《Inductive Representation Learning on Large Graphs》

《大型图的归纳表示学习》

摘要：

1. **本文的背景：**自 GCN 相关的论文提出后，图神经网络开启了新一轮的研究热潮。然而，当时的大多数训练算法并不能扩展到大型图中或者设计用于全图的分类，这使得当时的技术很难应用到实际场景之中，或存在很大的局限。
2. **本文的贡献：**有效解决了大规模动态图结构下的节点 embedding 的生成问题，并**在图神经网络领域上提出了通用的归纳式学习框架**，即可以利用已有的节点信息产生新加入节点的 embedding。
3. **主要创新点：**利用节点特征信息和结构信息，通过顶点的局部邻居**采样操作**(sampling)，以及聚合邻居节点特征的**聚合操作**(aggregation)，来获取（训练）顶点的 Graph Embedding。并且，上述这种方式可以应用到归纳式学习中。
4. **实验结果：**相比一些 baseline 算法，GraphSAGE 在性能上取得了更进一步的提升。此外，研究还比较了 GraphSAGE 的不同聚合函数的效果，其中 LSTM 聚合函数以及 Pooling 聚合函数取得了更优异的成绩。

直推式学习和归纳式学习：

——参考自 https://zhuanlan.zhihu.com/p/332289397

直推式学习（transductive learning）是指：模型在学习阶段除了能够看到训练数据（样本+标签）外，还能够接触到测试数据（样本），以期能够利用测试数据的内在结构，把测试数据作为一个整体来预测其标签，而不是孤立地、 one-by-one 地预测每一个测试样本。

归纳式学习（Inductive learning）是指：从特定任务到一般任务的学习，模型在学习阶段仅基于训练集，而后将其应用于测试集的预测任务中，训练集与测试集之间是相斥的，即测试集中的任何信息都是没有在训练集中出现过的。因此，模型本身具备一定的通用性和泛化能力。【实际上，传统的 supervised learning 都可以理解为是 Inductive learning 的范畴】

二者的区别：

1）模型训练：Transductive learning 在训练过程中已经用到测试集数据（不带标

签）中的信息，而 Inductive learning 仅仅只用到训练集中数据的信息。

2）模型预测：Transductive learning 只能预测在其训练过程中所用到的样本（特殊—> 特殊），而 Inductive learning，只要样本特征属于同样的欧拉空间，即可进行预测（特殊—>一般）

3）模型复用性：当有新样本出现时，Transductive learning 需要重新进行训练；而 Inductive Leaning 不需要。

4）模型计算量：显而易见，Transductive leaning 是需要更大的计算量的，即使其有时候确实能够取得相比 Inductive learning 更好的效果。

在图神经网络中，早前提出的 DeepWalk、node2vec、LINE 以及 GCN 都是直推式学习，因此都有很大的局限性。而本篇论文所提出的 **GraphSAGE 是一种归纳式学习框架，所以它有着更广泛的应用前景。**【通过训练聚合节点邻居的函数（卷积层），使 GCN 扩展成归纳式学习任务，对未知节点起到泛化作用】

GraphSAGE 框架：

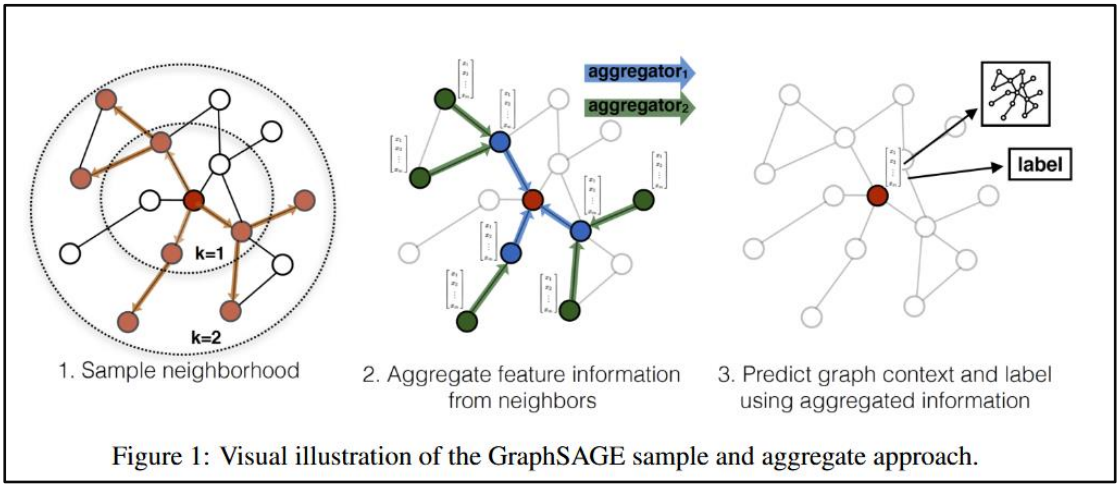


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

如图所示，这是 GraphSAGE 生成目标节点（红色）embededing 并提供给下游任务预测的过程：

1）对邻居节点进行随机采样，降低计算复杂度。图中一跳（$k=1$）的邻居采样数为 3，二跳（$k=2$）的邻居采样数为 5）。

2）生成目标节点 emebedding。图中先聚合了 2 跳的邻居特征，生成一跳的邻居 embedding，然后再聚合一跳的邻居 embedding，生成目标节点的 embedding。

3）将训练得到的 embedding 作为全连接层的输入，来预测目标节点的标签。

GraphSAGE 框架实现：

1）嵌入生成（即前向传播）算法

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

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

如图所示，这是嵌入生成算法部分的伪代码，其中$\{x_v, \forall v \in V\}$是算法的输入特征；K是深度；W^k是各深度的权重矩阵；σ是激活函数；$AGGREGATE_k$代表各深度的聚合器；$\mathcal{N}$是邻居采样函数。

而算法的整体流程是：首先将输入特征传递给h_v^0，代表每个目标节点的初始特征。然后，由浅到深，依次聚合每个目标节点的采样邻居的特征h_u^{k-1}，$\forall u \in \mathcal{N}(v)$，得到邻居聚合特征$h_{\mathcal{N}(v)}^k$。接着，将自身前一层的深度特征$h_v^{k-1}$和邻居聚合特征$h_{\mathcal{N}(v)}^k$拼接，再通过特征映射和激活函数得到新的深度特征$h_v^k$。最后，对每个得到的目标特征做归一化处理，防止出现梯度消失或爆炸。而第K层的深度特征h_v^K就是我们所需的嵌入向量z_v。

此外，作者提到在该算法背后的直觉是：在每次迭代或搜索深度时，节点都会从其本地邻居那里聚合信息。因此，随着整个过程的迭代，节点会从图的更远部分逐渐获得越来越多的信息。

2）邻居采样算法

首先，定义需要的邻居数量S，然后采用**有放回**的重采样/负采样方法达到指定的邻居数量S。即 GraphSage 采用的是**定长采样**。

原因：采用定长采样可以将节点和邻居拼成 Tensor 送到 GPU 中进行批训练，并且还可以方便研究者估计算法的时间和空间复杂度。

3）聚合函数$AGGREGATE$

由于在图中顶点的邻居是自然无序的，而研究者希望构造出的聚合函数是对称的（即改变输入的顺序，函数的输出结果不变），同时具有较高的表达能力。因此，他们构造了四种聚合函数来进行比较，分别是：

a. 平均聚合

先对邻居 embedding 中的每个维度取平均，然后与目标节点 embedding 拼接后进行非线性转换。

$$h_{\mathcal{N}(v)}^{k} = mean(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{\mathcal{N}(v)}^{k}))$$

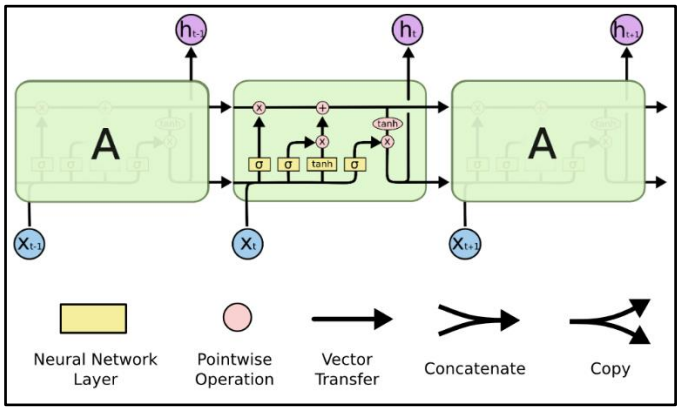
b. 归纳式聚合(GCN)

直接对目标节点和所有邻居 emebdding 中每个维度取平均（替换伪代码中第 5、6 行），后再进行非线性转换。

$$h_v^k = \sigma(W^k \cdot mean(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

c. LSTM 聚合

由于 LSTM 函数不符合“排序不变量”的性质，因此需要先对采样邻居进行随机排序，然后将随机邻居序列 embedding（如下所示）作为 LSTM 输入：



d. Pooling 聚合

先对每个邻居节点上一层 embedding 进行非线性转换（等价单个全连接层，每一维度代表在某方面的表示），再按维度应用 Max/Mean Pooling，捕获邻居集上在某方面的突出的/综合的表现，以此来表示目标节点 embedding。

$$h_{\mathcal{N}(v)}^{k} = max(\{\sigma(W_{pool} h_{ui}^{k-1} + b), \forall ui \in \mathcal{N}(v)\})$$

$$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{\mathcal{N}(v)}^{k}))$$

GraphSAGE 小批量前向传播算法：

Algorithm 2: GraphSAGE minibatch forward propagation algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
input features $\{\mathbf{x}_v, \forall v \in \mathcal{B}\}$;
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 sampling functions, $\mathcal{N}_k : v \to 2^{\mathcal{V}}, \forall k \in \{1, ..., K\}$

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

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

如图所示，该算法在原算法的基础上进行了一定的改进。（说明略）

无监督和有监督损失函数的设定：

损失函数根据具体应用情况，可以使用基于图的**无监督损失**和**有监督损失**。

1）基于图的无监督损失

$$J_{\mathcal{G}}(z_u) = -\log\left(\sigma(z_u^T z_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(z_u^T z_v)\right)$$

其中，z_u为节点u通过 GraphSAGE 生成的 embedding；节点v是节点u随机游走访达的“邻居”；$v_n \sim P_n(v)$ 表示负采样；Q为采样样本数；而 embedding 之间相似度是通过向量点积计算得到的。

解释：希望节点u与“邻居”v的 embedding 相似（对应公式的第一项），而与“没有交集”的节点v_n不相似（对应公式的第二项)。

2）有监督损失

利用无监督损失函数的设定来学习节点 embedding 可以供多个下游任务使用，若仅使用在特定某个任务上，则可以替代上述损失函数符合特定任务目标即可，如交叉熵函数。

模型参数的学习：

通过前向传播得到节点u的 embedding，然后梯度下降（使用 Adam 优化器）进行反向传播优化参数W^k和聚合函数内的参数，例如W_{pool}。

实验及结果：

1）实验目的

a. 比较 GraphSAGE 相比 baseline 算法的提升效果；

b. 比较 GraphSAGE 的不同聚合函数。

2）数据集和相关任务

a. Citation 论文引用网络（节点分类）

b. Reddit web 论坛（节点分类）

c. PPI 蛋白质网络（graph 分类）

3）比较方法

1)随机分类器；2)手工特征（非图特征）；3)deepwalk（图拓扑特征）；4)deepwalk+手工特征；5)GraphSAGE 的四个变种，并无监督生成 embedding 输入给 LR 和端到端的有监督学习。

4）GraphSAGE 的设置

• $K = 2$，即仅聚合两跳内的邻居特征。

• $S1 = 25, S2 = 10$，即对第一跳的邻居抽样 25 个，对第二跳邻居的抽样 10 个。

• 采用 RELU 激活函数。

• 采用 Adam 优化器。

• 对每个节点进行步长为 5 的 50 次随机游走。

• 采用负采样，按平滑 degree 进行，对每个节点采 20 个负样本。

• 保证公平性，即所有版本都采用相同的 minibatch 迭代器、损失函数以及邻居抽样器。

5）Baseline 实验结果

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%

如图所示，我们可以发现：1）GraphSAGE 相比 baseline 的效果有大幅度的提升；2）GraphSAGE 的有监督版本比无监督版本的效果更好；3）聚合函数 LSTM 和 Pooling 的效果较好。4）尽管 LSTM 是为有序数据而不是无序集设计的，可是基于 LSTM 的聚合函数仍然显示了强大的性能。

6）运行时间和参数敏感性研究

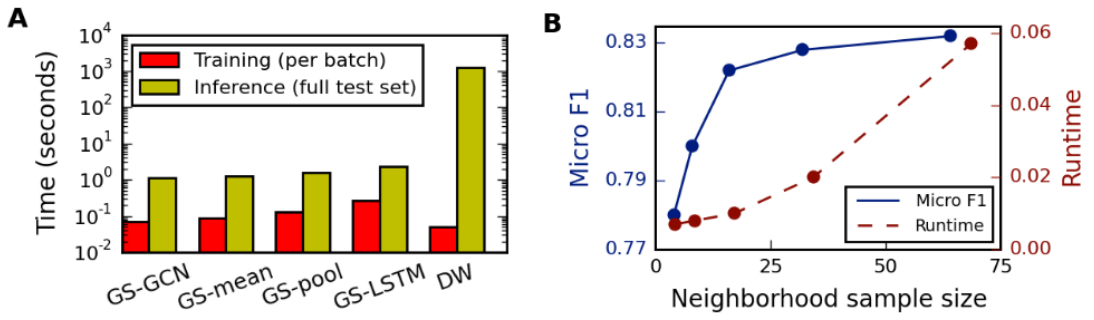


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each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

如图所示，我们可以发现：1）在计算时间上，GraphSAGE 中的 LSTM 训练速度最慢，但相比 DeepWalk，GraphSAGE 在预测时间上减少了 100—500 倍（因为对于未知节点，DeepWalk 要重新进行随机游走以及通过 SGD 学习 embedding）。2）在邻居抽样数量上，随着邻居抽样数量的递增，边际收益（F1）呈递减趋势，且计算时间也在变大。3）算法需要平衡 F1 得分和计算时间，例如在 GraphSAGE 上$K = 2$相比$K = 1$有$10 - 15\%$的提升；但如果将K设置超过 2，边际效果上只有$0 - 5\%$的提升，但是计算时间却扩大了$10 - 100$倍。

总结：

本文提出了一种全新的基于图的归纳式学习方法——GraphSAGE，该方法允许为看不见的节点有效地生成嵌入向量。并且，实验结果表明 GraphSAGE 的性能始终优于最先进的 Baseline。此外，基于 GraphSAGE 的许多扩展和潜在改进也都是可能的，例如我们可以扩展 GraphSAGE 以合并有向或多模态图。

对于本文的感悟：

GraphSAGE 的提出是在 GCN 的基础上做出了大胆的假设与改进，是对 GCN 的一种扩展。而如今看来，GraphSAGE 的贡献是不可估量的，它不仅催生了第一个基于 GCN 的工业级推荐系统 PinSAGE，还给众多在图深度学习领域的研究者们打下了一针强心剂。