GraphSAGE 图神经网络

原创 NLP与人工智能 NLP与人工智能 2020-07-13

收录于话题
#GNN 28 #图神经网络 139

> GraphSAGE
>
> GraphSAGE 是 2017 年提出的一种图神经网络算法，解决了 GCN 网络的局限性: GCN 训练时需要用到整个图的邻接矩阵，依赖于具体的图结构，一般只能用在直推式学习 Transductive Learning。GraphSAGE 使用多层聚合函数，每一层聚合函数会将节点及其邻居的信息聚合在一起得到下一层的特征向量，GraphSAGE 采用了节点的邻域信息，不依赖于全局的图结构。

前言

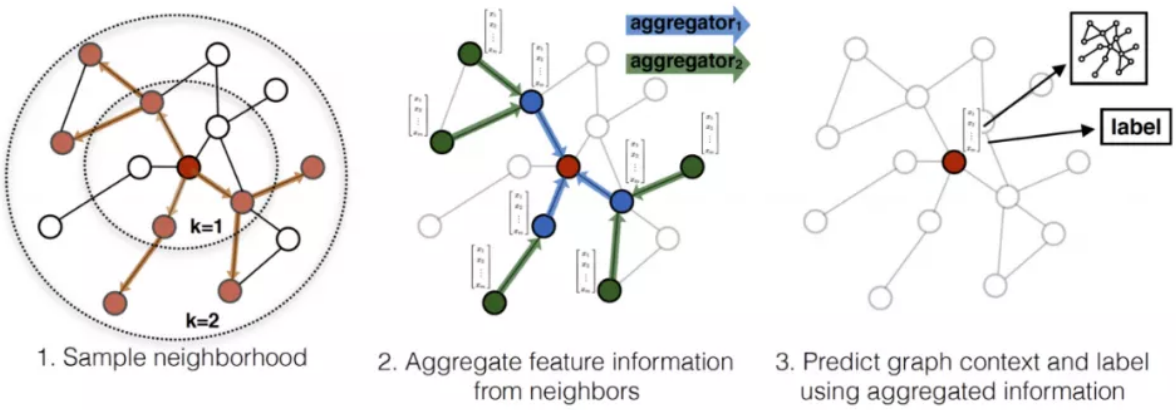
在之前的文章《图神经网络 GNN 之图卷积网络 (GCN)》和《GAT 图注意力网络 Graph Attention Network》分别介绍了 GCN 和 GAT，不熟悉的童鞋可以参考一下。

图神经网络的任务一般有 Transductive (直推式) 和 Inductive (归纳式)。Transductive 通常指要预测的节点在训练时已经出现过， 例如有一个作者关系网络，知道部分作者的类别，用整个网络训练 GCN，最后预测未知类别的作者。Inductive 指要预测的节点在训练时没有出现，例如用今天的图结构训练，预测明天的图。

GCN 利用了图的整个邻接矩阵和图卷积操作融合相邻节点的信息，因此一般用于 Transductive 任务而不能用于处理 Inductive 任务。因此 2017 年 GraphSAGE 算法被提出，用于解决 GCN 的问题，《Inductive Representation Learning on Large Graphs》。

GraphSAGE

GraphSAGE 包含采样和聚合 (Sample and aggregate)，首先使用节点之间连接信息，对邻居进行采样，然后通过多层聚合函数不断地将相邻节点的信息融合在一起。用融合后的信息预测节点标签。下图展示了 GraphSAGE 的聚合过程，采用了两层聚合层。



上图中的包括两层聚合，对应的聚合函数为 aggregator1 和 aggregator2。通过 k 层聚合之后，可以得到节点最终的表示向量，GraphSAGE 的伪代码如下：

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

伪代码中的 h0 表示节点 v 的初始特征向量，包含 K 层聚合操作。在第 k 次聚合生成 v 节点特征向量时，会采用聚合函数把 v 节点的邻居信息融合在一起。这一操作也可改成 minibatch 的，伪代码如下：

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, $\mathcal{N}_k : v \rightarrow 2^{\mathcal{V}}, \forall k \in \{1, ..., K\}$

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

$\mathcal{B}^K \leftarrow \mathcal{B}$;
for $k = K...1$ **do**
  $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^k$;
  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
$\mathbf{z}_u \leftarrow \mathbf{h}_u^K, \forall u \in \mathcal{B}$

上面的伪代码中，**B** = **B**K 为要生成向量的节点集合， **B**k-1 是深度为 1 的邻域，**B**0 为深度为 K 的邻域，**B**0 包含的节点最多。Nk(v) 表示 v 节点在第 k 次聚合时的邻域，节点在每一层的邻域数量都不同，通过采样得到。

GraphSAGE 聚合函数

GraphSAGE 提供了四种聚合节点的函数：

Mean aggregator: 对节点 v 进行聚合时，对节点 v 和邻域的特征向量求均值。

$$h_v^k \leftarrow \text{Mean}(\{h_u^{k-1}, \forall u \in N(v)\})$$

GCN aggregator: 采用了类似 GCN 卷积的方式进行聚合，公式和 Mean aggregator 类似：

$$h_v^k \leftarrow \sigma\left(W \cdot \text{Mean}(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in N(v)\})\right)$$

LSTM aggregator: 作者任务 LSTM 有比较好的抽取特征能力，因此也使用了 LSTM 进行聚合，但是因为节点之间没有明显的顺序关系，因此会打乱之后放入 LSTM。

Pooling aggregator: 先把所有邻居节点的特征向量传入一个全连接层，然后使用 max-pooling 聚合。

$$h_v^k \leftarrow \max\left(\left\{\sigma\left(W_{pool}h_u^{k-1}+b\right), \forall u \in N(v)\right\}\right)$$

GraphSAGE 训练

GraphSAGE 可以采用无监督训练或者有监督训练。**无监督训练**采用负采样算法，公式如下：

$$J_G(z_u) = -\log\left(\sigma(z_u^T z_v)\right) - \mathrm{Q}\cdot \mathrm{E}_{v_n\sim P_n(v)}\log\left(\sigma\left(-z_u^T z_{v_n}\right)\right)$$

公式中的 zu 是经过 GraphSAGE 聚合之后的特征向量，节点 v 是节点 u 邻域内的节点，而 Q 表示负采样次数。

对于**有监督训练**可以使用任务相关的目标函数，例如节点分类时采用交叉熵损失函数。

实验结果

作者对比了不同算法的性能，也对比了 GraphSAGE 四种聚合方式的效果，如下表所示。

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%

下图 A 是训练和测试时间的实验结果，B 是采样邻域大小对性能影响的结果。

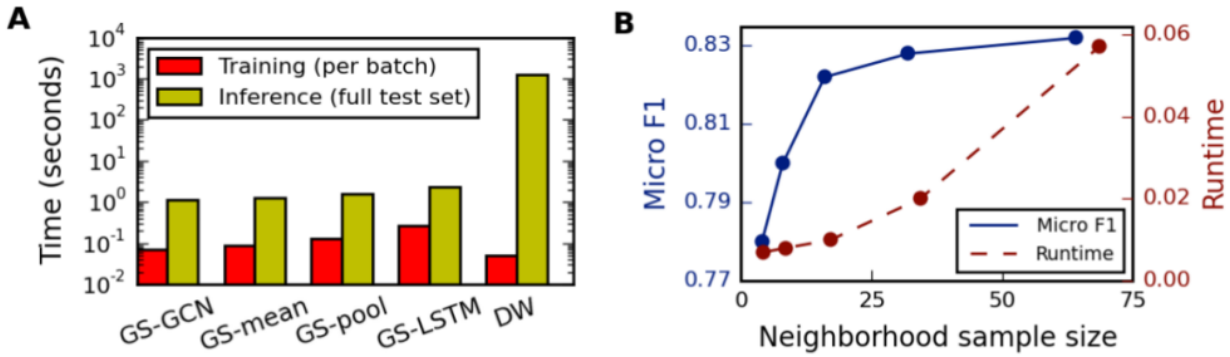


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each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

参考文献

Inductive Representation Learning on Large Graphs