图神经网络的介绍（基础，DeepWalk和GraphSage）

ronghuaiyang AI蜗牛车 2019-08-04

作者：Steeve Huang
编译：ronghuaiyang

导读

给大家介绍目前非常热门的图神经网络，包括基础和两个常用算法，DeepWalk和GraphSage。

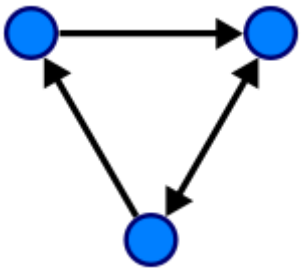
近年来，图神经网络(GNN)在社交网络、知识图谱、推荐系统甚至生命科学等各个领域得到了越来越广泛的应用。GNN具有对图中节点间依赖关系建模的强大功能，使得图分析相关研究领域取得了突破。本文会介绍图神经网络的基本原理，以及两种更高级的算法，DeepWalk和GraphSage。

图

在讨论GNN之前，让我们先了解什么是*Graph*。在计算机科学中，图是由顶点和边两部分组成的数据结构。一个图*G*可以由它所包含的顶点*V*和边*E*的集合很好地描述。

$$G = (V, E)$$

根据顶点之间是否存在方向依赖关系，边可以是有向的，也可以是无向的。



有向图

这些顶点通常称为节点。在本文中，这两个术语是可以互换的。

图神经网络

图神经网络是一种直接作用于图结构上的神经网络。GNN的一个典型应用是节点分类。本质上，图中的每个节点都与一个标签相关联，我们希望在没有ground-truth的情况下预测节点的标签。本节将说明本文中描述的算法。第一个提出的GNN常常被称为原始GNN。

在节点分类问题设置中，每个节点*v*的特征是*x_v*，并与一个ground-truth标签*t_v*关联。给定一个部分标记的图*G*，目标是利用这些标记的节点来预测未标记的标签。它学习用一个*d*维向量(状态)*h_v*表示每个节点，其中包含其邻域的信息。具体地说，

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

其中*x_co[v]*表示与*v*相连的边的特征，*h_ne[v]*表示与*v*相邻节点的嵌入，*x_ne[v]*表示与*v*相邻节点的特征。函数*f*是将这些输入投射到d维空间的转换函数。因为我们正在为h_v寻找一个惟一的解，所以我们可以应用Banach定点定理并将上面的等式重写为迭代更新过程。这种操作通常称为**消息传递**或**邻居聚合**。

$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X})$$

H和X分别表示所有H和X的级联。

通过将状态*h_v*和特征*x_v*传递给输出函数g来计算GNN的输出。

$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v)$$

这里的f和g都可以解释为前馈全连接神经网络。L1损失可以直接表述为:

$$loss = \sum_{i=1}^{p} (\mathbf{t}_i - \mathbf{o}_i)$$

可以通过梯度下降来优化。

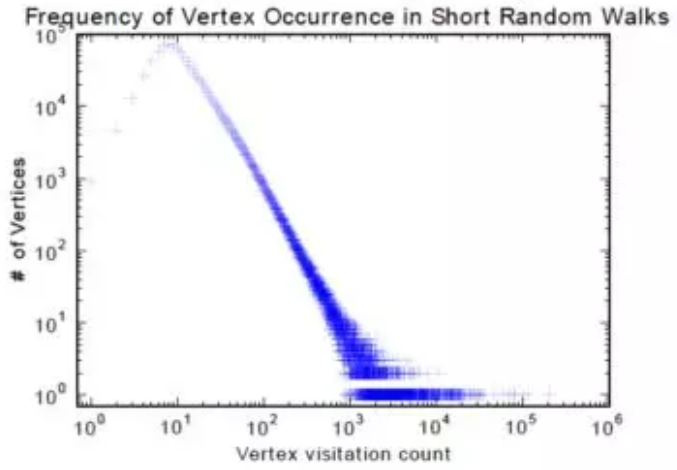
然而，有文章指出，GNN的这一原始方法存在三个主要限制：

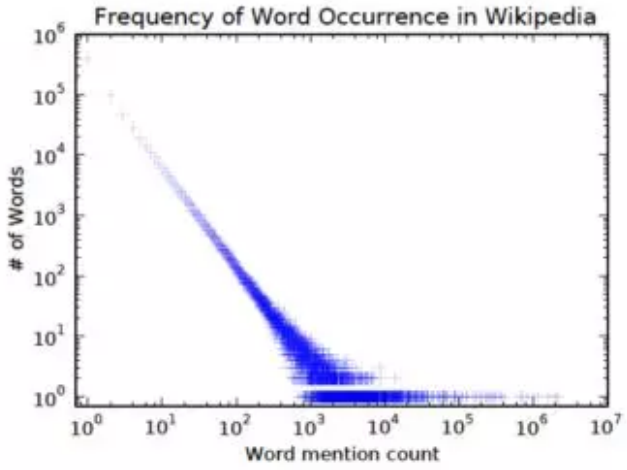
1. 如果放松“不动点”的假设，就有可能利用多层感知器来学习更稳定的表示，并消除迭代更新过程。这是因为，在原方案中，不同的迭代使用相同的转换函数f的参数，而MLP不同层中不同的参数允许分层特征提取。
2. 它不能处理边缘信息(例如，知识图中不同的边缘可能表示节点之间不同的关系)
3. 不动点会阻碍节点分布的多样化，因此可能不适合学习表示节点。

已经提出了几个GNN的变体来解决上述问题。然而，它们没有被涵盖，因为它们不是本文的重点。

DeepWalk

DeepWalk是第一个提出以无监督方式学习节点嵌入的算法。就训练过程而言，它非常类似于单词嵌入。其动机是图中节点和语料库中单词的分布遵循幂律，如下图所示:





算法包括两个步骤:

1. 在图中的节点上执行随机漫步以生成节点序列
2. 根据步骤1生成的节点序列，运行skip-gram，学习每个节点的嵌入

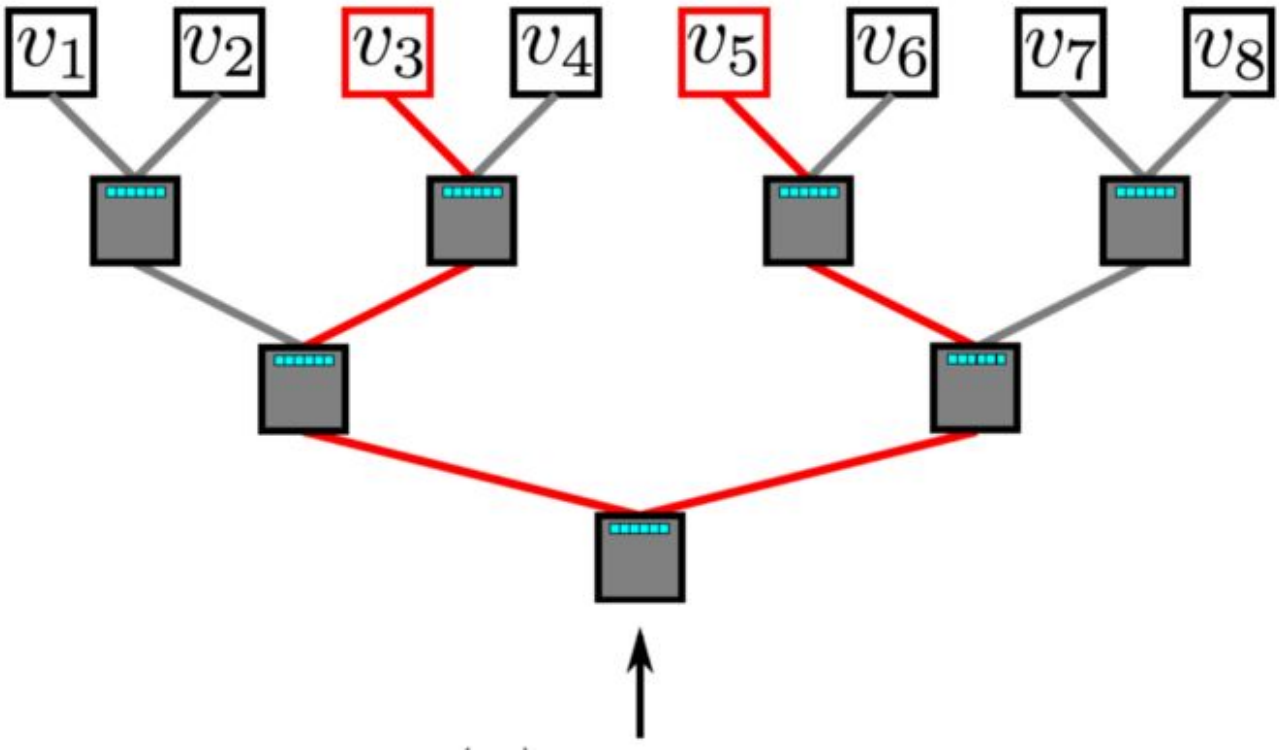
在随机游走的每个时间步长上，下一个节点均匀地从前一个节点的邻居中采样。然后将每个序列截断为长度*2|w| + 1*的子序列，其中w表示skip-gram的窗口大小。

本文采用层次softmax 算法，解决了节点数量大、计算量大的softmax问题。要计算每个单独输出元素的softmax值，我们必须计算所有元素k的所有e^xk。

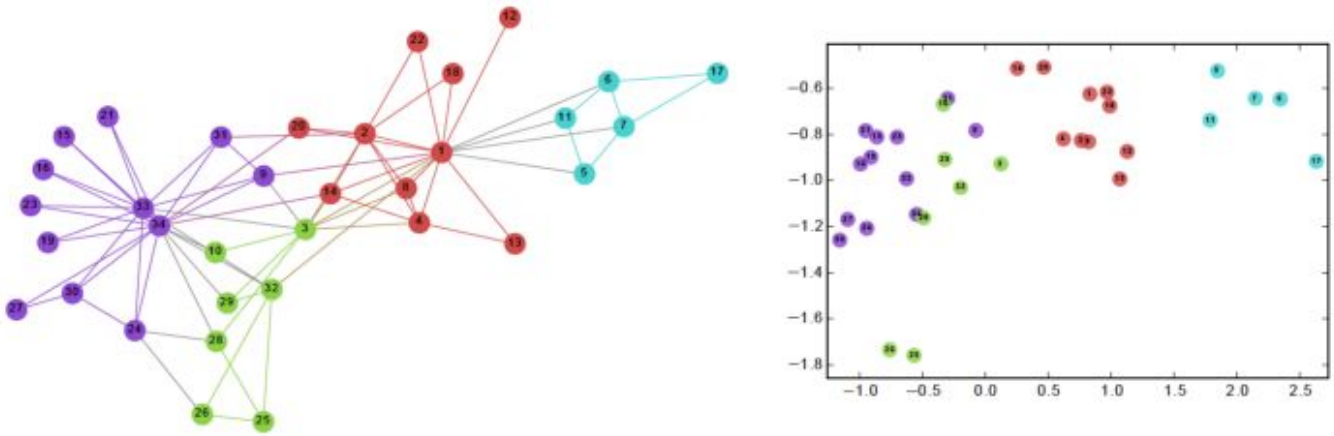
$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$$

因此，原始softmax的计算时间为*O(|V|)*，其中*V*表示图中顶点的集合。

层次softmax利用二叉树来处理该问题。在这个二叉树中，所有的叶子(上图中的v1 v2…v8)都是图中的顶点。在每个内部节点中，都有一个二进制分类器来决定选择哪条路径。要计算给定顶点*v_k*的概率，只需计算从根节点到左节点的路径上的每个子路径的概率*v_k*。由于每个节点的子节点的概率之和为1，所以所有顶点的概率之和等于1的性质在层次softmax中仍然成立。现在一个元素的计算时间减少到*O(log|V|)*，因为二叉树的最长路径以*O(log(n))*为界，其中*n*是叶子的数量。



经过DeepWalk GNN的训练，模型学习到每个节点的良好表示，如下图所示。不同的颜色表示输入图中不同的标签。我们可以看到，在输出图中(2维嵌入)，具有相同标签的节点聚集在一起，而具有不同标签的大多数节点被正确地分离。



(a) Input: Karate Graph (b) Output: Representation

然而，DeepWalk的主要问题是它缺乏泛化的能力。每当一个新节点出现时，它都必须对模型进行重新训练，才可以表示这个节点。因此，这种GNN不适用于图中节点不断变化的动态图。

GraphSage

GraphSage提供了一个解决上述问题的解决方案，以**归纳**的方式学习每个节点的嵌入。具体地说，每个节点由其邻域的聚合表示。因此，即使在图中出现了训练过程中没有出现过的新的节点，也可以用相邻的节点来表示。下面是GraphSage的算法。

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 $v \in \mathcal{V}$

1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
2 **for** $k = 1...K$ **do**
3 **for** $v \in \mathcal{V}$ **do**
4 $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
5 $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
6 **end**
7 $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
8 **end**
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

外循环表示更新迭代次数，*h^k_v*表示更新迭代时节点*v*的隐向量*k*。在每次更新迭代中，根据一个聚集函数、前一次迭代中*v*和*v*邻域的隐向量以及权矩阵*W^k*对*h^k_v*进行更新。本文提出了三个聚合函数:

1. Mean aggregator:

mean aggregator取一个节点及其所有邻域的隐向量的平均值。

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

与原始方程相比，它删除了上面伪代码第5行中的连接操作。这种操作可以看作是一种“跳跃连接”，本文稍后的部分证明了这种连接在很大程度上提高了模型的性能。

2. LSTM aggregator:

由于图中的节点没有任何顺序，它们通过遍历这些节点来随机分配顺序。

3. Pooling aggregator:

这个操作符在相邻的集合上执行一个元素池化函数。下面是最大池化的例子:

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

可以用均值池化或任何其他对称池化函数替换。池化聚合器性能最好，而均值池化聚合器和最大池化聚合器性能相近。本文使用max-pooling作为默认的聚合函数。

损失函数定义如下:

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

其中*u*和*v*共出现在固定长度的随机游动中，*v_n*是与*u*不共出现的负样本。这种损失函数鼓励距离较近的节点进行类似的嵌入，而距离较远的节点则在投影空间中进行分离。通过这种方法，节点将获得越来越多的关于其邻域的信息。

GraphSage通过聚合其附近的节点，为不可见的节点生成可表示的嵌入。它允许将节点嵌入应用于涉及动态图的领域，其中图的结构是不断变化的。例如，Pinterest采用GraphSage的扩展版本PinSage作为内容发现系统的核心。

总结

你学习了图形神经网络、DeepWalk和GraphSage的基础知识。GNN在复杂图形结构建模方面的能力确实令人吃惊。鉴于其有效性，我相信在不久的将来，GNN将在人工智能的发展中发挥重要的作用。

— END —

英文原文：https://towardsdatascience.com/a-gentle-introduction-to-graph-neural-network-basics-deepwalk-and-graphsage-db5d540d50b3