图表示学习起源: 从Word2vec到DeepWalk

原创 张备 图与推荐 2020-03-24

本文发表在知乎专栏<435算法研究所>,介绍的是2014年的一篇文章《DeepWalk: Online Learning of Social Representations》，附个链接https://arxiv.org/pdf/1403.6652.pdf，这是NLP中的表示学习算法第一次被引入到图结构当中。如标题，本文先来介绍Word2vec的基本知识，再来介绍下如何利用Word2vec来表示图结构。

一、Skip-Gram

一般提到word2vec有两种算法，cbow和Skip-Gram，相对来说Skip-Gram更为常用。目标是通过训练，用向量embedding来表示词，向量embedding中包含了词的语义和语法信息。

统计语言模型需要计算出一句话在词袋空间中出现的概率，通常的优化目标是最大化概率

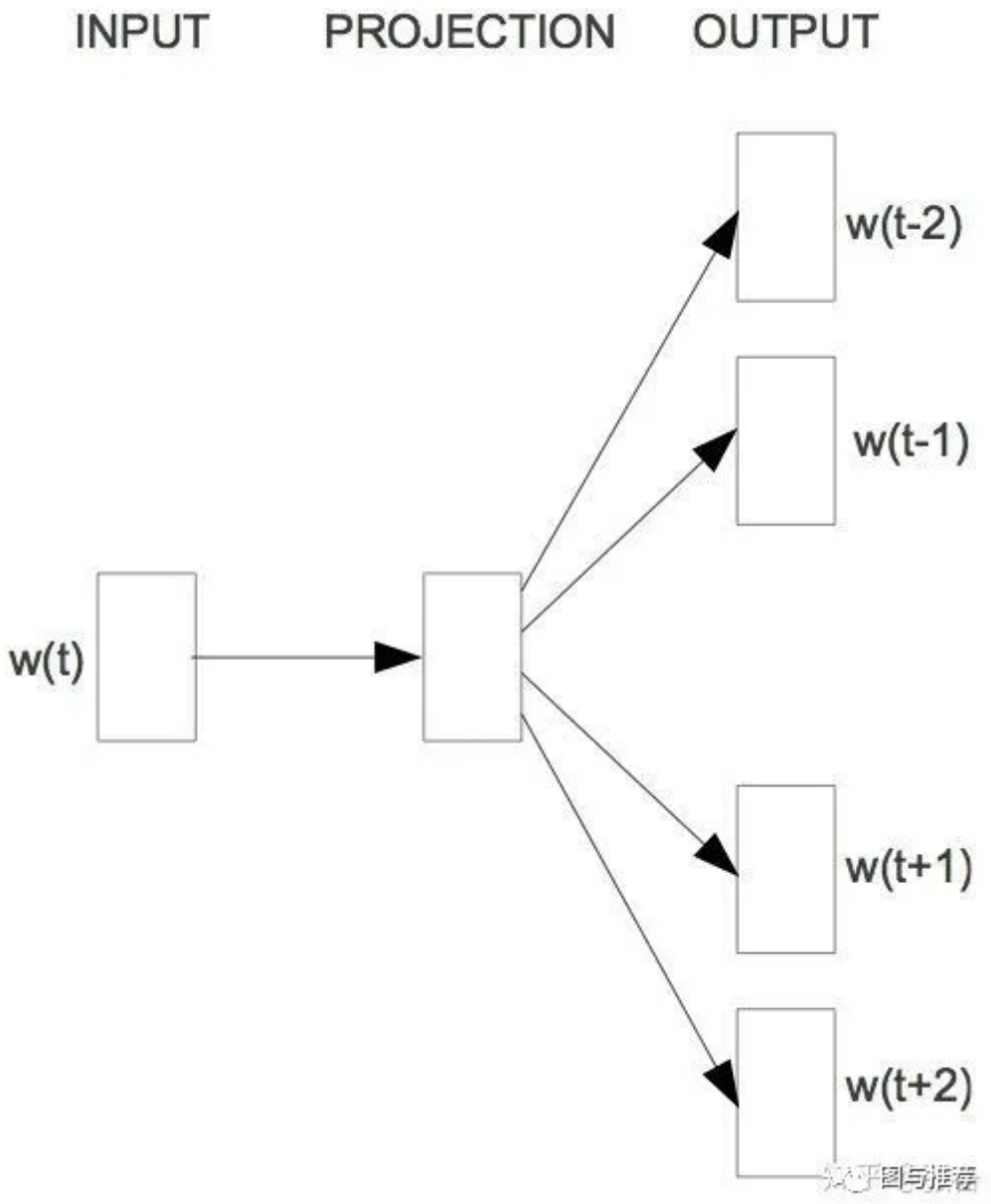
$$\Pr(w_n|w_0, w_1, \dots w_{n-1})$$

。w表示词袋中的单词，序号表示了词在句子中的顺序关系，但是这个目标实现起来比较困难。

Skip-Gram放宽了限制，做了一些调整。1）不再用句子中前面的词来预测下一个词，而是用当前词去预测句子中周围的词；2）周围的词包括当前词的左右两侧的词；3）丢掉了词序信息，并且在预测周围词的时候，不考虑与当前词的距离。优化目标是最大化同一个句子中同时出现的词的共现概率，优化目标如公式（1）所示，其中k表示滑窗大小。

$$\text{maxmize}(\log \Pr(\{w_{i-k}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+k}\}|w_i)), (1)$$

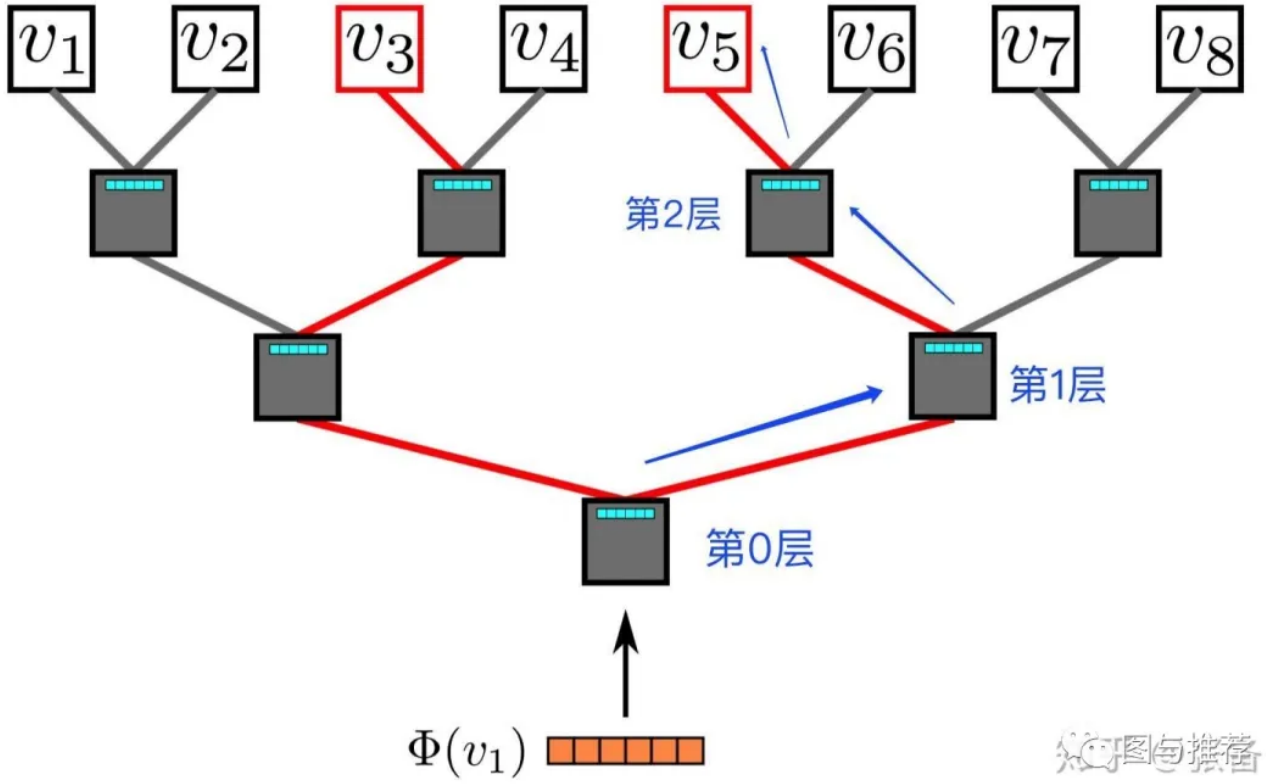
Skip-Gram的模型结构如下图所示，在整个词袋vocabulary中预测上下文词，在输出端需要做一个|V|维度的分类，|V|表示词袋vocabulary中词的数量。每一次迭代计算量非常庞大，这是不可行的。为了解决这个问题，采用了Hierarchical Softmax分类算法。



img

Hierarchical Softmax

如下图所示，构建一个二叉树，每一个词或者分类结果都分布在二叉树的叶子节点上，在做实际分类的时候，从根节点一直走到对应的叶子节点，在每一个节点都做一个二分类。假设这是一颗均衡二叉树，并且词袋的大小是|V|，那么从根走到叶子节点只需要进行$\log(|V|)$次计算，远远小于|V|的计算量。



img

具体如何计算？我们以上图中用v_1 预测v_5为例进行介绍。树的根部输入的是v_1的向量，用$\phi(v_1)$表示。在二叉树的每一个节点上都存放一个向量，需要通过学习得到，最后的叶子节点上没有向量。显而易见，整棵树共有|V|个向量。规定在第k层的节点做分类时，节点左子树为正类别，节点右子树是负类别，该节点的向量用$v(k)$表示。那么正负类的分数如公式(2)(3)所示。在预测的时候，需要按照蓝色箭头的方向做分类，第0层分类结果是负类，第1层分类结果是正类，第2层分类结果是正类，最后到达叶子节点 v_5 。最后把所有节点的分类分数累乘起来，作为v_1预测v_5的概率，如公式(4)所示，并通过反向传播进行优化。

$$p_k(left) = \text{sigmoid}(\phi(v_1) \cdot v(k)), (2)$$
$$p_k(\text{right}) = 1 - \text{sigmoid}(\phi(v_1) \cdot v(k)) = \text{sigmoid}(-\phi(v_1) \cdot v(k)), (3)$$
$$p(v_5|v_1) = \prod p_k = p_0(right) \cdot p_1(left) \cdot p_2(left), (4)$$

Huffman编码是一种熵编码方式，对于出现频率高的符号用较短的编码表示，出现频率较低的符号用较长的编码表示，从而达到编码压缩的目的。Hierarchical Softmax树也可以采用Huffman编码的方式生成，高频词用较短的路径到达，低频词用较长的路径到达，可以进一步降低整个训练过程的计算量。

顺便提一句，如果输出端是对所有词的softmax分类的话，那么在Skip-gram模型中，分别有输入和输出两个矩阵，一般是采用输出矩阵作为表示向量。但是如果采用Hierarchical Softmax分类的话，输出端就不存在输出矩阵了，就只能采用输入矩阵作为表示向量了。

二、DeepWalk

DeepWalk的思路很简单，将随机游走产生的序列当做句子，输入给skip-gram算法，从而得到节点的向量表示，具体算法过程如下图所示。其中，G表示表示网络，包括节点V和边E， [公式] 表示要随机游走的轮数，t表示每一次随机游走的长度。第1步是初始化节点的表示向量，第2步构建Hierarchical Softmax Tree，第5步表示对所有的节点，都要作为起点随机游走一次，第6步表示从v_i节点出发，随机游走t步得到序列 W_{v_i}，第7步表示将序列 W_{v_i} 进行SkipGram的训练。

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\mathcal{O} = \text{Shuffle}(V)$
5: **for each** $v_i \in \mathcal{O}$ **do**
6: $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
7: SkipGram$(\Phi, \mathcal{W}_{v_i}, w)$
8: **end for**
9: **end for**

img

前文提到过，Skip-Gram丢掉了句子中的词序信息，以及词与词之间的距离信息，这也适合网络表示学习，丢掉随机游走的顺序信息能够灵活地捕获节点之间的邻近关系。另外，如果两个节点具有相同的邻域，Skip-Gram学习出来的表示向量接近或者相似，有利于在下游任务上取得好的效果。

不管是在NLP中，还是在graph中，学习到的向量只是中间结果，用于作为下游任务的输入。例如在图中对节点做多标签分类任务时，第一步先通过DeepWalk进行无监督训练，得到所有节点的特征向量；第二步，通过一些分类器对节点进行分类。不同于传统的方法，DeepWalk将标签和表示空间分割开来，标签和表示向量相互独立，学习到的特征向量可以应用于各种不同的任务。而且试验证明，特征向量和最简单的分类算法相结合，比如逻辑回归，也能获得好的效果。

算法中有一个参数t，是随机游走的步长，即需要限定随机游走的长度，不要过长，有几个好处，1）可以捕获网络中局部区域的结构信息；2）易于实现并行化，多个线程，进程，甚至服务器，可以同时随机游走网络的不同部分，实现分布式计算，这个后边还会再提一下；3）能够适应网络的变化，网络局部发生变化时，可以只对局部网络进行学习和训练，而不需要对整个网络重新学习。

算法变种

1）streaming

训练前看不到整个网络，实时的将游走的序列丢到网络中进行训练。这对于构建hierarchical Softmax Tree比较麻烦，如果能够事先知道有多少个节点，以及节点的出现频率，就可以事先构建一个Huffman二叉树。否则的话，每次新遇到一个节点，将节点加到二叉树的叶子节点中。

2）Non-Random Walks

用户在访问某些网站的时候，可能会受到网站的引导而进行一些操作，因此访问过程可能并不随机，不过这并没有关系。对这些非随机的访问过程的训练，使得算法可以学习到网络的结构信息，以及访问路径的频次情况。

良好的可伸缩性

DeepWalk具有良好的可伸缩性，可以多台机器同时训练网络的不同部分。而且节点的出现频次符合指数分布，大部分低频节点都分布在长尾。多台机机器同时训练也不太会发生冲突，文中提出可以采用异步的随机梯度下降（ASGD）。下图中左边表示训练时间和机器数量的关系，服务器增加，训练时间也线性减少；右边表示服务器数量和效果的关系，随着服务器增加，训练出来的模型效果几乎没什么变化。

效果评估

为了衡量DeepWalk的效果，作者在3组数据上进行了测试，包括Blogcatalog，Flickr，Youtube。数据集中包含用户，用户间的关系，以及兴趣标签或者用户组标签。

整个流程应该是两阶段任务。第一阶段，在整个网络结构上做无监督训练，得到网络中每个节点的向量。第二阶段，根据已有的向量，划分一部分节点并给出标签作为训练集，剩下的节点作为测试集，训练一个分类器，预测测试集上的标签。

作者用$Micro-F_1$和 $Macro-F_1$ 做指标。分类器采用逻辑回归做one-vs-rest的分类，用的是LibLinear这个库。参数$\gamma=80,\quad w=10,\quad d=128$。

从实验结果来看，大部分情况下DeepWalk都要优于传统算法。在达到相同的效果时，DeepWalk 相比传统算法所需要的有标签训练数据更少。在大规模网络上，有些传统算法已经跑不动了，DeepWalk也有不错的表现。

附录：

介绍文中用于评估的几个指标：

(1)F_1

$$F_1 = 2.0 * \text{ precision } * \text{ recall } / \text{ precision } + \text{ recall })$$

$$\text{Macro} - F_1 = 1/n \sum_{i=0}^{n} (f_1^i)$$

(2)$Micro - F_1$

在多标签分类任务中，计算全部类别的所有样本的precision和recall，并且带入上面公式中计算得到Micro-F1，样本数量较多的类别可能会对指标影响较大。

3）$Macro - F_1$

在多标签分类任务中，对每个类别i分别计算 f_1^i，然后对所有类别求平均，如下公式所示，其中n表示类别的个数。每个类别的样本数量无论多少，在指标中都有相同的权重，因此即便是某些类别中样本数量较少，对于指标也有相当的影响。

$$\text{Macro} - F_1 = 1/n \sum_{i=0}^{n} (f_1^i)$$

文章已于2020-03-24修改