【图表示学习算法系列】一：GraphSAGE-归纳式图表示学习算法

原创 TwT 小韬学算法 2020-03-31

> 论文题目：Inductive Representation Learning on Large Graphs
>
> 论文链接：https://arxiv.org/pdf/1706.02216.pdf
>
> 论文发表在NIPS2017

0.摘要

在一个大型图中，对节点的低纬度嵌入表示是十分有用的，之前的算法需要图中所有的节点信息且对未知数据的泛化能力一般，这种方法被称为 `transductive` ；而本文的方法被称为 `inductive` ，本算法名为 `GraphSAGE` ，对未知数据有较好的泛化能力。简单的说就是通过一个节点的邻居信息进行特征的采样和聚合，然后学习到一个函数从而为当前节点生成一个嵌入向量。实验证明，该算法在可变信息图（就是说图不是一成不变的）上对未知数据的分类效果是很好的。

1.前言

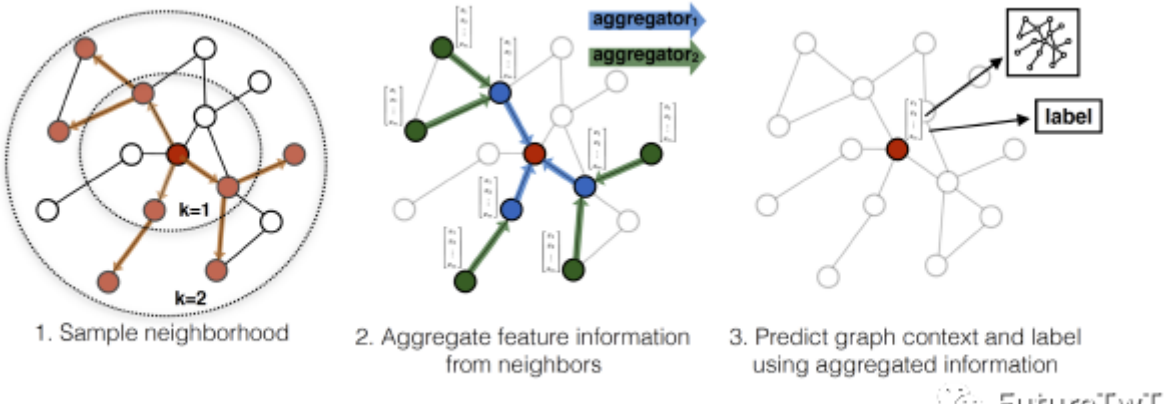

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

基础的方法是利用降维技术对节点邻居的高维信息进行提取得到一个密集的嵌入向量，从而实现对当前节点的表达。利用该算法可以作为上游处理过程（类似于预处理）得到节点的图嵌入并为后面的机器学习算法和相关任务提供信息。但是之前的方法都是对单一固定图进行处理，实际要求需要一个快速的节点嵌入学习方法，并且图是可变的，未知数据会源源不断地出现，

inductive 方法对相似形式特征的图具有很好的泛化能力。 inductive 方法对比 transdutive 方法，其节点嵌入学习更加困难，需要更好的识别全图的结构属性以及节点的邻域特性。现有的算法主要基于矩阵分解（MF），对于未知数据的泛化能力一般。目前有利用卷积策略进行嵌入学习的方法如 GCN ，本项工作将会把 GCN 拓展到归纳式的无监督学习场景中，并使用一种可训练的聚合函数替代简单的卷积过程。

本文工作

提出GraphSAGE，不同于基于矩阵分解的嵌入学习方法，该算法学习节点特征是为了提高对未知数据的泛化能力。学习每个节点的邻域的拓扑结构（连接结构）以及节点特征在邻域内的分布特性。并且该算法可以用于无节点特征的图中。算法的核心在于训练一组用于聚合邻域信息的聚合函数 AGGREGATOR ，每一个聚合函数聚合的邻域范围（跳数）是不同的，并且通过训练好一组聚合函数，就可以对未知节点进行嵌入生成。算法中设计了无监督的损失函数，并且也可以进行监督式训练。本文在三个节点分类数据集上测试了算法性能以及对未知数据的嵌入生成能力。

2.方法概述

方法的关键在于如何对一个节点的局部邻居信息进行汇聚。文章分别介绍了前向推理算法和反向传播算法。

2.1 嵌入生成

算法伪代码如下：

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
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

首先给出其中的输入（1~7）和输出信息（8）：

（1）图 G(V,E) 表示顶点集合和边集合；

（2）每一个节点的特征信息（一个向量） {x}

（3）迭代次数（搜索深度） K

（4）权重矩阵 W ，用于每一次迭代之间的信息传递

（5）非线性函数σ

（6）可微的聚合函数 AGGREGATE

（7）邻居函数 N ，得到每一个节点的所有邻居节点

（8）每一个节点的特征表达 {z}

循环次数k表示迭代次数，每一次迭代，节点都会聚合邻居的信息，随着迭代次数增加，节点会获得越来越多（可以理解为更远的邻居）的节点信息。具体流程如下：

第一行表示初始化， h 表示节点的嵌入表达，下标为节点，上标为当前迭代次数，一开始迭代次数为0，节点的嵌入表达初始化为其自身的特征信息；当进行第k次循环时，需要对每一个顶点集V中的节点进行更新，更新分为两步，第一步获取邻居信息，利用函数 AGGREGATE 将当前节点 v 的所有邻居节点 u∈N(v) 进行聚合（注意的是利用的节点信息均为上一次迭代的信息，所以在实际计算过程中，节点信息不能只用一个向量集合表示，会出现覆盖问题），第二步将邻居信息向量与当前节点的自身向量进行拼接，再利用全连接层和非线性激活函数处理得到新的嵌入表达。最后得到输出 z 。因此，算法关键在于聚合函数的选择。为了降低计算复杂度，同时因为Graph为非欧几里得结构，邻接数量不是固定的，所以对邻居函数 N(v) 进行重新定义，每次迭代都会选取固定的邻居数量，实验表明最佳参数为迭代次数K=2，两次迭代所选邻居数S1*S2<=5。

2.2 参数学习

首先明确的是该算法为无监督学习算法，利用SGD策略对相关参数进行调优，希望的是邻接近的点其嵌入表达更具有相似性，反之需要有很大的区分度。

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

如果该嵌入表达有固定的应用场景，可以适当替换损失函数，比如做节点分类，可以通过节点分类的损失值来更新这里的 embedding 结果。

2.3 聚合函数选择

1. Mean aggregator将邻居节点向量按位取均值，即

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

2. LSTM aggregator LSTM表达能力更强，但是对顺序有要求，所以实验中采用邻居节点随机排序的方式应用LSTM聚合器。

3. Pooling aggregator 每一个邻居的向量单独进行全连接层计算并对所有的处理后的向量进行一个按位的最大池化操作：

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

最大池化操作可以提取到每一个邻居节点不同层面的信息。

3.实验

实验主要分为三个部分：

（1）利用Web of Science引文数据集对学术论文进行分类；

（2）将贴子分类到不同的社区；

（3）利用生物蛋白之间的相互作用PPI对蛋白质功能进行分类。

实验过程中，所有预测节点的信息在训练过程中是不会出现的。实验设置如下，设置四个baseline，分别为

（1）随机分类器

（2）逻辑回归分类器

（3）DeepWalk算法

（4）原始特征+DeepWalk嵌入表达结果

同时对比四种不同的GraphSAGE算法（利用不同的聚合函数），对于GCN版本，无监督变体采用上文的损失函数，有监督变体利用分类交叉熵损失。非线性函数均采用 ReLU函数 ， K=2 ， S1=25 ， S2=10 。实验结果如下：

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%	49%

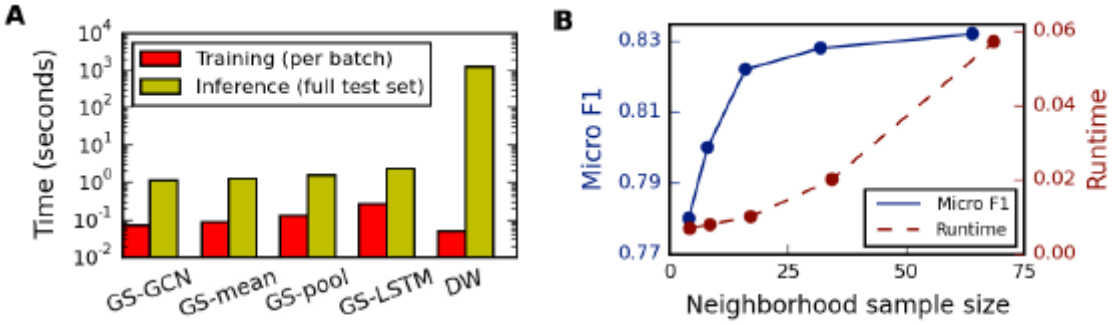

Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each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

【注】如果有理解存在偏差或者错误的地方，欢迎读者们给出指导意见！