Inductive Representation Learning on Large Graphs

William L. Hamilton[*] wleif@stanford.edu

Rex Ying[*] rexying@stanford.edu

Jure Leskovec jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

Abstract

Low-dimensional embeddings of nodes in large graphs have proved extremely useful in a variety of prediction tasks, from content recommendation to identifying protein functions. However, most existing approaches require that all nodes in the graph are present during training of the embeddings; these previous approaches are inherently *transductive* and do not naturally generalize to unseen nodes. Here we present GraphSAGE, a general *inductive* framework that leverages node feature information (e.g., text attributes) to efficiently generate node embeddings for previously unseen data. Instead of training individual embeddings for each node, we learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. Our algorithm outperforms strong baselines on three inductive node-classification benchmarks: we classify the category of unseen nodes in evolving information graphs based on citation and Reddit post data, and we show that our algorithm generalizes to completely unseen graphs using a multi-graph dataset of protein-protein interactions.

Abstract

Low-dimensional embeddings of nodes in large graphs have proved extremely useful in a variety of prediction tasks, from content recommendation to identifying protein functions. However, most existing approaches require that all nodes in the graph are present during training of the embeddings; these previous approaches are inherently transductive and do not naturally generalize to unseen nodes. Here we present GraphSAGE, a general inductive framework that leverages node feature information (e.g., text attributes) to efficiently generate node embeddings for previously unseen data. Instead of training individual embeddings for each node, we learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. Our algorithm outperforms strong baselines on three inductive node-classification benchmarks: we classify the category of unseen nodes in evolving information graphs based on citation and Reddit post data, and we show that our algorithm generalizes to completely unseen graphs using a multi-graph dataset of protein-protein interactions.

摘要

事实证明，在大型图中，节点的低维嵌入在从内容推荐到识别蛋白质功能的各种预测任务中极为有用。但是，大多数现有方法都要求在训练嵌入过程中存在图中的所有节点。这些先前的方法本质上是转导的，不能自然地推广到隐藏的节点。在这里，我们介绍 GraphSAGE，这是一个通用的归纳框架，它利用节点特征信息（例如，文本属性）为先前隐藏的数据有效地生成节点嵌入。我们将学习一个函数，该函数通过对节点本地邻居的特征进行采样和聚合来生成嵌入，而不是为每个节点训练单独的嵌入。我们的算法在三个归纳节点分类基准上均优于强基准：我们基于引文和 Reddit 上发布的数据对演化信息图中的隐藏节点类别进行分类，并且我们证明了我们的算法使用关于蛋白质-蛋白质相互作用的多图数据集将其归纳为完全隐藏图。

1 Introduction

Low-dimensional vector embeddings of nodes in large graphs have proved extremely useful as feature inputs for a wide variety of prediction and graph analysis tasks [5, 11, 28, 35, 36].

1 引言

大图中节点的低维向量嵌入已被证明对各种预测和图形分析任务的特征输入非常有用.

The basic idea behind node embedding approaches is to use dimensionality reduction techniques to distill the high-dimensional information about a node’s graph neighborhood into a dense vector embedding. These node embeddings can then be fed to downstream machine learning systems and aid in tasks such as node classification, clustering, and link prediction [11, 28, 35].

节点嵌入方法背后的基本想法是使用降维技术来,将有关节点图邻域的高维信息提取为密集的矢量嵌入,然后这些节点嵌入可以提供给下游的机器学习系统并协助完成诸如结点分类,聚类,链接预测等的任务.

However, previous works have focused on embedding nodes from a single fixed graph, and many real-world applications require embeddings to be quickly generated for unseen nodes, or entirely new (sub)graphs. This inductive capability is essential for high-throughput, production machine learning systems, which operate on evolving graphs and constantly encounter unseen nodes (e.g., posts on Reddit, users and videos on Youtube). An inductive approach to generating node embeddings also facilitates generalization across graphs with the same form of features: for example, one could train an embedding generator on protein-protein interaction graphs derived from a model organism, and then easily produce node embeddings for data collected on new organisms using the trained model.

但是，以前的工作集中于从单个固定图嵌入节点，并且许多实际应用程序需要为隐藏节点或全新的（子）图快速生成嵌入。这种归纳能力对于高吞吐量高生产力的机器学习系统至关重要，这些系统在进化图上运行，并不断遇上隐藏节点（例如，Reddit，用户和 YouTube 上的视频）。一个生成节点嵌入的归纳方法促进具有相同形式特征的图的泛化：例如，可以在源自模型生物的蛋白质-蛋白质相互作用图上训练一个嵌入生成器，然后使用训练后的模型轻松生成已收集的新生物数据的节点嵌入。

The inductive node embedding problem is especially difficult, compared to the transductive setting, because generalizing to unseen nodes requires “aligning” newly observed subgraphs to the node embeddings that the algorithm has already optimized on.

与传导设置相比，归纳结点嵌入问题等价困难，因为泛化到隐藏节点需要将新观察到的子图与该算法已对其进行优化的结点嵌入进行对齐。

An inductive framework must learn to recognize structural properties of a node’s neighborhood that reveal both the node’s local role in the graph, as well as its global position.

归纳框架必须学会识别节点邻域的结构属性，以揭示节点在图中的局部角色及其全局位置。

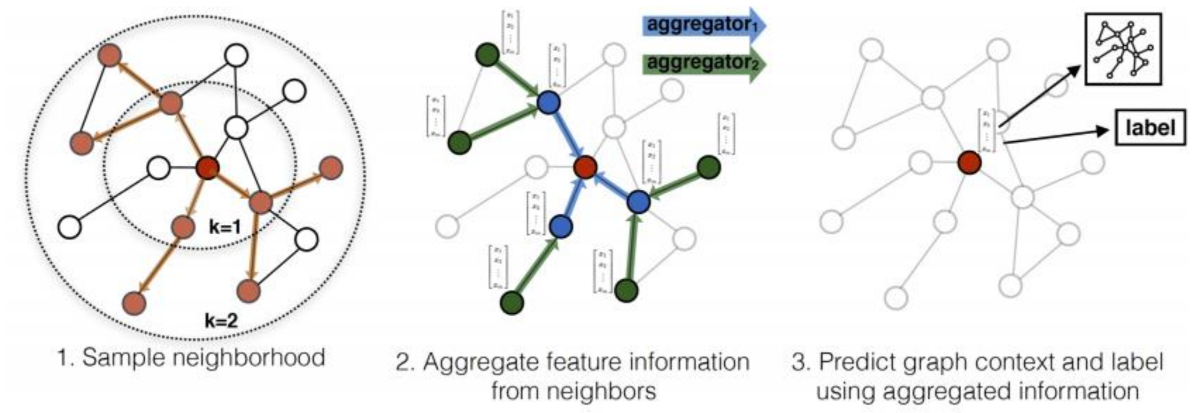

aggregator1
aggregator2
label
k=1
k=2
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

Most existing approaches to generating node embeddings are inherently transductive. The majority of these approaches directly optimize the embeddings for each node using matrix-factorization-based objectives, and do not naturally generalize to unseen data, since they make predictions on nodes in a single, fixed graph [5, 11, 23, 28, 35, 36, 37, 39]. These approaches can be modified to operate in an inductive setting (e.g., [28]), but these modifications tend to be computationally expensive, requiring additional rounds of gradient descent before new predictions can be made. There are also recent approaches to learning over graph structures using convolution operators that offer promise as an embedding methodology [17]. So far, graph convolutional networks (GCNs) have only been applied in the transductive setting with fixed graphs [17, 18]. In this work we both extend GCNs to the task of inductive unsupervised learning and propose a framework that generalizes the GCN approach to use trainable aggregation functions (beyond simple convolutions).

大多数现有的生成节点嵌入的方法本质上都是传导的。这些方法中的大多数使用基于矩阵分解的目标直接优化了每个节点的嵌入，并且自然不会将其归纳为隐藏数据，因为它们是对单一、固定的图上的结点进行预测。可以修改这些方法以操作归纳设置，但这些修改在计算上往往很昂贵，在做出新的预测之前，需要进行另外几轮梯度下降。

最近也有卷积运算符学习图结构的方法嵌入方法，它们为嵌入技术提供了前景。到目前为止，图卷积网络（GCN）仅被应用在带有固定图形的传导设置中[17，18]。在这项工作中，我们都将 GCN 扩展到了归纳式无监督学习，并提出一个框架，将 GCN 方法推广到使用可训练的聚合函数（除了简单的卷积）。

Present work. We propose a general framework, called GraphSAGE (SAmple and aggreGatE), for inductive node embedding. Unlike embedding approaches that are based on matrix factorization, we leverage node features (e.g., text attributes, node profile information, node degrees) in order to learn an embedding function that generalizes to unseen nodes. By incorporating node features in the learning algorithm, we simultaneously learn the topological structure of each node's neighborhood as well as the distribution of node features in the neighborhood. While we focus on feature-rich graphs (e.g., citation data with text attributes, biological data with functional/molecular markers), our approach can also make use of structural features that are present in all graphs (e.g., node degrees). Thus, our algorithm can also be applied to graphs without node features.

目前的工作。 我们提出了一个通用的框架，称为 GraphSAGE（SAmple 和 aggreGatE），用于归纳节点嵌入。 与基于矩阵分解的嵌入方法不同，我们利用节点功能（例如，文本属性，节点配置文件信息，节点度）来学习一个推广到隐藏节点的嵌入函数。 通过将节点功能合并到学习算法，我们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。虽然我们专注于特征多样的图（例如具有文本属性的引文数据，具有功能/分子标记的生物学数据），该方法还可以利用所有图形中都存在的结构特征（例如，节点度）。因此，我们的算法也可以应用于没有节点特征的图。

Instead of training a distinct embedding vector for each node, we train a set of aggregator functions that learn to aggregate feature information from a node's local neighborhood (Figure 1). Each aggregator function aggregates information from a different number of hops, or search depth, away from a given node. At test, or inference time, we use our trained system to generate embeddings for entirely unseen nodes by applying the learned aggregation functions. Following previous work on generating node embeddings, we design an unsupervised loss function that allows GraphSAGE to be trained without task-specific supervision. We also show that GraphSAGE can be trained in a fully supervised manner.

我们训练了一组学会从节点的本地邻域聚合特征信息的聚合函数，而不是为每个节点训练不同的嵌入向量。 每个聚合器函数从给定的节点意外的不同跳数或搜索深度聚合信息。 在测试或推断时，我们使用受过训练的系统,,通过应用学习到的聚合函数，来完全隐藏的节点生成嵌入。继以前的生成节点嵌入的工作，我们设计了一种无监督的损失函数，该函数可以使 GraphSAGE 无需特定任务的监督来完成训练。 我们还证明了 GraphSAGE 可以在完全监督下进行训练。

We evaluate our algorithm on three node-classification benchmarks, which test GraphSAGE's ability to generate useful embeddings on unseen data. We use two evolving document graphs based on citation data and Reddit post data (predicting paper and post categories, respectively), and a multigraph generalization experiment based on a dataset of protein-protein interactions (predicting protein functions). Using these benchmarks, we show that our approach is able to effectively generate representations for unseen nodes and outperform relevant baselines by a significant margin: across domains, our supervised approach improves classification F1-scores by an average of 51% compared to using node features alone and GraphSAGE consistently outperforms a strong, transductive baseline [28], despite this baseline taking ∼100× longer to run on unseen nodes. We also show that the new aggregator architectures we propose provide significant gains (7.4% on average) compared to an aggregator inspired by graph convolutional networks [17]. Lastly, we probe the expressive capability of our approach and show, through theoretical analysis, that GraphSAGE is capable of learning structural information about a node's role in a graph, despite the fact that it is inherently based on features (Section 5).

2 Related work

Our algorithm is conceptually related to previous node embedding approaches, general supervised approaches to learning over graphs, and recent advancements in applying convolutional neural networks to graph-structured data.

我们在三个节点分类基准上评估我们的算法，以测试 GraphSAGE 在隐藏数据上生成有用的嵌入的能力。我们使用两个基于引文数据和 Reddit 发布数据（分别预测论文和职位类别），以及基于蛋白质-蛋白质相互作用数据集（预测蛋白质功能）的多图概括实验。使用这些基准，我们表明我们的方法能够有效地产生隐藏节点的表示形式，并且以明显的优势超过了相关基线：

在各个领域中，相比起单独使用节点特征，我们的监督方法将分类 F1 分数平均提高了 51％，并且 GraphSAGE 始终优于强大，传导的基准，尽管这个基线需要花费约 100 倍的时间才能在隐藏节点上运行。我们还表明，与受图卷积网络启发的聚合器相比，我们提出的新聚合器架构可带来显着的收益（平均 7.4％）最后，我们探讨该方法的表达能力，并通过理论分析表明 GraphSAGE 具有学习关于节点在图中的角色的结构信息的能力，尽管事实上它是基于特征的。

2 相关工作

我们的算法在概念上与以前的节点嵌入方法，一般有监督的图学习方法和应用到图结构数据的卷积神经网络的最新进展有关。

Factorization-based embedding approaches. There are a number of recent node embedding approaches that learn low-dimensional embeddings using random walk statistics and matrix factorization-based learning objectives [5, 11, 28, 35, 36]. These methods also bear close relationships to more classic approaches to spectral clustering [23], multi-dimensional scaling [19], as well as the PageRank algorithm [25]. Since these embedding algorithms directly train node embeddings for individual nodes, they are inherently transductive and, at the very least, require expensive additional training (e.g., via stochastic gradient descent) to make predictions on new nodes. In addition, for many of these approaches (e.g., [11, 28, 35, 36]) the objective function is invariant to orthogonal transformations of the embeddings, which means that the embedding space does not naturally generalize between graphs and can drift during re-training. One notable exception to this trend is the Planetoid-I algorithm introduced by Yang et al. [40], which is an inductive, embedding-based approach to semi-supervised learning. However, Planetoid-I does not use any graph structural information during inference; instead, it uses the graph structure as a form of regularization during training. Unlike these previous approaches, we leverage feature information in order to train a model to produce embeddings for unseen nodes.

基于因子分解的嵌入方法。最近有许多节点嵌入方法使用随机游走统计和基于矩阵分解的学习目标来学习低维嵌入。这些方法还与更经典的光谱聚类方法，多维缩放，以及 PageRank 算法密切相关。由于这些嵌入算法对于单个节点的嵌入直接训练节点，它们本质上是可传导的，至少需要昂贵的额外训练（例如，通过随机梯度下降）以对新节点进行预测。此外，对于许多此类方法，目标函数对于嵌入的正交变换都是不变的，这意味着嵌入空间不会自然地在图之间进行概括，并且在重新训练期间可能会移动。一个值得注意的例外是 Yang 等人提出的 Planetoid-I 算法，这是一种基于归纳，基于嵌入的半监督学习方法。但是，Planetoid-I 在推理过程中不使用任何图形结构信息；相反，在训练期间它使用图结构作为正则化的形式。与这些先前的方法不同，我们利用特征信息来训练模型，为隐藏节点生成嵌入。

Supervised learning over graphs. Beyond node embedding approaches, there is a rich literature on supervised learning over graph-structured data. This includes a wide variety of kernel-based approaches, where feature vectors for graphs are derived from various graph kernels (see [32] and references therein). There are also a number of recent neural network approaches to supervised learning over graph structures [7, 10, 21, 31]. Our approach is conceptually inspired by a number of these algorithms. However, whereas these previous approaches attempt to classify entire graphs (or subgraphs), the focus of this work is generating useful representations for individual nodes.

有监督的图学习。 除了节点嵌入方法外，还有丰富的关于图结构数据的监督学习的文献资料，包括各种基于内核的方法，其中图的特征向量是从各种图内核派生的。 最近还有许多神经网络方法用于有监督的图学习。 我们的方法在概念上受到许多这些算法的启发。 但是，尽管这些先前的方法试图对整个图（或子图）进行分类，这项工作的重点是为各个节点生成有用的表示。

Graph convolutional networks. In recent years, several convolutional neural network architectures for learning over graphs have been proposed (e.g., [4, 9, 8, 17, 24]). The majority of these methods do not scale to large graphs or are designed for whole-graph classification (or both) [4, 9, 8, 24]. However, our approach is closely related to the graph convolutional network (GCN), introduced by Kipf et al. [17, 18]. The original GCN algorithm [17] is designed for semi-supervised learning in a transductive setting, and the exact algorithm requires that the full graph Laplacian is known during training. A simple variant of our algorithm can be viewed as an extension of the GCN framework to the inductive setting, a point which we revisit in Section 3.3.

图卷积网络。 近年来，几种用于学习图的卷积神经网络架构已经提出。 这些方法大多数不按比例放大大图或专为整图分类（或两者）而设计。但是，我们的方法与 Kipf 等人引入的图卷积网络（GCN）密切相关。 原始的 GCN 算法在传导设置中被设计用于半监督学习，并且确切的算法要求在整个训练过程中全图拉普拉斯算子已知。 我们算法的一个简单变体可以看作是 GCN 框架的在传导设置方面的扩展，我们将在第 3.3 节中重新讨论这一点。

3 Proposed method: GraphSAGE

The key idea behind our approach is that we learn how to aggregate feature information from a node's local neighborhood (e.g., the degrees or text attributes of nearby nodes). We first describe the GraphSAGE embedding generation (i.e., forward propagation) algorithm, which generates embeddings for nodes assuming that the GraphSAGE model parameters are already learned (Section 3.1). We then describe how the GraphSAGE model parameters can be learned using standard stochastic gradient descent and backpropagation techniques (Section 3.2).

3.1 Embedding generation (i.e., forward propagation) algorithm

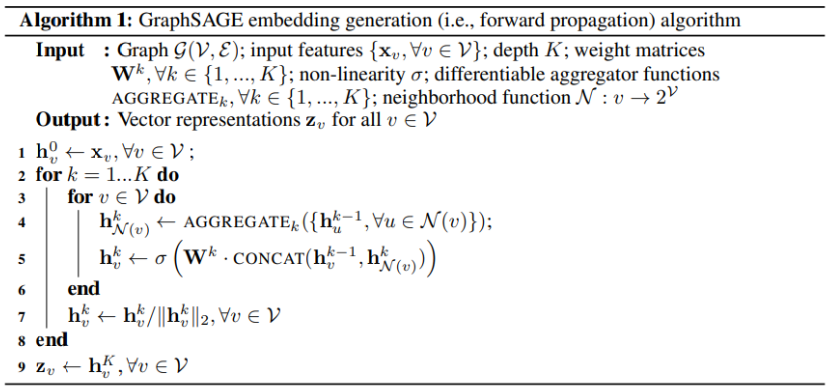
In this section, we describe the embedding generation, or forward propagation algorithm (Algorithm 1), which assumes that the model has already been trained and that the parameters are fixed. In particular, we assume that we have learned the parameters of K aggregator functions (denoted AGGREGATEk, ∀k ∈ {1, ..., K}), which aggregate information from node neighbors, as well as a set of weight matrices Wk , ∀k ∈ {1, ..., K}, which are used to propagate information between different layers of the model or "search depths". Section 3.2 describes how we train these parameters.

3 提出：GraphSAGE

我们的方法背后的关键思想是，我们将学习如何从节点的本地邻域（例如，附近节点的度数或文本属性）聚合特征信息。我们首先描述 GraphSAGE 嵌入生成（即前向传播）算法， 假设已经学习了 GraphSAGE 模型参数，则该算法为节点生成嵌入。 然后，我们描述使用标准的随机梯度下降和反向传播技术学习，GraphSAGE 模型参数是如何学习的。

3.1 嵌入生成（即前向传播）算法

在本节中，我们描述嵌入生成或前向传播算法（算法 1），它假设模型已经过训练并且参数是固定的。 特别地，我们假设我们已经了解了 K 个聚合函数的参数（表示为$AGGREGATE_k$，∀k∈{1，…，K}），它汇总了来自节点邻居的信息以及一个集合权重矩阵W^k，∀k∈{1，…，K}，用于在不同模型层或"搜索深度"之间传播信息。
3.2 节介绍了我们如何训练这些参数。



Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V},\mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1,...,K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1,...,K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

```
$\mathbf{h}^0_v \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
for $k = 1...K$ do
    for $v \in \mathcal{V}$ do
        $\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\})$;
        $\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$
    end
    $\mathbf{h}^k_v \leftarrow \mathbf{h}^k_v / \|\mathbf{h}^k_v\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}^K_v, \forall v \in \mathcal{V}$
```

The intuition behind Algorithm 1 is that at each iteration, or search depth, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gain more and more information from further reaches of the graph.

Algorithm 1 describes the embedding generation process in the case where the entire graph, G = (V, E), and features for all nodes xv, ∀v ∈ V, are provided as input. We describe how to generalize this to the minibatch setting below. Each step in the outer loop of Algorithm 1 proceeds as follows, where k denotes the current step in the outer loop (or the depth of the search) and h k denotes a node's representation at this step: First, each node v ∈ V aggregates the representations of the nodes in its immediate neighborhood, {h k−1 u , ∀u ∈ N (v)}, into a single vector h k−1 N(v) . Note that this aggregation step depends on the representations generated at the previous iteration of the outer loop (i.e., k − 1), and the k = 0 ("base case") representations are defined as the input node features. After aggregating the neighboring feature vectors, GraphSAGE then concatenates the node's current representation, h k−1 v , with the aggregated neighborhood vector, h k−1 N(v) , and this concatenated vector is fed through a fully connected layer with nonlinear activation function σ, which transforms the representations to be used at the next step of the algorithm (i.e., h k v , ∀v ∈ V). For notational convenience, we denote the final representations output at depth K as zv ≡ h K v , ∀v ∈ V. The aggregation of the neighbor representations can be done by a variety of aggregator architectures (denoted by the AGGREGATE placeholder in Algorithm 1), and we discuss different architecture choices in Section 3.3 below.

算法 1 的直觉是每次迭代或搜索深度时都会聚合信息，节点从本地邻居聚集信息。随着这个过程的反复进行，节点从图的进一步延伸中增量地获得越来越多的信息。

算法 1 描述了在整个图形 G =(V,E)和所有节点 xv, ∀v∈V 的特征作为输入的情况下的嵌入生成过程。下面我们描述了如何概括这些到小批量设置。算法 1 外循环的每个步骤如下进行：

其中 k 表示外循环中的当前步长（或搜索深度），h^k表示节点此步骤的表示形式：首先，每个节点 v∈V 聚合其邻域节点中的表示，$\{h_u^{k-1}, \forall\, u \ \in N(v)\}$，表示成单个向量$h_{N(v)}^{k-1}$。注意，此聚合步长取决于外循环先前迭代（即 k − 1）生成的表示形式。k = 0（"基本情况"）的形式被定义为输入节点特征。在聚合邻域特征向量后，，GraphSAGE 连接该节点的当前表示，h_v^{k-1}，结合聚合的邻域向量$h_{N(v)}^{k-1}$

，并且该连接层向量通过一个具有非线性激活函数σ的全连接层馈入，它将在算法的下一步使用的表示形式进行进行转换。（即$h_v^k, \forall\, v \ \in V$）。为了符号上的方便，我们将深度 K 的最终表示输出为 $z_v \equiv \ h_v^K, \forall\, v \ \in V$。邻居的聚合

表示可以通过多种聚合器架构（以算法 1 中的 AGGREGATE 占位符表示）完成。我们将在下面的第 3.3 节中讨论不同的体系结构选择。

To extend Algorithm 1 to the minibatch setting, given a set of input nodes, we first forward sample the required neighborhood sets (up to depth K) and then we run the inner loop (line 3 in Algorithm 1), but instead of iterating over all nodes, we compute only the representations that are necessary to satisfy the recursion at each depth (Appendix A contains complete minibatch pseudocode).

Relation to the Weisfeiler-Lehman Isomorphism Test. The GraphSAGE algorithm is conceptually inspired by a classic algorithm for testing graph isomorphism. If, in Algorithm 1, we (i) set K = |V|, (ii) set the weight matrices as the identity, and (iii) use an appropriate hash function as an aggregator (with no non-linearity), then Algorithm 1 is an instance of the Weisfeiler-Lehman (WL) isomorphism test, also known as “naive vertex refinement” [32]. If the set of representations {zv, ∀v ∈ V} output by Algorithm 1 for two subgraphs are identical then the WL test declares the two subgraphs to be isomorphic. This test is known to fail in some cases, but is valid for a broad class of graphs [32]. GraphSAGE is a continuous approximation to the WL test, where we replace the hash function with trainable neural network aggregators. Of course, we use GraphSAGE to generate useful node representations–not to test graph isomorphism. Nevertheless, the connection between GraphSAGE and the classic WL test provides theoretical context for our algorithm design to learn the topological structure of node neighborhoods.

为了将算法 1 扩展到小批量设置，给定一组输入节点，我们首先转发样本所需的邻域集（直到深度 K），然后运行内部循环（算法 1 中的第 3 行），但是我们不会遍历所有节点，而是只计算必要满足每个深度递归的表示形式。（附录 A 包含完整的小批量伪代码）。

与 Weisfeiler-Lehman 同构测验的关系。GraphSAGE 算法在概念上是受测试同构图的经典算法的启发。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（不是非线性）

则算法 1 是 Weisfeiler-Lehman（WL）同构测试的一个实例，也称为“朴素顶点细化”。通过算法 1，如果表示集$z_v \forall v \in V$输出对于两个子图是相同的，则 WL 测试声明两个子图为同构。已知该测试在某些情况下会失败，但对于广泛的图形类型来说是有效的 GraphSAGE 是 WL 测试的连续近似，其中我们替换了哈希函数为一个可训练的神经网络聚合器。当然，我们使用 GraphSAGE 生成有用的节点表示，不是为了测试图同构。尽管如此，GraphSAGE 域经典 WL 测试之间的联系为我们学习节点邻域的拓扑结构的算法设计提供了理论背景。

Neighborhood definition. In this work, we uniformly sample a fixed-size set of neighbors, instead of using full neighborhood sets in Algorithm 1, in order to keep the computational footprint of each batch fixed. That is, using overloaded notation, we define N (v) as a fixed-size, uniform draw from the set {u ∈ V : (u, v) ∈ E}, and we draw different uniform samples at each iteration, k, in Algorithm 1. Without this sampling the memory and expected runtime of a single batch is unpredictable and in the worst case O(|V|). In contrast, the per-batch space and time complexity for GraphSAGE is fixed at O(QK i=1 Si), where Si , i ∈ {1, ..., K} and K are user-specified constants. Practically speaking we found that our approach could achieve high performance with K = 2 and S1 · S2 ≤ 500 (see Section 4.4 for details).

邻域定义。 在这项工作中，我们统一采样一组固定大小的邻域，而不是使用算法 1 中的完整邻域集，以保持每个批次的计算足迹固定。也就是说，使用重载符号，我们将$N(v)$定义为固定大小，从集合$\{u\ \in V : (u, v) \in\ \varepsilon\}$中均匀绘制。同时我们在算法 1 的每次迭代 k 中绘制了不同的均匀样本。如果没有此采样，单个批次的内存和预期运行时间将无法预测，并且

最坏的情况$O(|V|)$。相反，GraphSAGE 的每个批处理空间和时间复杂度固定为

$O(\prod_{i=1}^{K} S_i)$，其中$S_i, i\ \in \{1, \dots, K\}$与$K$是用户指定的常数。实际上来说我们发现，当 K = 2 且 S1·S2≤ 500 时，我们的方法可以到达较高性能（详情请参阅第 4.4 节。）

3.2 Learning the parameters of GraphSAGE

In order to learn useful, predictive representations in a fully unsupervised setting, we apply a graph-based loss function to the output representations, zu, ∀u ∈ V, and tune the weight matrices, Wk , ∀k ∈ {1, ..., K}, and parameters of the aggregator functions via stochastic gradient descent. The graph-based loss function encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct:

3.2 学习 GraphSAGE 的参数

为了在完全不受监督的情况下学习有用的预测表示，我们应用了基于图的损失函数到输出表示$z_u, \forall\ u\ \in V$，并调整权重矩阵，$W^k, \forall\ k\ \in \{1 \dots K\}$，以及聚合函数的参数通过随机梯度下降来实现。

基于图的损失函数鼓励附近的节点具有相似的表示，同时强制不同的节点的表示形式非常不同：

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

where v is a node that co-occurs near u on fixed-length random walk, σ is the sigmoid function, Pn is a negative sampling distribution, and Q defines the number of negative samples. Importantly, unlike previous embedding approaches, the representations zu that we feed into this loss function are generated from the features contained within a node’s local neighborhood, rather than training a unique embedding for each node (via an embedding look-up). This unsupervised setting emulates situations where node features are provided to downstream machine learning applications, as a service or in a static repository. In cases where representations are to be used only on a specific downstream task, the unsupervised loss (Equation 1) can simply be replaced, or augmented, by a task-specific objective (e.g., cross-entropy loss)

其中 v 是在固定长度且随机游走的节点 u 附近共同出现的一个节点，σ是 S 型函数，P_n是负采样分布，而 Q 定义了负采样样本数。 重要的是，与以前的嵌入方法不同，我们输入到此损失函数中的z_u

由节点本地邻域中包含的特征生成，而不是从训练每个节点的唯一嵌入（通过嵌入查找）生成。此无人监督的设置模拟了向下游机器学习应用程序提供节点特征的情况，如作为一个服务或在一个静态存储库中。 在有表示的情况下仅用于特定的下游任务，无监督损失（等式 1）可以简单地由特定任务的目标（例如，交叉熵损失）替代或增强。

3.3 Aggregator Architectures

Unlike machine learning over N-D lattices (e.g., sentences, images, or 3-D volumes), a node’s neighbors have no natural ordering; thus, the aggregator functions in Algorithm 1 must operate over an unordered set of vectors. Ideally, an aggregator function would be symmetric (i.e., invariant to permutations of its inputs) while still being trainable and maintaining high representational capacity. The symmetry property of the aggregation function ensures that our neural network model can be trained and applied to arbitrarily ordered node neighborhood feature sets. We examined three candidate aggregator functions:

Mean aggregator. Our first candidate aggregator function is the mean operator, where we simply take the elementwise mean of the vectors in {h k−1 u , ∀u ∈ N (v)}.

3.3 聚合器架构

与通过 N-D 格（例如句子，图像或 3-D 体）进行机器学习不同，节点的邻居没有自然的秩序；因此，算法 1 中的聚合函数必须在操作在无序的向量集合上。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），同时仍是可训练的并保持较高的代表能力。

聚合器函数的对称性确保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。我们检查了三个候选聚合器函数：

均值聚合器。我们的第一个候选聚合器函数是均值运算符，在这里我们简单地

取向量$h_u^{k-1}, \forall u \in N(v)$的元素均值。

The mean aggregator is nearly equivalent to the convolutional propagation rule used in the transductive GCN framework [17]. In particular, we can derive an inductive variant of the GCN approach by replacing lines 4 and 5 in Algorithm 1 with the following:

平均聚合器几乎等效于传导 GCN 框架中使用的卷积传播规则。特别地，我们可以像下面这样通过替换算法 1 中第 4 行和第 5 行来推导 GCN 方法的归纳变量：

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

We call this modified mean-based aggregator convolutional since it is a rough, linear approximation of a localized spectral convolution [17]. An important distinction between this convolutional aggregator and our other proposed aggregators is that it does not perform the concatenation operation in line 5 of Algorithm 1—i.e., the convolutional aggregator does concatenate the node's previous layer representation h k−1 v with the aggregated neighborhood vector h k N(v) . This concatenation can be viewed as a simple form of a "skip connection" [13] between the different "search depths", or "layers" of the GraphSAGE algorithm, and it leads to significant gains in performance (Section 4)

我们称其为修正的基于均值的聚合器卷积，因为它是的粗略线性近似局部频谱卷积。 这种卷积聚合器之间的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 的第 5 步中的串联操作，即卷积聚合器确实将节点的上一层表示h_v^{k-1}用聚集的邻域向量$h_{N(v)}^k$ 连接在一起。 此串联可以是被视为在不同"搜索深度"或 GraphSAGE 中的"层"之间"跳过连接" [13] 的简单形式，这样可显着提高性能（第 4 节）

LSTM aggregator. We also examined a more complex aggregator based on an LSTM architecture [14]. Compared to the mean aggregator, LSTMs have the advantage of larger expressive capability. However, it is important to note that LSTMs are not inherently symmetric (i.e., they are not permutation invariant), since they process their inputs in a sequential manner. We adapt LSTMs to operate on an unordered set by simply applying the LSTMs to a random permutation of the node's neighbors

LSTM 聚合器。 我们还研究了基于 LSTM 架构的更复杂的聚合器。 与平均聚合器相比，LSTM 具有更大的表达能力优势。但是，请务必注意，LSTM 不是天生对称的（即它们不是置换不变的），因为它们以顺序方式处理其输入。

通过简单地应用 LSTEM 到节点邻居的随机排列，我们改进 LSTM 使其可以操作在一个无序集上。

Pooling aggregator. The final aggregator we examine is both symmetric and trainable. In this pooling approach, each neighbor's vector is independently fed through a fully-connected neural network; following this transformation, an elementwise max-pooling operation is applied to aggregate information across the neighbor set:

池聚合器。 我们最后检测的池聚合器既对称又可训练。 在这个池化方法，每个邻域向量通过完全连接的神经元网络独立地馈入； 进行此转换之后，将一个元素化最大池操作应用于聚合跨邻域集的信息：

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

where max denotes the element-wise max operator and σ is a nonlinear activation function. In principle, the function applied before the max pooling can be an arbitrarily deep multi-layer perceptron, but we focus on simple single-layer architectures in this work. This approach is inspired by recent advancements in applying neural network architectures to learn over general point sets [29]. Intuitively, the multi-layer perceptron can be thought of as a set of functions that compute features for each of the node representations in the neighbor set. By applying the max-pooling operator to each of the computed features, the model effectively captures different aspects of the neighborhood set. Note also that, in principle, any symmetric vector function could be used in place of the max operator (e.g., an element-wise mean). We found no significant difference between max- and mean-pooling in developments test and thus focused on max-pooling for the rest of our experiments.

其中 max 表示面向元素的 max 运算符，而σ是非线性激活函数。原则上来说，最大池化之前应用的函数可以是任意深度的多层感知器，但在此工作中，我们专注于简单的单层体系结构。 这种方法的灵感来自于应用神经网络体系结构来学习通用点集的最新进展。直观地讲，多层感知器可以被认为是计算每个节点在邻域集中表示特征的一组函数。 通过将最大池运算符应用于每个根据计算出的特征，模型可以有效地捕获邻域集的不同方面。 注意，原则上，可以使用任何对称矢量函数代替 max 运算符（例如，按元素表示的均值）。 我们发现最大池化和平均池化之间在开发测试中没有显着差异，因此在其余的实验中将着重于最大池化。

4 Experiments

We test the performance of GraphSAGE on three benchmark tasks: (i) classifying academic papers into different subjects using the Web of Science citation dataset, (ii) classifying Reddit posts as belonging to different communities, and (iii) classifying protein functions across various biological protein-protein interaction (PPI) graphs. Sections 4.1 and 4.2 summarize the datasets, and the supplementary material contains additional information. In all these experiments, we perform predictions on nodes that are not seen during training, and, in the case of the PPI dataset, we test on entirely unseen graphs.

Experimental set-up. To contextualize the empirical results on our inductive benchmarks, we compare against four baselines: a random classifer, a logistic regression feature-based classifier (that ignores graph structure), the DeepWalk algorithm [28] as a representative factorization-based approach, and a concatenation of the raw features and DeepWalk embeddings. We also compare four variants of GraphSAGE that use the different aggregator functions (Section 3.3). Since, the "convolutional" variant of GraphSAGE is an extended, inductive version of Kipf et al's semi-supervised GCN [17], we term this variant GraphSAGE-GCN. We test unsupervised variants of GraphSAGE trained according to the loss in Equation (1), as well as supervised variants that are trained directly on classification cross-entropy loss. For all the GraphSAGE variants we used rectified linear units as the non-linearity and set K = 2 with neighborhood sample sizes S1 = 25 and S2 = 10 (see Section 4.4 for sensitivity analyses)

4 实验

我们在三个基准任务上测试 GraphSAGE 的性能：

（i）使用 Web of Science 引文数据集，对学术论文进行分类，分类为不同的主题，（ii）将 Reddit 帖子分类为不同的社区，（iii）基于蛋白质-蛋白质相互作用（PPI）图对各种生物的蛋白质功能进行分类

4.1 节和 4.2 节总结了数据集，补充材料包含的其他信息。 在所有这些实验中，我们对训练期间的隐藏节点进行预测，对于 PPI 数据集，我们在完全隐藏图上进行测试。

实验建立

为了将归纳基准上的经验结果进行情境化，我们与四个基线进行比较：

(1)随机分类器

(2)基于逻辑回归特征的分类器（忽略图结构）

(3)基于分解方法代表的 DeepWalk 算法

(4)原始功能和 DeepWalk 嵌入的串联

我们还比较了四个使用不同聚合函数的 GraphSAGE 变体（第 3.3 节）。由于 GraphSAGE 的"卷积"变体是 Kipf 等人的半监督 GCN 的扩展、归纳的版本。我们将其称为 GraphSAGE-GCN。我们测试根据公式（1）中的损失进行训练的 GraphSAGE 的无监督变体，以及受分类交叉熵损失直接训练的有监督变体。对于所有 GraphSAGE 变体，我们使用整流线性单位作为非线性和设置 K = 2，加上邻域样本大小为 S1 = 25 和 S2 = 10（请参见 4.4 进行敏感性分析）

For the Reddit and citation datasets, we use "online" training for DeepWalk as described in Perozzi et al. [28], where we run a new round of SGD optimization to embed the new test nodes before making predictions (see the Appendix for details). In the multi-graph setting, we cannot apply DeepWalk, since the embedding spaces generated by running the DeepWalk algorithm on different disjoint graphs can be arbitrarily rotated with respect to each other (Appendix D).

对于 Reddit 和引文数据集，如 Perozzi 等所述，我们对 DeepWalk 使用"在线"训练

，其中我们在进行预测之前将运行新一轮的 SGD 优化去嵌入新测试的节点（详细信息见附录）。在多图设置中，我们无法应用 DeepWalk，因为通过在不同的不相交处的图运行 DeepWalk 算法生成的嵌入空间可以相对于彼此任意地旋转（附录 D）。

All models were implemented in TensorFlow [1] with the Adam optimizer [16] (except DeepWalk, which performed better with the vanilla gradient descent optimizer). We designed our experiments with the goals of (i) verifying the improvement of GraphSAGE over the baseline approaches (i.e., raw features and DeepWalk) and (ii) providing a rigorous comparison of the different GraphSAGE aggregator architectures. In order to provide a fair comparison, all models share an identical implementation of their minibatch iterators, loss function and neighborhood sampler (when applicable). Moreover, in order to guard against unintentional "hyperparameter hacking" in the comparisons between GraphSAGE aggregators, we sweep over the same set of hyperparameters for all GraphSAGE variants (choosing the best setting for each variant according to performance on a validation set). The set of possible hyperparameter values was determined on early validation tests using subsets of the citation and Reddit data that we then discarded from our analyses. The appendix contains further implementation details.

所有模型均使用 Adam 优化器在 TensorFlow 中实现（除了 DeepWalk，使用批梯度下降优化器效果更好）。我们设计了实验，目的是

（i）验证 GraphSAGE 与基准方法相比的改进（即原始特征和 DeepWalk）

（ii）对不同的 GraphSAGE 聚合架构进行严格的比较

公平起见，所有模型都共享其最小批量迭代器，损失函数和邻域采样器（如果适用）的相同实现。此外，为了在 GraphSAGE 聚合器之间进行比较时防止意外的"超参数黑客"行为，我们对所有 GraphSAGE 的变体进行了相同的超参数设置（根据验证集的性能为每个变体选择最佳设置）。在早期验证测试中我们使用

引用的子集和 Reddit 数据确定了一组可能的超参数值，然后我们从分析中将其丢弃。附录还包含进一步的实施细节。

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	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
% gain over feat.	39%	46%	55%	63%	19%	45%

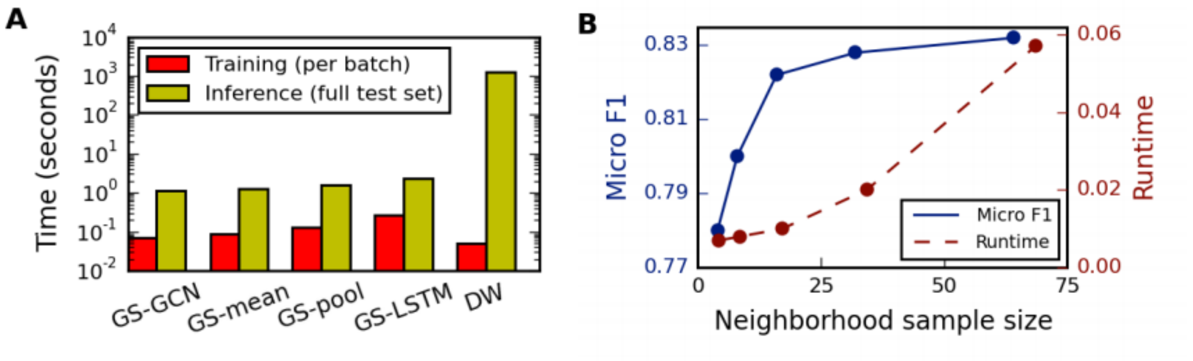


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each depth for $K = 2$ with $S_1 = S_2$ (on the citation data using GraphSAGE-mean).

4.1 Inductive learning on evolving graphs: Citation and Reddit data

Our first two experiments are on classifying nodes in evolving information graphs, a task that is especially relevant to high-throughput production systems, which constantly encounter unseen data.

4.1 演化图的归纳学习：引文和 Reddit 数据

我们的前两个实验是在不断发展的信息图中对节点进行分类，这一任务与与高吞吐量的生产系统密切相关，这些系统经常遇到隐藏数据。

Citation data. Our first task is predicting paper subject categories on a large citation dataset. We use an undirected citation graph dataset derived from the Thomson Reuters Web of Science Core Collection, corresponding to all papers in six biology-related fields for the years 2000-2005. The node labels for this dataset correspond to the six different field labels. In total, this is dataset contains 302,424 nodes with an average degree of 9.15. We train all the algorithms on the 2000-2004 data and use the 2005 data for testing (with 30% used for validation). For features, we used node degrees and processed the paper abstracts according Arora et al.'s [2] sentence embedding approach, with 300-dimensional word vectors trained using the GenSim word2vec implementation [30].

引文数据。 我们的首要任务是在大型引用数据集上预测论文主题类别。 我们使用一个源自 Thomson Reuters Web of Science 核心集合的无向图数据集，对应于 2000-2005 年六个生物学相关领域的所有论文。 该数据集的节点标签对应于六个不同的字段标签。 总的来说，这是包含 302,424 个节点，平均度为 9.15 的数据集。 我们根据 2000-2004 年的数据训练所有算法，并使用 2005 年的数据进行测试（其中 30％用于验证）。 对于特征，我们使用节点度，并根据 Arora 等人的句子嵌入方法，使用 GenSim word2vec 实现训练的 300 维单词向量，处理论文摘要。

Reddit data. In our second task, we predict which community different Reddit posts belong to. Reddit is a large online discussion forum where users post and comment on content in different topical communities. We constructed a graph dataset from Reddit posts made in the month of September, 2014. The node label in this case is the community, or "subreddit", that a post belongs to. We sampled 50 large communities and built a post-to-post graph, connecting posts if the same user comments on both. In total this dataset contains 232,965 posts with an average degree of 492. We use the first 20 days for training and the remaining days for testing (with 30% used for validation). For features, we use off-the-shelf 300-dimensional GloVe CommonCrawl word vectors [27]; for each post, we concatenated (i) the average embedding of the post title, (ii) the average embedding of all the post's comments (iii) the post's score, and (iv) the number of comments made on the post.

Reddit 数据。 在第二个任务中，我们预测不同 Reddit 帖子属于哪个社区。Reddit 是一个大型在线讨论论坛，用户可以在其中发布和评论不同主题社区的内容。 我们根据 2014 年 9 月的 Reddit 帖子构建了一个图形数据集。在这种情况下，节点标签是帖子所属的社区或" subreddit"。 我们取样了 50 个大型社区，并构建了一个 post-to-post 图，如果同一用户对两者都发表了评论，则将这些帖子连接起来。该数据集总共包含 232,965 个帖子，平均度为 492。我们使用 20 天的训练时间，剩余的时间用于测试（其中 30％用于验证）。对于特征，我们使用现成的 300 维 GloVe CommonCrawl 词向量； 对于每个帖子，我们将（i）帖子标题的平均嵌入（ii）所有帖子评论的平均嵌入

（iii）帖子的分数（iv）对帖子的评论数量

连接起来。

The first four columns of Table 1 summarize the performance of GraphSAGE as well as the baseline approaches on these two datasets. We find that GraphSAGE outperforms all the baselines by a significant margin, and the trainable, neural network aggregators provide significant gains compared to the GCN approach. For example, the unsupervised variant GraphSAGE-pool outperforms the concatenation of the DeepWalk embeddings and the raw features by 13.8% on the citation data and 29.1% on the Reddit data, while the supervised version provides a gain of 19.7% and 37.2%, respectively. Interestingly, the LSTM based aggregator shows strong performance, despite the fact that it is designed for sequential data and not unordered sets. Lastly, we see that the performance of unsupervised GraphSAGE is reasonably competitive with the fully supervised version, indicating that our framework can achieve strong performance without task-specific fine-tuning.

表 1 的前四列总结了 GraphSAGE 的性能以及这两个数据集的基线接近程度。 相比之下，我们发现 GraphSAGE 的性能优于所有基线，同时可训练的神经网络聚合器，相比起 GCN 方法，提供了显著的利润。例如，在引文数据与 Reddit 数据上，无监督变体 GraphSAGE-pool 的性能分别比 DeepWalk 嵌入和原始特征的连接优于 13.8％与 29.1%，而有监督版本的性能分别提高了 19.7％和 37.2％。

有趣的是，基于 LSTM 的聚合器展现了强大的性能，尽管它是为顺序数据而不是无序集而设计的。 最后，我们看到无监督的 GraphSAGE 与完全受监督的版本相比具有相当的竞争力，这表明我们的框架无需特定于任务的微调就可以实现强大的性能。

4.2 Generalizing across graphs: Protein-protein interactions

We now consider the task of generalizing across graphs, which requires learning about node roles rather than community structure. We classify protein roles—in terms of their cellular functions from gene ontology—in various protein-protein interaction (PPI) graphs, with each graph corresponding to a different human tissue [41]. We use positional gene sets, motif gene sets and immunological signatures as features and gene ontology sets as labels (121 in total), collected from the Molecular Signatures Database [34]. The average graph contains 2373 nodes, with an average degree of 28.8. We train all algorithms on 20 graphs and then average prediction F1 scores on two test graphs (with two other graphs used for validation).

4.2 跨图归纳：蛋白质-蛋白质相互作用

现在，我们考虑跨图进行归纳的任务，这需要了解节点角色而不是社区结构。我们根据蛋白质的细胞功能从基因本体中对蛋白质角色进行分类—在各种蛋白质-蛋白质相互作用（PPI）图中，每个图都对应

到不同的人体组织。我们使用位置基因集，基序基因集和免疫学标记作为特征，基因本体集作为标记（共 121 个），从分子标记数据库中收集。图平均包含 2373 个节点，平均度为 28.8。我们在 20 张图上训练所有算法，然后在两张测试图上平均预测 F1 得分（另外两个用于验证的图表）。

The final two columns of Table 1 summarize the accuracies of the various approaches on this data. Again we see that GraphSAGE significantly outperforms the baseline approaches, with the LSTM- and pooling-based aggregators providing substantial gains over the mean- and GCN-based aggregators.

表 1 的最后两列总结了各类方法在这些数据上的准确性。再次，我们发现 GraphSAGE 明显优于基准方法，基于 LSTM 和基于池的聚合器比基于均值和 GCN 的聚合器具有可观的性能提升。

4.3 Runtime and parameter sensitivity

Figure 2.A summarizes the training and test runtimes for the different approaches. The training time for the methods are comparable (with GraphSAGE-LSTM being the slowest). However, the need to sample new random walks and run new rounds of SGD to embed unseen nodes makes DeepWalk 100-500× slower at test time.

For the GraphSAGE variants, we found that setting K = 2 provided a consistent boost in accuracy of around 10-15%, on average, compared to K = 1; however, increasing K beyond 2 gave marginal returns in performance (0-5%) while increasing the runtime by a prohibitively large factor of 10-100×, depending on the neighborhood sample size. We also found diminishing returns for sampling large neighborhoods (Figure 2.B). Thus, despite the higher variance induced by sub-sampling neighborhoods, GraphSAGE is still able to maintain strong predictive accuracy, while significantly improving the runtime.

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是可比的（GraphSAGE-LSTM 最慢）。但是，需要采样新的随机游走并运行新一轮的 SGD 去嵌入隐藏节点，使 DeepWalk 测试时间慢 100-500 倍。

对于 GraphSAGE 变体，我们发现设置 K = 2 始终可以提高精度，与 K = 1 相比，平均约为 10-15％； 但是，将 K 增加到 2 以上会降低 0-5%的性能，同时运行时间延长了 10-100 倍，取决于邻域样本量。 我们还发现抽样大型社区的收益递减（图 2.B）。因此，尽管子采样导致较高的方差，GraphSAGE 仍然能够保持强大的预测准确性，同时显著地改善了运行时间。

4.4 Summary comparison between the different aggregator architectures

Overall, we found that the LSTM- and pool-based aggregators performed the best, in terms of both average performance and number of experimental settings where they were the top-performing method (Table 1). To give more quantitative insight into these trends, we consider each of the six different experimental settings (i.e., (3 datasets) × (unsupervised vs. supervised)) as trials and consider what performance trends are likely to generalize. In particular, we use the non-parametric Wilcoxon Signed-Rank Test [33] to quantify the differences between the different aggregators across trials, reporting the T-statistic and p-value where applicable. Note that this method is rank-based and essentially tests whether we would expect one particular approach to outperform another in a new experimental setting. Given our small sample size of only 6 different settings, this significance test is somewhat underpowered; nonetheless, the T-statistic and associated p-values are useful quantitative measures to assess the aggregators’ relative performances.

We see that LSTM-, pool- and mean-based aggregators all provide statistically significant gains over the GCN-based approach (T = 1.0, p = 0.02 for all three). However, the gains of the LSTM and pool approaches over the mean-based aggregator are more marginal (T = 1.5, p = 0.03, comparing LSTM to mean; T = 4.5, p = 0.10, comparing pool to mean). There is no significant difference between the LSTM and pool approaches (T = 10.0, p = 0.46). However, GraphSAGE-LSTM is significantly slower than GraphSAGE-pool (by a factor of ≈2×), perhaps giving the pooling-based aggregator a slight edge overall.

4.4 不同聚合器架构之间的摘要比较

总体而言，我们发现基于 LSTM 和基于池的聚合器在这两个方面均表现最佳：

平均性能和实验设置的数量，这里它们是表现最佳的方法（表 1）。为了更定量地了解这些趋势，我们考虑了六种不同的实验设置（即（3 个数据集）×（无监督与有监督））作为试验和考虑哪些性能趋势可能会泛化。特别地，通过试验，我们使用非参数 Wilcoxon Signed-Rank 测试以量化跨不同聚合器之间的差异，并在合适时候报告 T 统计量和 p 值。请注意，此方法是基于排名的，本质上是测试我们是否期望一种特定的方法优于在一个新实验设置中的另一种方法。鉴于我们只有 6 种不同设置的小样本量，显着性检验显得有些

动力不足；尽管如此，T 统计量和相关的 p 值还是有用的评估聚合器相对性能的措施。

我们看到，与基于 GCN 的方法相比，基于 LSTM 的，基于池的和基于均值的聚合器均提供了统计上的显着的收益（这三种方法中都是 T = 1.0，p = 0.02）。但是，与基于均值的聚合器相比，LSTM 和池化方法的收益更为微不足道（T = 1.5，p = 0.03，LSTM 与均值比较； T = 4.5，p = 0.10，池化与均值比较）。LSTM 和池化方法之间没有显着差异（T = 10.0，p = 0.46）。但是，GraphSAGE-LSTM 比 GraphSAGE-pool 慢得多（约 2 倍），这可能会使基于池的聚合器总体上略有优势。

5 Theoretical analysis

In this section, we probe the expressive capabilities of GraphSAGE in order to provide insight into how GraphSAGE can learn about graph structure, even though it is inherently based on features. As a case-study, we consider whether GraphSAGE can learn to predict the clustering coefficient of a node, i.e., the proportion of triangles that are closed within the node's 1-hop neighborhood [38]. The clustering coefficient is a popular measure of how clustered a node's local neighborhood is, and it serves as a building block for many more complicated structural motifs [3]. We can show that Algorithm 1 is capable of approximating clustering coefficients to an arbitrary degree of precision:

Theorem 1. Let xv ∈ U, ∀v ∈ V denote the feature inputs for Algorithm 1 on graph G = (V, E), where U is any compact subset of R d . Suppose that there exists a fixed positive constant C ∈ R + such that kxv − xv 0k2 > C for all pairs of nodes. Then we have that ∀> 0 there exists a parameter setting Θ∗ for Algorithm 1 such that after K = 4 iterations |zv − cv| < , ∀v ∈ V, where zv ∈ R are final output values generated by Algorithm 1 and cv are node clustering coefficients.

5 理论分析

在本节中，我们将探讨 GraphSAGE 的表达能力，以便深入了解 GraphSAGE 如何学习图结构，即使它本质上是基于特征的。作为案例研究，我们考虑 GraphSAGE 是否可以学习预测节点的聚类系数，即在节点的 1 跳邻域内闭合的三角形的比例。聚类系数是一种流行的衡量节点本地邻域聚类程度的度量，并且它是许多更复杂的结构图案的基石。我们可以证明算法 1 能够将聚类系数近似为任意精度：

定理 1。设x_v $\in U, \forall v$ $\in \mathcal{V}$表示图 $G = (\mathcal{V}, \mathcal{E})$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 $\in$ $\mathbb{R}^+$，这样$|| x_v - x_{v'} ||_2 >$ C对所有节点对都成立。然后我们有$\forall \epsilon > 0$，且存在一个参数

设定Θ^*， 对于算法 1，使得在 K = 4 次迭代后有：

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

where zv ∈ R are final output values generated by Algorithm 1 and cv are node clustering coefficients.

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

Theorem 1 states that for any graph there exists a parameter setting for Algorithm 1 such that it can approximate clustering coefficients in that graph to an arbitrary precision, if the features for every node are distinct (and if the model is sufficiently high-dimensional). The full proof of Theorem 1 is in the Appendix. Note that as a corollary of Theorem 1, GraphSAGE can learn about local graph structure, even when the node feature inputs are sampled from an absolutely continuous random distribution (see the Appendix for details). The basic idea behind the proof is that if each node has a unique feature representation, then we can learn to map nodes to indicator vectors and identify node neighborhoods. The proof of Theorem 1 relies on some properties of the pooling aggregator, which also provides insight into why GraphSAGE-pool outperforms the GCN and mean-based aggregators.

定理 1 指出，对于任何图，如果每个节点的特征是不同（同时加上条件如果模型足够高维），都存在算法 1 的参数设置，使得它可以近似于该图中的聚类系数达到任意精度。 定理 1 的充分证明在附录中。请注意，作为定理 1 的推论，GraphSAGE 可以了解局部图结构，即使从绝对连续随机分布样本中采样节点特征输入（详见附录）。 证明背后的基本思想是，如果每个节点都有一个独特的特征表示，然后我们可以学习将节点映射到指标向量并识别节点邻域。 定理 1 的证明依赖于池聚合器的某些属性，这些属性还提供了有关 GraphSAGE 池为何胜过 GCN 和基于均值的聚合器的相关了解。

6 Conclusion

We introduced a novel approach that allows embeddings to be efficiently generated for unseen nodes. GraphSAGE consistently outperforms state-of-the-art baselines, effectively trades off performance and runtime by sampling node neighborhoods, and our theoretical analysis provides insight into how our approach can learn about local graph structures. A number of extensions and potential improvements are possible, such as extending GraphSAGE to incorporate directed or multi-modal graphs. A particularly interesting direction for future work is exploring non-uniform neighborhood sampling functions, and perhaps even learning these functions as part of the GraphSAGE optimization.

6 结论

我们引入了一种新颖的方法，可以有效地为隐藏节点生成嵌入。GraphSAGE 始终优于最新基准，通过对节点邻域进行采样，有效地权衡了性能和运行时间。我们的理论分析了我们的方法是如何学习局部图结构的。 许多扩展和有效的改进是可能的，例如扩展 GraphSAGE 以合并定向或多模态图。 未来工作的一个特别有趣的方向是探索非均匀邻域的采样函数，甚至可能学习这些函数作为 GraphSAGE 优化的一部分。