第 3 章

图表示学习

在本章中，将介绍图表示学习（Graph Representation Learning，GRL）的基本概念和相关算法。图表示学习是表示学习与图结构数据相结合产生的方法，其目的是将高维稀疏的图结构数据映射到低维稠密向量，同时来捕获网络拓扑结构及网络中节点的内在特征。现有的图表示学习方法按照图的学习技术可以大致分为三类：基于矩阵分解的图表示学习、基于随机游走的图表示学习和基于深度学习的图表示学习。按照研究图的类型可以分为同质图表示学习和异质图表示学习。本章将围绕这些图表示学习方法进行详细介绍。

3.1 图表示学习的意义

图是一种简单、易于理解的数据表现形式。一般而言，工业图中的节点数量巨大，而且边与边之间的连接并不稠密，如微博社交网络。在如此大而稀疏的图数据上，直接进行深度学习存在一定的局限性。图上的节点和边的关系，一般只能使用统计或者特定的子集进行表示，或者使用邻接矩阵表示。若直接在节点集合上做计算，或者直接采用维度较高的邻接矩阵来做计算都不太合适。例如，一个具有百万节点的图，其邻接矩阵的图占据的存储空间将至少达到数百吉比特。直接用邻接矩阵做计算将带来严重的内存消耗问题，同时也会影响计算效率。基于这些弊端，人们提出了图表示学习方法。如图 3-1 所示，图表示学习是一种将图数据（通常为高维稀疏的矩阵）映射为低维稠密向量的方法，同时通过图表示学习来捕获网络拓扑结构及图中节点的本质属性，用来加速图数据的计算效率，提高建模灵活性。图表示学习需要保持原有图的拓扑结构特征，例如，原图连接的节点，在嵌入向量空间中需

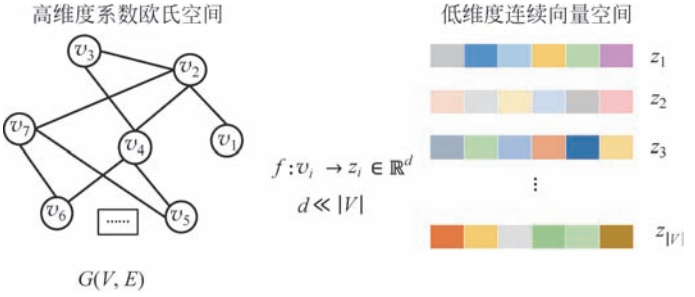


图 3-1　顶点表示学习示意图

要保持彼此靠近。

根据嵌入粒度的程度，可以将图表示学习分为两类，即顶点嵌入和整图表示学习。顶点嵌入是针对每个节点生成一个低维度的向量表征。这种图表示学习方式粒度较细，一般用于在节点层面上进行预测，如节点分类。如图 3-1 所示，将 $G(V,E)$ 的稀疏的节点信息，映射到低维的向量空间中，这里 V 是顶点集合，E 为边集合，映射到 $|V|$ 个维度为 $\mathbb{R}^d$ 的向量空间中。整图表示学习则是针对全图生成一个向量，这种图表示学习方式粒度较粗，一般用于在整图层面上进行预测或比较的场景。

在过去的数十年里，学术界一直没有停止对图表示学习方法的研究。总结过去的研究成果，可以将图表示学习按方法分为三类，如图 3-2 所示，包括基于矩阵分解的图表示学习方法、基于随机游走的图表示学习方法和基于神经网络的图表示学习方法。在本章中，介绍各类方法的一些具体算法，帮助读者全面深入地了解图表示学习。

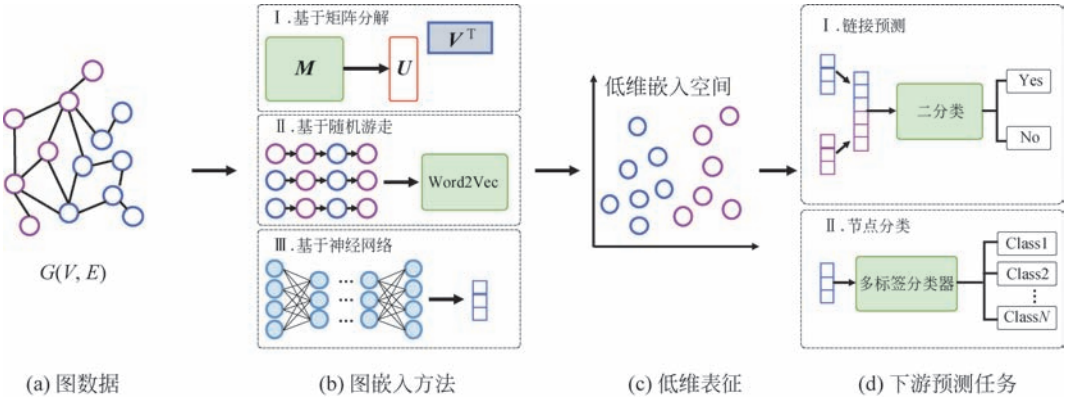


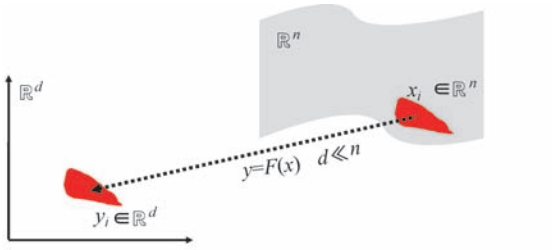
图 3-2　图表示学习算法的分类

3.2　基于矩阵分解的图表示学习方法

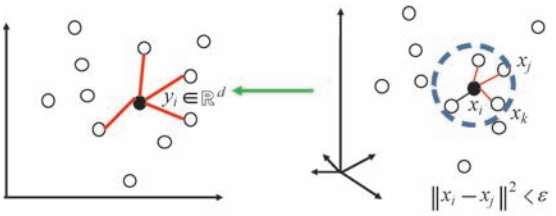
图结构的矩阵表示通常包括邻接矩阵(Adjacency Matrix)和拉普拉斯矩阵(Laplacian Matrix)。基于矩阵分解的图表示学习通过矩阵分解获取图中的节点向量表征。在本节中将详细介绍一个典型的基于分解的图表示学习：拉普拉斯映射(Laplacian Eigenmaps，LE)。

LE 算法是 Mikhail Belkin 和 Partha Niyogi 于 2002 年提出的基于图的降维方法，其核心思想是希望相互有连接的点在降维后的空间中能尽可能地被拉近，从而尽量维持降维之前的数据结构特征。如图 3-3 所示，假设在一个维度为 $\mathbb{R}^n$ 的流形空间中存在点集合 $X=\{x_1,x_2,\cdots,x_N\}$，其中 $x_i\in\mathbb{R}^n$，LE 算法的目标是将这些点映射到一个低维度空间 $\mathbb{R}^d$ 中 $(d\ll n)$，对应的映射坐标集合为 $Y=\{y_1,y_2,\cdots,y_N\}$，其中 $y_i\in\mathbb{R}^d$，且保证高维空间中离得很近的点投影到低维空间中的像也应该离得很近。LE 算法的主要目的是学习映射关系 $y=F(x)$。

给定的流形空间中的数据点集 $X=\{x_1,x_2,\cdots,x_N\}$ 属于散点集合，可以被视为图中顶点集合 V，然而边却不是预先给定的。那么如何构建点之间的边呢？这里介绍一种基于近



(a) 高维坐标系的点映射到低维坐标系



(b) 两个空间的点相对位置保持一致

图 3-3 LE 算法

邻距离的构建边的方法，即如果点 x_i 和 x_j 足够接近，则在这两点之间添加一条边来构建边集合 E。如图 3-3(b)所示，如果两个点落在以$\sqrt{\varepsilon}$为半径的球内则构建边，即 $\| x_i - x_j \|^2 < \varepsilon$ 作为判断条件决定是否在点 x_i 与点 x_j 之间构建边。当然也可以给边赋予一定的权重，一种方式是存在边则 $W_{ij}=1$，否则为 0，称为二元权重(Binary Weight)，另一种方式是给边赋予高斯权重 $W_{ij}=\mathrm{e}^{-\frac{\| x_i - x_j \|^2}{2\sigma^2}}$，$\sigma^2$ 为标准差。

上述过程已经可以构建起图 $G(V,E)$。接下来具体讲解 LE 算法利用图结构来实现降维的原理。按照 LE 算法的核心思想，降维的关键在于尽可能使原空间中相近的点在降维后的目标空间中也尽可能相近，可以将此问题转化为优化目标，公式如下：

$$\min_{y \in \mathbb{R}^d} \sum_i \sum_j W_{ij} \| y_i - y_j \|^2 \tag{3.1}$$

其中，$\| y_i - y_j \|^2$ 表示目标空间内两个数据点间的距离，W_{ij} 表示原空间内两个数据点间的边权重，这里采用高斯权重，即连续型变量 $W_{ij} \in (0,1]$。原空间中 x_i 与 x_j 越靠近，W_{ij} 越大，此时 y_i 和 y_j 也必须靠近；原空间中 x_i 与 x_j 越远，W_{ij} 越小，y_i 和 y_j 相对位置则比较灵活。

然而，式(3.1)并不能很好地约束 y，例如 $y_i = y_j = 0$。为了避免所有新坐标均为 0 或受缩放效应的影响，加一个约束 $\| y_i \|^2 = 1$。需要指出的是，式(3.1)可以做进一步转化。

$$\sum_i \sum_j W_{ij} \| y_i - y_j \|^2 = 2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{Y} \tag{3.2}$$

其中，$\boldsymbol{Y}=[y_1, y_2, \cdots, y_n] \in \mathbb{R}^{n \times d}$，$\boldsymbol{L}$ 为图 $G(V,E)$ 的拉普拉斯矩阵，可以改写为

$$\min_{\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Y}=1} 2\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{Y} \tag{3.3}$$

该问题便转化为一个广义特征值的问题，即求解如下特征方程：

$$\boldsymbol{L}_{rw} v_i = \lambda_i v_i \tag{3.4}$$

其中，$\boldsymbol{L}_{rw}=\boldsymbol{D}^{-1}\boldsymbol{L}=\boldsymbol{D}^{-1}(\boldsymbol{D}-\boldsymbol{W})=\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{W}$。$\boldsymbol{D}$ 为度矩阵。根据式(3.4)计算拉普拉斯矩阵的特征向量和特征值，并以此得出降维后的结果输出。$\boldsymbol{Y}=[v_2,\cdots,v_{d+1}]\in\mathbb{R}^{n\times d}$。

LE 算法可以将已有的特征高维变换获取简约特征。LE 算法可用于高维的数据降维和可视化，例如降低维度为二维或者三维。然而，LE 算法的复杂度为节点数量的平方，导致该算法不能适用于大规模图。

3.3　基于随机游走的图表示学习

本节重点讲解基于随机游走的图表示学习，随机游走学习节点相似度是基于领域节点相似的。随机游走的图表示学习借鉴了经典的词嵌入算法 Word2Vec，通过随机游走等采样方法从图中采样若干条由节点序列，将其视为自然语言中的句子，然后使用 Word2Vec 模型训练得到节点的嵌入向量。首先介绍 Word2Vec 算法，然后介绍 DeepWalk 和 Node2Vec 两个典型的基于随机游走的模型，最后介绍随机游走算法的优化技术。

3.3.1　Word2Vec 算法

Word2Vec 算法在 2013 年由 Google 提出，该算法将语料中的词的高维度独热(One-Hot)表征形式，通过神经网络学习得到低维词向量。在自然语言模型中，单词是划分后的最细粒度，单词组成句子，句子再组成段落和文章。对于语料中的单词，一种最简单的词向量方式是独热表征，即采用一个很长的向量来表示一个词。独热向量的长度为词汇表的大小，词典中的位置表示为 1，其他均为 0。如图 3-4 所示。‘我’表示为$[1,0,0,0,0]^{\mathrm{T}}$，‘在’表示为$[0,1,0,0,0]^{\mathrm{T}}$。然而每一种语言都有成千上万个单词，如果想用独热的方式表示这些单词，就会导致每一个单词的维度非常大，造成维度灾难。同时词语之间的独热编码是正交的，无法计算词之间的相似性。为此就需要用词嵌入算法找到一个合适的映射函数，能使高维度的索引向量嵌入到一个相对低维的向量空间内。Word2Vec 算法的最终目的是为了得到各个词的低维向量表达，解决独热编码的不足。Word2Vec 学习到的词向量的意义更为清晰，相关联的词在词向量空间中是相互靠近的。在数学上，意义相关的词之间的距离比无关的词之间的距离小。如图 3-5 所示，意义相近的词“美丽”与“漂亮”之间的距离会比较靠近。Word2Vec 包含连续词袋模型(Continuous Bag Of Words，CBOW)和跳字模型(Skip-Gram)两种算法。

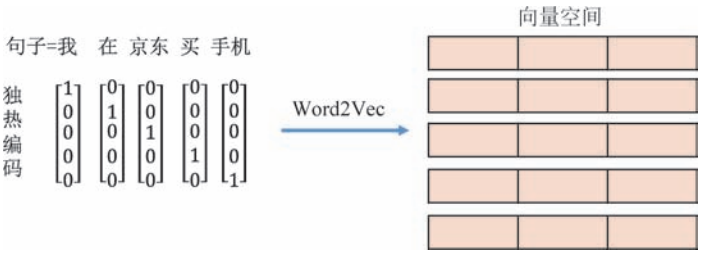


图 3-4　独热词表达转化低维词向量示意图

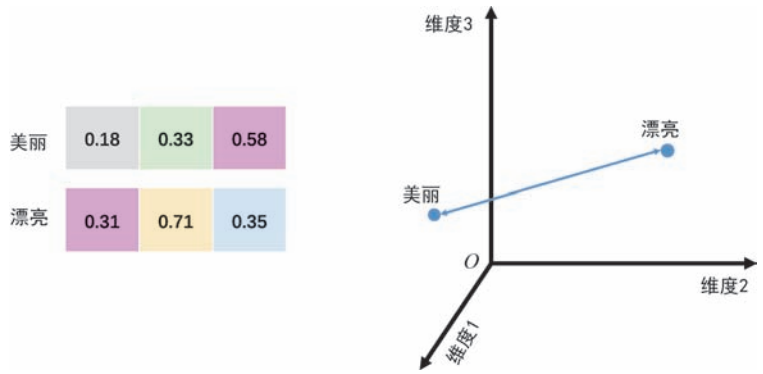


图 3-5　词向量空间中词的相对关系示意图

强调一下，Word2Vec 学习的目的是为了获得词在低维空间的向量表达，学习到的低维度的向量表达需要能描述词之间近似或者关联。因此，设计的模型要预测上下文，从而调整好模型参数，得到最终的词向量表达。

（1）连续词袋模型。

CBOW 是一个用上下文单词预测当前词的模型，如图 3-6 所示。当前语料的词序列为 $w(1),w(2),\cdots,w(T)$，其中包含 $|V|$ 个相异的词，构成词表集合 $W=\{w_1,w_2,\cdots,w_{|V|}\}$。词 w_k 的独热表征为 $\boldsymbol{X}_k=[x_1,x_2,\cdots,x_{|V|}]$，向量 $\boldsymbol{X}_k$ 的分量中只有 $x_k=1$，其余均为 0。CBOW 算法的目的是根据背景词 $w(t-n),w(t-n+1),\cdots,w(t+n-1),w(t+n)$ 简写为 $w_B(t)$ 来预测中心词 $w(t)$ 的概率，即 $p(w(t)|w_B(t))$，n 表示背景窗口大小，即句子内中心词前后 n 个词作为背景。算法结构如图 3-6 所示，首先将 $2n$ 个背景词的独热表征形式 $\boldsymbol{X}_{b_1},\boldsymbol{X}_{b_2},\cdots,\boldsymbol{X}_{b_{2n}}$ 通过共享权重矩阵 $\boldsymbol{W}\in\mathbb{R}^{|V|\times d}$ 映射到维度为 $\mathbb{R}^d$ 的向量：

$$z_{b_k}=\boldsymbol{W}^{\mathrm{T}}X_{b_k}=\boldsymbol{W}^{\mathrm{T}}_{(b_k,:)} \tag{3.5}$$

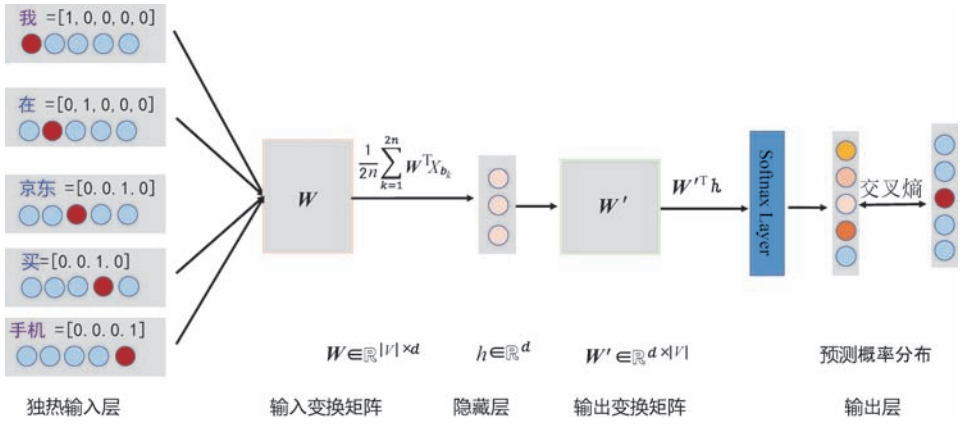


图 3-6　CBOW 计算结构示意图

$z_{b_k}\in\mathbb{R}^d$ 对应矩阵 $\boldsymbol{W}^{\mathrm{T}}$ 的第 b_k 行，称为词 w_{b_k} 的输入词向量（Input Embedding Vector），得到 $\{z_{b_1},z_{b_2},\cdots,z_{b_{2n}}\}$。隐藏层 $h\in\mathbb{R}^d$，是直接对周围词向量的求和 $\left(h=\sum_{i=1}^{2n}z_{b_i}\right)$ 或者平均 $\left(h=\frac{1}{2n}\sum_{i=1}^{2n}z_{b_i}\right)$。从隐藏层到输出层的共享权重矩阵为 $\boldsymbol{W}'\in\mathbb{R}^{d\times|V|}$，设中心词 $w(t)$ 对

应词表中的第 j^* 个词，$w(t)$对应的词向量为$\boldsymbol{W}'$的第 j^* 列 $\boldsymbol{z}'_{j^*}$，$\boldsymbol{z}'_{j^*}$ 称为词 $w(t)$的输出词向量(Output Embedding Vector)。将隐藏层 h 映射到维度$\mathbb{R}^{|V|}$的向量 $\boldsymbol{u}=\boldsymbol{W}'^{\mathrm{T}}h\in\mathbb{R}^{|V|}$，并通过一个 Softmax 层计算词表中的词为中心词的概率：

$$P(w(t)\mid w_B(t))=\frac{\exp(u_j)}{\sum_i^{|V|}\exp(u_i)}=\frac{\exp(\boldsymbol{z}'^{\mathrm{T}}_{j^*}\boldsymbol{h})}{\sum_i^{|V|}\exp(\boldsymbol{z}'^{\mathrm{T}}_i\boldsymbol{h})} \tag{3.6}$$

(2) 跳字模型。

跳字模型根据中心词 $w(t)$来预测背景词 $w_B(t)$的条件概率，用数学公式可以描述为 $P(w_B(t)|w(t))\in\mathbb{R}^1$。若假设中心词预测周围词的概率是相互独立的，则有

$$P(w_B(t)\mid w(t))=\prod_{-n\leqslant j\leqslant n,j\neq 0}P(w(t+j)\mid w(t)) \tag{3.7}$$

举个例子，在句子“我在京东买手机”中，我们根据“京东”来预测周围词的概率，则可以表述为：P(我，在，买，手机|京东)$=P$(我|京东)P(在|京东)P(买|京东)P(手机|京东)。

如图 3-7 所示，Skip-Gram 算法结构与 CBOW 模型类似，但是其输入为中心词 w_c，首先根据中心词的独热表征 $\boldsymbol{X}_c$ 映射得到隐藏层 $h\in\mathbb{R}^d$。

$$h=\boldsymbol{W}^{\mathrm{T}}\cdot\boldsymbol{X}_c \tag{3.8}$$

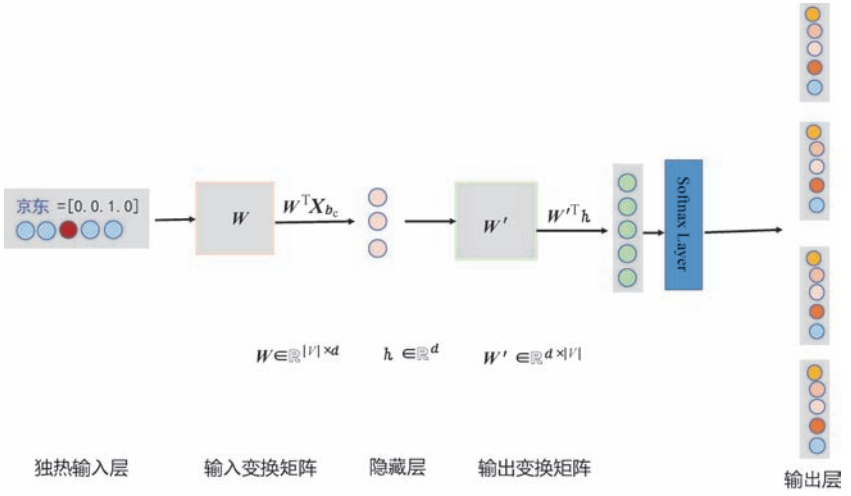


图 3-7　Skip-Gram 算法结构示意图

通过输出变换矩阵 $\boldsymbol{W}'\in\mathbb{R}^{d\times|V|}$，将隐藏层 h 映射到维度$\mathbb{R}^{|V|}$的向量 $\boldsymbol{u}=\boldsymbol{W}'^{\mathrm{T}}h$，并通过一个 Softmax 层计算词表中各词的概率：

$$\begin{bmatrix}P(w_1\mid w_c)\\P(w_2\mid w_c)\\P(w_3\mid w_c)\\\vdots\\P(w_{|V|}\mid w_c)\end{bmatrix}=\frac{\exp(\boldsymbol{W}'^{\mathrm{T}}\cdot h)}{\sum_{i=1}^{V}\exp(\boldsymbol{W}'^{\mathrm{T}}\cdot h)}=\frac{\exp(u)}{\sum_i^{|V|}\exp(u_i)} \tag{3.9}$$

设背景词 $w_B(t)$对应词表的序列为 $j_1^*,j_2^*,\cdots,j_{2n}^*$，对应 $\boldsymbol{W}'$的第 $j_1^*,j_2^*,\cdots,j_{2n}^*$ 列，可以

得到背景词的预测概率：

$$P(w_B(t) \mid w(t)) = \prod_{k=1}^{2n} \frac{\exp(u_{j_k^*})}{\sum_{i}^{|V|} \exp(u_i)} \tag{3.10}$$

(3) 模型训练。

在深度学习中，一般方式是最小化损失函数，而不是最大化损失函数。为了与之对应，在式(3.6)与式(3.10)上加一个负号。同时，对 $P(w(t)|w_B(t))$和 $P(w_B(t)|w(t))$采用单调递增的自然对数是为了降低计算复杂度。在整个语料 $w(1),w(2),\cdots,w(T)$上计算平均损失函数，公式如下：

$$L_{\text{CBOW}}(\boldsymbol{W},\boldsymbol{W}') = -\sum_{t=1}^{T} \ln(P(w(t) \mid w_B(t))) \tag{3.11}$$

$$L_{\text{SkipGram}}(\boldsymbol{W},\boldsymbol{W}') = -\sum_{t=1}^{T} \ln(P(w_B(t) \mid w(t))) \tag{3.12}$$

为了得到权重参数 $\boldsymbol{W}$、$\boldsymbol{W}'$，只需要利用反向传播算法来训练这个神经网络即可，反向传播算法可参考第 1 章。由损失函数形式可知，CBOW 模型比 Skip 模型训练速度更快。一般而言，Skip-Gram 模型对词频较低的词优于 CBOW 模型。需要强调的是，Word2Vec 模型的最终目的是得到词表的向量化表达，即矩阵 $\boldsymbol{W}^{\mathrm{T}}$ 的列向量(输入词向量)或者 $\boldsymbol{W}'^{\mathrm{T}}$ 的行向量(输出词向量)。

3.3.2 DeepWalk

DeepWalk 由 Bryan Perozzi 等人[①]于 2014 年提出的。DeepWalk 借鉴了 Word2Vec 的思想，希望借助两个点之间的共现关系(Co-occurrences)来学习向量表征表示。Word2Vec 中所对应的数据输入格式是由一个个单词组成的句子，通过句子中单词的共现关系学习表征，但对于图数据结构，如何获取类似的数据呢？由于图是非线性的，所以需要一个策略将其转换为线性输入。DeepWalk 通过随机游走的方式提取顶点序列，根据顶点和顶点的共现关系，学习顶点的向量表示。DeepWalk 抽取顶点序列的方法是随机游走(Random Walk)，可以说随机游走是整个 DeepWalk 中最重要也是最具有开创性的一部分算法。

随机游走是一种可重复访问已访问节点的深度优先遍历算法。对于给定的一个图中的起始节点，随机游走算法会从其邻居节点中随机抽取一个节点作为下一个访问点，游走到下一个节点后重复上述过程，直到访问序列达到预设长度(需要注意的是，被访问的点不会被标记删除，也不会更新访问它的概率，因而随机游走可以重复访问已访问的节点)。

在图 $G(V,E)$，顶点 $v_i \in V$ 的邻居节点集合记作 $N(v_i)$，v_i 的度记为 $D(v_i)$。对于顶点 $v_i \in V$ 为随机游走起点，记作 $tr_{v_i}^{(0)}$，生成一个随机序列 $\mathrm{Tr}=tr_{v_i}^{(0)},tr_{v_i}^{(1)},\cdots,tr_{v_i}^{(T-1)}$，如图 3-8 所示，其中 T 是一个超参数，表示游走序列长度。这里按照对邻居节点等概率采样机序列

① Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. 2014: 701-710.

$$P(tr_{v_i}^{(t+1)} \mid tr_{v_i}^{(t)}) = \begin{cases} \dfrac{1}{d(tr_{v_i}^{(t)})}, & tr_{v_i}^{(t+1)} \in N(tr_{v_i}^{(t)}) \\ 0, & \text{其他} \end{cases} \tag{3.13}$$

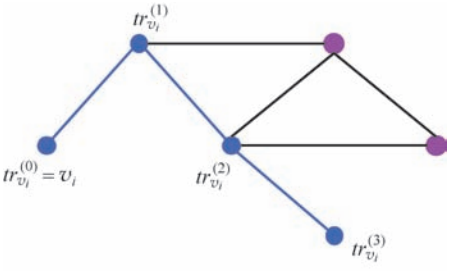


图 3-8　随机游走示意图

其中，$N(tr_{v_i}^{(t)})$表示节点 $tr_{v_i}^{(t)}$ 的邻居集合，$d(tr_{v_i}^{(t)})$表示节点 $tr_{v_i}^{(t)}$ 的度。依次从上一个访问顶点的邻居节点中均匀随机采样一个顶点作为序列的下一个点。经 T 步随机游走，为了充分挖掘节点 v_i 在全图中的上下文信息，DeepWalk 算法中，可以每个节点 $v_i \in V$ 为中心，生成 γ 个独立随机游走序列，记作 $\mathrm{Tr}_{v_i}^{(0)}, \mathrm{Tr}_{v_i}^{(1)}, \cdots, \mathrm{Tr}_{v_i}^{(\gamma-1)}$，总共会产生 $\gamma|V|$ 个随机游走序列，类似于 Word2Vec 语料中的 $\gamma|V|$ 个句子。

随机游走还有并行化和适应性两大优势。并行化体现在一个大的网络中可以同时开始多个从不同顶点开始的随机游走，这可以大大减少运行时间；适应性指的是由于随机游走具有局部性，使其能很好地适应网络的局部变化，网络在演变过程中通常只会有部分点和边产生变化，这些变化只会对一部分随即游走的路径产生影响，因此在网络变化后只需要重新采样这部分改变的点和边，不需要重新计算整图的随机游走。

如图 3-9 所示，与 Skip-Gram 算法相对应，在 DeepWalk 中，对于随机游走序列 $\mathrm{Tr} = tr_{v_i}^{(0)}, tr_{v_i}^{(1)}, \cdots, tr_{v_i}^{(T-1)}$，设观测窗口大小为 n，顶点 $tr_{v_i}^{(j)}$ 作为中心词其前后 n 个顶点 $tr_{v_i}^{(j-n)}, tr_{v_i}^{(j-n+1)}, \cdots, tr_{v_i}^{(j+1)}, \cdots, tr_{v_i}^{(j+n)}$ 是 $tr_{v_i}^{(j)}$ 的上下文。按照 Skip-Gram 算法，以 $tr_{v_i}^{(j)}$ 为输入，预测邻居节点 $tr_{v_i}^{(j+k)}$ 的条件概率，可写为 $P(tr_{v_i}^{(j+k)} \mid tr_{v_i}^{(j)})$。若节点 $tr_{v_i}^{(j)}$ 在顶点表中的序号为 s_j，类似于 Word2Vec 算法，需要将独热形式的节点表征 $X_{s_j} \in \mathbb{R}^{|V|}$ 通过共享权重矩阵 $\boldsymbol{W} \in \mathbb{R}^{|V| \times d}$ 映射为 $z_{s_j} = \boldsymbol{W}^{\mathrm{T}} X_{s_j} \in \mathbb{R}^d$ 的低维向量表征。$\boldsymbol{W}' \in \mathbb{R}^{d \times |V|}$ 为输出权重矩阵，设 $z_i' \in \mathbb{R}^d$ 为输出权重矩阵 $\boldsymbol{W}'^{\mathrm{T}}$ 的第 i 行，则通过 Softmax 函数来计算概率：

$$P(tr_{v_i}^{(j+k)} \mid tr_{v_i}^{(j)}) = \frac{z'_{s_{j+k}} \cdot z_{s_j}}{\sum\limits_{v_i \in V} \exp(z'_i \cdot z_{s_j})} \tag{3.14}$$

对于采样序列 Tr 的优化目标为

$$\max_{\boldsymbol{W}, \boldsymbol{W}'} \prod_{tr_{v_i}^{(j)} \in \mathrm{Tr}} \prod_{-n \leqslant k \leqslant n, k \neq 0} \frac{z'_{s_{j+k}} \cdot z_{s_j}}{\sum\limits_{v_i \in V} \exp(z'_i \cdot z_{s_j})} \tag{3.15}$$

通过随机游走获得序列数据之后，便可以使用 Word2Vec 中的两个模型来进行训练。这里只介绍使用 Skip-Gram 模型获取这些节点的向量嵌入，Skip-Gram 模型的细节见 3.2

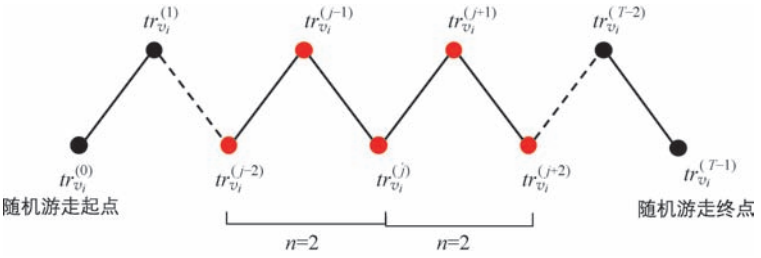


图 3-9 随机游走序列 Tr 中节点 $tr_{v_i}^{(j)}$ 的上下文关系示意图

节相关部分。

整体过程如图 3-10 所示，由于图是非线性的，DeepWalk 先采用随机游走生成节点的随机序列，后将游走序列视为句子，然后采用 Skip-Gram 算法对顶点共现关系建模，来学习节点表征。

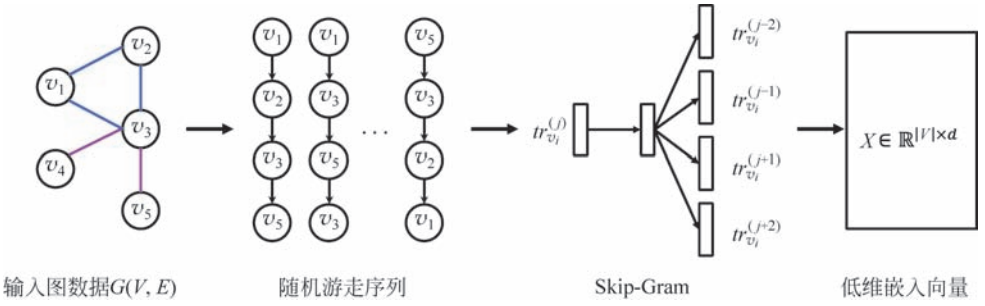


图 3-10 DeepWalk 计算示意图

3.3.3 Node2Vec

首先回顾一下图算法中经典的采样策略，即广度优先采样（Breadth-first Sampling，BFS）和深度优先采样（Depth-first Sampling，DFS）两种采样策略。如图 3-11 所示，以 u 节点为起点深度优先采样出 s_4、s_5、s_6，而广度优先则采样出 s_1、s_3、s_4。

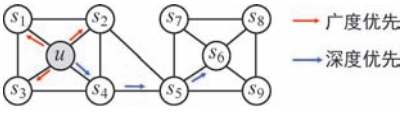


图 3-11 源节点为 u 的深度优先和广度优先遍历策略示意图

图表征的目的是得到节点间的相似性。网络的相似性有同质性（Homophily）和结构等价性（Structural Equivalence）两个评价标准。同质性是指属于同一集聚结构的两个节点更加相似，如图 3-11 中的 s_1 和 u。结构等价性是指在两个近似的集聚结构中扮演类似角色的节点间更具有相似性，如节点 s_6 和 u。那么采样策略对图节点的向量表征的学习有什么影响呢？

广度优先可以获得每个节点的所有邻居，强调的是局部微观视图，所以通过 BFS 采样的网络更能体现网络的局部结构，从而产生的表征更能体现结构等价性；而 DFS 可以探索

更大的网络结构，只有从更高的角度才能观察到更大的集群，所以其嵌入结果更能体现同质性。

Node2Vec 是另一种基于随机游走的图表示学习算法，于 2016 年被 Aditya Grover 等人[①]提出。Node2Vec 的最大突破是改进了 DeepWalk 中的随机游走采样策略。接下来将详细介绍 Node2Vec 中采用的有偏向的游走策略。Node2Vec 是一种综合考虑 DFS 邻域和 BFS 邻域的图表示学习方法。此处设计了一种灵活的邻居节点抽样策略，它允许用户在 BFS 和 DFS 间进行平衡。Node2Vec 可以通过参数设置来控制搜索策略，从而有效地平衡图表示学习的同质性和结构有效性。

Node2Ve 引入了两步随机游走算法，如图 3-12 所示。第一步从节点 t 游走到节点 v，第二步从节点 v 游走至其邻居节点，如 x_1、x_2、t 等。节点 v 跳转到邻居节点的概率不再是均匀分布的，而是根据节点 t 与节点 x 来共同确定，可以表示为 $\alpha(v_{t+1}|v_t,v_{t-1})$。这里是根据节点 t 与节点 x 的最短路径距离来确定，可将候选游走节点分为三类，并由两个超参数 p 和 q 来调控，$\alpha(v_{t+1}|v_t,v_{t-1})$具体为

$$\alpha(v_{t+1}\mid v_t,v_{t-1})=\begin{cases}\dfrac{1}{p}, & d_{(v_{t-1},v_{t+1})}=0\\ 1, & d_{(v_{t-1},v_{t+1})}=1\\ \dfrac{1}{q}, & d_{(v_{t-1},v_{t+1})}=2\end{cases}\tag{3.16}$$

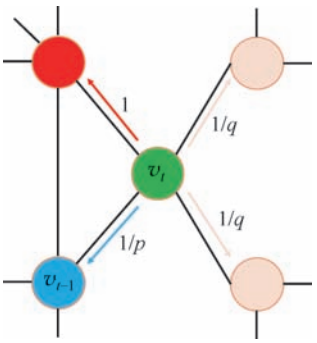


图 3-12　Node2Vec 随机游走

其中，$d_{(v_{t-1},v_{t+1})}$表示从节点 v_{t+1} 到节点 v_{t-1} 的最短路径距离。当 $d_{(v_{t-1},v_{t+1})}=0$ 时，即节点 v_t 跳转回节点 v_{t-1}，$\alpha(v_{t+1}|v_t,v_{t-1})=1/p$；若 $d_{(v_{t-1},v_{t+1})}=1$，即 $v_{t+1}\in N(v_{t-1})$，也就是说 v_{t+1} 为 v_{t-1} 的邻居节点，此时 $\alpha(v_{t+1}|v_t,v_{t-1})=1$；若 $d_{(v_{t-1},v_{t+1})}=2$，即 $v_{t+1}\notin N(v_{t-1})$，此时 $\alpha(v_{t+1}|v_t,v_{t-1})=1/q$。其中 p 被称为返回参数(Return Parameter)，q 被称为进出参数(In-out Parameter)。

对于参数 q：若 $q>1$，随机游走会倾向于访问与 t 相连的节点，从而体现出 BFS 特性；若 $q<1$，那么随机游走会倾向于访问远离 t 的节点，即朝着更深的节点游走，从而体现出 DFS 特性。对于返回参数 p：若 $p>\max(1,q)$，则返回的概率会变得相对较小，这时候游

① Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 855-864.

走会倾向于不往回走，更倾向 DFS 特性；若 $p<\max(1,q)$，则返回的概率会变得相对较大，这时候游走会倾向于往回走，多步游走会倾向于围绕在起始点附近，更倾向 DFS 特性。

获得采样序列后，接下来的处理流程与 DeepWalk 相同，总体流程如图 3-13 所示。当 Node2Vec 设置 $p=q=1$ 时，等价于 DeepWalk。

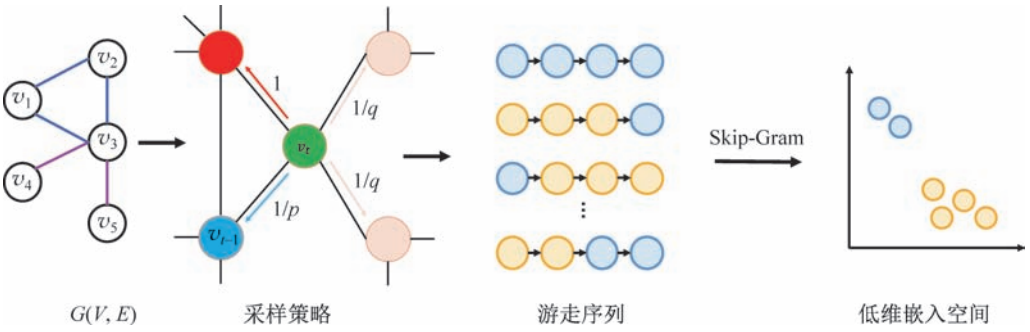


图 3-13 Node2Vec 算法流程

3.3.4 随机游走模型的优化策略

随机游走策略的图表示学习方法，预测共现关系的模块为 Skip-Gram 算法。在传统模型的预测阶段，需要利用一个 $\mathbb{R}^{d\times|V|}$ 的权重参数将 $\mathbb{R}^d$ 维的中间向量转回一个 $|V|$ 维的向量，再通过 Softmax 归一化计算每个节点的预测概率，最后进行损失计算和预测。这种方法在处理大量节点的图时会有速度过慢的问题。下面介绍分层 Softmax（Hierarchical Softmax）和负采样（Negative Sampling）两种优化算法。

1. 分层 Softmax

分层 Softmax 算法重新设计了 Skip-Gram 的输出层。以图 $G(V,E)$ 中的节点为叶子节点，以节点的度为权重构造一棵霍夫曼树（Huffman Tree），如图 3-14 所示，用此霍夫曼二叉树结构的神经网络代替原先的 Skip-Gram 神经网络层。在这棵二叉树中，叶子节点共有 $|V|$ 个，非叶子节点共有 $|V|-1$。在分层 Softmax 算法中，隐藏层到预测概率的输出不是一下子完成的，而是沿着霍夫曼树从根节点一步步向前传播至叶子节点完成的，相应传播路径上的内部节点则起到隐藏层神经元的作用。

如图 3-14 所示，假设输入节点为 v_i 预测节点 v_p 图中示例预测目标为（$v_p=v_3$），先将独热编码的输入节点向量 $\boldsymbol{X}_i\in\mathbb{R}^{|V|}$ 转换为低维度的隐藏层向量 $\boldsymbol{h}\in\mathbb{R}^d$，再沿着霍夫曼树向前传播至叶子节点。对二叉树中的每一个叶子节点，存在唯一由根节点到该叶子节点的路径。在分层 Softmax 中，我们利用这条路径估计当前预测值是否为该叶子节点表示的单词的概率 $P(v_p|v_i)$。在图 3-14 中，从根节点传播至 v_p 的路径如红色路线所示。在分层 Sotfmax 模型中，假设这条传播路径是随机游走形成的，每步游走过程是相互独立的，于是根据独立事件的联合概率原则可以得到

$$P(v_p=v_3\mid v_i)=P(n(v_p,1),\text{left})\cdot P(n(v_p,2),\text{right})\cdot P(n(v_p,1),\text{left}) \quad (3.17)$$

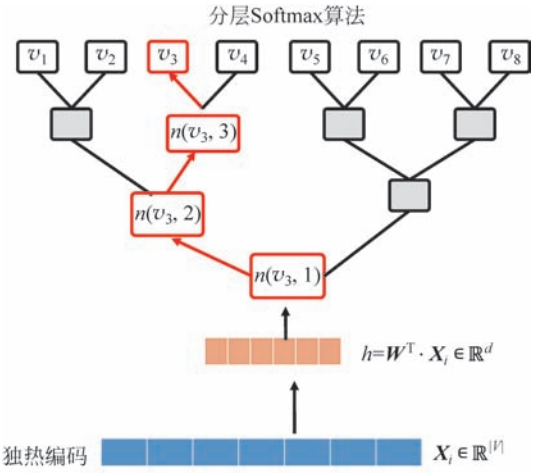


图 3-14　霍夫曼树示意图

其中，$n(v_p, j)$表示从根节点到叶子节点 v_p 路径上的第 j 个霍夫曼树节点。$P(n(v_p, j), \text{left})$表示节点 $n(v_p, j)$指向它的左孩子的概率，而 $P(n(v_p, j), \text{left})$则表示节点 $n(v_p, j)$指向它的右孩子的概率。此处向左向右的选择，可以视为一个二分类问题，每一个内部节点向左孩子节点游走的概率为

$$P(n(v_p, j), \text{left}) = \sigma(\boldsymbol{\theta}_j^{v_p} \cdot \boldsymbol{h}^{\mathrm{T}}) \tag{3.18}$$

其中，$\boldsymbol{\theta}_j^{v_p} \in \mathbb{R}^d$ 表示内部节点 $n(v_p, j)$的可学习向量，$\sigma(x) = \dfrac{1}{1+\mathrm{e}^{-x}}$表示 Sigmoid 激活函数。在二叉树中，每个内部节点只会存在左孩子或者右孩子节点，则节点 $n(v_p, j)$指向右孩子的概率为

$$P(n(v_p, j), \text{right}) = 1 - \sigma(\boldsymbol{\theta}_j^{v_p} \cdot \boldsymbol{h}^{\mathrm{T}}) = \sigma(-\boldsymbol{\theta}_j^{v_p} \cdot \boldsymbol{h}^{\mathrm{T}}) \tag{3.19}$$

式(3.17)更一般的形式为

$$P(v_p \mid v_i) = \prod_{j=1}^{L(v_p)-1} \sigma([n(v_p, j+1) = \mathrm{ch}(n(v_p, j))] \cdot \boldsymbol{\theta}_j^{v_p\,\mathrm{T}} \cdot \boldsymbol{h}) \tag{3.20}$$

$\mathrm{ch}(n(v_p, j))$表示节点 $n(v_p, j)$的左孩子节点，$L(v_p)$是根节点到叶子节点的路径长度，$[x]$为一个符号函数：

$$[x] = \begin{cases} 1, & x \text{ 为 true} \\ -1, & \text{其他} \end{cases} \tag{3.21}$$

$[n(v_p, j+1) = \mathrm{ch}(n(v_p, j))]$用来指示节点 $n(v_p, j)$下一步游走至左孩子还是右孩子。为了最大化预测概率 $P(v_p \mid v_i)$，在分层 Softmax 模型中的损失函数可以定义为

$$L = -\ln(P(v_p \mid v_i)) = -\sum_{j=1}^{L(v_p)-1} \ln\sigma([n(v_p, j+1) = \mathrm{ch}(n(v_p, j))] \cdot \boldsymbol{\theta}_j^{v_p\,\mathrm{T}} \cdot \boldsymbol{h}) \tag{3.22}$$

训练的复杂度，从原来的 $O(|V|)$下降到 $O(\ln(|V|))$。

2. 负采样(Negative Sampling)

在 DeepWalk 和 Node2Vec 算法中,使用随机游走嵌入算法在图 $G(V,E)$采样,生成线性序列,然后使用中心节点来预测观测窗口内的共现节点出现的概率。节点预先采用维度为$\mathbb{R}^{|V|}$独热编码,训练后得到$\mathbb{R}^d$ 的低维度表征。如图 3-15 所示,观测窗口为 2 时,我们使用训练样本(输入节点为 $tr_{v_i}^{(j)}$,预测节点为 $tr_{v_i}^{(j-2)}$、$tr_{v_i}^{(j-1)}$、$tr_{v_i}^{(j+1)}$、$tr_{v_i}^{(j+2)}$)。出现在观测窗口中的样本称为正样本(Positive Sample)。不在窗口内的样本称为负样本(Negative Sample)。对于正样本,其对应的神经元的期望值为 1,剩下的元素则为 0。图的节点个数$|V|$决定 Skip-Gram 神经网络会存在大规模的权重矩阵,这些权重需要通过大量的训练样本进行调整,会消耗大量的计算资源。负采样方法中,选择留下正样本及部分采样得到的负样本,并不对每个样本更新所有权重,而是每次让一个训练样本仅仅更新一小部分权重,其他权重全部固定,从而降低梯度下降过程中的计算量,提升训练速度。

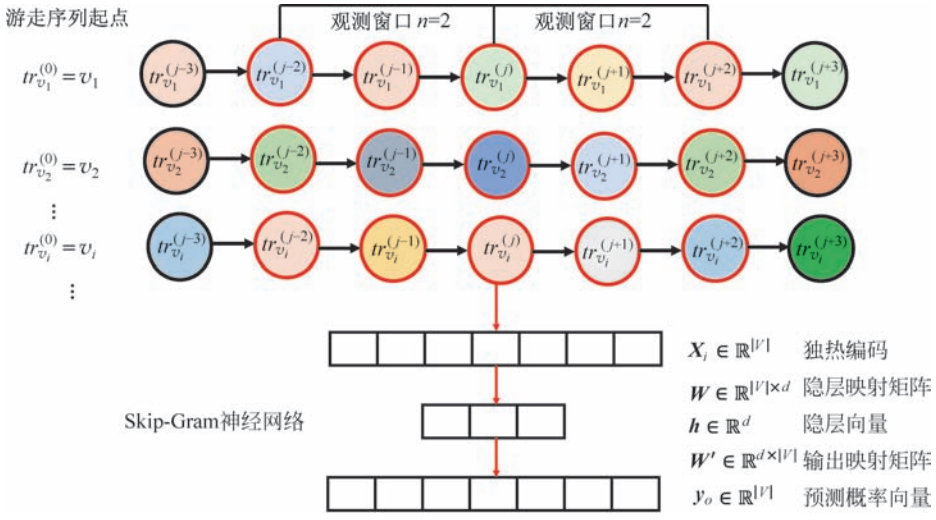


图 3-15 随机游走嵌入算法示意图

为了提升训练权重参数矩阵的质量,负采样修改目标函数,既考虑正样本也考虑负样本,需要最大化成对的概率,而最小化不关联节点对的概率,对应的损失函数为

$$L=\sum_{j=-n}^{n}-p(tr_{v_i}^{(j+k)}\mid tr_{v_i}^{(j)})+\sum_{j+kn\in tr_{\text{neg}}}p(tr_{v_i}^{(j+kn)}\mid tr_{v_i}^{(j)}) \tag{3.23}$$

其中,tr_{neg} 表示负样本集合,即不处在节点 $tr_{v_i}^{(j)}$ 观测窗口内的节点集合,这里只抽取部分负样本来计算,故称为负采样。如图 3-15 所示,正样本是处于观测邻居内的节点集合。负采样因为只计算部分负样本,故能加速计算,使得随机游走模型能适应规模更大的图。

这里通过图 3-15 讲解负采样的过程。假设图的节点个数为$|V|$,当输入节点 $tr_{v_i}^{(j)}$ 进入预测模型时,对其共现节点 $tr_{v_i}^{(j+1)}$ 的输出向量维度为$\mathbb{R}^{|V|}$,在此向量中我们希望 $tr_{v_i}^{(j+1)}$ 对应的位置为 1,其余位置为 0。如果按照传统的训练方式,这里所有的输出向量对应的权重参数都需要更新,而负采样的对象是$|V|-1$ 位,负采样策略是从中选择一小部分,并用这一

小部分负样本进行权重更新。在小规模数据集上，一般对每个顶点选择 5～20 个负样本会比较好，而在大规模数据集上可以仅选择 2～5 个负样本。假设负采样的个数为 k，则计算复杂度为全部采样的$\frac{k}{|V|}$。

从负样本中抽取样本时，需要按照合适的概率分布来抽取，才能让学习到的权重参数更好。这里介绍一种常用的采样概率分布：$P(v)\sim d(v)^{3/4}$，具体为

$$P(v_i)=\frac{d(v_i)^{\frac{3}{4}}}{\sum_{v\in V}(d(v)^{\frac{3}{4}})} \tag{3.24}$$

其中，$d(v_i)$代表顶点 v_i 出现的度。

3.3.5　其他随机游走方法

DeepWalk 可以视作一个基于深度优先的随机游走算法，Node2Vec 则综合了深度优先与广度优先。这两种算法都是基于邻域相似假设的方法，即网络中相似的点在向量表示中的距离比较近，但是这两种算法只考虑了成边的顶点之间的相似度，并未对不成边顶点之间关系的建模。LINE(Large-scale Information Network Embedding)模型既考虑成边的顶点对之间的关系(称为局域相似度)，也考虑未成边顶点的相似度(称为全局相似度)。LINE 算法为图的局域相似度和全局相似度设计了专门的度量函数，用来学习得到保持图局域和全局结构的低维节点表征 $u_i\in\mathbb{R}^d$，且能适用于无向图和有向图。

局域相似度用一阶相似度(First-order Proximity)描述，表示图中直接相连的节点之间的相似度。具体来说，存在边的两个节点之间的相似度为其边的权重 $e_{i,j}$，对于不存在边的节点间，则权重为 0，图 3-16 中 6 和 7 就是一阶相似，因为 6 和 7 直接相连。对于每一条无向边(v_i,v_j)，节点 v_i、v_j 之间的一阶联合概率为

$$P_1(v_i,v_j)=\frac{1}{1+\exp(-\boldsymbol{z}_i^{\mathrm{T}}\cdot\boldsymbol{z}_j)} \tag{3.25}$$

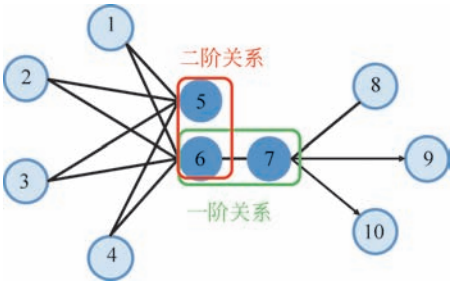


图 3-16　网络中一阶关系和二阶关系示意图

其中，$\boldsymbol{z}_i\in\mathbb{R}^d$ 是节点 v_i 的需要学习的低维向量表征。同时定义需要拟合经验概率(Empirical Probability)，即归一化的边权重：

$$\hat{P}_1(v_i,v_j)=\frac{W_{ij}}{\sum_{(m,n)\in E}W_{mn}} \tag{3.26}$$

优化目标就是尽可能拉近这两个分布的距离。常用的衡量两个概率分布差异的指标为 KL 散度，目标函数可写为

$$O_1 = -\sum_{(i,j)\in E} W_{ij} \ln P_1(v_i, v_j) \tag{3.27}$$

需要注意的是，一阶相似度只能用于无向图。

全局相似度是用二阶相似度(Second-order Proximity)来衡量两个节点的邻居节点之间的相似度。二阶相似度假设那些具有相同邻居节点的节点在特征上较为相似，具体来说，若两个节点的邻居节点集合 N_i 与 N_j 之间有许多重叠，则认为两个节点之间拥有很高的二阶相似度。直观地来解释就是拥有共享邻居的节点更为相似，在许多现实的例子中可以印证这一点，例如，拥有相同社交网络的两个人很可能有共同的兴趣。图中的 5 和 6 就是二阶相似，因为它们虽然没有直接相连，但是它们连接的其他节点中有重合(1,2,3,4)，因此节点 5 和 6 在特征空间内的表示也会十分接近。

在二阶相似度中，对于任意顶点 $v \in V$，算法中维护两个向量，一个是该节点本身的表示向量 $\boldsymbol{u} \in \mathbb{R}^d$，另一个是该节点作为其他节点的邻居时的表示向量 $\boldsymbol{u}' \in \mathbb{R}^d$，即作为其他节点的上下文。对于任意一对有向边 (v_i, v_j)，我们可以定义在给定一个节点 v_i 的条件下，产生邻居节点 v_j 的概率为

$$P_2(v_j \mid v_i) = \frac{\exp(\boldsymbol{u}_j'^{\mathrm{T}} \cdot \boldsymbol{u}_i)}{\sum_{k=1}^{|V|} \exp(\boldsymbol{u}_k'^{\mathrm{T}} \cdot \boldsymbol{u}_i)} \tag{3.28}$$

其中，$\boldsymbol{u}_i$ 为给定节点 v_i 对应的表征向量，此时观测其邻居为 v_j 的概率，$\boldsymbol{u}_j'$ 为顶点 v_j 充当邻居角色的向量表征。式(3.28)实际上定义了 $P_2(\cdot \mid v_i)$ 对所有上下文的条件分布。为了能够保留二阶相似度，需要使上下文的条件概率分布和其经验分布 $\hat{P}_2(\cdot \mid v_i)$ 之间的距离尽可能的小，其经验分布为

$$\hat{P}_2(v_j \mid v_i) = \frac{W_{ij}}{\sum_{k\in V} w_{ik}}$$

二阶相似度的目标是尽可能让两个节点的邻居节点集合，所以其优化目标为

$$O_2 = \sum_{i\in V} \lambda_i d(\hat{P}_2(\cdot \mid v_i), P_2(\cdot \mid v_i)) \tag{3.29}$$

其中 $d(\cdot)$ 代表两个分布的距离，同样使用 KL 散度来进行度量，另外引入了控制节点重要性的因子 λ_i，这里将 λ_i 定义为顶点的出度 d_i，则二阶相似度的目标函数可以优化为

$$O_2 = -\sum_{(i,j)\in E} \omega_{ij} \ln p_2(v_j \mid v_i) \tag{3.30}$$

之后通过合并一阶、二阶相似度的优化目标完成 LINE 的模型优化。

在计算二阶相似度中的条件概率 $p_2(v_j \mid v_i)$ 时，需要对所有顶点进行计算，这样的计算成本是非常高的，为了解决这个问题，原论文[①]中提出了负采样的方法，通过使用一些噪声分布进行负样本采样，于是可以将对于每一条边 (i,j) 的优化目标转化为

---

① Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th international conference on world wide web. 2015: 1067-1077.

$$\ln\sigma(\boldsymbol{u}'^{\mathrm{T}}_j \cdot \boldsymbol{u}_i) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)}[\ln\sigma(-\boldsymbol{u}'^{\mathrm{T}}_n \cdot \boldsymbol{u}_i)] \tag{3.31}$$

其中，激活函数为 $\sigma(x)=1/(1+\exp(-x))$，第一项为对一阶相似度进行建模，第二项是对从噪声分布中提取的负样本进行二阶相似度建模，K 为负样本的个数。

同时，模型中还存在着另一个问题，在 O_1 和 O_2 的优化函数中，ln 前有一个权重参数 w_{ij}，在使用梯度下降优化参数时，w_{ij} 会直接与梯度相乘，这时如果 w_{ij} 的方差过大，就会很难选择一个合适的学习率，如果学习率选择过大，则较大的 w_{ij} 可能出现梯度爆炸，而如果学习速率选择过小，则较小的 w_{ij} 可能出现梯度过小。

对于上述问题，一种最简单的方法是将带权边拆成等权边，如果所有边的权相同，那么选择一个合适的学习率就会比较方便，举例来说，这种方法可以将一条权重为 w 的边拆成 w 条权重为 1 的边。

3.4　基于深度学习的图表示学习

本节将重点介绍基于深度学习的图表示学习。随着时代的发展和算力的提升，深度学习也再次成为了热门的研究内容，利用深层的神经网络对图结构进行嵌入表征也成为了图研究学习中的一类不可忽视的重要方法。在本节内容中，我们将以 SDNE 为例，向读者介绍深度学习是如何被运用到图表示学习中的。

结构深层网络嵌入(Structural Deep Network Embedding，SDNE)是 Daixin Wang 等人[①]于 2016 年提出的一种图表示学习方法，同时也是第一种将深度学习应用于网络表征学习的方法。在 SDNE 中，使用了一个自动编码器优化图中节点的一阶、二阶相似度，采用一阶相似度学习局域网络结构，采用二阶相似度来学习全局的相似度，这使得该方法所获得的表征向量能同时保留图的局部和全局结构。

3.4.1　局域相似度和全局相似度

假设有图 $G(V,E)$，其中 $V=\{v_1,v_2,\cdots,v_n\}$ 表示图中的 n 个节点，$E=\{e_{i,j}\}_{i,j=1}^{n}$ 表示边，其中每一条边都有其权重 $s_{i,j}$，若两个节点间没有边，则其权重为 0。一阶相似度用来表示图中成对节点之间的相似度，用来学习局域图结构。具体来说，两个节点 i、j 之间的相似度即为两点之间边的权重 $s_{i,j}$，若之间没有连接，则相似度为 0。一阶相似度可以直接表示节点对之间的相似度，然而真实环境下的信息网络往往存在大量的信息缺失，导致存在许多一阶相似度为 0 的节点，而它们相似度也很高。例如，构建的人际档案信息中，如果两个人存在很多共同的朋友，那么这两个人很有可能也认识。为此，基于节点邻居的相似度，提出了二阶相似度，用来表征全局相似度。二阶相似度是两个节点的邻居节点之间的相似度。具体来说，若两个节点的邻居节点集合之间有许多重叠，则认为两个节点之间拥有很高的二阶相似度。假设 $N_i=\{s_{i,1},\cdots,s_{i,|V|}\}$ 表示节点 v_i 的一阶邻居，$N_j=\{s_{j,1},\cdots,s_{j,|V|}\}$ 表示

① Wang D, Cui P, Zhu W. Structural deep network embedding[C]//Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016: 1225-1234.

节点 v_j 的一阶邻居，则节点 i 和 j 的二阶相似度表示为 N_i 与 N_j 之间的相似度。引入二阶相似度后，更能表征网络结构，使得稀疏网络更具有健壮性。

3.4.2 SDNE 算法结构图

图 3-17 是 SDNE 算法的结构，SDNE 分为两部分来实现局域相似和全局相似。局域相似度采用拉普拉斯映射的方法，在嵌入空间中保留原图的节点的相对距离。SDNE 采用节点邻居间的相似度来表示全局相似度。对于节点 v_i 对应的邻居信息为邻接矩阵 $\boldsymbol{S}\in\mathbb{R}^{|V|\times|V|}$ 的第 i 行，表示为 $s_i\in\mathbb{R}^{|V|}$。同理，对于节点 v_j，邻居信息表示为 s_j。s_i 和 s_j 已经蕴含了两个节点间的二阶相似度，这里并未直接给出二者相似度的大小，而是希望在嵌入向量 $\boldsymbol{y}^{(K)}\in\mathbb{R}^d$ 中保存节点的邻接特性。

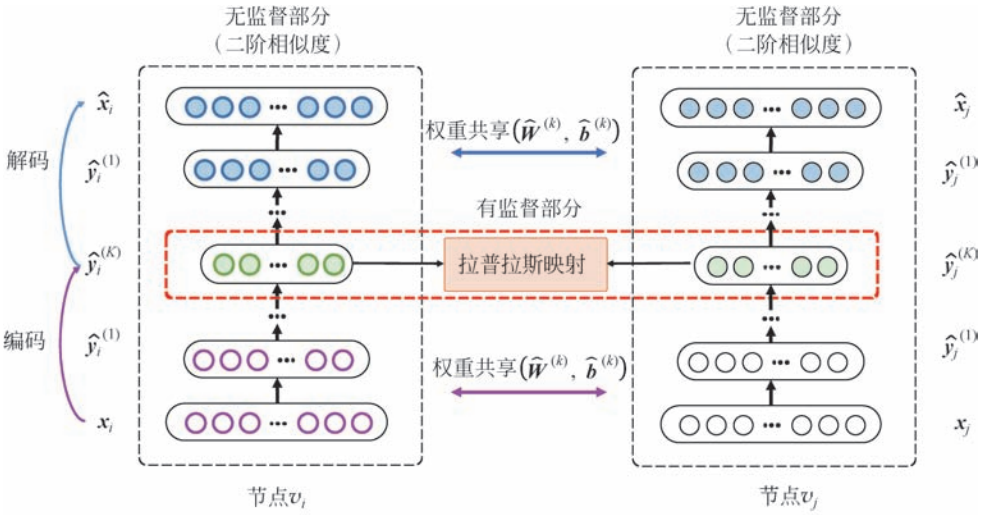


图 3-17 SDNE 的模型框架图

SDNE 使用了自编码器对图的邻接矩阵 $\boldsymbol{S}$ 进行解码重构，通过这样的重构过程能够使得结构相似的顶点具有相似的表示向量。图 3-17 中的自编码器的结构包含一个编码(Encoder)部分和一个解码(Decoder)部分。编码部分的作用是将节点 v_i 对应的输入向量 $\boldsymbol{x}_i=s_i\in\mathbb{R}^{|V|}$，压缩到一个低维向量 $\boldsymbol{y}_i^{(K)}\in\mathbb{R}^d$，解码部分的作用是将压缩后的向量还原到原始输入向量空间 $\hat{\boldsymbol{x}}_i\in\mathbb{R}^{|V|}$，$d\ll|V|$。此处采用 K 层神经网络进行编码，公式如下：

$$\boldsymbol{y}_i^{(1)}=\sigma(\boldsymbol{W}^{(1)}x_i+b^{(1)}) \tag{3.32}$$

$$\boldsymbol{y}_i^{(k)}=\sigma(\boldsymbol{W}^{(k)}y_i^{(k-1)}+b^{(k)}),\quad k=2,3,\cdots,K \tag{3.33}$$

其中，$\sigma(\cdot)$为 Sigmoid 激活函数。经过编码后得到低维度向量 $\boldsymbol{y}_i^{(K)}\in\mathbb{R}^d$，然后经过编码的逆过程解码得到 $\hat{\boldsymbol{x}}_i\in\mathbb{R}^{|V|}$。$\boldsymbol{W}^{(k)}$表示编码第 k 层的权重矩阵。一般而言，根据解码还原的 $\hat{\boldsymbol{x}}_i$ 与输入 $\boldsymbol{x}_i$，可以构建如下损失函数：

$$L=\sum_{i=1}^{|V|}\|\hat{\boldsymbol{x}}_i-\boldsymbol{x}_i\|_2^2 \tag{3.34}$$

由于数据的稀疏性，在邻接矩阵中，存在着很多 0 值，这使得自编码器在学习过程中学

习了过多的 0，直接导致模型的效果不佳。考虑到由于数据的稀疏性，图的邻接矩阵可能会缺少很多实际上潜在的连接。以推荐系统中用户—商品的购买二部图为例，用户没有购买商品（即这两个节点之间没有边）的原因可能是用户不喜欢该商品，也有可能是用户没有浏览到该商品，因此用户对此商品是有潜在购买可能的。SDNE 在传统自编码器的损失函数上进行了一些改动：

$$L_{2\text{nd}}=\sum_{i=1}^{|V|}\|(\hat{\boldsymbol{x}}_i-\boldsymbol{x}_i)\odot b_i\|_2^2 \tag{3.35}$$

其中，$b_i\in\mathbb{R}^{|V|}$，如果 $s_{i,j}=0$，$b_{i,j}=1$，否则 $b_{i,j}=\beta>1$，$\odot$ 表示哈达玛积（Hadamard Product）。这样设置可以使模型在将一个非 0 值的节点预测为 0 时受到更多惩罚。

局域相似度部分使用了 3.3.2 节介绍的拉普拉斯特征映射方法，让图中相邻的两个顶点对应的嵌入向量在嵌入空间接近，提取图中的局部特征。在 SDNE 中，这种想法被融入深度模型中，使得有连接的两个点被映射到附近空间中。损失函数定义如下

$$L_{1\text{st}}=\sum_{i}^{|V|}\sum_{j}^{|V|}s_{i,j}\|\boldsymbol{y}_i^{(K)}-\boldsymbol{y}_j^{(K)}\|_2^2=2\text{trace}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{Y}) \tag{3.36}$$

其中，$\boldsymbol{Y}=[\boldsymbol{y}_1^{(K)};\boldsymbol{y}_2^{(K)};\cdots;\boldsymbol{y}_{|V|}^{(K)}]$，除了这两个损失函数之外，SDNE 还增加了一个 L2 正则化项来防止过拟合：

$$L_{\text{reg}}=\frac{1}{2}\sum_{k=1}^{|V|}(\|\hat{\boldsymbol{W}}^{(k)}\|_F^2+\|\boldsymbol{W}^{(k)}\|_F^2) \tag{3.37}$$

其中，$\boldsymbol{W}^{(k)}$，$\hat{\boldsymbol{W}}^{(k)}$ 分别表示编码和解码层中第 k 层的权重矩阵。于是可以得到总的损失函数，即

$$L_{\text{mix}}=L_{1\text{st}}+\alpha L_{2\text{nd}}+\beta L_{\text{reg}} \tag{3.38}$$

其中，α 和 β 是超参数。损失函数可以通过反向传播算法进行优化。

3.5　异质图表示学习

上述算法虽然可以用于同构网络（Homogeneous Networks），即仅适合只包含单一顶点类型和边类型的网络表示学习，但并不能很好地用于包含多种顶点类型和边类型的复杂关系网络。而在现实世界中，节点类型或者关系类型是多种多样的。如图 3-18 所示的异质图中，存在机构、作者、论文和期刊 4 种节点，以及作者与学术机构的归属关系、作者与论文的发表关系、论文与学术期刊的归属关系。针对同构网络设计的模型很多都没法应用于异质网络，例如，对于一个学术网络而言，如何根据上下文信息表征不同类型的节点？为此需要给异质图来设计图表示学习算法。

给定异质图 $G=(V,E,X)$，其中，$V=\{v_1,v_2,\cdots,v_{|V|}\}$ 为顶点集合，对应的点类型集合为 T_v，顶点映射到对应类型的函数为 ϕ_v，边集合为 $E=\{e_1,e_2,\cdots,e_{|E|}\}$，对应的边类型集合为 T_e，边映射到对应类型的函数为 ϕ_e。在具体实践中，为了分辨异质的特点，引入了元路径（meta-path）的概念。元路径是在异质图 G 上按照元路径模式 ψ：$A_1\xrightarrow{R_1}A_2\xrightarrow{R_2}A_3\cdots\xrightarrow{R_{K-1}}A_K$ 来游走产生路径，其中，类型 $A_i\in T_v$，关系 $R_i\in T_e$。复合关系组合为可表示为 R_1。

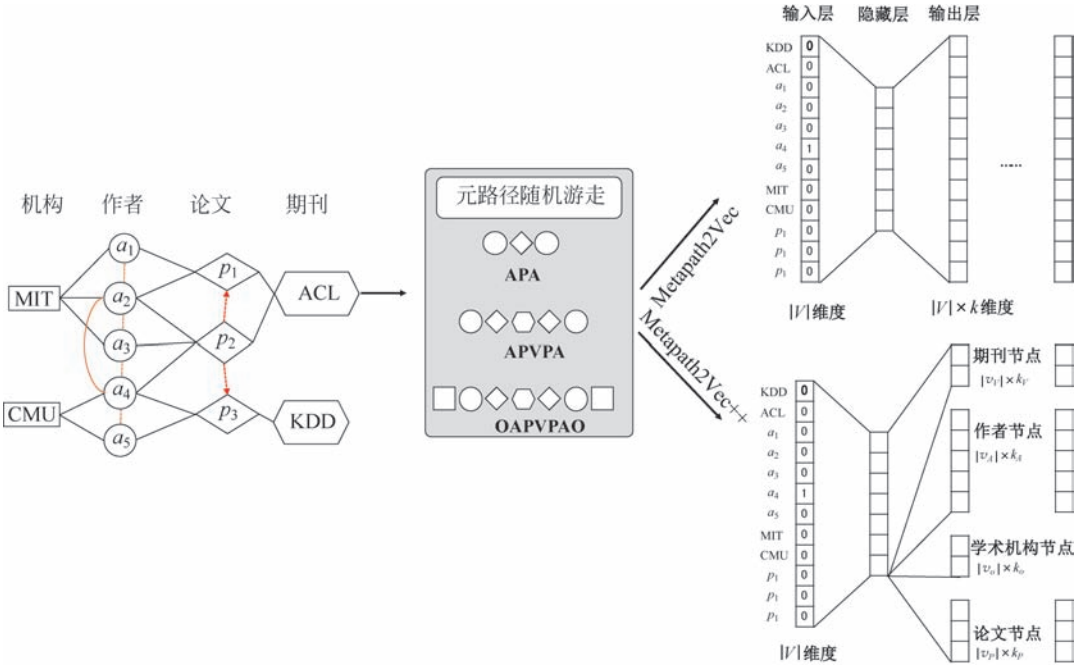


图 3-18　异质图学术网络的 Metapath2Vec 与 Metapath2Vec++算法结构图

$R_2 \circ \cdots \circ R_{K-1}$，其中$\circ$表示关系运算符，$R_i$ 表示第 $K-1$ 个节点与第 K 个节点之间的关系。

先介绍基于元路径的随机游走，元路径的模式 ψ 被用来限制随机游走的决策，即每一步游走需要按照模式 ψ 设计的节点类型和边类型进行游走，给定类型为 A_i 节点 v_i^{ψ}，跳转概率为

$$P(v_{i+1}^{\psi} \mid v_i^{\psi}) = \begin{cases} \dfrac{1}{|N_{i+1}^{R_i}(v_i^{\psi})|}, & v_{i+1}^{\psi} \in N_{i+1}^{R_i}(v_i^{\psi}) \\ 0, & \text{其他} \end{cases} \tag{3.39}$$

其中，$N_{i+1}^{R_i}(v_i^{\psi})$表示节点 v_i^{ψ} 的邻居中满足边关系为 R_i 且节点类型为 A_{i+1} 的邻居顶点集合。除此之外，元路径通常以一种对称的方式使用，如图 3-18 中的元路径是对称的，即顶点 A_1 的类型和 A_K 的类型相同。根据对称性的元路径，$P(v^{i+1} \mid v_t^i) = p(v^{i+1} \mid v_1^i)$，如果 $t = K$。可以看出，元路径的受限随机采样过程中，对顶点类型与关系都具有限定作用。基于元路径的随机游走可保证不同类型顶点之间的语义关系可以适当地融入 Skip-Gram 模型中。

基于元路径的表示学习有两种方法，分别是 Metapath2Vec 和 Metapath2Vec++。在 Metapath2Vec 中使用基于元路径的随机游走来构建每个顶点的异质邻域，然后使用 Skip-Gram 模型，通过在顶点 v 的领域 $N_t(v)$，$t \in T_v$ 最大化保留一个异质网络的结构和语义信息的似然：

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_v} \sum_{c_t \in N_t(v)} \ln P(c_t \mid v; \theta) \tag{3.40}$$

其中，c_t 为节点类型为 R_i 的邻居集合 $N^{R_i}(v)$，θ 为待学习参数，似然概率为

$$P(c_t \mid v; \theta) = \frac{e^{\boldsymbol{X}_{ct} \cdot \boldsymbol{X}_v}}{\sum_{u \in V} e^{\boldsymbol{X}_u \cdot \boldsymbol{X}_v}} \tag{3.41}$$

图 3-18 中，沿着元路径 OAPVPAO 的节点 a_4 的邻居包括作者(Author)类型节点$\{a_2, a_3, a_5\}$、期刊(Venue)类型节点{ACL，KDD}、机构(Org)类型节点{CMU}、论文(Paper)类型节点$\{p_2, p_3\}$，k 为路径中各种类型邻居的个数之和，即 $k=k_V+k_A+k_O+k_P$。为了加速优化过程，可以引入负采样，改进为

$$\arg\max_{\theta}\sum_{v\in V}\sum_{t\in T_v}\sum_{c_t\in N_t(v)}\ln\sigma(\boldsymbol{X}_{c_t}\cdot\boldsymbol{X}_v)+\sum_{m=1}^{M}\mathbb{E}_{u^m\sim P(u)}[\ln\sigma(-\boldsymbol{X}_{u^m}\cdot\boldsymbol{X}_v)] \tag{3.42}$$

$P(u)$是负采样的节点 u^m 在 M 次采样中的预定义分布。

Metapath2Vec 在为每个顶点构建领域时，通过元路径来指导随机游走过程向指定类型的顶点进行有偏游走。但是在 Softmax 环节中，没有分辨顶点的类型，而是将所有的顶点视作同一类型的顶点，如式(3.41)。也就是说，Metapath2Vec 在负采样环节采样的负样本并没有考虑顶点的类型。

在 Metapath2Vec++中，Softmax 函数根据不同类型的顶点的上下文 c_t 进行归一化。也就是说 $P(c_t|v;\theta)$根据固定类型的顶点进行调整，即

$$P(c_t\mid v;\theta)=\frac{\mathrm{e}^{\boldsymbol{X}_{c_t}\cdot\boldsymbol{X}_v}}{\sum_{u_t\in N_t}\mathrm{e}^{\boldsymbol{X}_{u_t}\cdot\boldsymbol{X}_v}} \tag{3.43}$$

其中，N_t 是网络中 t 类型顶点集合。Metapath2Vec++给 Skip-Gram 模型的每种类型的领域指定特定的集合。如图 3-18 所示。最终得到如下的目标函数：

$$O(\boldsymbol{X})=\ln\sigma(\boldsymbol{X}_{c_t}\cdot\boldsymbol{X}_v)+\sum_{m=1}^{M}\mathbb{E}_{u_t^m\sim P_t(u_t)}[\ln\sigma(-\boldsymbol{X}_{u_t^m}\cdot\boldsymbol{X}_v)] \tag{3.44}$$

3.6　本章小结

本章介绍了以拉普拉斯特征映射为代表的矩阵分解方法，DeepWalk、Node2Vec 和 LINE 的随机游走算法，以及基于深度学习的 SNDE 算法，最后介绍了 Metapath2Vec 和 Metapath2Vec++的异质图表示学习算法。

拉普拉斯特征映射算法依据原有高维空间数据的相对位置来构图，优化的目标是在低维空间中尽可能使原空间中相近的点在降维后的目标空间中也尽可能相近，从而得到节点的低维度表示。随机游走算法是一种基于邻域相似假设的算法，受启发于 Word2Vec，来学习节点的向量表示。DeepWalk 通过截断深度优先随机游走获取游走序列，Node2Vec 的随机游走方式则综合了深度优先和广度优先游走，二者都是通过节点的局域网络特征来学习节点的低维表示。LINE 也是一种基于邻域相似假设的方法，可以看作是一种使用广度优先构造邻域的算法。LINE 算法同时考虑网络的局域结构相似度(一阶相似度)和节点邻域的全局相似度(二阶相似度)，然而这两种相似度是分开考虑的，只是一种简单的组合。SDNE 算法是一种深度学习算法，利用深度自编码器学习图中节点的表征向量，结合一阶和二阶相似度进行联合训练。相对于同质图，异质图的存在更为广泛，在 Metapth2Vec 和 Metapath2Vec++中采用元路径指导随机游走的邻居节点的选择，即为有偏置的随机游走构建邻居集合，在 Skip-Gram 学习阶段，Metapath2Vec 与 DeepWalk 和 Node2Vec 一样不考虑节点类型，而 Metapath2Vec++则对不同类型节点单独处理。