8% 和 29.1%，而有监督版本则分别高出 19.7% 和 37.2%。有趣的是，基于 LSTM 的聚合器表现出了强劲的性能，尽管它是为顺序数据而不是无序集而设计的。最后，我们看到无监督 GraphSAGE 的性能与完全监督版本相比具有相当的竞争力，这表明我们的框架无需针对特定任务进行微调就能取得很强的性能。

4.2 跨图泛化蛋白质与蛋白质之间的相互作用

现在我们来考虑跨图泛化的任务，这需要学习节点作用而不是群落结构。我们根据基因本体论中的细胞功能对各种蛋白质-蛋白质相互作用（PPI）图中的蛋白质角色进行分类，每个图对应不同的人体组织[41]。我们使用分子特征数据库（Molecular Signatures Database）[34]中收集的位置基因集、主题基因集和免疫特征作为特征，基因本体集作为标签（共 121 个）。平均图包含 2373 个节点，平均度数为 28.8。我们在 20 个图上训练所有算法，然后在两个测试图（另外两个图用于验证）上平均预测 F1 分数。

表 1 的最后两列总结了各种方法在这些数据上的准确度。我们再次看到，GraphSAGE 的表现明显优于基线方法，基于 LSTM 和池化的聚合器比基于均值和 GCN 的聚合器有大幅提升。[6]

4.3 运行时间和参数敏感性

图 2.A 总结了不同方法的训练和测试运行时间。这些方法的训练时间相当（GraphSAGE-LSTM 最慢）。不过，由于需要采样新的随机游走并运行新一轮 SGD 来嵌入未见节点，DeepWalk 在测试时的速度要慢 100-500 倍。

对于 GraphSAGE 变体，我们发现，与 $K = 1$ 相比，设置 $K = 2$ 可持续提高准确率，平均约为 10%-15%；但是，将 K *增加*到 2 以上，性能回报微乎其微（0%-5%），而运行时间却增加了 10-100 倍，这取决于邻域样本大小。我们还发现，对大型邻域进行采样的收益也会递减（图 2.B）。因此，尽管对邻域进行子采样会导致更高的方差，但 GraphSAGE 仍能保持较高的预测精度，同时显著缩短运行时间。

4.4 不同聚合器架构的简要比较

总体而言，我们发现基于 LSTM 和池的聚合器在平均性能和实验设置数量方面都表现最佳（表 1）。为了更定量地了解这些趋势，我们将六种不同的实验设置（即（3 个数据集）×（无监督与有监督））中的每一种都视为试验，并考虑哪些性能趋势可能具有普遍性。特别是，我们使用非参数 Wilcoxon Signed-Rank 检验[33]来量化不同聚合器在不同试验中的差异，并在适用时报告 T 统计量和 p *值*。需要注意的是，这种方法是基于等级的，主要测试在新的实验环境中，我们是否预期一种特定的方法会优于另一种方法。尽管如此，T 统计量和相关的 p 值仍是评估聚合器相对性能的有用量化指标。

我们发现，与基于 GCN 的方法相比，基于 LSTM、池和均值的聚合器都有显著的统计增益（$T = 1.0$，三者的 $p = 0.02$）。不过，与基于均值的聚合器相比，基于 LSTM 和池的聚合器的收益较为微弱（$T = 1.5$，$p = 0.03$，比较：$T = 1.5$，$p = 0.03$）。

[6]请注意，在最近的后续工作中，Chen 和 Zhu [6] 通过专门针对 PPI 任务优化 GraphSAGE 超参数，并采用新的训练技术（如 dropout、层归一化和新的采样方案），取得了更优异的性能。关于使用 GraphSAGE 方法的变体在 PPI 数据集上获得的最新数据，我们请读者参阅他们的研究成果。

LSTM 与平均值比较；池与平均值比较，$T = 4.5$，$p = 0.10$）。LSTM 与池方法之间没有明显差异（$T = 10.0$，$p = 0.46$）。不过，GraphSAGE-LSTM 的速度明显慢于 GraphSAGE-pool（系数≈2×），这可能使基于池的聚合器在整体上略胜一筹。

5 理论分析

在本节中，我们将探究GraphSAGE的表达能力，以便深入了解GraphSAGE如何学习图结构，尽管它本质上是基于特征的。作为一个案例研究，我们将考虑 GraphSAGE 能否学会预测节点的聚类系数，即在节点的 1 跳邻域内封闭三角形的比例[38]。聚类系数是衡量节点本地邻域聚类程度的常用指标，也是许多更复杂结构图案的基础[3]。我们可以证明，算法 1 能够以任意精度逼近聚类系数：

定理 1. *让 $\mathbf{x}_v \in U, \forall v \in V$ 表示图 $G = (V, E)$ 上算法 1 的特征输入，其中 U 是 $\mathbb{R}^d$ 的任意紧凑子集。假设存在一个固定的正常数 $C \in \mathbb{R}^+$，使得 $\|\mathbf{x}_v - \mathbf{x}'_v\|_2 > C$ 适用于所有节点对。那么我们可以得出 $\forall \epsilon > 0$，算法 1 存在一个参数设置 Θ^*，使得 $K = 4$ 次迭代后*

$$|z_v - c_v| < \epsilon, \quad \forall v \in V,$$

其中，$z_v \in \mathbb{R}$ 是算法 1 生成的最终输出值，c_v 是节点聚类系数。

定理 1 指出，对于任何图形，如果每个节点的特征都是不同的（并且模型足够高维），算法 1 都存在一个参数设置，从而可以将该图形中的聚类系数近似到任意精度。定理 1 的完整证明见附录。请注意，作为定理 1 的推论，即使节点特征输入是从绝对连续的随机分布中采样，GraphSAGE 也能学习局部图结构（详见附录）。证明背后的基本思想是，如果每个节点都有唯一的特征表示，那么我们就可以学习将节点映射到指示向量并识别节点邻域。定理 1 的证明依赖于池化聚合器的一些特性，这也让我们了解到 GraphSAGE-pool 优于 GCN 和基于均值的聚合器的原因。

6 结论

我们引入了一种新方法，可为未见节点高效生成嵌入。GraphSAGE 的性能始终优于最先进的基线，通过对节点邻域进行采样，有效地在性能和运行时间之间进行了权衡。我们还可以进行一些扩展和潜在的改进，例如将 GraphSAGE 扩展到有向图或多模式图。未来工作的一个特别有趣的方向是探索非均匀邻域采样函数，甚至将学习这些函数作为 GraphSAGE 优化的一部分。

致谢

作者感谢 Austin Benson、Aditya Grover、Bryan He、Dan Jurafsky、Alex Ratner、Marinka Zitnik 和 Daniel Selsam 对初稿的讨论和评论。作者还要感谢本-约翰逊（Ben Johnson）对我

们的代码提出了许多有用的问题和意见，感谢尼基尔-梅塔（Nikhil Mehta）和丁宇辉（Yuhui Ding）纠正了上一版附录中的一些小错误。本研究部分得到了国家自然科学基金 IIS-1149837、DARPA SIMPLEX、斯坦福数据科学计划、华为和陈-扎克伯格生物中心的支持。WLH 还得到了 SAP 斯坦福研究生奖学金和 NSERC PGS-D 基金的支持。本材料中表达的观点和结论仅代表作者本人，不应被解释为代 表 上述资助机构、公司或美国和加拿大政府的官方政策或认可（无论明示或暗示）。

参考资料

[1] M.M. Abadi、A. Agarwal、P. Barham、E. Brevdo、Z. Chen、C. Citro、G. S. Corrado、A. Davis、

J.Dean, M. Devin, et al：异构分布式系统上的大规模机器学习。*ArXiv preprint* , 2016.

[2] S.Arora, Y. Liang, and T. Ma.简单但难以超越的句子嵌入基准。在 *ICLR*, 2017.

[3] A.R. Benson, D. F. Gleich, and J. Leskovec.复杂网络的高阶组织。*科学*》，353（6295）：163-166，2016。

[4] J.布鲁纳、W. 扎伦巴、A. 施拉姆和 Y. 勒存。图上的谱网络和局部连接网络。In *ICLR*, 2014.

[5] S.Cao, W. Lu, and Q. Xu.Grarep：利用全局结构信息学习图表示。In *KDD*, 2015.

[6] J.Chen and J. Zhu.图卷积网络的随机训练》，*arXiv preprint arXiv:1710.10568*, 2017.

[7] H.Dai, B. Dai, and L. Song.结构化数据潜变量模型的判别嵌入。In *ICML*, 2016.

[8] M.Defferrard, X. Bresson, and P. Vandergheynst.具有快速局部频谱滤波功能的图上卷积神经网络。In *NIPS*, 2016.

[9] D.K. Duvenaud、D. Maclaurin、J. Iparraguirre、R. Bombarell、T. Hirzel、A. Aspuru-Guzik 和 R.P. Adams.用于学习分子指纹的图上卷积网络。In *NIPS*, 2015.

[10] M.Gori, G. Monfardini, and F. Scarselli.图域学习的新模型。In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 729-734, 2005.

[11] A.Grover 和 J. Leskovec：可扩展的网络特征学习。*KDD*, 2016.

[12] W.L. Hamilton, J. Leskovec, and D. Jurafsky.异时词嵌入揭示语义变化的统计规律。In *ACL*, 2016.

[13] K.He, X. Zhang, S. Ren, and J. Sun.深度残差网络中的身份映射。*EACV*, 2016.

[14] S.Hochreiter 和 J. Schmidhuber.Long short-term memory.*神经计算*，9（8）：1735- 1780，1997.

[15] K.Hornik.多层前馈网络的逼近能力。*神经网络*，4（2）：251-257，1991.

[16] D.Kingma 和 J. Ba.亚当：一种随机优化方法。In *ICLR*, 2015.

[17] T.N. Kipf 和 M. Welling.用图卷积网络进行半监督分类。In *ICLR*, 2016.

[18] T.N. Kipf 和 M. Welling.变异图自动编码器。2016 年 *NIPS 贝叶斯深度学习研讨会。*

[19] J.B. Kruskal.通过优化与非度量假设的拟合度进行多维缩放。*Psychometrika*, 29(1):1-27, 1964.

[20] O.Levy 和 Y. Goldberg.作为隐式矩阵因式分解的神经词嵌入。*NIPS*, 2014.

[21] Y.Li、D. Tarlow、M. Brockschmidt 和 R. Zemel。门控图序列神经网络。在 *ICLR*, 2015.

[22] T.Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean.单词和短语的分布式表示及其构成性。In *NIPS*, 2013.

[23] A.A. Y. Ng, M. I. Jordan, Y. Weiss, et al：分析与算法。在 *NIPS*, 2001.

[24] M.Niepert, M. Ahmed, and K. Kutzkov.学习图的卷积神经网络。在 *ICML*, 2016.

[25] L.Page、S. Brin、R. Motwani 和 T. Winograd。pagerank 引用排名：为网络带来秩序。技术报告，斯坦福信息实验室，1999 年。

[26] F.Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel、P.P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher、M.Perrot 和 E. Duchesnay。Scikit-learn：Python 中的机器学习。*机器学习研究期刊》*，12:2825-2830，2011 年。

[27] J.Pennington, R. Socher, and C. D. Manning.手套：用于单词表示的全局向量。见 *EMNLP*，2014 年。

[28] B.Perozzi, R. Al-Rfou, and S. Skiena.Deepwalk：社会表征的在线学习。在 *KDD*, 2014.

[29] C.C. R. Qi、H. Su、K. Mo 和 L. J. Guibas。Pointnet：用于 3D 分类和分割的点集深度学习。In *CVPR*, 2017.

[30] R.Řehůřek 和 P. Sojka.使用大型语料库进行主题建模的软件框架。在 *LREC*, 2010.

[31] F.Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini.图神经网络模型。*IEEE 神经网络论文集*，20（1）：61-80，2009.

[32] N.Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt.Weisfeiler-lehman 图核。*机器学习研究期刊》*，12:2539-2561，2011 年。

[33] S.Siegal.*行为科学的非参数统计》*。McGraw-hill, 1956.

[34] A.Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette、A.Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles.美国*国家科学院院刊》*，102（43）：15545-15550，2005 年。

[35] J.Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei.行：大规模信息网络嵌入。*WWW*, 2015.

[36] D.Wang, P. Cui, and W. Zhu.结构性深度网络嵌入。In *KDD*, 2016.

[37] X.Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang.社区保护网络嵌入。In *AAAI*, 2017.

[38] D.J. Watts and S. H. Strogatz.小世界 "网络的集体动力学。*自然》*，393（6684）：440-442，1998 年。

[39] L.Xu、X. Wei、J. Cao 和 P. S. Yu.社交网络的身份和兴趣嵌入。在 *WWW*, 2017.

[40] Z.Yang, W. Cohen, and R. Salakhutdinov.用图嵌入重新审视半监督学习。In *ICML*, 2016.

[41] M.Zitnik 和 J. Leskovec.通过多层组织网络预测多细胞功能。*生物信息学》*，33（14）：190-198，2017。

附录

A 微型批处理伪代码

为了使用随机梯度下降算法，我们对算法进行了调整，允许节点和边的小批量前向传播和后向传播。在此，我们将重点讨论与算法 1 类似的小批量前向传播算法。在 GraphSAGE 的前向传播中，小批量 B 包含我们要生成表示的节点。算法 2 给出了迷你批方法的伪代码。

算法 2： GraphSAGE 小批量前向传播算法

输入： 图 G（V，E）；
输入特征 $\{\mathbf{x}_v, \forall v \in B\}$；
深度 K；权重矩阵 $\mathbf{W}^k, \forall k \in \{1, \ldots, K\}$；非线性 σ；
可微聚合函数 $AGGREGATE_k$，$\forall k \in \{1, \ldots, K\}$；邻域采样函数 $N_k : v \rightarrow 2^V, \forall k \in \{1, \ldots, K\}$

输出： 所有 $v \in B$ 的矢量表示 $\mathbf{z}_v$

1 $B^K \leftarrow B$；
2 **对于** $k = K \ldots 1$ **做**
3 $B^{k-1} \leftarrow B^k$；
4 **for** $u \in B^k$ **do**
5 $B^{k-1} \leftarrow B^{k-1} \cup N_k(u)$；
6 **结束**
7 **结束**
8 $\mathbf{h}_u^0 \leftarrow \mathbf{x}_v, \forall v \in B^0$；
9 **对于** $k = 1 \ldots K$ **做**
10 **for** $u \in B^k$ **do**
11 $\mathbf{h}_{N(u)}^k \leftarrow$ 集料$_k(\{\mathbf{h}_{u'}^{k-1}, \forall u' \in N_k(u)\})$；
12 $\mathbf{h}_u^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_u^{k-1}, \mathbf{h}_{N(u)}^k))$；
13 $h_u^k \leftarrow h_u^k / \|h_u^k\|_2$；
14 **结束**
15 **结束**
16 $\mathbf{z}_u \leftarrow \mathbf{h}_u^K, \forall u \in B$

其主要思路是首先对计算所需的所有节点进行采样。算法 2 的第 2-7 行对应采样阶段。每个集合 B^k 包含计算节点 $v \in B^{k+1}$ 的表示所需的节点，即算法 1 第 $(k+1)$- 次迭代或 "层 "中的节点。第 9-15 行对应的是聚合阶段，与批量推理算法几乎相同。请注意，在第 12 和 13 行中，可以计算出集合 B^k 中任意节点在第 k 次迭代时的表示，因为其在第 k - 1 次迭代时的表示及其在第 k - 1 次迭代时采样邻居的表示已在上一循环中计算完毕。因此，该算法避免了计算不在当前迷你批中、在当前随机梯度下降迭代中未使用的节点的表示。我们使用 $N_k(u)$ 来表示一个确定性函数，它指定了节点邻域的随机样本（即假设随机性是在映射中预先计算的）。我们用 k 对该函数进行索引，以表示随机样本在 k 次迭代中是独立的。我们在本研究中使用了均匀采样函数，并在样本大小大于节点度数的情况下进

行替换采样。

请注意，算法 2 中的采样过程与算法 1 *中 k 的*迭代过程在概念上是相反的：我们从要生成表示的 "K 层 "节点（即 B 中的节点）开始，然后对它们的邻居（即算法中 "K-1 层 "的节点）进行采样，以此类推。这样做的一个结果是，邻域采样大小的定义可能有点违背直觉。特别是，如果我们使用 $K = 2$ 的总迭代次数，采样大小为 S_1

和 S_2，那么这意味着我们在算法 1 的迭代 $k = 1$ 期间对 S_1 节点进行采样，在迭代 $k = 2$ 期间对 S_2 节点进行采样，从我们希望在迭代 $k = 2$ 后生成 B 中 "目标 "节点表示的角度来看，这相当于对其近邻的 S_2 和其 2 跳邻居的 $S_1 \cdot S_2$ 进行采样。

B 其他数据集详情

在本节中，我们将提供一些额外的相关数据集详情。完整的 PPI 和 Reddit 数据集可从以下网址获取：http://snap.stanford.edu/graphsage/。科学网数据集（WoS）由汤姆森路透社授权，可提供给拥有有效 WoS 授权的团体。

Reddit 数据 为了对社区进行抽样，我们按照社区在 2014 年的评论总数进行了排名，并选择了排名[11,50]（含）的社区。我们省略了最大的社区，因为它们是大型的通用默认社区，会严重偏离类别分布。我们选择了定义在这些社区联盟上的图的最大连接部分。我们对 2014 年 10 月和 11 月的数据进行了早期验证实验和模型开发。

有关 Reddit 数据来源的详细信息，请访问：`https://archive.org/details/FullRedditSubmissionCorpus2006ThruAugust2015` 和 `https://archive.org/details/2015_reddit_comments_corpus`。

WoS 数据 我们手动选择了以下子领域，因为它们的规模相对相等，而且都是与生物学相关的领域。我们在神经科学子领域（代码=RU，不包括在最终数据集中）进行了早期验证和模型开发。我们没有在 WoS 数据的其他子集上进行任何实验。我们选取了这些领域联合图中最大的连通部分。

- 免疫学（代码：NI，文件数：77356）
- 生态学（代码：GU，文件数：37935）
- 生物物理学（代码：DA，文件数：36688）
- 内分泌学和新陈代谢（代码：IA，文件编号：52225）。
- 细胞生物学（代码：DR，文件数：84231）
- 生物学（其他）（代码：CU，文件数：13988）

PPI 组织数据 为了进行训练，我们随机选择了 20 个至少有 15,000 条边的 PPI 网络。为了进行测试和验证，我们选择了 4 个大型网络（2 个用于验证，2 个用于测试，每个网络至少有 35,000 条边）。模型设计和开发的所有实验都是在相同的 2 个验证网络上进行的，我们在所有实验中都使用了相同的随机训练集。

我们选择的特征至少包括任何一个 PPI 图中出现的 10%的蛋白质。需要注意的是，数据集的特征数据非常稀疏（42% 的节点没有非零特征值），因此利用邻域信息至关重要。

C 实验设置和超参数调整详情

用于无监督目标的随机行走 在所有设置中，我们从每个节点运行 50 次长度为 5 的随机行走，以获得无监督损失（公式 1）所需的线对。我们用纯 Python 实现了随机行走，并直接基于 Perozzi 等人提供的 Python 代码[28]。

逻辑回归模型 对于纯特征模型和对无监督模型输出的嵌入进行预测，我们使用了 scikit-learn Python 软件包[26]中的逻辑 SGDClassifier，并使用了所有默认设置。需要注意的是，该模型始终只在训练节点上进行优化，而不会在测试数据生成的嵌入式数据上进行微调。

超参数选择 在所有设置中，我们都对学习率和模型维度进行了超参数选择。除 DeepWalk 外，我们对有监督模型的初始学习率 {0.01, 0.001, 0.0001} 和无监督模型的初始学习率 {2×10^{-6}, 2×10^{-7}, 2×10^{-8}} 进行了参数扫描。[7]在适当的情况下，我们测试了每个模型的 "大 "和 "小 "版本，尽量保持模型的整体大小相当。对于池化聚合器，"大 "模型的池化维度为 1024，而 "小 "模型的维度为 512。对于 LSTM 聚合器，"大 "模型的隐藏维度为 256，而 "小 "模型的隐藏维度为 128；请注意，由于不同门的权重不同，LSTM 的实际参数数大约是这个数字的 4 倍。在所有实验和所有模型中，我们指定递归每深度 k 的 $\mathbf{h}^k_i$ 向量的输出维度为 256。所有模型都使用整流线性单元作为非线性激活函数。所有无监督 GraphSAGE 模型和 DeepWalk 都使用了 20 个负样本，并按照 [11、22、28] 的方法，使用 0.75 的平滑参数对节点度进行上下文分布平滑。最初的实验表明，DeepWalk 在学习率较大的情况下表现更好，因此我们在学习率集合 {0.2, 0.4, 0.8} 中进行了扫频。对于有监督的 GraphSAGE 方法，我们对所有模型都运行了 10 个历元。除 DeepWalk 外，所有方法的批次大小均为 512。我们发现，DeepWalk 在使用较小的 64 批次时，壁钟收敛速度更快。

硬件 除DeepWalk外，我们在一台配备4个英伟达Titan X Pascal GPU（12Gb内存，10Gbps速度）、16个英特尔至强CPU（E5-2623 v4 @ 2.60GHz）和256Gb内存的机器上进行了实验。在配备 144 个英特尔至强处理器（E7-8890 v3 @ 2.50GHz）和 2Tb 内存的 CPU 密集型机器上，DeepWalk 的运行速度更快。总体而言，在共享资源环境下，我们的实验耗时约 3 天。我们预计，一台消费级的单 GPU 机器（例如，配备 Titan X GPU 的机器），如果能充分利用全部资源，可以在 4-7 天内完成全套实验。

关于DeepWalk实现的说明 现有的DeepWalk实现[28, 11]只是对专用word2vec代码的简单封装，它们不容易支持嵌入新节点和其他变化。此外，这也使得我们难以比较这些方法的运行时间和其他统计数据。为此，我们使用 TensorFlow word2vec 教程中介绍的向量初始化等方法，在纯 TensorFlow 中重新实现了 DeepWalk。[8]

我们发现，DeepWalk 的收敛速度比其他方法慢得多，由于它的训练速度是其他方法的 2-5 倍，因此我们对随机行走数据进行了 5 次训练，而不是 1 次。为了在新数据上更新 DeepWalk 方法，我们运行了 50 次长度为 5 的随机漫步（如上所述），并对新节点的嵌入式进行更新，同时保持已训练的嵌入式固定不变。我们还测试了两种变体，一种是限制随机游走 "上下文节点 "的采样只能来自已训练过的节点集（这样可以减轻统计漂移），另一种是不做这种限制的方法。我们始终选择性能更好的变体。需要注意的是，尽管 DeepWalk 在归纳任务中表现不佳，但在反演环境中进行测试时，它的竞争力要强得多，因为在反演环境中，它可以在单个固定图上进行广泛训练。(尽管如此，Kipf 等人[17][18] 发现，基于 GCN 的方法在链接预测方面的表现始终优于 DeepWalk，即使在转导环境中也是如此，而这一任务在理论上更有利于 DeepWalk）。我们确实观察到，DeepWalk 的性能*会*随着进一步的训练而提高，而且在某些情况下，如果让它运行的时间比其他方法多 1000 *倍以上*（以测试集上预测的挂钟时间计算），它就能与无监督 GraphSAGE 方法（而不是有监督方法）相

抗衡；不过，我们认为这对于归纳任务来说并不是有意义的比较。

请注意，DeepWalk 也等同于 $p = q = 1$ 的 node2vec 模型 [11]。

邻域采样注意事项 由于阶数分布的重尾性质，我们在将所有图中的边输入 GraphSAGE 算法之前都会对其进行下采样。特别是，我们对边进行了子样本处理，这样就不会有节点的阶数大于 128。由于我们最多只能对每个节点的 25 个邻居进行采样，因此这是一个合理的权衡。通过这种降低采样率的方法，我们可以将邻接信息存储为密集的邻接列表，从而大大提高了计算效率。对于 Reddit 数据，作为预处理步骤，我们还对原始图的边进行了下采样，这是因为

[7]请注意，这些值与我们之前报告的预发表值不同，因为它们经过了校正，考虑到了 批次大小的无关归一化。我们感谢 Ben Johnson 指出了这一差异。

[8]https://github.com/tensorflow/models/blob/master/tutorials/embedding/word2vec.py

原始图形密度极高。所有实验都是在下采样版本上进行的，但我们在项目网站上发布了完整版本以供参考。

D DeepWalk 和相关方法的 对齐问题和正交不变性

DeepWalk [28]、node2vec [11] 以及最近其他成功的节点嵌入方法都采用了这种形式的目标函数：

$$\alpha \sum_{i,j \in A} f(\mathbf{z}_i \mathbf{z}^{\mathsf{T}}_j) + \beta \sum_{i,j \in 5} g(\mathbf{z}_i \mathbf{z})^{\mathsf{T}}_j \tag{4}$$

其中，f, g 是平滑、连续的函数，$\mathbf{z}_i$ 是直接优化（即通过嵌入查找）的节点表示，A、B 是节点对的集合。请注意，在很多情况下，在这些方法的作者使用的实际代码实现中，节点与两个唯一的嵌入向量相关联，并且 f 和 g 中点乘的参数是为不同的嵌入查找而绘制的（例如，[11, 28]）；不过，这并不会从根本上改变学习算法。大多数方法都会将学习到的嵌入向量归一化为单位长度，因此我们也假设进行了这种后处理。

通过与词嵌入方法和 [20] 的论证联系起来，这些方法也可以被看作是随机的隐式矩阵因式分解，我们试图学习一个矩阵 $\mathbf{Z} \in \mathbf{R}^{|P| \times d}$，使得

$$\mathbf{Z}\mathbf{Z}^{\mathsf{T}} \approx \mathbf{M}、 \tag{5}$$

其中，$\mathbf{M}$ 是某个包含随机漫步统计量的矩阵。

这种结构的一个重要结果是，嵌入可以通过任意正交矩阵进行旋转，而不会影响目标：

$$\mathbf{z}\mathbf{q}\ \mathbf{q}\mathbf{z}^{\mathsf{TT}} = \mathbf{z}\mathbf{z}、^{\mathsf{T}} \tag{6}$$

其中 $\mathbf{Q} \in \mathbf{R}^{d \times d}$ 是任意正交矩阵。由于嵌入空间在其他方面不受约束，唯一的误差信号来自正交不变目标 (4)，因此整个嵌入空间在训练过程中可以任意旋转。

这样做的两个明显后果是

1. 假设我们使用相同的输出维度，在两个独立的图 A 和 B 上运行基于 (4) 的嵌入方法。如果没有一些明确的惩罚措施来强制对齐，那么这两个图的嵌入空间将在训练后任意地相互旋转。因此，对于任何以图 A 的单个嵌入空间为基础进行训练的节点分类方法来说，输入图 B 的嵌入空间基本上是随机的。由于这些图的 $\mathbf{M}$ 矩阵完全不相交，因此这一事实也是真实的。当然，如果我们有办法匹配图之间的 "相似 "节点，那么就有可能使用对齐程序来共享图之间的信息，例如 [12] 提出的用于对齐词嵌入算法输出的程序。研究这种对齐程序是未来工作的一个有趣方向；不过与 GraphSAGE 等无需额外训练或对齐就能为新节点生成嵌入的方法相比，这些方法在新数据上的运行速度不可避免地会比较慢。
2. 假设我们在 t 时刻在图 C 上运行基于 (4) 的嵌入方法，并根据学习到的嵌入结果训

练分类器。然后，在时间 $t+1$ 时，我们向 C 添加更多节点，运行新一轮 SGD 并更新所有嵌入。这时会出现两个问题：首先，类比上文第 1 点，如果新节点只与极少数旧节点相连，那么新节点的嵌入空间就会相对于原始嵌入空间发生旋转。此外，如果我们在训练过程中更新所有的嵌入空间（而不仅仅是新节点的嵌入空间），就像文献[28]对 DeepWalk 采用的流式方法所建议的那样，那么与我们训练分类器的嵌入空间相比，嵌入空间可以任意旋转，这只会进一步加剧问题的严重性。

需要注意的是，这种旋转不变性对于只依赖成对节点距离的任务（如通过点积进行链接预测）来说并不存在问题。此外，缓解统计漂移问题的一些合理方法是：(1) 在为新测试节点优化嵌入式时，不更新已训练好的嵌入式；(2) 在采样随机游走中只保留现有节点作为 "上下文节点"，即确保跳格目标中的每个点积都是已训练好的节点与新/测试节点的乘积。我们在这项工作中尝试了这两种方法，并始终选择性能最好的 DeepWalk 变体。

另外需要注意的是，根据经验，DeepWalk 在引文数据上的表现要优于 Reddit 数据（第 4.1 节），因为与引文图相比，Reddit 数据的统计漂移更严重。特别是，Reddit 数据中从测试集到训练集的边较少，这有助于防止误对齐：2005 年引文链接中有 96% 连接回 2000-2004 年的数据，而 Reddit 测试集中只有 73% 的边连接回训练数据。

E 定理 1 的证明

为了证明定理 1，我们首先要证明三个定理：

- 定理 1 指出，存在一个连续函数，它能保证只在固定点周围的闭合球中为正，并具有一定的噪声容限。
- 例证 2 指出，我们可以使用单隐层的多层感知器将例证 1 中的函数逼近到任意精度。
- 假设所有的输入特征向量都足够独特，那么，在前两个定理的基础上，定理 3 可以证明池化架构可以学会将节点映射到唯一的指标向量上。

我们还依赖于这样一个事实，即最大池化算子（至少有一个隐藏层）能够以任意ϵ的精度逼近任何豪斯多夫连续对称函数[29]。

我们注意到，以下所有论点本质上都是*可识别性*论证。我们表明，存在一种参数设置，在这种参数设置下，算法 1 可以学习节点聚类系数，这一点并不明显，因为算法 1 是通过聚合特征信息来运行的。所述函数的*高效可学习*性是未来工作的主题。我们还注意到，这些证明是保守的，因为聚类系数实际上可能在比我们规定的更少的迭代次数或更少的限制条件下就能识别出来。此外，由于我们依赖两个普遍近似定理[15, 29]，所需维度原则上为$O(|V|)$。我们可以就某些特定层所需的输出维度提供一个更有参考价值的约束（例如，Lemma 3）；然而，在最坏的情况下，这种可识别性论证依赖于$O(|V|)$维度。但值得注意的是，Kipf 等人的 "无特征 "GCN 方法的参数维数为$O(|V|)$，因此这一要求并非完全不合理[17, 18]。

根据定理 1，我们让 $\mathbf{x}_v \in U, \forall v \in V$ 表示图上算法 1 的特征输入 $G = (V, E)$，其中 U 是 $\mathbb{R}^d$ 的任意紧凑子集。

定理 1. *设 $C \in \mathbb{R}^+$ 是一个固定的正常数。那么对于节点的任何非空有限子集 $D \subseteq V$，存在一个连续函数 $g : U \to \mathbb{R}$，使得*

$$g(\mathbf{x}) > \epsilon, \quad if \|\mathbf{x} - \mathbf{x}_v\|_2 = 0 \; for \; some \; v \in D$$
$$g(\mathbf{x}) \leq -\epsilon, \; if \; x \not\in \mathbf{x}_{v2} > C, \; \forall v \in D,$$

其中，$\epsilon < 0.5$

是选定的误差容限。

(7)

*证明。*存在许多这样的函数。为具体起见，我们提供一种满足这些标准的构造。让 $x \in U$ 表示 g 的任意输入，让 $d_v = \|\mathbf{x} - \mathbf{x}_v\|_2$ ，$\forall v \in D$，让 g 定义为 $g(x) = \sum_{v \in D} g_v(\mathbf{x})$，其中

$$g_v(\mathbf{x}) = \frac{3|D|\epsilon}{bd2_v + 1} - 2\epsilon \tag{8}$$

根据构造，$b = 3|D| - 1 > 0$：

1. g_v 在 $d_v = 0$ 时有一个唯一的最大值 $3|D|\epsilon - 2\epsilon > 2|D|\epsilon$。

2. 限 $_{d_v \to \infty} \frac{3|D|\epsilon}{\text{bd}2+1} - 2\epsilon = -2\epsilon$

   制 $_{v}$

3. $\frac{3|D|\epsilon}{\text{bd}2_{v}+1} - 2\epsilon \leq -\epsilon$ 如果 $d_v \geq C$.

还要注意的是，g 在其域（$d_v \in \mathbf{R}^+$）上是连续的，因为它是有限连续函数集的和。此外，对于给定输入 $x \in U$，如果所有点 $v \in D$ *的* $d_v \geq C$，那么根据上述性质 3，$g(\mathbf{x}) = \sum_{v \in D} g_v(\mathbf{a}) \leq -\epsilon$。而且，如果 $d_v = 0$ 对于任意 $v \in D$，那么根据性质 1 和 2，g *在*构造上是正的，因为在这种情况下、

$$\begin{aligned} g_v(\mathbf{x}) + \sum_{v' \in D \setminus v} g_{v'}(\mathbf{x}) &\geq g_v(\mathbf{x}) - (|D| - 1)2\epsilon \\ &> g_v(\mathbf{x}) - 2(|D|)\epsilon \\ &> 2(|D|)\epsilon - 2(|D|)\epsilon \\ &> 0, \end{aligned}$$

因此我们知道，只要任何节点的 $d_v = 0$，g 就是正值；只要所有节点的 $d_v > C$，g 就是负值。□

定理 2. *函数 $g : U \to \mathbf{R}$ 可以用标准多层感知器（MLP）近似到任意精确度，MLP 至少有一个隐层和一个非恒定单调递增激活函数（例如，一个整流线性单元）。准确地说，如果让 f_{θ_σ} 表示这种 MLP，ϑ_σ 表示它的参数，那么我们就有这样的条件：$\forall \epsilon, \exists \vartheta_\sigma$，即 $|f_{\theta_\sigma}(\mathbf{x}) - g(\mathbf{x})| < \epsilon, \forall x \in U$。*

*证明*这是 [15] 中定理 2 的直接结果。□

定理 3. *设 $\mathbf{A}$ 是 G 的邻接矩阵，让 $N^2(v)$ 表示节点 v 的 2 跳邻域，并定义 $\chi(G^4)$ 为具有邻接矩阵 $\mathbf{A}$ 的图的色度数[4]（忽略自循环）。假设存在一个固定的正常数 $C \in \mathbf{R}^+$，使得 $x\|_v - \mathbf{x}'_v\|_2 > C$ 适用于所有节点对。那么，算法 1 的参数设置就存在，使用深度 $k = 1$ 的集合聚合器，该集合聚合器有 ≥ 2 个具有整流非线性单元的隐藏层，从而*

$$\mathbf{h}^1_v = \mathbf{h}^1_{v'}, \forall (v, v') \in \{(v, v') : \exists u \in V, v, v' \in N^2(u)\}, \quad _{v,I} \in E^{\chi(G^4)}$$

其中，$E^{\chi(G^4)}$ 是维数为 $\chi(G^4)$ 的单热指示向量集合。

*证明*根据色度数的定义，我们知道可以用 $\chi(G^4)$ 种唯一的颜色来标记 V 中的每个节点，这样在任何节点的 2 跳邻域中共同出现的两个节点都不会被分配相同的颜色。[4]这样，我们就能为每个节点分配一个唯一的单次触发指示向量，而在任何 2 跳邻域中共同出现的两个节点都没有相同的向量。换句话说，每种颜色都定义了一个节点子集 $D \subseteq V$，而这个节点子集可以全部映射到相同的指示向量上，而不会引入冲突。

根据 Lemma 1 和 2 以及所有节点对 $x\|_v - \mathbf{x}'_v\|_2 > C$ 的假设，我们可以选择一个 $\epsilon < 0.5$，并且存在一个单层 MLP f_{θ_σ}，这样对于任意节点子集 $D \subseteq V$：

$$
\begin{aligned}
&f_{\theta_\sigma}(\mathbf{x}_v) > 0, \quad \forall v \in \mathsf{D} \\
&f_{\theta_\sigma}(\mathbf{x}_v) < 0, \ \forall v \in \mathsf{V} \setminus \mathsf{D}.
\end{aligned}
\tag{9}
$$

通过将该 MLP 设为更深一层，并特别使用整流线性激活函数，我们可以只为子集 D 中的节点返回一个正值，否则返回零值，而且，由于我们在应用聚合层后进行了归一化处理，因此这个单一的正值可以映射到一个指标向量。此外，我们还可以创建 $\chi(\mathsf{G}^4)$ 这样的 MLP，其中每个 MLP 对应不同的颜色/子集；相当于正文公式 3 中每个 MLP 对应不同的最大池化维度。

现在我们重述定理 1 并给出证明。

定理 1. *让 $\mathbf{x}_v \in \mathbb{R}^d, \forall v \in \mathsf{V}$ 表示图 $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ 上算法 1 的特征输入，其中 U 是 $\mathbb{R}$ 的任意紧凑子集[4]。假设存在一个固定的正常数 $\mathsf{C} \in \mathbb{R}^+$，使得 $\|\mathbf{x}_v - \mathbf{x}_{v'}\|_2 > \mathsf{C}$ 适用于所有节点对。那么我们可以得出，$\forall \epsilon > 0$ 存在算法 1 的参数设置 Θ^*，使得 $\mathsf{K} = 4$ 次迭代后*

$$|z_v - c_v| < \epsilon, \quad \forall v \in \mathsf{V},$$

其中，$z_v \in \mathbb{R}$ 是算法 1 生成的最终输出值，c_v 是节点聚类系数，定义见 [38]。

*证明*在不失一般性的前提下，我们将描述如何计算任意节点 v *的*聚类系数。为方便记述，我们使用 $\oplus$ 表示向量连接，使用 d_v 表示节点 v *的*度数。本证明需要算法 1 的 4 次迭代，其中我们在所有深度都使用了池聚合器。为了清晰起见，我们忽略了与向量归一化相关的问题，并使用了池聚合器可以以任意 ϵ*的*精度逼近任何豪斯多夫连续函数这一事实[29]。请注意，我们总是可以通过让聚合器在所有输出表示中预置一个单位值来考虑归一化常数（算法 1 中的第 7 行）；归一化常数可以在后面的层中通过取预置值的倒数来恢复。还要注意的是，几乎可以肯定存在下面描述的对称函数可以由池化聚合器（或其变体）精确计算的情况，但 [29] 的对称通用近似定理以及利普斯基茨连续性论证足以证明聚类系数的可识别性（达到任意精度）。特别是，我们需要近似计算聚类系数的下述函数，在其域上都是 Lipschitz 连续的（假设我们只在具有正度的节点上运行），因此近似引入的误差仍受固定常数的约束（可以任意变小）。

我们假定深度 $k = 2$ 和 $k = 3$ 的权重矩阵 $\mathbf{W}^1, \mathbf{W}^2$ 是相同的，并且所有非线性都是经过矫正的线性单元。此外，对于最后一次迭代（即 $k = 4$），我们完全忽略邻域信息，并简单地将该层视为具有单隐层的 MLP。定理 1 可以等同于需要算法 1 的 $\mathsf{K} = 3$ 次迭代，然后将表示输入单层 MLP。

根据 Lemma 3，我们可以假设在深度 $k = 1$ 时，v 的 2 跳邻域中的所有节点都有唯一的、单次触发的指示向量，即 $\mathbf{h}^1_v \in \mathsf{E}_I$。因此，在算法 1 中的深度 $k = 2$ 处，假设我们将相邻节点的非规范化表示相加。那么在不失一般性的前提下，我们将得到 $\mathbf{h}^2_v = \mathbf{h}^1_v \oplus \mathbf{A}_v$，其中 $\mathbf{A}_v$ 是包含所有连接节点的子图的邻接矩阵。

到 G^4 中的 v，而 $\mathbf{A}_v$ 是与 v 对应的邻接矩阵的行。那么，在深度 $k = 3$ 时，再次假设我们将邻接表示相加（权重矩阵为同一值），那么我们将得到

$$\mathbf{h}^3_v = \mathbf{h}^1_v \oplus \mathbf{A}_v \oplus \sum_{v \in \mathsf{N}(v)} \mathbf{h}^1_v \oplus \mathbf{A}_v \quad 。 \tag{10}$$

让 m 表示 $\mathbf{h}^1$ 向量的维度（即，根据阶式 3，$m \equiv \chi(G^4)$），并用方括号表示向量索引，我们可以发现

- $\mathbf{a} \equiv \mathbf{h}^3_v[0 : m]$ 是 v 的单击指示向量。
- $\mathbf{b} \equiv \mathbf{h}^3_v[m : 2m]$ 是 v 在邻接矩阵 $\mathbf{A}$ 中的行。
- $\mathbf{c} \equiv \mathbf{h}^3_v[3m : 4m]$ 是 v 的邻居邻接行的总和。

因此，我们可以得出 $\mathbf{b}^\mathsf{T}\mathsf{c}$ 是只包含 v 的子图中的边的数量，它直接邻居，$\sum_{i=0}^{m} \mathbf{b}[i] = d_v$。 最后，我们可以计算出

$$\frac{2(\mathbf{b}^\mathsf{T}\mathsf{c} - d_v)}{(d_v)(d_v - 1)} = \frac{2|\{e_{v,v'} : v, v' \in \mathsf{N}(v), e_{v,v'} \in \mathsf{E}\}|}{(d_v)(d_v - 1)}$$

(11)

$$= c_v \text{、} \tag{12}$$

由于这是 **h** 的连续函数[3]，我们可以用单层 MLP 将其近似到任意ϵ *的*精度（或者等价于算法 1 的多一次迭代，忽略邻域信息）。最后一步同样直接沿用 [15]。 □

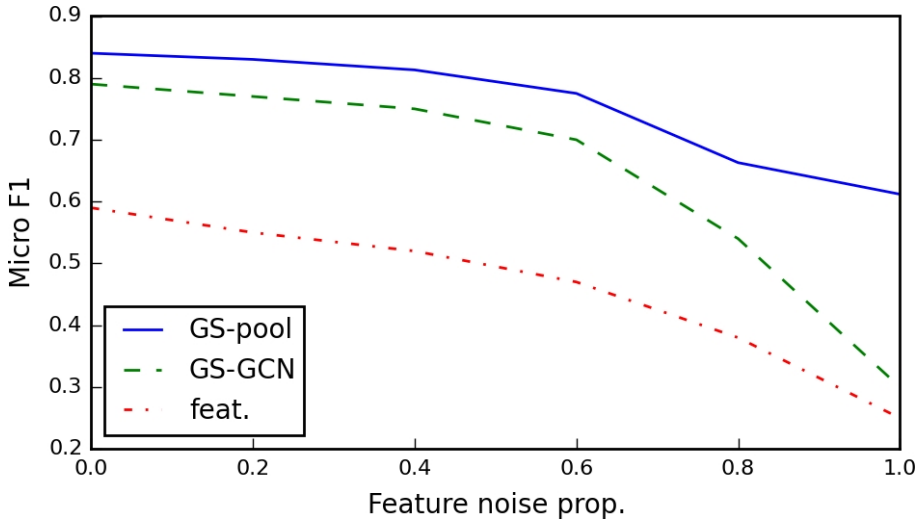

图 3：当特征矩阵逐渐被随机高斯噪声取代时，不同方法在引用数据上的准确率（以 F1 分数表示）。

推论 2 *假设我们从 $x \in U$ 的任意概率分布 μ 中抽取节点特征，其中 μ 相对于 Lebesgue 度量是绝对连续的。那么在特征输入 $\mathbf{x}_v \sim \mu$ 的情况下，定理 1 的条件几乎肯定满足。*

推论 2 是定理 1 和以下事实的直接结果：对于任何与 Lebesgue 测量绝对连续的概率分布，采样两个相同点的概率为零。根据经验，我们发现 GraphSAGE-pool 事实上能够利用图结构保持适度的性能，即使是完全随机的特征输入也是如此（见图 3）。然而，GraphSAGE-GCN 的性能却不那么稳健，这在直观上是合理的，因为推理 1、2 和 3 直接依赖于池聚合器的通用表达能力。

最后，我们注意到，定理 1 和推论 2 是针对特定给定图形表达的，因此具有一定的转导性。对于归纳环境，我们可以指出

推论 3. *假设对于属于某类图 G^* 的所有图 $S = (U, S)$，我们有 $\forall k, d \geq 0, k, d \in \mathbf{z}$ 这样的条件：*

$$\mathbf{h}_v^k = \mathbf{h}_v^{k'}, \forall (v, v') \in \{(v, v') : \forall u \in U, v, v' \in N^3(u)\}, \mathbf{h}_v^k, \mathbf{h}_v^{k'} \in S \text{、} {}^d$$

那么在算法 1 的 $K = k + 4$ 次迭代之后，我们就可以将聚类系数近似为任意的 ε。

推论 3 简单地指出，如果在算法 1 的 k 次迭代之后，我们能够学会唯一地识别一类图形的节点，那么我们也可以将这一类图形的聚类系数近似到任意精度。