Inductive Representation Learning on Large Graphs

NIPS 2017

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

GraphSAGE (sample and aggregate) 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.

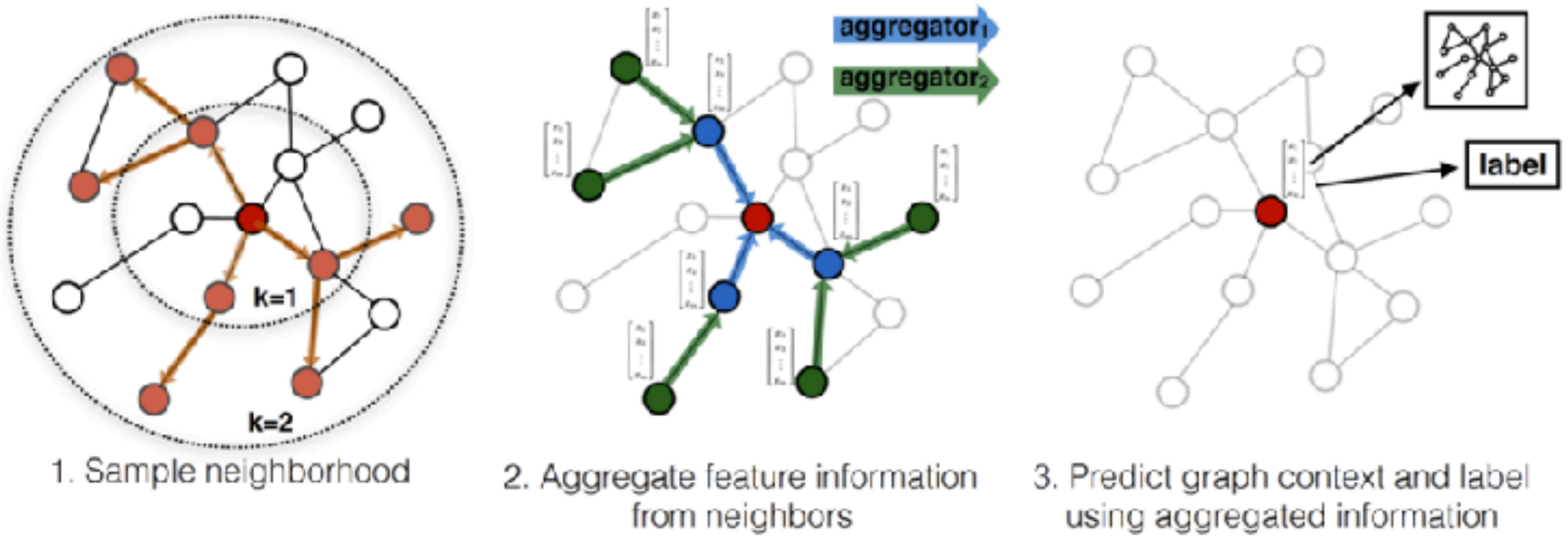


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

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 $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

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

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

Relation to the Weisfeiler-Lehman Isomorphism Test

1.2.1 图　核　图核是一种衡量两个图结构相似程度的度量技术。近年来，许多研究者提出了多种图核构建的方法，其中包括基于步数[15]、基于路径[16]和基于子树的[17]等。图核技术被广泛应用于图像分类[18]和蛋白质功能预测[19]等领域中。

本文利用了文献[17]提出的图核构建方法，其基本思想是利用 Weisfeiler-Lehman 同构测试。对于给定的两个图，Weisfeiler-Lehman 测试的基本步骤为：在每一步迭代步骤中，每个节点的标号根据它之前的标号及其相邻节点的标号进行迭代更新。然后，将更新的标号压缩成新的标号。这个过程一直迭代到两个图的顶点标号变为一样，或者达到设定的迭代次数。

定义 3.2（Weisfeiler-Lehman 图核）图 G 和图 G'之间的　WL 图核定义为

$$k_{WL}^{(h)}(G,G') = \left|\{(s_i(v),s_i(v')) \mid f(s_i(v)) = f(s_i(v')), i\{0,\ldots,h\}, v \in V, v \in V'\}\right.$$

其中 Si（v）为节点 v 在第 i 次迭代中的多分类标签集，f 是一个映射标签压缩函数，对于所有的 $i \neq j$，集合 $\{f(s_i(v)) \mid v \in V \cup V'\}$ 和集合 $\{f(s_j(v)) \mid v \in V \cup V'\}$ 是不相交的。S_0（v）是在标签图 v 和非标签图中的初始标签并且 $f(s_0(v)) = s_0(v)$。

Relation to the Weisfeiler-Lehman Isomorphism Test

(i) set K = |V|,

(ii) set the weight matrices as the identity,

(iii) use an appropriate hash function as an aggregator (with no non-linearity)

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.

Aggregator Architectures

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:

(1) symmetric (i.e., invariant to permutations of its inputs)
(2) trainable and maintaining high representational capacity

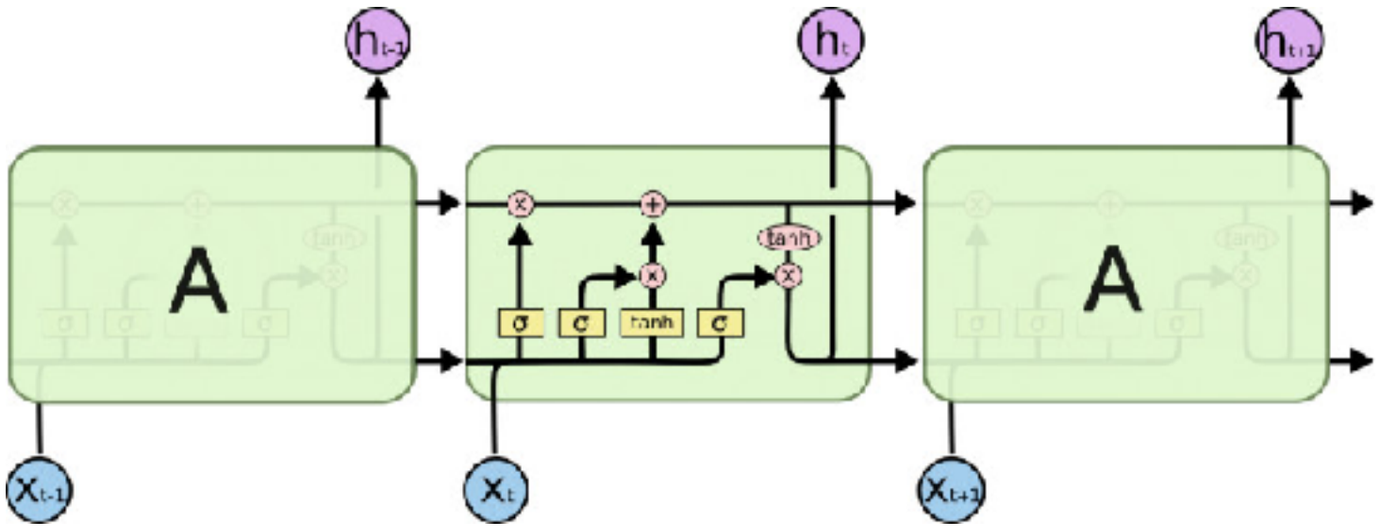
The symmetry property of the aggregation function ensures that our neural network model can be trained and applied to arbitrarily ordered node neighborhood feature sets. We examined three candidate aggregator functions:

Mean aggregator

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

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).

LSTM aggregator



We also examined a more complex aggregator based on an LSTM architecture. 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.

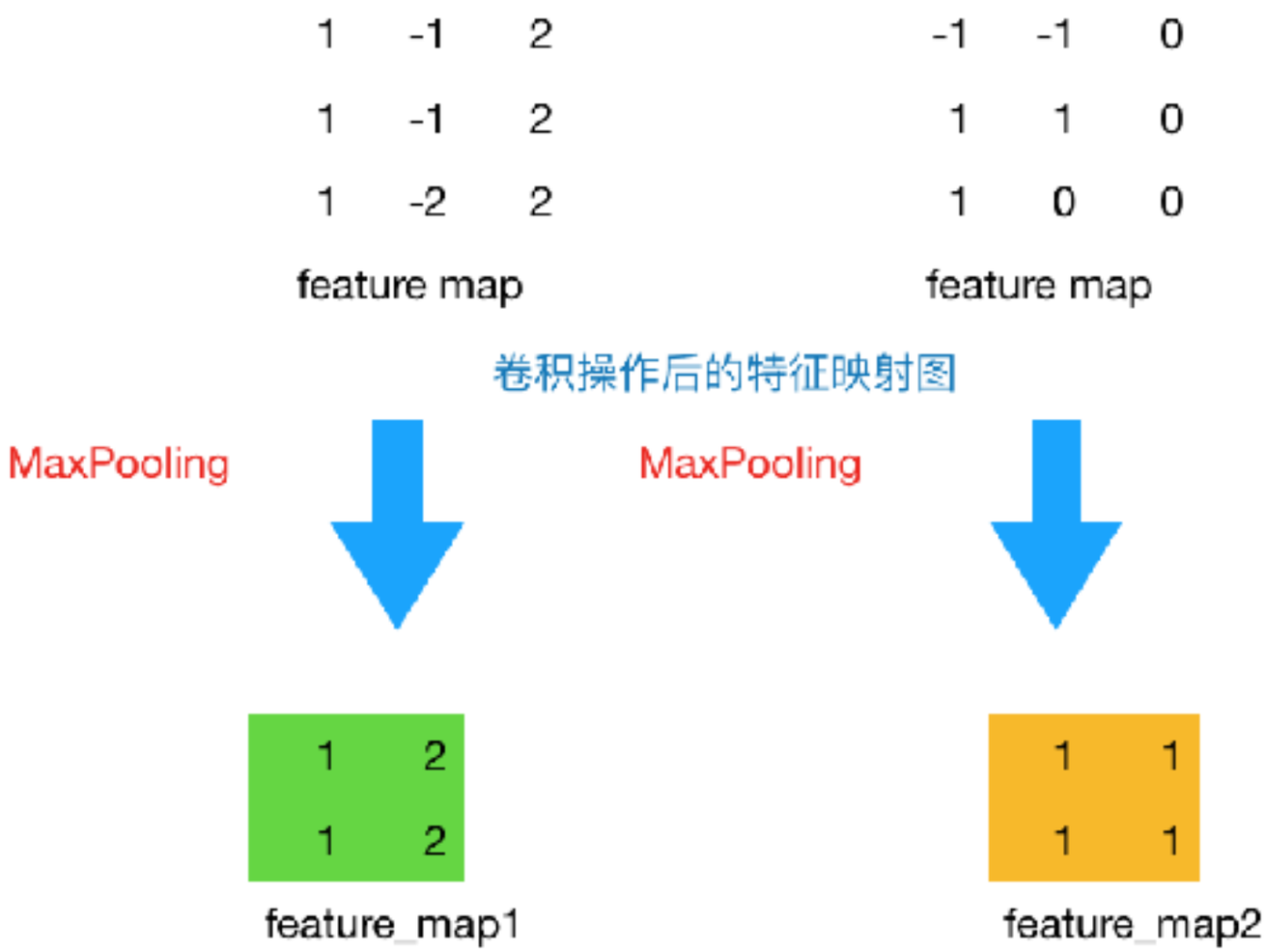
Pooling aggregator

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:

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

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.

Pooling aggregator



Learning the parameters of GraphSAGE

The graph-based loss function encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct:

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

where v is a node that co-occurs near u on fixed-length random walk, σ is the sigmoid function, P_n is a negative sampling distribution, and Q defines the number of negative samples. Importantly, unlike previous embedding approaches, the representations z_u 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).

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,

(iii) classifying protein functions across various biological protein-protein interaction (PPI) graphs

Inductive learning on evolving graphs: Citation and Reddit data

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).

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).

Inductive learning on evolving graphs: Citation and Reddit data

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%

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.

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 . 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. 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).

Runtime and parameter sensitivity

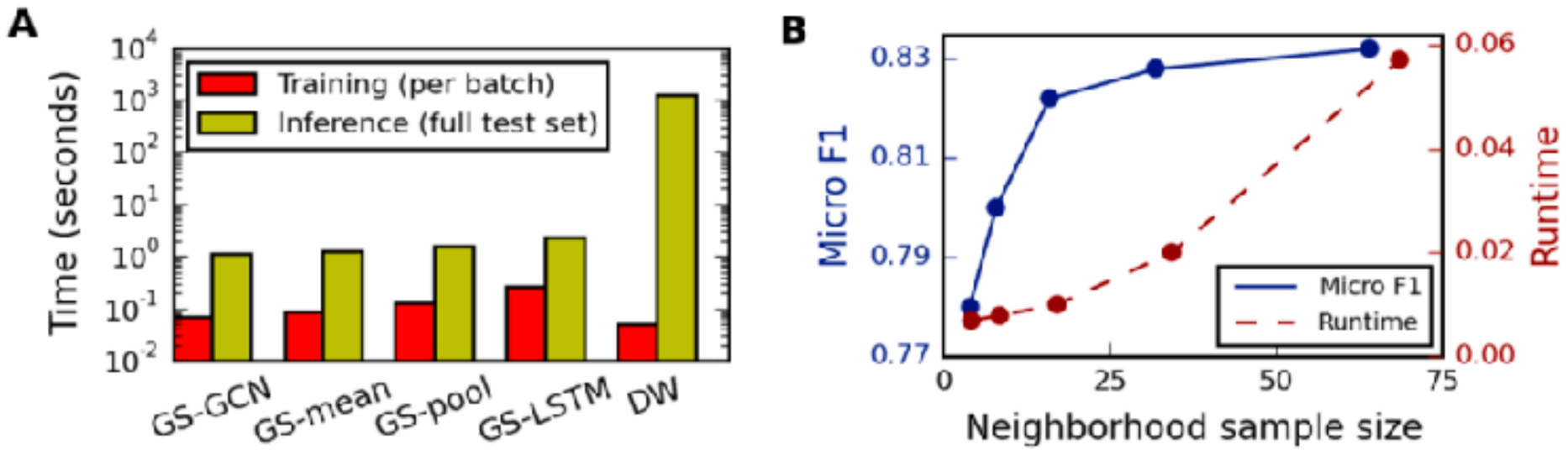


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each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

Theoretical analysis

Theorem 1. *Let $\mathbf{x}_v \in U, \forall v \in \mathcal{V}$ denote the feature inputs for Algorithm 1 on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where U is any compact subset of $\mathbb{R}^d$. Suppose that there exists a fixed positive constant $C \in \mathbb{R}^+$ such that $\|\mathbf{x}_v - \mathbf{x}_{v'}\|_2 > C$ for all pairs of nodes. Then we have that $\forall \epsilon > 0$ there exists a parameter setting $\mathbf{\Theta}^*$ for Algorithm 1 such that after $K = 4$ iterations*

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

where $z_v \in \mathbb{R}$ are final output values generated by Algorithm 1 and c_v are node clustering coefficients.

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.