第 3 章

图神经网络

3.1 图数据简介

图数据通过节点(实体)和边(实体之间的关系)的形式,能够表示各种类型的信息和关系网络,例如社交网络、分子结构等,然而,图是一种极其强大和普遍的数据表示方法,接下来将展示两种读者可能认为不可以用图来建模的数据类型:图像和文本。

通常认为图像是具有图像通道的矩形网格,并将它们表示为数组(例如 224×224×3 的矩阵)。另一种思考图像的方式是具有规则结构的图,以 224×224×3 的图片举例,其中每个像素代表一个节点,并通过边缘与相邻的像素相连。每个非边界像素正好有 8 个邻居,每个节点存储的信息是一个代表像素 RGB 值的三维向量。

通过邻接矩阵来可视化一幅图形的连通性的方法。首先对节点进行排序,在这种情况下,在一个简单的 5×5 的笑脸图像中,每个节点有 25 像素,如果两个节点共享一条边,就用一个条目填充一个 $n_{\text{nodes}} \times n_{\text{nodes}}$ 的矩阵。需要注意,如图 3-1 所示,这 3 种表示方法都是对同一份数据(这张笑脸图像)的不同看法。

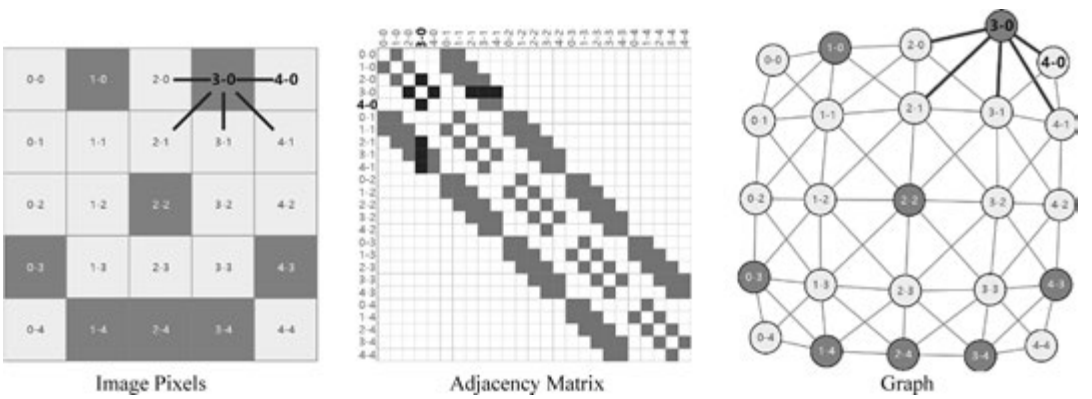


图 3-1 图像的 3 种表示(像素矩阵/邻接矩阵/图)

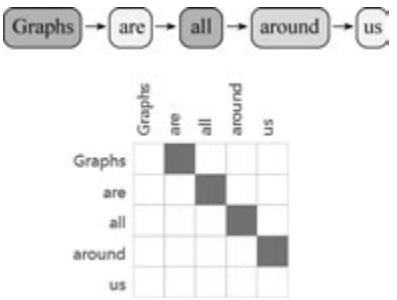


图 3-2　文本表示成邻接矩阵

至于文本数据，可以通过为每个字符、词关联一个索引来数字化文本，并将文本表示为这些索引的一个序列。这就形成了一个简单的有向图，其中每个字符或索引都是一个节点，并通过一条边与后面的节点相连，如图 3-2 所示。

当然，在实践中，GNN 通常不是处理文本和图像的最佳算法。因为文本和图像都是规则数据，例如，图像的像素只与周边像素相关，文本中每个词只与前一个词和后一个词相连接，从邻接矩阵的表示上就可以看出明显的规律。针对这些规律明显的数据，有更好的算法可以处理它们，例如卷积神经网络和循环神经网络等。这两个例子仅是为了展示图数据强大的表征能力。

GNN 更擅长处理不规则的数据，更专业的说法是异质化的数据（Heterogeneously Structure），例如社交网络、分子结构等，在这些例子中，每个节点的邻居数量是可变的（与图像和文本的固定邻居大小相反）。这种数据除了可以用图来表达外，很难用其他方式来表达，例如分子数据，分子是物质的组成部分，由三维空间中的原子和电子构成。所有的粒子都是相互作用的，但是当一对原子彼此卡在一个稳定的距离上时，我们说它们有一个共价键。不同的原子对和键有不同的距离（例如单键、双键）。把这个三维物体描述成一幅图，其中节点是原子、边是共价键，这是一个非常方便和常见的抽象概念，如图 3-3 所示。

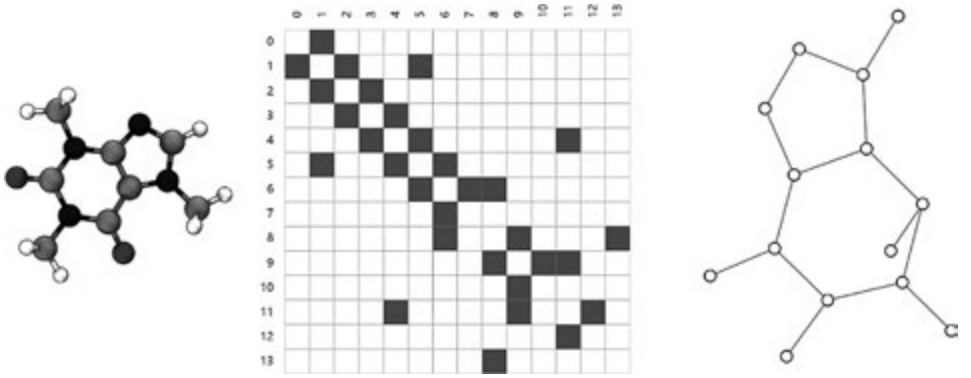


图 3-3　分子结构的 3 种表示（图像/邻接矩阵/图）

此外，社交网络也可以看作一种图。社会网络是研究人、机构和组织的集体行为模式的工具。可以通过将个人建模为节点，将他们的关系建模为边来建立一幅代表人群的图，如图 3-4 所示的人物关系图。

其实，生活中能表示成图结构的数据有很多种，其真实应用也远不止如此。互联网和 Web 搜索、推荐系统、知识图谱、网络安全、物流和供应链管理、交通网络优化等应用都涉及图数据处理。图的灵活性决定了它在数据表示中的重要地位，对图的研究是必要且意义重大的。下面详细展开讲解的图神经网络章节与几何图神经网络章节便是基于神经网络算法

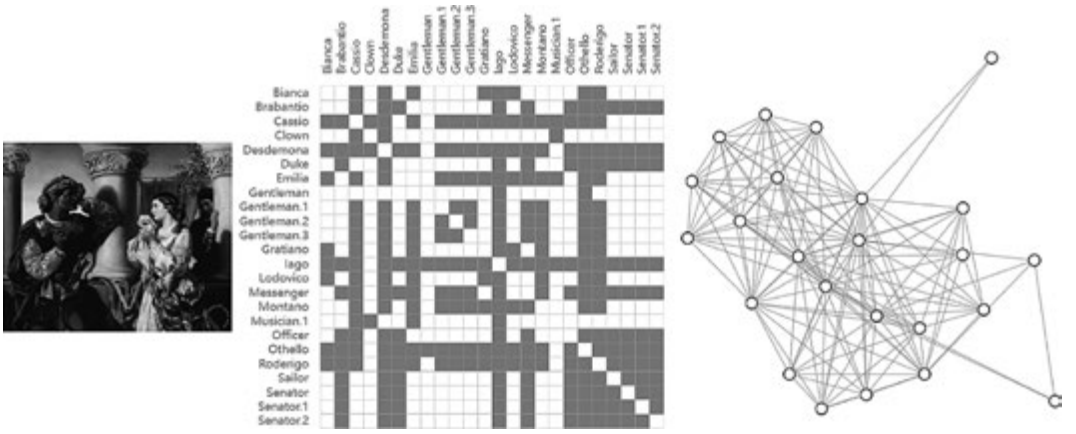


图 3-4　社交网络的 3 种表示(图像/邻接矩阵/图)

专门针对图数据设计的深度学习模型。

3.1.1　图数据的任务类型

图数据的任务类型主要有 4 种：全图级别的预测、节点级别的预测、边缘级别的预测和生成类任务。在全图层面的任务中，主要对整幅图的单一属性进行预测。对于节点层面的任务，主要预测图中每个节点的一些属性。对于边缘层面的任务，要预测图中边缘的属性或存在，而生成类的任务，主要根据已有的图数据生成一些类似的图结构，进行数据探索和发现。

1. Graph-level task

在全图层面的任务中，目标是预测整幅图的属性，例如，对于一个以图表示的分子，可能想预测该分子的气味，或者预测它是否具有某些特殊的药理性质可以用来治疗某些疾病，图 3-5 所示的是生物化学领域的常见应用。

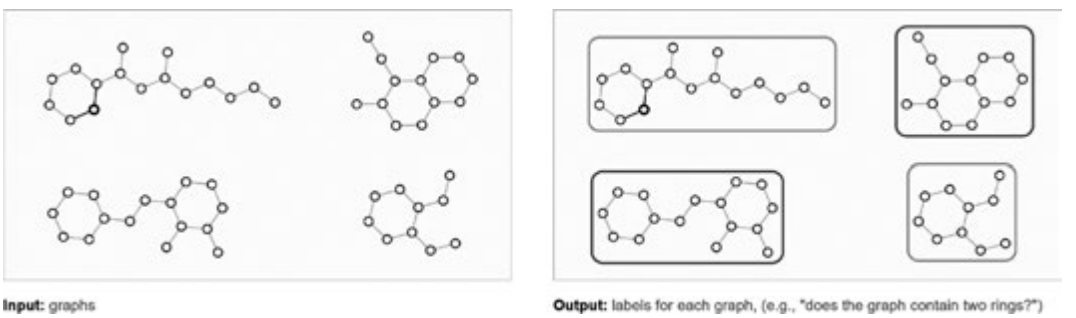


图 3-5　Graph-level task

图级别任务在其他领域也有广泛的应用，例如，在材料科学中，利用图级别任务来预测新材料的性质或发现新材料，通过分析原子或分子间的联系来寻找具有特定特性的材料。在物理领域中，图模型用于粒子物理、量子计算和宇宙学的研究，例如通过粒子碰撞事件中

的粒子相互作用图来识别新的物理现象或计算量子系统的性质。图数据也被广泛地用于研究社会网络、经济系统、交通领域等。通过分析社交网络图的结构，研究者可以识别社群结构、意见领袖，或预测信息传播模式和社会行为趋势。在经济领域，图模型可用于分析金融网络的稳定性、预测市场风险或探究企业间的合作与竞争关系。在交通领域中，图级别任务对于优化交通流、规划城市交通网络和提高物流效率至关重要。利用交通图，可以分析和优化道路、铁路、航空和航海网络，实现更有效的路径规划、拥堵管理和事故响应等。

2. Node-level task

节点级别任务关注的是预测图中每个节点的身份或角色。节点级别预测问题的一个典型例子是 Zach 的空手道俱乐部，如图 3-6 所示。该数据集是一个单一的社会网络图，问题是两名教员决裂了，学生必须选择其中一人效忠。正如故事所言，Hi 先生（教练）和 John H（管理员）之间的争执在空手道俱乐部中造成了分裂。节点代表空手道练习者个人，边则代表这些成员在空手道之外的互动关系。预测问题是对一个给定的成员是否会效忠于 Hi 先生或 John H 进行分类。在这种情况下，两个节点之间的距离表示两人之间关系的远近。

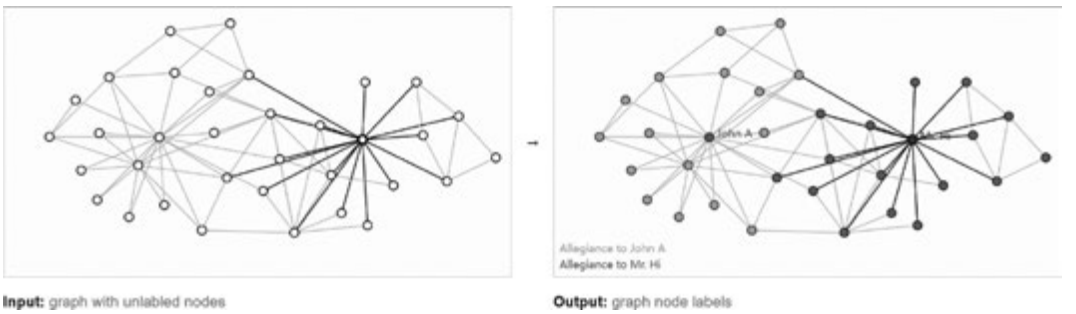


图 3-6　Node-level task

另一个例子是近期非常火热的蛋白质三维结构预测模型：AlphaFold，如图 3-7 所示。在这个例子里，我们把蛋白质中的每个氨基酸看作节点，如果两个氨基酸相连，则连一条边。由于蛋白质是会折叠的，所以模型 AlphaFold 的目的是预测每个节点在三维空间中的位置，解决了人类在这个领域 50 年没有解决的问题：蛋白质三维结构预测。

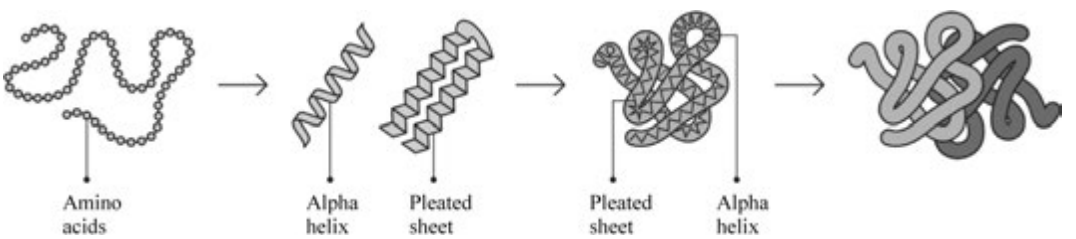


图 3-7　AlphaFold

在生物化学中，节点级别任务还可用于预测蛋白质功能，通过分析蛋白质相互作用网络中的节点（蛋白质）来预测其功能分类或疾病关联性。这对于理解生物过程和开发新药具有

重要意义。在材料科学中，节点级别分析可以帮助预测材料中原子或分子的特性，例如电荷、磁性或化学活性，这对于设计新材料和改善现有材料性能至关重要。在物理领域，特别是在粒子物理和凝聚态物理中，节点级别任务可以用于识别和分类粒子类型，或预测原子在特定物理环境下的行为等。

3. Edge-level task

边缘级别任务的一个例子是在图像场景理解中。除了可以识别图像中的物体，深度学习模型还可以用来预测它们之间的关系。可以将其表述为边缘级别分类：给定代表图像中物体的节点，希望预测这些节点中哪些节点共享一条边缘，或者该边缘的价值是什么，如图 3-8 所示，原始图像(左上角)已经被分割成 5 个实体：两名拳手、裁判、观众和垫子。

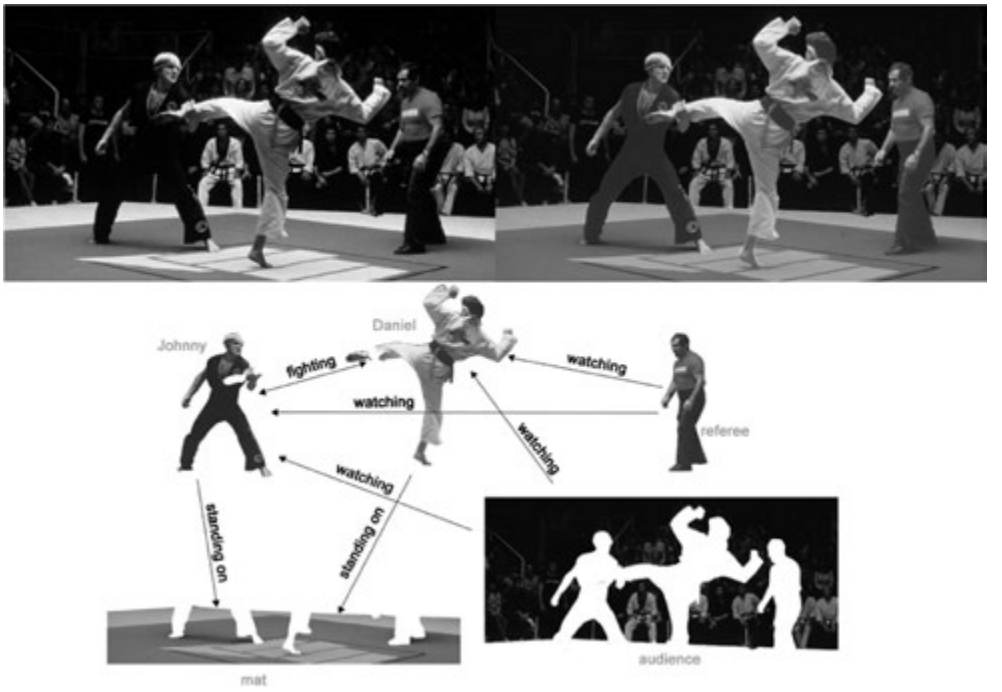
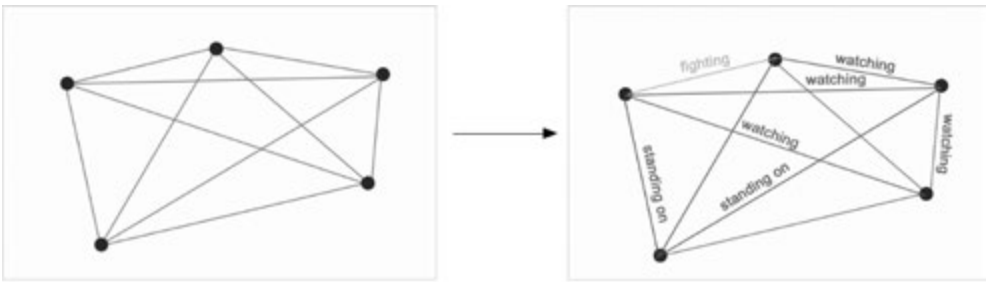


图 3-8 Edge-level task(1)

从先前的视觉场景中建立的初始图如图 3-9(a)所示，此图的一个可能的边缘标签如图 3-9(b)所示。

另一个例子是蛋白质相互作用网络的边缘级别预测问题，例如，将药物分子和人体蛋白质作为图的节点，如果药物分子和蛋白质直接存在反应，则连一条边，如图 3-10 所示。根据已有的图结构，可以预测药物辛伐他汀和环丙沙星一起服用时分解肌肉组织的可能性有多大。

此外，在材料科学领域，通过分析原子或分子间的相互作用，边缘级别任务可以帮助预测材料中的化学键类型或强度，这对于设计具有特定性质的新材料非常重要。在物理中，边缘级别任务可以应用于预测粒子间的相互作用，例如量子纠缠或力的作用，这对于理解基本



(a) 初始图　　(b) 边缘标签

图 3-9　Edge-level task(2)

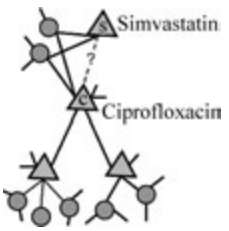


图 3-10　Edge-level task(3)

物理规律和发展新的物理理论有重要意义。在社会科学中，边缘级别分析可用于探索人际关系的性质，例如友谊、合作或信任关系。在网络分析中，这有助于揭示社会网络的结构和演化，以及信息、影响力和资源的流动路径。在交通领域，边缘级别任务通常用于预测路段的交通流量或拥堵情况，以及分析运输网络中的关键连接路径。这对于交通规划、拥堵管理和提高交通系统的效率具有重要意义。

4. Generation tasks

除了上述的预测类任务以外，图神经网络也可以用于生成类的任务，例如，在生物化学中，在药物发现和生物学研究中，图生成模型可以用于设计新的分子结构。通过学习现有分子的图表示，这些模型能够生成具有预期属性(例如药效、稳定性)的新分子结构，加速新药的开发。在材料科学中，利用图生成技术，研究人员可以设计出具有特定特性(例如超高强度、导电性)的新材料。通过模拟原子或分子间的连接方式，可以探索未知材料的可能结构。在社会科学中，通过生成社会网络的图结构，研究人员可以模拟社会群体的形成、信息传播路径及观点动态，这对于理解社会现象和行为模式有重要意义。在交通规划中，图生成模型可以用于设计和优化城市交通网络，通过生成不同的路网结构来评估交通流、拥堵情况和事故风险，从而指导城市规划和交通管理。

图生成任务通过算法自动构造图结构，不仅能帮助我们模拟和理解现有的网络系统，还能够创造出全新的结构和模式，促进创新和发现。这些任务依赖于复杂的算法和模型，例如图神经网络和生成对抗网，它们通过学习现实世界数据中的规律和关系，生成具有特定特性和功能的图。

5. Transductive tasks/Inductive tasks

上述 4 种任务类型的分类方式依赖于应用问题本身的性质。如果从模型训练和使用阶段的数据集是否一致的角度来分类，又可以被分为 Transductive 任务和 Inductive 任务两种类型。这两种任务的主要区别在于模型训练阶段与模型使用阶段的图数据是否相同。具体来讲，Transductive 任务是指：训练阶段与训练后的使用阶段都基于同样的图结构，如图 3-11 所示。

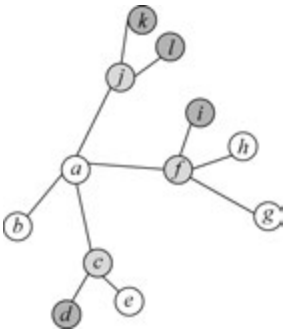


图 3-11 Transductive 任务

在整个图数据中，假设 k、i、l、d 节点是带有真实标签的节点，而其他节点不带标签，现在的任务是预测不带标签的节点属性，也就是说，要学习不带标签节点向量的合理表示。通过 GNN 的处理，每个节点的向量信息都可以进行交流和更新，在进行损失函数计算时，只需计算带有标签的节点损失。因为一个合理的直觉是，想要 k、i、l、d 节点经学习得到一个正确的向量表示，那么它们邻居节点的向量表示必须也要学好，想要邻居节点的向量表示学好，邻居的邻居节点也要学到一个好的向量表示，进而推广到全图，所以，基于图数据的这个特点，即使只训练了 k、i、l、d 节点的损失，整幅图中所有的节点都会得到学习。简单来说，Transductive 任务在训练阶段的目标是根据有限的带标签的节点信息，对全图信息进行 Embedding 向量的优化，优化后的向量可进一步进行一些预测或分类任务。这种任务在生活中的应用场景还是很多的，尤其是在一些体量比较大的图数据中，已知的节点信息一般很难涵盖全图，此时便需要通过这些已知信息来预测其他的未知节点的信息。

另一方面，Inductive 任务是指：训练阶段与使用阶段需要处理的图结构不同。举一个生化领域的经典问题：训练阶段处理的是一些已知标签的药物分子活性，测试阶段要预测其他不带标签的药物分子活性。在这个问题中，训练阶段和测试阶段分子图数据的结构是明显不同的，Inductive 便代指这一类的任务。

下面将介绍最经典的图神经网络消息传递神经网络（Message Passing Neural Network，MPNN）及它的 3 种经典的变体：Graph Convolutional Network（GCN）、Graph Sample and Aggregate Network（GraphSAGE）和 Graph Attention Network（GAT），其中，GCN 更适合处理 Transductive 任务，而 MPNN、GraphSAGE 和 GAT 对于两种任务都可以进行处理。

3.1.2 图数据的 Embedding

图数据的嵌入（Embedding）是将图结构中的节点、边或整幅图转换为低维、连续、密集的向量表示的过程，这个过程也被称为向量化。这些向量表示（也称为嵌入向量）捕捉了图中的实体及其关系的本质特征，使图数据可以被用于各种机器学习和数据分析任务。图嵌入技术的目的是保留原始图结构的信息，例如节点间的邻接关系、路径长度和网络拓扑特性，同时将这些信息压缩到一个低维空间中。

1. 节点的向量化

独热（One-Hot）编码是节点向量化的一个基础方法。简单来说就是用一个长度为节点数量的向量来表示节点信息，这个向量的绝大部分元素都是零，只有一个位置是 1，所以称为独热编码，如图 3-12 所示。

这种表示方式虽然可以保证每个节点对应不同的向量，但是有两个致命的缺点。

（1）图的稀疏性，向量中绝大部分元素是 0，而且随着图大小的增加，稀疏性会跟着增加。

（2）独热编码的形式不能抓取节点中的相关性，例如节点刘备和其他 5 个节点都相连，但这个关系在独热编码的形式上是体现不出来的。实际上，这 6 位猛将都有内在联系，例如刘备、关羽、张飞在桃园结义过；张飞、关羽、马超、赵云和黄忠同为五虎上将；黄忠比较年迈，而赵云、马超比较年轻；五虎上将的武力值都是顶配的；马超和黄忠的出身较好等，所以可以构建查找表矩阵，如图 3-13 所示。

图 3-12　独热编码

	结义	五虎	年龄	武力	出身
刘备	[1	0	0.60	0.70	0.3]
关羽	[1	1	0.55	0.99	0.3]
张飞	[1	1	0.50	0.89	0.3]
赵云	[0	1	0.40	0.92	0.4]
马超	[0	1	0.46	0.87	0.6]
黄忠	[0	1	0.72	0.81	0.7]

图 3-13　查找表

通过上述形式，我们把 One-Hot 编码从稀疏态变成了密集态，并且让相互独立的向量变成了有内在联系的关系向量。上述这种将数据点转换为数值向量并使用矩阵表示的方法被称为查找表（Lookup Table）。这种方法通过为每个唯一的类别（例如单词、标签或符号）分配一个固定长度的向量来工作，使模型可以处理和分析这些数据，其中每行代表一个类别的向量表示。矩阵的大小由两个因素决定：一是唯一类别的数量，二是向量的维度。每个类别被分配一个唯一的索引，通过这个索引，可以在查找表中检索对应的向量。

Lookup Table 经常被应用于一些预测或分类任务。具体来讲，不同的任务一般对应不同的 Lookup Table，通过 Lookup Table 可以将当前任务数据中的每个样本转化成向量的形式，再送入一些机器学习模型中（例如神经网络、SVM 等）执行分类或预测任务。这其中的重点其实是如何得到合理的 Lookup Table。一般来讲可以有 3 种方式：自定义方法、有监督训练和无监督训练。

自定义方法最简单，可以根据特定的数据特征和任务需求设计自定义嵌入方法，例如图 3-13 所表述的形式就是一种自定义嵌入的方法。在后续的几何图卷积神经网络模型中，一种常见的自定义嵌入的方法是将笛卡儿坐标系下的空间表示经过径向模型和球谐函数处理得到新的向量表示，以此来帮助模型具备可以表述分子数据在几何空间对称性的能力，这种方法在处理具有明确几何或物理属性的数据时尤为有效，因为它能够精确地捕捉数据点之间的空间关系和结构特征，详见第 4 章。

在几何深度学习中，可以使用图神经网络算法，或几何图神经网络通过有监督训练生成

合理的 Lookup Table。具体来讲，这个过程大体上分为几个简单的步骤：首先准备数据，确保每个数据点都有明确的输入特征和期望输出（输入和对应的真值），例如如果数据是分子数据，每个输入节点的信息可能是原子的位置和初始化的特征向量，输出则是想预测的属性，例如分子的活性等，然后选择合适的 GNN 模型，并进行训练，这些模型将在后续章节进行详细介绍。在这个阶段，模型会不断地更新初始化的特征向量，尝试使用更新后的特征向量来预测正确的输出结果。训练完成后，用模型对所有可能的输入进行计算，模型计算得到的节点特征向量会被用来填充 Lookup Table。值得注意的是，关于送入 GNN 的输入信息中的初始化特征向量，初始化的方式有多种，既可以是 One-Hot 形式，也可以是上述的自定义方式。换句话说，可以先使用自定义方式初始化 Lookup Table 表格，再通过 GNNs 模型在特定任务上以有监督训练的方式来优化 Lookup Table，使其更适配该任务。

最后无监督建模是生成 Lookup Table 的常用方法，按数据类型可以分为序列和图两类，针对序列数据，即自然语言处理领域，生成 Embedding 常采用 word2vec 或类似算法（item2vec、doc2vec 等）。针对图数据，也就是本书讲解的几何深度学习领域，生成 Embedding 的算法称为 Graph Embedding，这类算法包括 deepwalk、node2vec、struc2vec 等，它们大多采用随机游走方式生成序列，下面以随机游走（Random Walk）为例进行介绍。

在介绍随机游走算法之前，一个合理的向量化应该首先考虑相似性，即在图中相互临近的节点经过向量化后得到的向量也应该是相似的。也就是说，希望向量化之后的节点向量点乘之后的值接近于原图中的节点相似度，公式如下所示。

$$\text{similarity}(u,v)=\boldsymbol{z}_v^{\mathrm{T}}\boldsymbol{z}_u \tag{3-1}$$

在随机游走算法中，当给定一幅图和一个起始节点 u，然后按照一定概率随机选择一个邻居节点，走到该处后再随机选择一个邻居，重复 length 次，到最终的终止节点 v。length 是一个超参数，是指随机游走的长度，如图 3-14 所示。

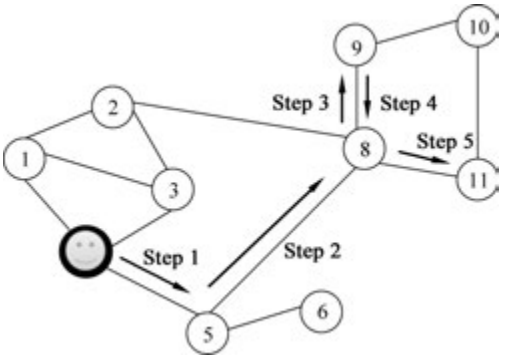


图 3-14 Random Walk

在多次进行随机游走后，随机游走从起始节点 u 到终止节点 v 的次数，除以随机游走的总次数得到的概率值，实际上就可以用来表示相似度。也就是说，从 u 节点到 v 节点的概率值，应该正比于 u 节点与 v 节点向量化之后的点乘结果，即如果从节点 u 开始的随机游走以高概率访问 v，则 u 和 v 相似。公式如下：

$$\begin{cases}\text{similarity}(u,v)=\boldsymbol{z}_v^{\mathrm{T}}\boldsymbol{z}_u \\ \text{similarity}(u,v)\propto P(v\mid u)\end{cases} \tag{3-2}$$

这种方法主要有两个优点：

(1) 相似度的定义结合了图的局部信息。

(2) 只需考虑随机游走的节点，不需要考虑全局信息，节省计算复杂度、效率高。

接下来，进行随机游走模型的参数优化，给定图 $G=(V,E)$，定义 $N_R(u)$ 表示在随机游走过程中从节点 u 出发，在一定步数内能够到达的所有节点集合。通过调整随机游走的转移概率，从而让节点间的相似度计算更加符合网络的实际连接情况。这一过程涉及最大化下述目标函数：

$$\max_\theta \sum_{u'\in V}\log P(N_R(u)\mid u',\theta) \tag{3-3}$$

式(3-3)是一个优化问题，其目的是最大化关于模型参数 θ 的一个函数。这个函数涉及图中所有节点 u'的累积对数概率，其中 $P(N_R(u)|u',\theta)$是条件概率，表示在给定模型参数 θ 的情况下，从节点 u'开始进行随机游走到达节点集合 $N_R(u)$的概率。这个优化问题可以看作寻找参数 θ，使随机游走的结果(访问不同节点的概率分布)尽可能地与数据中观察到的结构相匹配。

为了将上述优化问题表述为一个损失函数，我们通常会寻找最小化损失而不是最大化收益。这意味着，可以将最大化对数概率的问题转换为最小化负对数概率的问题，即最小化损失函数。这样做是因为在优化过程中，我们常常是最小化而不是最大化一个目标函数，因此损失函数可以定义为

$$L(\theta)=-\sum_{u'\in V}\log P(N_R(u)\mid u',\theta) \tag{3-4}$$

其中，$L(\theta)$是损失函数，训练目标是找到参数 θ，以最小化损失函数。在机器学习中，通常通过梯度下降或其他优化算法来实现。在图数据和随机游走的上下文中，这个损失函数可以帮助我们调整参数 θ，从而学习到能够反映节点之间真实关系的随机游走策略。

在图嵌入模型的随机游走中，参数 θ 是初始化的节点向量本身。在学习过程中，这些初始化的向量会不断地调整以最优化上述目标，最小化式(3-4)中的损失函数。值得注意的是，随机游走的每步都是无偏游走，也就是说走到下一个邻居节点的概率都是相同的，那么游走的结果可能会只关注局部信息，类似宽度优先搜索(甚至是两个节点来回跳)；或者只关注全局信息，类似深度优先搜索。那么有没有什么方法能控制其游走的策略呢？一种策略是新增两个参数。

(1) p 参数：用来控制返回上一个节点的概率。

(2) q 参数：用来控制远离上一个节点的概率。该参数也可以理解为采用宽度优先搜索还是深度优先搜索的一个比例值。q 值大，更偏向深度优先搜索，q 值小，更偏向广度优先搜索。剩下的操作与随机游走类似，定义损失函数，并采用 SGD 进行参数优化即可。

2. 图全局信息的向量化

图全局信息的向量化就是考虑怎么把整幅图的信息映射成一个图向量。聚合是最简单

的得到图向量的方法。就是简单地对图中所有的节点向量求和或求平均，并将这个结果作为图向量，即

$$\boldsymbol{z}_G = \sum_{v \in G} \boldsymbol{z}_v \tag{3-5}$$

这种方法不仅简单，在实际操作时，效果还是很好的。另一种方法是在整幅图的基础上，创造一个虚拟节点(Virtual Node)，如图 3-15 所示。这个虚拟节点与全图所有节点相连。在图模型的训练阶段，虚拟节点会与全图的节点进行信息交互，因此它可以在一定程度上表征全图信息。当训练结束后，可以取该虚拟节点的节点向量作为图向量。

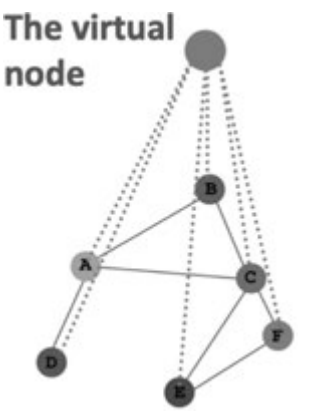


图 3-15 Virtual Global Node

3. 边的向量化

至于边的向量化也比较简单。简单地将该边相连的两个节点向量做聚合即可。除此之外，边的向量化也可以具体问题具体对待，即通过自定义的方式，例如，如果图结构表示的是一个分子，其中边表示两个原子是否相连。此时边的性质包括是否是双键、是否是环结构、是否是共价键、相连的原子是否是碳原子等。那么可以用一个长度是四维的相连来表示这个边，例如 [1，0，0，1] 可以表示这个边是一个连接碳原子的双键。

3.2 图神经网络模型

考虑 3.1.2 节讲解的图数据的 Embedding，其目的在于将节点映射成 d 维特征向量，使在图中相似的节点在向量域中也相似。本节介绍的图神经网络，也就是从深度学习的角度，对图的所有属性(节点、边、全局信息)的一种可优化的转换。优化后的特征向量便可以通过神经网络等模型继续进行一系列的预测或分类任务。

GNNs 是一种专为图数据设计的深度学习模型。它们能够直接在图结构上操作，捕捉节点间的复杂关系和图的全局结构特征。图神经网络在多种任务中表现出色，包括节点分类、图分类、链接预测和图生成等。在图神经网络中，图被定义为一组节点和连接节点的边。每个节点和边都可以拥有它们的属性。GNN 的目标是学习图中节点的低维向量表示(嵌入)，这些向量捕获了节点的特征信息及它们在图结构中的位置。

GNN 的工作原理基于邻域聚合(也被称为消息传递)策略，其中每个节点通过聚合和转换其邻居节点的信息来更新自己的表示。这个过程通常包括以下几个步骤。

(1) 消息聚合：对于给定的节点，从其邻居收集信息。涉及对邻居节点的特征向量进行聚合操作，例如求和、平均或最大化。

(2) 更新：结合当前节点的特征和聚合来自邻居的信息来更新节点的表示。更新过程通常通过一个神经网络(例如全连接层)实现。

(3) 重复：上述过程可以重复多次，每次迭代允许信息传递更远的距离，从而捕获更宽范围内的图结构特征。

(4) 输出：最终，对节点的嵌入可以通过各种方式被利用，例如直接用于节点级别任务，或者通过汇总所有节点的表示来完成图级任务。

3.2.1 消息传递神经网络

Gilmer 等提出的消息传递神经网络(Message Passing Neural Network，MPNN)框架是 GNN 中最简单最基础的框架。下面介绍如何使用 MPNN 来完成图预测任务。

消息传递神经网络采用“图进图出”架构，意味着消息传递神经网络接受一幅图作为输入，将信息加载到其节点、边和全局上下文中，并逐步优化这些节点向量、边向量和全局信息，让其向量表示更具意义，并且不改变输入图的结构。

如图 3-16 所示，MPNN 在图的每个图属性(节点 U、边 V、全局信息 E)上使用一个单独的多层感知器对特征向量进行优化。由于 MPNN 不改变图的结构，因此可以用与输入图相同的邻接列表和相同数量的特征向量来描述 MPNN 的输出图，但是，输出图的每个节点、边缘和全局背景的向量表示实际上经过了 MLP 的多次映射学习而变得更有意义。至此，一个简单的 GNN 建立完成，接着我们讨论如何用 GNN 的输出结果来进行预测。

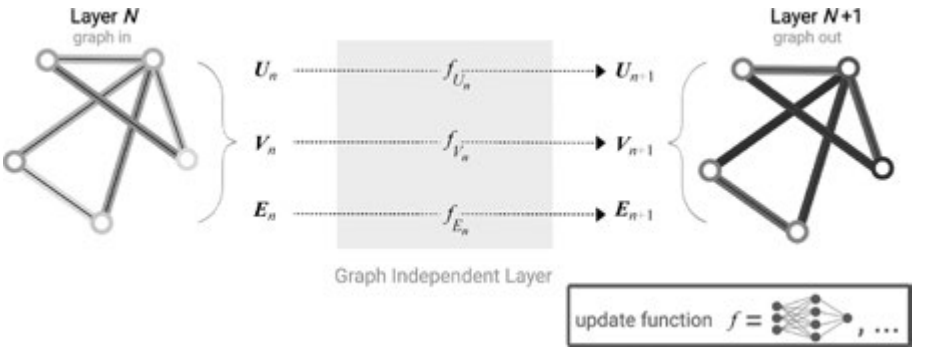


图 3-16 MPNN

以二元分类的问题进行举例。如果任务是对节点进行二元预测，而图中已经包含了节点信息，则这些节点信息是经过 MPNN 网络优化以后的节点特征表示，对于每个节点的向量，应用一个线性分类器进行二元分类即可，如图 3-17 所示。

然而，实际上没有那么简单，例如节点间的关系信息可能存储在边向量中，而不是存储在节点向量中。这时，仅仅依靠节点向量进行分类预测就显得有些单薄了。此时，可以通过池化操作来实现。池化分两步进行：首先对于每个要汇聚的属性，收集它们的向量表示，并将它们串联成一个矩阵，然后对这个矩阵进行某种池化操作，通常通过一个求和操作把两条边向量的信息和当前节点向量的信息汇聚在一起并映射成一个新的汇聚向量，如图 3-18 所示。

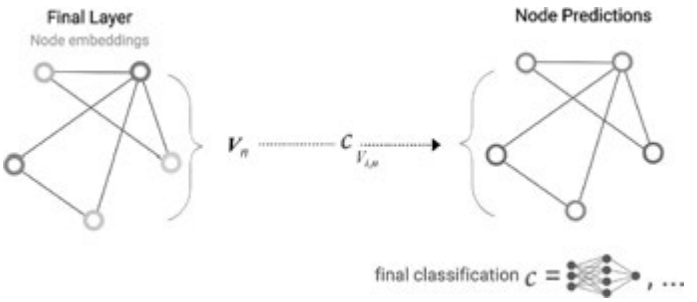


图 3-17 GNN 二元分类

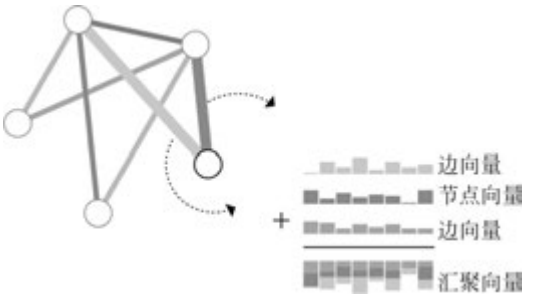


图 3-18 将边向量的信息汇聚到节点向量中

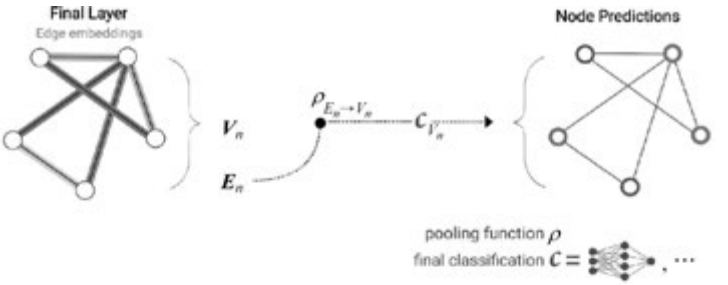
同样的思想也可以运用在边向量和全局向量上，通过池化来解决最后关于边属性的预测问题，以及关于全局属性的预测问题，如图 3-19 所示，其中 ρ 表示池化操作，例如图 3-19(a) $\rho_{E_n \to V_n}$ 表示把边向量的信息汇聚到节点向量中。

上述讲解便是关于 MPNN 来完成图预测任务的网络结构。整体的计算流程如图 3-20 所示，GNN 首先接受图数据作为网络的输入，这些图数据是经过 Embedding 生成的节点向量或边向量，可以看作数据的预处理。众所周知，预处理的质量是可以直接影响模型最终结果的，因此 Embedding 是 GNNs 模型设计中很重要的一个组成部分。接下来，GNN block 针对初始化的特征向量，例如节点向量进一步地进行训练学习，其中包含不同节点间信息的交互与融合，以及映射处理得到新的向量表示(Transformed Graph)。GNN block 一般是一些基于神经网络的模块，既可以是简单的 MLP，也可以是 Transformer 或其他算法组成的网络结构，这部分往往作为一个模型的创新点，在不同模型中的具体表现是不同的。最后，更新得到的图信息可以用于后续的分类或预测任务，通过一些简单的分类器即可实现，一般是采用几层神经网络计算得到最后的预测结果。

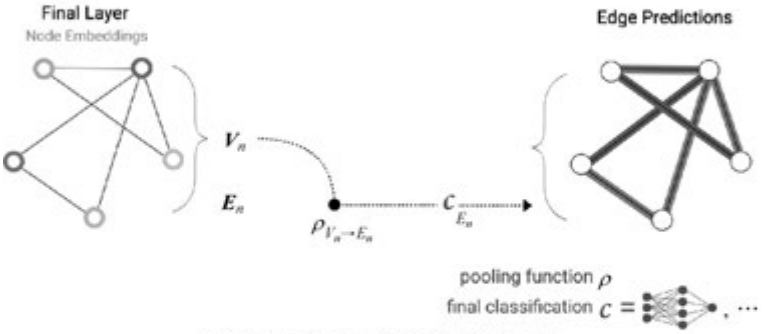
需要注意，在这个 MPNN 中，GNN blocks 内根本没有使用图的连接性(邻接矩阵/列表)。每个节点都是独立处理的，每条边也是如此，还有全局环境。只在汇集信息进行预测时，即池化得到预测结果的阶段，才涉及连接性的使用。

3.2.2 图神经网络的层结构与连接性

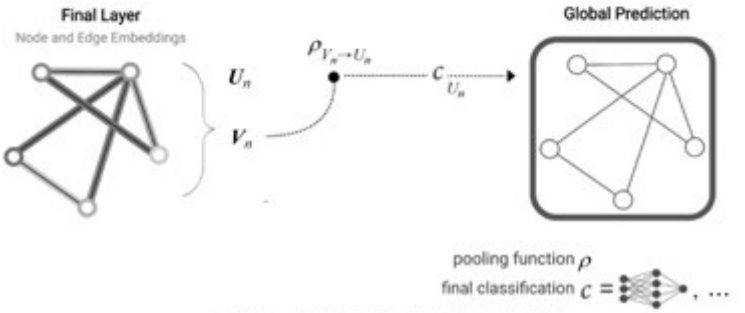
3.2.1 节 MPNN 是最简单的图神经网络的实现，并没有把图的连接性考虑在 GNN



(a) 将边向量的信息汇聚到节点向量中



(b) 将节点向量的信息汇聚到边向量中



(c) 将节点向量的信息汇聚到全局向量中

图 3-19　信息汇聚

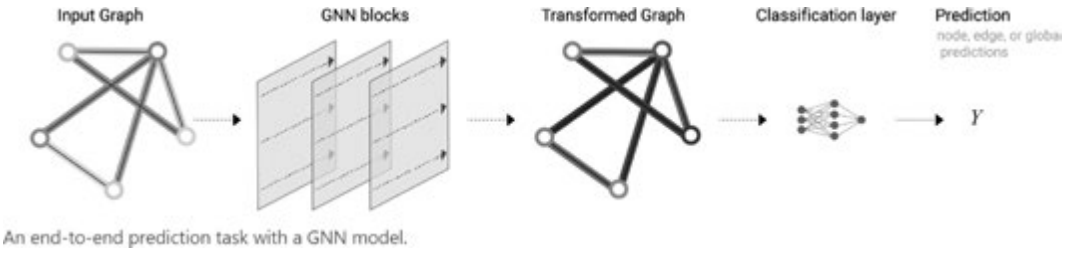


图 3-20　解决二分类问题的图神经网络

blocks 的网络设计中。那么应该如何融入邻接矩阵的信息呢？一个很朴素的方法：将邻接矩阵和特征合并在一起作为 GNN blocks 的输入，如图 3-21 所示。

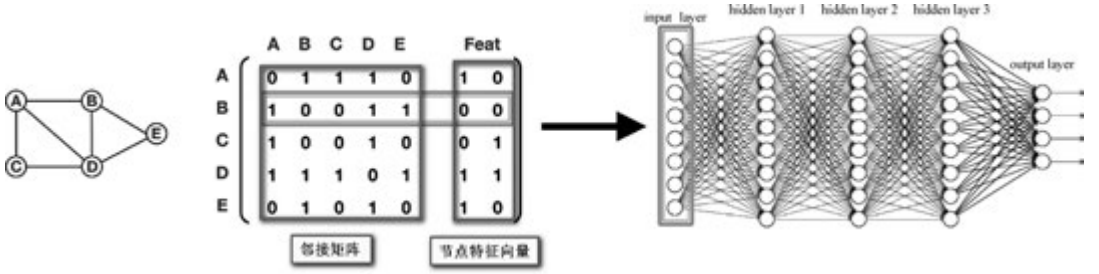
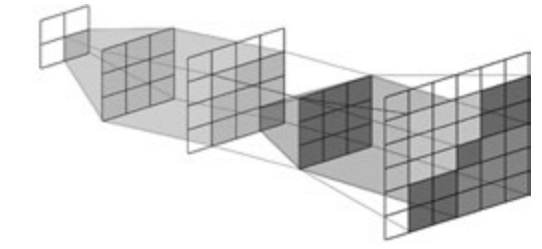


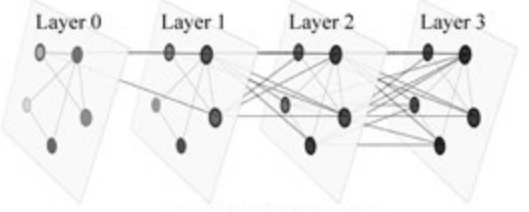
图 3-21　邻接矩阵＋特征的图神经网络

但这种方式存在几个问题。首先，参数复杂度增加：这种方法要求网络同时处理图的结构和特征信息，导致模型需要更多的参数来学习如何有效地整合这些信息，增加了模型的训练难度和计算成本。更重要的是，不适用于不同大小的图，即 Inductive 任务。这是因为直接使用全部的邻接矩阵意味着输入的维度依赖于训练阶段整幅图的结构，一旦图的大小或结构改变，训练出的 GNN 模型将不再适用，因此这使模型难以处理测试或使用阶段不同大小的图，限制了其应用的灵活性。最后，对节点顺序敏感：邻接矩阵的表示依赖于节点的顺序，不同的节点排列会导致矩阵不同，因此模型可能对输入的节点顺序高度敏感，影响了模型的泛化能力。这些问题表明：简单的合并邻接矩阵和特征可能不是处理图数据的最佳方法。

一个更好的解决方法是将卷积神经网络的思想（局部相关性和层级结构）泛化到图神经网络的 GNN blocks 的结构设计上。先从宏观上来观察一下，CNN 的结构如图 3-22(a)所示，将卷积思想泛化到 GNN 中的网络结构如图 3-22(b)所示。



(a) 卷积神经网络结构图



(b) 图神经网络结构图

图 3-22　CNN 和 GNN

需要注意的是，处理图片数据的卷积核适用于规则的数据结构，通过设定卷积核大小来定义局部相关性的范围，然而，图数据的结构本质上是不规则的，这使传统的卷积核方法不适用于图数据。在图神经网络中，我们通常通过聚合邻居节点的信息来实现局部相关性的处理，这种方法基于邻接矩阵的信息进行操作，例如，在 GNN 的第 0 层，一个节点仅与其直接相邻的节点(1-hop 邻居)进行信息聚合。到了第 1 层，这个节点可以进一步聚合由其 1-hop 邻居连接的那些节点的信息(2-hop 邻居)，这样逐层扩展，层次越高，能够汇聚的邻居范围越广，最终可能涉及全图的节点。这个过程类似于卷积神经网络中“感受野”的扩展。通过在 GNN 中堆叠多层，可以逐渐增加感受野的大小，实现从局部到几乎全局的信息交互。

从本质上讲，GNN 中的消息传递和 CNN 中的卷积核操作都是处理一个元素的邻居的信息，以便更新该元素的值。在图中，元素是一个节点，而在图像中，元素是一像素。不同的是，图中相邻节点的数量可以是可变的，不像在图像中，每个像素都有固定数量的相邻元素(8 个)。

如果实际任务还是需要很多层 GNN 网络，则可以在 GNN 模型中增加 Skip Connections。这个想法来源于 CNN 算法中的 ResNet 模型。通过 Skip Connections 可以解决因网络层结构加深而带来的网络退化问题，如图 3-23 所示。

用数学公式的方法来表达，一个朴素的 GNN 为

$$h_v^{(l)}=\sigma\left(\sum_{u\in N(v)}W^{(l)}\frac{h_u^{(l-1)}}{|N(v)|}\right) \tag{3-6}$$

一个带有 Skip Connections 的 GNN 为

$$h_v^{(l)}=\sigma\left(\sum_{u\in N(v)}W^{(l)}\frac{h_u^{(l-1)}}{|N(v)|}+h_v^{(l-1)}\right) \tag{3-7}$$

Skip Connections 也可以跨多层，如图 3-24 所示。

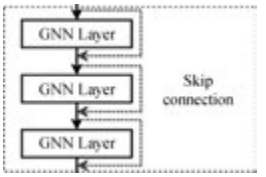


图 3-23 Skip Connections

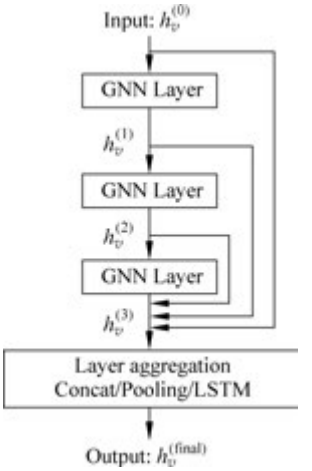
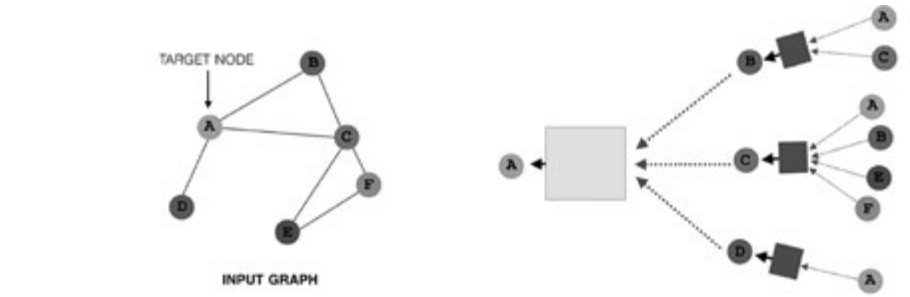


图 3-24 跨多层 Skip Connections

上述是类比卷积算法的解释，实际上从另一个角度来讲，汇聚节点邻居信息这一思想与神经网络的分层结构是非常相似的，如图 3-25 所示。



下图表示：每个节点都可以根据邻接矩阵中存储的边连接信息对节点向量信息进行汇聚。

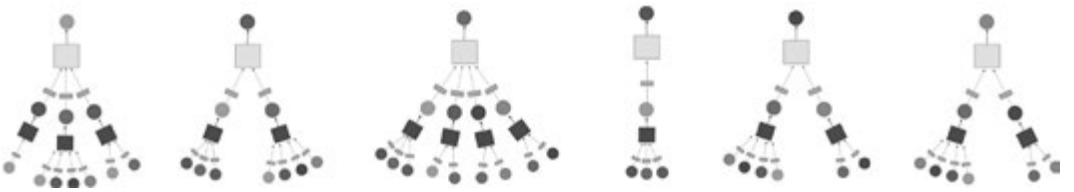


图 3-25　图神经网络

当根据邻接矩阵对节点的邻居信息进行汇聚之后，接下来的操作是什么？如图 3-26 所示。

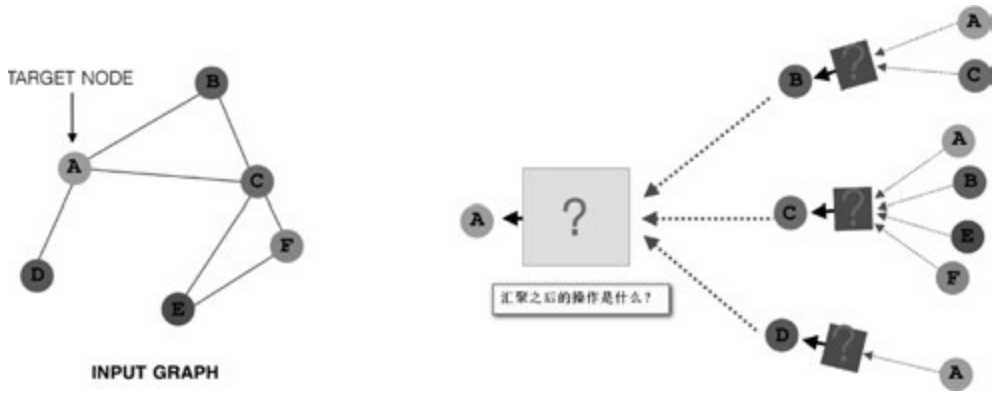


图 3-26　节点向量的信息交流

这个问题的答案是“神经网络”，GNN 在图的每个组成部分上（节点、边、全局）都分别使用一个单独的多层感知器进行映射，如图 3-27 所示。

值得注意的是，这个 MLP 是应用在单独的节点、边或全局向量上的，这意味着节点和边之间不能通过此 MLP 进行信息交互，这肯定是不合理的，其实解决办法也很简单，也就是在向量输入 MLP 之前，先在节点和边之间进行汇聚操作，如图 3-28 所示。

在图 3-28 中，在向量信息送入神经网络 f 之前，先进行边到节点（$\rho_{E_n \to V_n}$）和节点到边（$\rho_{E_n \to V_n}$）的信息汇聚，以此来完成边和节点之间的信息交互。同理，边、节点和全局向量三者之间也可以相互进行信息交互，如图 3-29 所示。

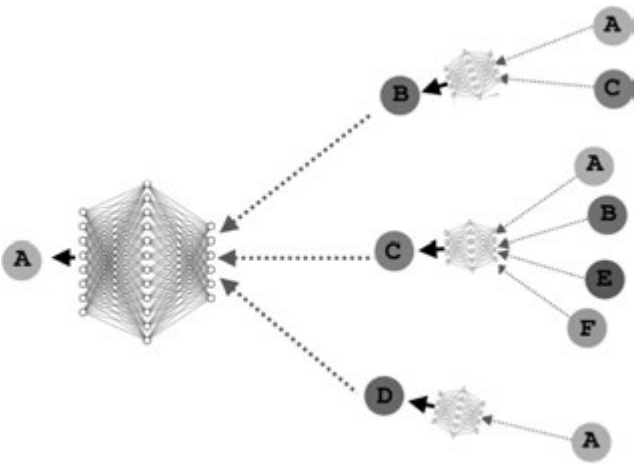


图 3-27　节点向量的信息交流

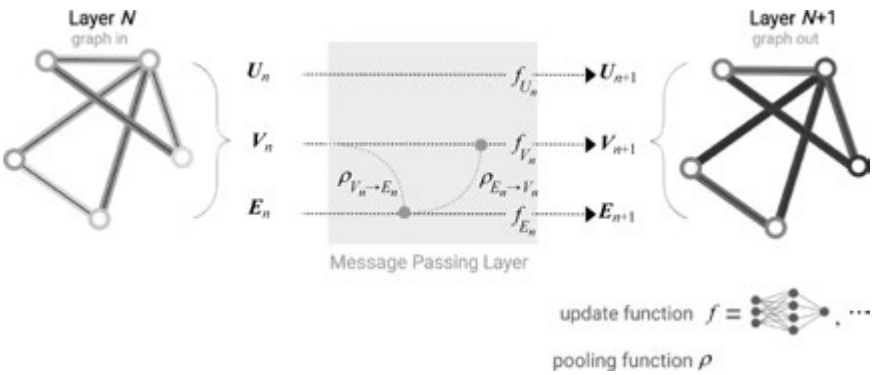


图 3-28　边和节点之间的信息交流

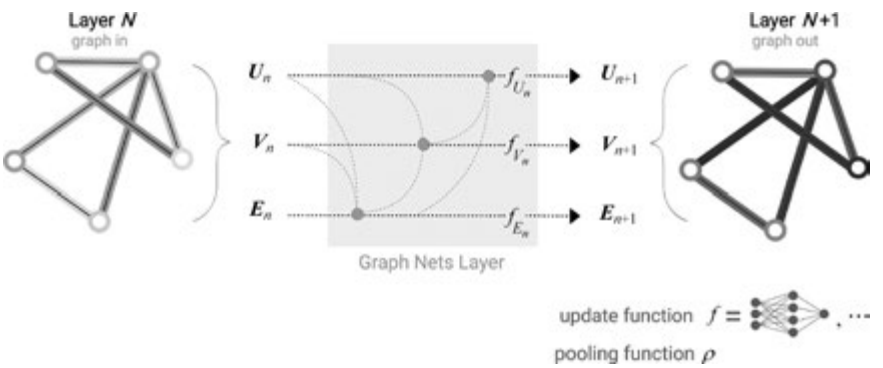


图 3-29　边、节点和全局向量之间的信息交流

值得注意的是，GNN 的层数往往不会太深，这与图数据的性质有关，读者还记得那个地球村有趣的说法吗？只需通过 6 个人，就可以找到世界上任何一个你不认识的人。如果

把全球人的关系绘制成一个巨大的社交图，这意味着在一个只有 6 层的 GNN 中，一个节点最终就可以纳入整幅图的信息。

3.2.3 图神经网络模型的训练

GNN 的训练流程如下，流程图如图 3-30 所示。

(1) 输入数据。

(2) 用 GNN 训练数据。

(3) 得到节点向量。

(4) 送入 Predictor(本质上是一个 MLP，将节点向量转换为最终需要的预测向量)。

(5) 得到预测向量。

(6) 选取损失函数和标签计算损失。

(7) 根据损失更新模型参数直到收敛。

(8) 选取评估指标测试模型。

(9) 使用模型解决实际问题。

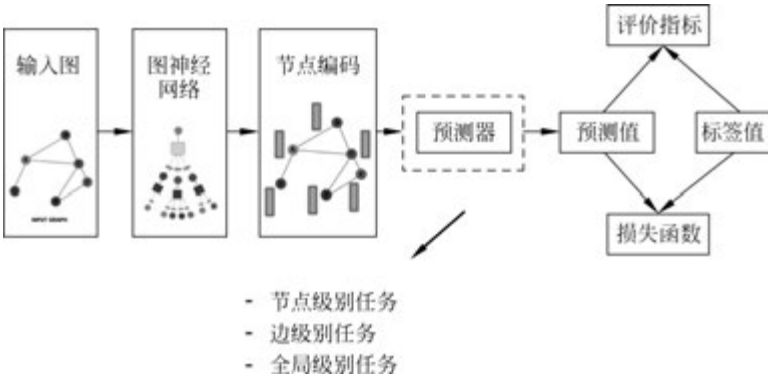


图 3-30 GNN 训练流程

Predictor 其实就算一个 MLP，是用来改变向量维度的，其目的是变化成想要的预测向量的形状。解释一下不同粒度任务下的 Predictor：假设 GNN 得到的节点是 d 维的，如果是节点级别的预测任务，例如在 k 个类别之间进行分类，可以直接用节点向量作为 Predictor 的输入，Predictor 将 d 维的节点向量映射到 k 维输出即可。如果是边级别的预测任务，如图 3-31 所示，则该任务是预测节点 u，v 之间是否有边。

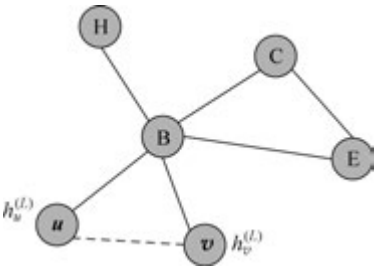


图 3-31 边级别预测

这时可以将节点向量 $\boldsymbol{u}$ 和 $\boldsymbol{v}$ 进行拼接，拼接后的向量在送入 Predictor 进行维度变化即可，即 $\hat{y}_{uv}=\mathrm{MLP}(\mathrm{Concat}(h_u^{(L)},h_v^{(L)}))$。此外，在边级别的任务上，Predictor 除了可以是 MLP 之外，还可以替换成没有任何可训练参数的点积操作(Dot Product)。因为两个向量

的点积结果意味着两个向量的相关程度，是一个常数。当两个向量的点积结果大时，意味着这两个节点之间有很大可能存在边。如果是图级别的任务，可以聚合图中所有节点(Global Pooling)的节点向量来做预测，即 $\hat{y}_G=\text{Head}_{\text{graph}}(h_v^{(L)}\in\mathbf{R}^d,\forall v\in G)$。

其中 Head Graph 有很多可选方式，常见的聚合方式如下。

(1) Global Mean Pooling：$\hat{y}_G=\text{Mean}(h_v^{(L)}\in\mathbf{R}^d,\forall v\in G)$；

(2) Global Max Pooling：$\hat{y}_G=\text{Max}(h_v^{(L)}\in\mathbf{R}^d,\forall v\in G)$；

(3) Global Sum Pooling：$\hat{y}_G=\text{Sum}(h_v^{(L)}\in\mathbf{R}^d,\forall v\in G)$。

这些聚合方式其实有一定的选择技巧，例如如果想比较不同大小的图，则 Mean 方法可能比较好，因为结果不受节点数量的影响；如果关心图的大小等信息，则 Sum 方法可能比较好，因为可以体现图的节点数量。如果关心图的某些重要特征，则 Max 方法会好一些，因为可以体现图中最重要的节点信息。这些方法都在小图上表现很好，但是在大图上的 Global Pooling 方法可能会面临丢失信息问题。

【例 3-1】 G_1 的节点嵌入为 $\{-1,-20,0,1,20\}$；G_2 的节点嵌入为$\{-10,-20,0,10,20\}$，显然两幅图的节点嵌入差别很大，图结构很不相同，但是经过 Global Mean Pooling 后，不管是求平均、求最大或求和。这两幅图的表示向量一样了，此时无法做出区分，显然是不行的。

为了解决这一问题，解决方法是分层聚合节点向量(Hierarchical Global Pooling)。具体来讲，可以使用 ReLU(Sum(·))做聚合，先分别聚合前两个节点和后 3 个节点的嵌入，然后聚合这两个嵌入，举例如下。

G_1：

第 1 轮聚合：$\hat{y}_a=\text{ReLU}(\text{Sum}(\{-1,-20\}))=0,\hat{y}_b=\text{ReLU}(\text{Sum}(\{0,1,20\}))=21$

第 2 轮聚合：$\hat{y}_G=\text{ReLU}(\text{Sum}(\{y_a,y_b\}))=21$

G_2：

第 1 轮聚合：$\hat{y}_a=\text{ReLU}(\text{Sum}(\{-10,-20\}))=0,\hat{y}_b=\text{ReLU}(\text{Sum}(\{0,10,20\}))=30$

第 2 轮聚合：$\hat{y}_G=\text{ReLU}(\text{Sum}(\{y_a,y_b\}))=30$

这样就可以将 G_1 和 G_2 区分开了，其实，这种分层聚合得到图级别预测结果的方式，在某种程度上非常类似于 CNN 处理图像识别问题的层级结构，如图 3-32 所示。

两种方法的区别在于图 3-32 中 CNN 处理图像识别问题的模型架构是用来训练的，是一个模型，而 GNN 的层次聚合是一个操作，不是一个模型。具体来讲，是对 GNN 模型的计算结果进行分层池化操作，其中没有任何可学习参数。将池化后的结果送入 Predictor 中进行结果向量预测，在 Predictor 中才存在可学习参数。

以上解释的是不同粒度的预测任务，接下来解释一些不同的训练方法，GNN 的训练方式分为以下两种。

(1) 有监督学习(Supervise Learning)：直接给出标签(例如一个分子图的药理活性)。

(2) 无监督学习(Unsupervised Learning/Self-supervised Learning)：使用图自身的信

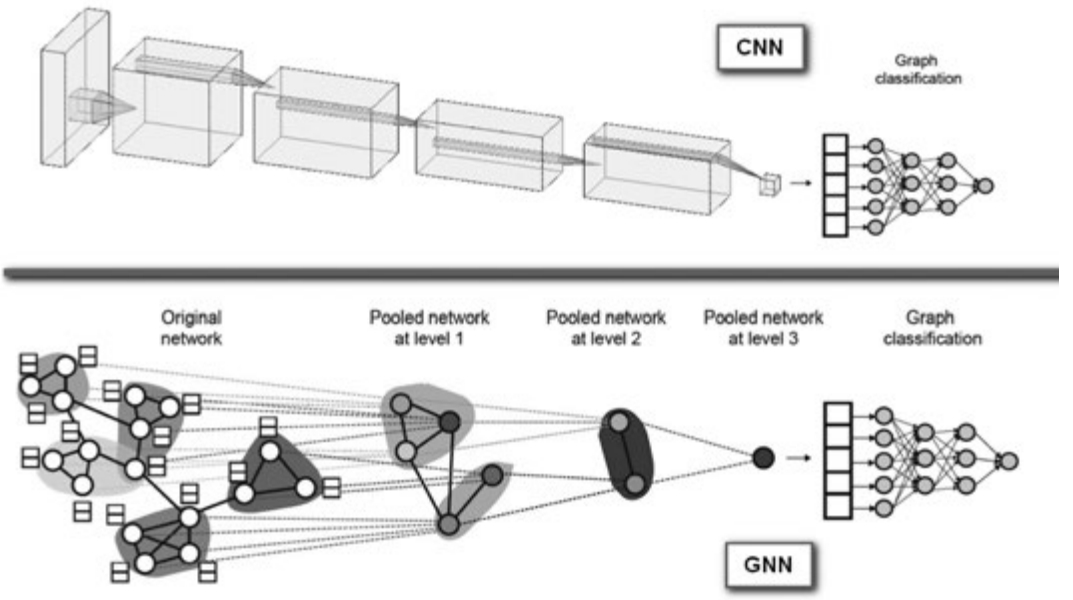


图 3-32　CNN 与 GNN 流程图

号(例如预测两节点间是否有边)。

有监督学习：按照实际情况而来,不同任务下标签是不同的,以下是具体的应用例子。

(1) 节点级别预测——引用网络中,节点(论文)属于哪一学科。

(2) 边级别预测——交易网络中,边(交易)是否有欺诈行为。

(3) 图级别预测——分子图的药理活性。

无监督学习：在没有外部标签时,可以使用图自身的信号作为有监督学习的标签。举例来讲,GNN 可以按以下级别进行预测。

(1) 节点级别：节点统计量(例如 Clustering coefficient、PageRank 等)。

(2) 边级别：链接预测(隐藏两节点间的边,预测此处是否存在链接)。

(3) 图级别：图统计量(例如预测两幅图是否同构)。

3.2.4　图数据的数据增强

数据增强是一种技术,用于通过对现有数据集进行各种变换来增加训练模型时的数据量和多样性。在图像数据处理领域,这种技术尤其重要,因为它可以帮助改善模型的泛化能力,减少过拟合。图数据的数据增强(Graph Augmentation for GNN)简称图增强,可以从结构层面或特征层面进行展开。

从结构方面来讲,图的数据增强主要分为增加虚拟信息、随机丢弃(DropOut)和子图抽样 3 种方案。增加虚拟信息主要处理图结构过度稀疏的问题,这可能会导致信息传递效率低。此时可以为两个 1-hop 跳邻居和目标节点之间增加一条虚拟边,如图 3-33 所示。

当然，也可以增加虚拟点，如图 3-34 所示。

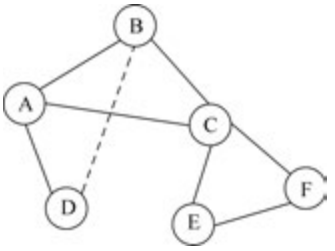


图 3-33　虚拟边

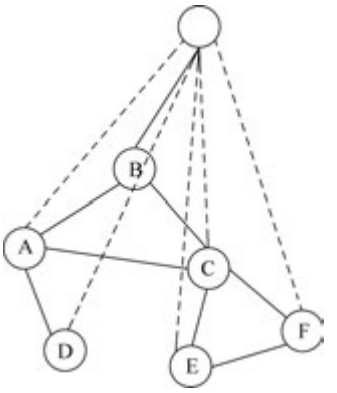


图 3-34　虚拟点

增加虚拟边可以帮助模型学习到更多可能的节点间关系，这在某些场景中可以模拟潜在的未观察到的联系。增加虚拟节点也是类似的道理，这些虚拟节点可能代表一种抽象的概念或集合，通过这种方式可以扩展图的表达能力，增强模型对不同图结构的适应能力。不管是虚拟边还是虚拟节点都可以提升在稀疏图中的信息传递能力。

另一方面，如果是针对图结构过度稠密的情况，则可以随机移除图中的部分节点，以及这些节点的边。这种方式有助于模型在部分信息丢失的情况下仍能做出准确的预测，增强模型对信息不完整的稳健性。当然，也可以随机移除部分边，保留节点。这可以帮助模型学习在连接信息不全的情况下有效地进行信息传递和决策。

关于子图抽样，通过从原始图中提取较小的子图来生成新的训练样本。这种方法尤其适用于处理大规模图数据，可以有效地减少计算资源的需求，同时增加数据的多样性。在子图抽样中，可以基于不同的标准选择节点和边，例如随机选择、基于节点度的选择或社区结构的选择。这样的抽样不仅有助于模型在更小的管理更容易的数据集上进行训练，而且能够让模型学习到从局部信息中推断全局结构的能力。此外，子图抽样还可以模拟在实际应用中的不完整数据情况，提高模型对不完整或有损图结构的适应性和稳健性。这种策略在社交网络分析、生物信息学和推荐系统等领域尤为重要，因为这些领域常常需要处理大型复杂网络，并从中抽取有价值的信息。

从特征层面来讲，图增强技术主要有属性掩码、节点置换和特征扰动 3 种方案，其中，属性掩码(Attribute Masking)方法通过随机隐藏或掩码节点的某些属性来增强数据。这可以迫使模型不过度依赖某些特征，并增强其利用节点的其他特征或结构信息来进行预测的能力，例如，在处理社交网络数据时，可以随机掩盖用户的年龄或兴趣点，以测试和增强模型在缺乏这些信息时的表现。节点置换(Node Shuffling)技术涉及重新排序或打乱节点的标签，但保持图的结构不变。这样做可以帮助模型学习到更通用的特征，而不是仅仅依赖于特定节点的标签或位置。节点置换有助于增强模型在面对不同节点身份时的泛化能力。特征扰动(Feature Perturbation)最容易理解，这种方法通过向节点的特征添加随机噪声来进行数

据增强。这不仅可以模拟数据采集过程中的潜在误差，还可以让模型更加稳健，减少对干净且精确数据的依赖。特征扰动尤其适用于那些特征信息可能因外部环境变化而受到影响的应用场景，例如环境监测或经济预测。这 3 种从特征层面的图数据增强方法都旨在通过模拟真实世界数据的不确定性和不完整性来改善模型的性能和适应性。它们既可以单独使用，也可以与结构层面的增强技术（例如子图抽样或边掉落等）结合使用，以进一步提升图神经网络在复杂应用中的效果。

3.3 图神经网络算法基础的变体

3.1 节和 3.2 节介绍了图的基本属性和 Embedding 的概念，并讲解了最简单的图神经网络模型——消息传递网络（MPNN），MPNN 是图进图出的网络设计，目标是得到图数据更好的 Embedding 信息表示。下面将介绍 3 种经典的 GNN 的变体：Graph Convolutional Network（GCN）、Graph Sample and Aggregate Network（GraphSAGE）和 Graph Attention Network（GAT），其中 GCN 更适合完成 Transductive 任务，而 GraphSAGE 和 GAT 在 Transductive 和 Inductive 这两种任务中都可以进行处理。

3.3.1 Graph Convolutional Network

先回顾一下之前讲解的朴素图神经网络，如图 3-35 所示。

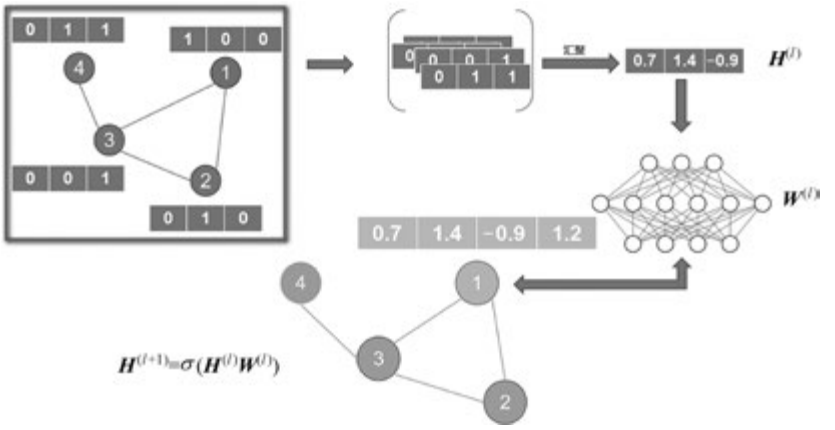


图 3-35 朴素图神经网络

从图 3-35 左上角方框部分可以看作图神经网络的初始状态。以 1 号节点为例，在图神经网络中，信息的传递是先汇聚一号节点的邻居节点信息，得到汇聚后的新向量，这个向量可以看作图神经网络第 1 层的输入信息 $\boldsymbol{H}^{(l)}$，然后 $\boldsymbol{H}^{(l)}$ 经过一个 MLP 的映射，得到一个新的输出向量 $\boldsymbol{H}^{(l+1)}$，这个向量则作为第 2 层图神经网络的输入信息，以此类推，可以定义出一个多层的图神经网络。层与层之间的信息传递公式可以写作：

$$\boldsymbol{H}^{(l+1)}=\sigma(\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{3-8}$$

而图卷积神经网络的计算公式则为

$$\boldsymbol{H}^{(l+1)}=\sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{3-9}$$

两者最主要的区别就是图卷积神经网络比图神经网络多了一个 $\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$，其中，$\widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N$，$\boldsymbol{A}$ 就是图的邻接矩阵，$\boldsymbol{I}_N$ 是一个全一的对角矩阵，即

$$\begin{cases}\boldsymbol{A}=\begin{bmatrix}0. & 1. & 1. & 0.\\ 1. & 0. & 1. & 0.\\ 1. & 1. & 0. & 1.\\ 0. & 0. & 1. & 0.\end{bmatrix} \quad \boldsymbol{I}_N=\begin{bmatrix}1. & 0. & 0. & 0.\\ 0. & 1. & 0. & 0.\\ 0. & 0. & 1. & 0.\\ 0. & 0. & 0. & 1.\end{bmatrix}\\ \widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N=\begin{bmatrix}1. & 1. & 1. & 0.\\ 1. & 1. & 1. & 0.\\ 1. & 1. & 1. & 1.\\ 0. & 0. & 1. & 1.\end{bmatrix}\end{cases} \tag{3-10}$$

邻接矩阵 $\boldsymbol{A}$ 表示的是节点与节点之间的关系，全一的对角矩阵 $\boldsymbol{I}_N$ 表示的是节点自身，所以 $\widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N$ 表示考虑了节点自身信息的邻接矩阵。

先将 $\boldsymbol{A}$ 矩阵加入图卷积神经网络的计算公式中，得到：

$$\boldsymbol{H}^{(l+1)}=\sigma(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{3-11}$$

其中，$\boldsymbol{H}^{(l)}$ 的值是图 3-35 中的节点初始化向量：

$$\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}=\begin{bmatrix}1. & 1. & 1. & 0.\\ 1. & 1. & 1. & 0.\\ 1. & 1. & 1. & 1.\\ 0. & 0. & 1. & 1.\end{bmatrix}*\begin{bmatrix}1. & 0. & 0.\\ 0. & 1. & 0.\\ 0. & 0. & 1.\\ 0. & 1. & 1.\end{bmatrix}=\begin{bmatrix}1. & 1+1. & 1.\\ 1. & 1. & 1.\\ 1. & 1+1. & 1+1.\\ 0. & 1. & 1+1.\end{bmatrix} \tag{3-12}$$

式(3-12)表示每个节点向量要同时考虑自身节点和它相邻节点的信息。$\widetilde{\boldsymbol{D}}$ 是度矩阵，表示的是每个节点的度数，例如，图 3-35 中的一号和二号节点，考虑其自连接，则度数等于 3；同理，三号节点的度数为 4，而四号节点度数为 2，因此得到下述度矩阵。

$$\widetilde{\boldsymbol{D}}=\begin{bmatrix}3. & 0. & 0. & 0.\\ 0. & 3. & 0. & 0.\\ 0. & 0. & 4. & 0.\\ 0. & 0. & 0. & 2.\end{bmatrix} \tag{3-13}$$

$\widetilde{\boldsymbol{D}}^{-1}$ 表示对度矩阵中的元素取倒数，得到：

$$\widetilde{\boldsymbol{D}}^{-1}=\begin{bmatrix}1/3. & 0. & 0. & 0.\\ 0. & 1/3. & 0. & 0.\\ 0. & 0. & 1/4. & 0.\\ 0. & 0. & 0. & 1/2.\end{bmatrix} \tag{3-14}$$

$\widetilde{\boldsymbol{D}}^{-1}(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)})$ 相当于对 $\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}$ 结果的行进行归一化操作，它有助于避免由于图中节点度

的不均匀分布而引起的梯度爆炸或消失问题。确保了所有节点的特征贡献在聚合时被适当地缩放，从而提高了学习的稳定性和效果。公式计算如下：

$$\begin{aligned}\widetilde{\boldsymbol{D}}^{-1}(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}) &= \begin{bmatrix}1/3. & 0. & 0. & 0.\\ 0. & 1/3. & 0. & 0.\\ 0. & 0. & 1/4. & 0.\\ 0. & 0. & 0. & 1/2.\end{bmatrix} * \begin{bmatrix}1. & 1+1. & 1.\\ 1. & 1. & 1.\\ 1. & 1+1. & 1+1.\\ 0. & 1. & 1+1.\end{bmatrix}\\ &= \begin{bmatrix}1/3*1. & 1/3*(1+1). & 1/3*1.\\ 1/3*1. & 1/3*1. & 1/3*1.\\ 1/4*1. & 1/4*(1+1). & 1/4*(1+1).\\ 1/2*0. & 1/2*1. & 1/4*(1+1).\end{bmatrix}\end{aligned} \tag{3-15}$$

同理，$(\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)})\widetilde{\boldsymbol{D}}^{-1}$ 相当于对 $\widetilde{\boldsymbol{A}}\boldsymbol{H}^{(l)}$ 结果的列做归一化。最后，式(3-9)中左乘和右乘的矩阵使用的是 $\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ 而不是 $\widetilde{\boldsymbol{D}}^{-1}$ 的原因是：当对行列元素各做一次归一化后，相当于对节点向量的每个元素都做了两次归一化，也就是多做了一次，因此这里使用作归一化的矩阵的是 $\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$。最终，图卷积神经网络的计算公式则为

$$\boldsymbol{H}^{(l+1)} = \sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{3-16}$$

尽管图卷积网络(GCN)和卷积神经网络(CNN)都是“卷积”概念的扩展，但它们在处理数据的方式和结构上有很大的不同。在 CNN 中，卷积是定义在规则的网格数据(例如图像)上的。这种卷积操作通常涉及固定大小的滤波器(卷积核)在输入数据上滑动，提取局部特征。在 GCN 中，卷积操作是在图结构上定义的，这意味着它需要考虑图的拓扑结构。GCN 中的“卷积”是通过聚合一个节点及其邻居的特征来实现的，这种聚合考虑了图的连接性而非空间的临近性。更重要的是在聚合邻居节点信息时，引入了自身节点的信息，并通过节点的度对聚合的特征进行归一化。这种归一化处理有助于避免节点度的大小对特征聚合造成过大的影响，使特征表示更加平滑和稳定。总结来讲，尽管 GCN 中的“卷积”与传统的 CNN 在技术细节上有所不同，但其核心思想都是利用局部连接结构信息进行特征学习和聚合，这也是为什么将这种操作称为“卷积”的原因。

另外，GCN 更适用于 Transductive 任务，因为 GCN 在更新节点向量时，利用了基于整个邻接矩阵信息的拉普拉斯矩阵($\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$)，从这种意义上讲它聚合邻居特征的时候，训练出来的权重 $\boldsymbol{W}$ 考虑了整个图结构，如果图结构改变，那么就不再适用，因此 GCN 在处理 Inductive 任务时表现不佳。

3.3.2 Graph Sample and Aggregate Network

相比 GCN，GraphSAGE 可以处理 Inductive 类型的任务。Inductive 任务的关键在于设计能够良好泛化到未见过的图或节点上的模型，其挑战在于模型需要能够处理图的结构和特征在训练集和测试集之间可能有显著差异的情况，因此适用于 Inductive 任务的 GNN

模型在设计时，应该侧重于学习能够代表单个节点的通用表示，而不是依赖于整幅图的结构。这意味着模型应该能够从节点的局部邻域信息中学习其表示，而不是从整幅图的全局结构中学习，GCN 利用整个邻接矩阵这一做法就违反了这一点。GraphSAGE 对此则进行了改进，更新节点信息的方式就不依赖于图的全局结构。具体来讲，其更新过程主要分三步：

（1）聚合邻居节点的信息，聚合函数有 3 种，将在下文展开解释。

（2）将聚合后的信息与自身的节点信息进行拼接，进行特征融合。

（3）送入神经网络模型中进行映射，得到更新后的节点信息。

【例 3-2】 图数据如图 3-36 所示，现在使用 GraphSAGE 对节点 1 进行更新。

节点 1 特征向量的更新步骤如下。

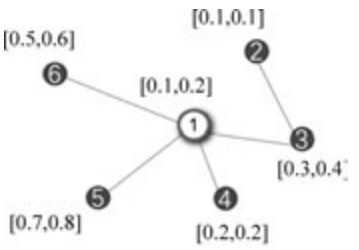


图 3-36 图数据

（1）聚合邻居节点：$h^1_{N(1)} \leftarrow \text{AGGREGATE}(h^0_3, h^0_4, h^0_5, h^0_6)$。

（2）拼接自身信息：$h^1_1 \leftarrow \text{CONCAT}(h^0_1, h^0_{\mathcal{N}(1)})$。

（3）经过神经网络映射：$h^1_1 \leftarrow \sigma(W^1 \cdot \text{CONCAT}(h^0_1, h^0_{\mathcal{N}(1)}))$。

假设步骤（1）中聚合函数 AGGREGATE 是 Mean 函数，则代数得

$$h^1_{\mathcal{N}(1)} \leftarrow \text{AGGREGATE}(h^0_3, h^0_4, h^0_5, h^0_6) = \text{Mean}([0.3, 0.4], [0.2, 0.2], [0.7, 0.8], [0.5, 0.6]) \tag{3-17}$$

另外，在这个计算流程中有两个地方需要特别注意。第一，GraphSAGE 在聚合某节点邻居信息的时候，并不是聚合全部的邻居，而是聚合 K 个邻居，K 是一个超参数。举例，在图 3-36 中，若 K 等于 3，则在聚合节点 1 的周围邻居时，随机从节点 3、4、5、6 中选择 3 个进行聚合。若 K 等于 5，则除了选择节点 1 的周围 4 个邻居以外，再重复从这 4 个邻居中抽样一个节点。这样做的好处是，当图数据非常庞大时，选取某节点的全部邻居做聚合是非常耗时耗力的，若只选择其中的 K 个邻居，则可以更快地进行计算。超参数 K 本质上是计算精度和计算速度之间的一种权衡。

第 2 个需要注意的是 GraphSAGE 定义了 3 种不同的聚合函数。

（1）Mean：$\text{AGG} = \sum_{u \in N(v)} \frac{h^{(l)}_u}{|N(v)|}$。

（2）Pool：$\text{AGG} = \gamma(\{\text{MLP}(h^{(l)}_u), \forall u \in N(v)\})$。

（3）LSTM：$\text{AGG} = \text{LSTM}([h^{(l)}_u, \forall u \in \pi(N(v))])$。

Mean 操作就是简单地对节点的邻居信息求平均，上文也举例说明过。Pool 操作就是先把节点的邻居节点向量送入一个 MLP 中，对 MLP 的输出结果做 γ 操作得到聚合后的节点向量，这个 γ 就是池化的算子，既可以是 Mean，也可以是 Max，分别对应平均池化和最大池化，它们的实际使用效果都差不多。至于第 3 种 LSTM 的聚合方式与第 2 种 Pool 聚合

类似，区别在于把 MLP 换成了 LSTM 模型。LSTM 模型的特征是能够对序列信息更好地进行建模，通过维护一个内部状态（记忆单元）来捕捉序列中的顺序依赖性。在图结构中，这意味着如果节点的连接具有某种逻辑或时间顺序，则 LSTM 可能更好地利用这种信息，而且，LSTM 能够根据序列中每个元素的重要性动态地调整其内部状态，这允许它在聚合邻居特征时给予不同邻居不同的重视程度。最后，LSTM 可以灵活地处理任何长度的输入序列，它在处理具有不同数量邻居的节点时不需要额外的填充或截断操作，这是处理不规则图数据的一个优势。

总结一下，GraphSAGE 的重要性体现在灵活的节点聚合框架上，展示了在图神经网络中节点信息聚合的多样化可能性。GraphSAGE 不仅局限于使用简单的平均操作来聚合节点特征，还提出了使用带参数的网络结构，例如 MLP（多层感知器）和 LSTM（长短期记忆网络），以增强聚合过程的表达能力和灵活性。这开启了使用更复杂的模型来处理图中节点的可能性。当然也可以尝试换成其他的网络结构，可以根据具体任务的需求和数据的特点进行定制，以实现最佳性能，例如，对于那些节点间交互非常复杂或图结构快速变化的应用场景，使用高级的聚合策略（例如 Transformer）可能更加有效，3.3.3 节讲解的 GAT 模型便是由此得到启发而被提出的模型。

3.3.3 Graph Attention Network

最后来学习图注意力网络 GAT。GAT 可以有两种运算方式，一种被称为全局注意力（Global Graph Attention），顾名思义，也就是每个顶点对于图上任意顶点都进行 Attention 运算。这样做的优点是完全不依赖于图的结构，即不依赖图的邻接矩阵，对于 Inductive 任务无压力。缺点也很明显：首先，丢掉了图结构的特征，其次，当图数据规模很大的时候，运算面临着高昂的成本。第 2 种被称为掩码图注意力机制（Mask Graph Attention），每个节点的 Attention 计算仅限于其邻接节点（邻接矩阵所定义的直接连接）。这样做保留了图的结构特性，并利用了局部邻域信息。与全局注意力相比，掩码图注意力机制大大地降低了计算成本，因为每个节点只与其直接邻居进行交互，而不是图中的所有节点。这种方式有效地利用了图的拓扑结构，可以捕捉到节点之间的局部关系，这对于许多图数据分析任务是非常有价值的。值得注意的是，虽然在掩码图注意力机制的 GAT 中用到了邻接矩阵，但是并不像 GCN 一样利用全部的邻接矩阵信息，而是仅利用邻接矩阵查询某节点的邻居是谁，因此可以来完成 Inductive 任务。换句话说，GAT 中节点可以通过注意力权重动态地选择其信息的重要来源，即它的邻居。这些权重是由模型通过学习自动确定的，并不依赖于预先定义的结构。这一点区别于 GCN，后者使用的是固定的邻接矩阵加自环的归一化形式来确定节点之间信息传递的权重。

接下来主要讲解 GAT 中的 Attention 机制。类似于 Transformer 中的注意力机制，GAT 的计算也分为两步：计算注意力系数（Attention Coefficient）和加权求和（Aggregate）进行特征重要程度的重分配。对于顶点 i，逐个计算它的邻居（$j \in N_i$）和它的注意力系数 $e_{ij}=\boldsymbol{\alpha}([\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j])$，$j \in \mathcal{N}_i$，具体流程如图 3-37 所示。

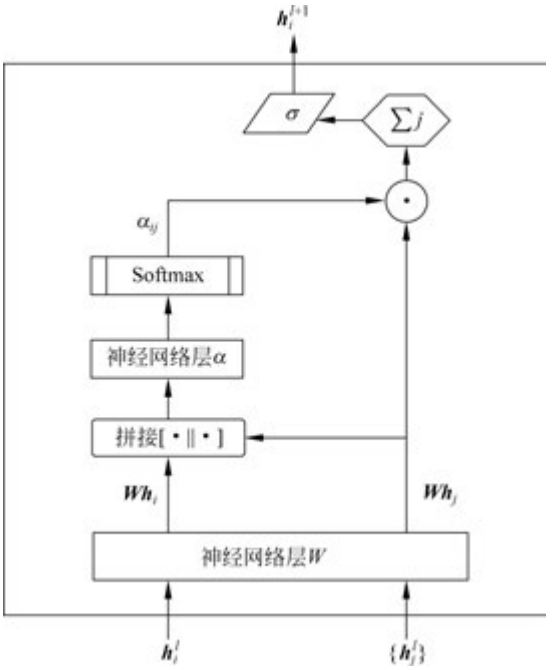


图 3-37 GAT Attention 机制

在图 3-37 中 $\boldsymbol{h}_i^l$ 是节点 i 的特征向量，$\{\boldsymbol{h}_j^l\}$是节点 j 所有邻居的特征向量的集合，$\boldsymbol{W}$ 是一个共享参数，通过一个单层的神经网络层来实现。[·‖·]代表对节点 i、j 经过神经网络层 $\boldsymbol{W}$ 变换后的特征进行拼接操作(Concat)。最后通过$\boldsymbol{\alpha}$ 把拼接后的高维特征映射到一个实数上，也是通过一个单层的神经网络层来实现的。显然学习节点 i、j 之间的相关性系数，也就是通过可学习的神经网络层的参数 $\boldsymbol{W}$ 和 $\boldsymbol{\alpha}$ 映射完成的。有了相关系数，再对其进行 Softmax 归一化操作即可得到注意力系数 α_{ij}。至于加权求和的实现也很简单，根据计算好的注意力系数，把特征加权求和聚合(Aggregate)一下，即

$$\boldsymbol{h}'_i=\sigma\Big(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\boldsymbol{W}\boldsymbol{h}_j\Big) \tag{3-18}$$

$\boldsymbol{h}'_i$就是 GAT 输出的对于每个节点 i 的新特征，这个新特征的向量表示融合了邻域信息，$\sigma(\cdot)$是激活函数。最后，与 Transformer 一样，GAT 也可以用多头注意力机制来进化增强：

$$\boldsymbol{h}'_i(K)=\Big\|_{k=1}^{K}\sigma\Big(\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\boldsymbol{W}^k\boldsymbol{h}_j\Big) \tag{3-19}$$

其中，K 是注意力机制的头数，每个头都会维护更新自己的参数，计算得到自己的结果，$\Big\|_{k=1}^{K}$ 表示对所有头的计算结果进行拼接(Concat)，从而得到最后更新好的新节点向量。多头注意力机制也可以理解成用了集成学习的方法，就像卷积中，也要靠大量的卷积核才能有比较好的特征提取效果一样。最后通过一个示例来复习一下 GAT 的计算过程，图数据如图 3-38 所示。

计算注意力系数的两个神经网络层的参数分别是 $\boldsymbol{W}$ 和 $\boldsymbol{\alpha}$，假设其初始化的值为 $\boldsymbol{W}=[1,1]$，$\boldsymbol{\alpha}=[1,1,1,1]$。注意，这些参数都是可学习的，随着网络的训练而更新。首先，计算注意力系数 $e_{ij}=\boldsymbol{\alpha}(\boldsymbol{W}\boldsymbol{h}_i,\boldsymbol{W}\boldsymbol{h}_j)$，以节点 1 为例，与其他节点的相关性系数为

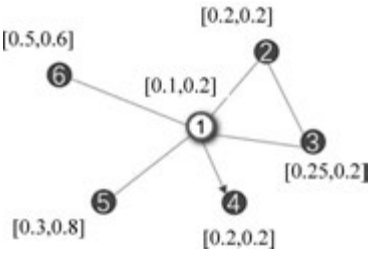


图 3-38　图数据

$$\begin{aligned}
e_{12}&=\boldsymbol{\alpha}\cdot[0.1,0.2,0.2,0.2]=0.7\\
e_{13}&=\boldsymbol{\alpha}\cdot[0.1,0.2,0.25,0.2]=0.75\\
e_{14}&=0\\
e_{15}&=\boldsymbol{\alpha}\cdot[0.1,0.2,0.3,0.8]=1.4\\
e_{16}&=\boldsymbol{\alpha}\cdot[0.1,0.2,0.5,0.6]=1.4
\end{aligned}\tag{3-20}$$

e_{14} 由于单向性，即节点 1 指向 4，因此在计算更新节点 1 的信息时，节点 4 的信息不会传递给节点 1，因此其相关性 e_{14} 为 0，然后通过 Softmax 公式经计算得到相关性系数 $\alpha_{ij}=\dfrac{\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i\,||\,\boldsymbol{W}\boldsymbol{h}_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\mathrm{LeakyReLU}(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i\,||\,\boldsymbol{W}\boldsymbol{h}_k]))}$，例如 α_{12}：

$$\alpha_{12}=\frac{\exp(\mathrm{LeakyReLU}(e_{12}))}{\exp(\mathrm{LeakyReLU}(e_{12}))+\exp(\mathrm{LeakyReLU}(e_{13}))+\cdots+\exp(\mathrm{LeakyReLU}(e_{16}))}\tag{3-21}$$

然后通过加权求和对某节点的邻居做重要程度的重分配，即

$$\boldsymbol{h}'_i=\sigma\Big(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\boldsymbol{W}\boldsymbol{h}_j\Big)\tag{3-22}$$

以节点 1 为例：

$$\boldsymbol{h}_1'^{Y}=\sigma(\alpha_{12}\cdot\boldsymbol{W}\cdot\boldsymbol{h}_2)+\sigma(\alpha_{13}\cdot\boldsymbol{W}\cdot\boldsymbol{h}_3)+\sigma(\alpha_{15}\cdot\boldsymbol{W}\cdot\boldsymbol{h}_5)+\sigma(\alpha_{16}\cdot\boldsymbol{W}\cdot\boldsymbol{h}_6)\tag{3-23}$$

本质上而言：GCN 与 GAT 都是将邻居节点的特征聚合到中心节点上，利用图数据上的局部连接性学习新的节点特征表达。不同的是 GCN 利用了拉普拉斯矩阵（$\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$）计算得到聚合过程中每个节点的权重，这个过程是一个预定义的数学方法，不涉及可学习参数，而 GAT 利用 attention 系数（α_{ij}），这其中设计了两个神经网络层的可学习参数 $\boldsymbol{W}$ 和 $\boldsymbol{\alpha}$。

最后探讨一下为什么 GAT 适用于 Inductive 任务，因为 GAT 聚合邻居特征的时候仅需考虑邻居特征，训练出来的参数 $\boldsymbol{W}$ 和 $\boldsymbol{\alpha}$ 是对邻居特征的线性变换参数矩阵，这个参数矩阵针对每个节点的每个邻居都是一样的，也就是所谓的共享权重参数。这些仅与节点特征相关，没有直接用到邻接矩阵进行计算，所以在测试任务中改变图的结构，对于 GAT 的影响并不大，只需更新节点的邻居集合 N_i，然后用 GAT 模型重新计算注意力系数和节点的表示。这里的重新计算并不是指重新训练模型，只需用训练好的权重参数对更新后的邻居集合进行前向传播。这是因为在前向传播过程中，注意力系数是基于节点和邻居的特征动态计算的，而不是静态地依赖于原始图的结构，因此即使图的结构发生了变化，这些计算仍

然是有效的，且在 GAT 的训练中，由于权重参数是在不同节点间共享的，并且在聚合过程中主要依赖于节点的特征而非图的全局结构，所以这些权重能够根据节点间的特征对某些属性进行预测，并不强依赖于图的结构，使模型能够很好地泛化到新的未见过的图结构上。

3.4 进阶图神经网络

3.4.1 进阶图神经网络概述

在讲解进阶图神经网络(Graph Isomorphism Network，GIN)之前，有必要先学习及了解 GNN 模型的表达能力(Expressive Power)，即 GNN 将不同图数据表示为不同嵌入向量的能力。这里涉及一个被称为计算图的概念，图神经网络(GNN)的计算图是一种数据结构，它表示图中的节点及它们之间的连接关系，用于有效地实施图数据上的学习算法。在这种结构中，每个节点聚合其邻居节点的信息，通过多次迭代更新其状态，从而捕捉和利用图的局部连接性质。为了研究 GNN 模型的表达能力，先来了解一下图数据的局部信息结构，如图 3-39 所示。

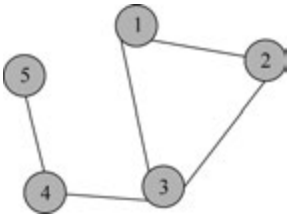


图 3-39 图结构示例

考虑图 3-39 两跳(2-hop)范围内计算图的几种特殊情况：

(1) 节点 1 和节点 4 具有相同的度数，但到其两跳(2-hop)邻居的信息上度数不同，导致计算图结构不同，因此可以区分这两个节点，如图 3-40 所示。

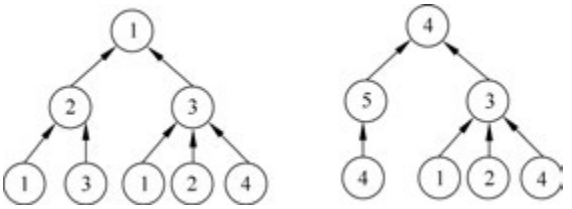


图 3-40 节点 1 和 4 的计算图

(2) 节点 1 和节点 5，因其自身度数不同而具有不同的局部结构信息，如图 3-41 所示。

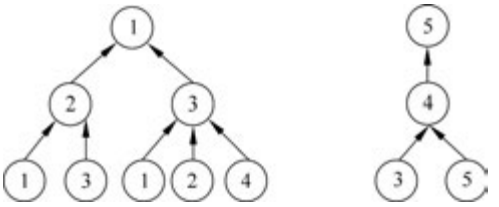


图 3-41 节点 1 和 5 的计算图

(3) 节点 1 和节点 2 既具有相同的度数，也具有相同的邻居结构，在两跳邻居上都具有相同的局部结构，如图 3-42 所示。

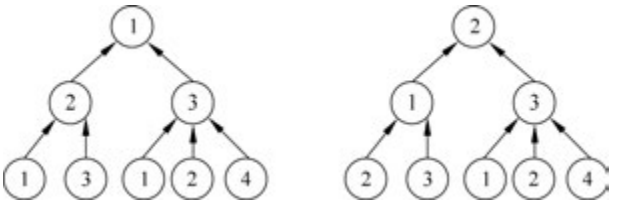


图 3-42　节点 1 和 2 的计算图

在图 3-42 的情况下，假设所有节点的初始化特征向量相同，并且聚合节点信息时仅考虑两跳范围内的信息，那么节点 1 和节点 2 将被嵌入同一个表示向量，GNN 无法区分这两个节点。相比之下节点 3、节点 4、节点 5 由于计算图不同，理论上是可能由 GNN 区分开的。GNN 模型应该有将不同的计算图映射到不同的节点向量的能力。想要区分不同的计算图，聚合的方式是关键，聚合函数表达能力越强，GNN 表达能力越强。

然而，不管是求和、求平均，还是求最大值这些广泛被使用的聚合方式都不是最佳的选择，如图 3-43 所示，两种图数据的计算图是不同的，但针对图 3-43(a)的初始化节点向量，求平均值和求最大值的聚合方式都不能有效地区分出不同的计算图。类似地，对于图 3-43(b)的初始化节点向量，求和的聚合方式也不能有效地区分这两种计算图。这是因为图通过聚合节点向量后却得到了相同的新的节点向量。

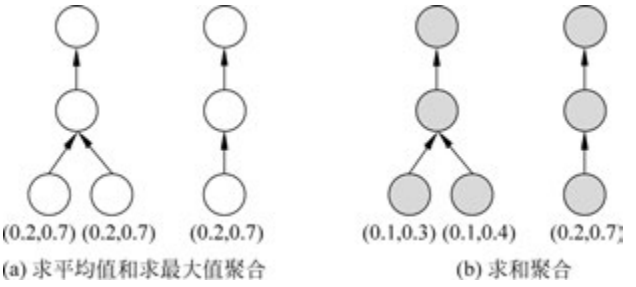


图 3-43　求和、求平均值和求最大值的聚合示例

从数学的角度上来讲，希望 GNN 的聚合函数是一种单射函数(Injective Function)，即可以把不同的元素映射成不同输出。换句话说，希望 GNN 的聚合函数能把不同的计算图映射成不同的节点向量。显然，求和函数、求平均值函数、求最大值函数都不是单射函数。根据 Xu et al. 在论文“Graph Wavelet Neural Network”中得到定理可知：任一单射函数都可以表示为

$$\phi\left(\sum_{x\in S} f(x)\right) \tag{3-24}$$

其中，ϕ 和 f 表示某种非线性函数，S 是输入信息 x 的集合。问题的关键是这个非线性函数 ϕ 和 f 该怎么求解？此时深度学习的魅力来了，对于神经网络模型而言，理论上它可以拟合任意连续函数，所以既然不清楚要求解的函数 ϕ 和 f 是什么，可以直接通过神经网络让它自己学，

通过定义损失，使用梯度下降更新网络的参数就好了。此时，这个单射函数的形式变成了：

$$\mathrm{MLP}_{\phi}\left(\sum_{x\mathrm{inS}}\mathrm{MLP}_{f}(x)\right) \tag{3-25}$$

使用式(3-25)这个函数作为聚合函数的图神经网络被称为 Graph Isomorphism Network (GIN)。在 GIN 的实际实现中，ϕ 和 f 这两个非线性函数是通过一个 MLP 实现的，因为 MLP 本身可以表示函数的组合。在第 1 次迭代中，如果输入特征是独热编码形式，在求和之前不需要 MLP，因为独热编码求和本身是单射的，最后，使用一个可学习参数 ϵ（也可以是一个固定的缩放因子）来决定聚合节点自身向量时的比重。完整的 GIN 层公式如下：

$$h_v^{(k)}=\mathrm{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot h_v^{(k-1)}+\sum_{u\in N(v)}h_u^{(k-1)}\right) \tag{3-26}$$

在式(3-26)中，每个节点 v 的新表示 $h_v^{(k)}$ 在第 k 轮迭代中是通过以下步骤经计算得出的：首先对邻居信息进行聚合，对节点 v 的所有邻居节点 u 的当前表示 $h_u^{(k-1)}$ 进行求和聚合，这表示为 $\sum_{u\in N(v)}h_u^{(k-1)}$，其中 $N(v)$ 是节点 v 的邻居节点集合，然后加上节点 v 自身在上一轮迭代中的信息 $h_v^{(k-1)}$，并将其乘以一个权重 $1+\epsilon^{(k)}$。这里的 $\epsilon^{(k)}$ 是在第 k 轮迭代中一个可学习的参数，它允许模型调整自身节点信息与邻居信息的相对重要性。最后，将上述两部分的和作为输入传递给一个多层感知器 $\mathrm{MLP}^{(k)}$，这是一个可学习的非线性函数，用来生成节点 v 的新表示 $h_v^{(k)}$。如果输入特征不是独热编码形式，则存在重复的特征向量表示，GIN 的形式可以写成：

$$h_v^{(k)}=\mathrm{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot h_v^{(k-1)}+\sum_{u\in N(v)}\mathrm{MLP}(h_u^{(k-1)})\right) \tag{3-27}$$

从式(3-27)可以看出，如果输入特征可能存在重复的特征向量表示，那么只需在聚合完节点 v 的邻居信息之后，在与节点 v 自身信息求和之前，对每个邻居节点的表示先进行一个 MLP 变换。

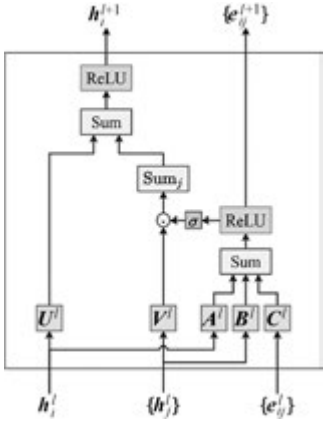


图 3-44 Gated GCN

3.4.2 Residual Gated Graph Convents

Gated GCN 结合了 GCNs 的空间聚合能力和 LSTM 中的门控机制。这种结构设计旨在解决传统 GCN 在处理复杂图数据时面临的信息过载和信息损失问题。每个节点的状态更新不仅依赖于其邻居信息，还受到门控机制的控制。这种操作使网络可以在每层学习到如何最有效地聚合邻近节点的信息，同时根据任务的需要丢弃无关信息，使其能够适应不同的图结构和动态变化的图数据，提供更精细的信息处理能力。Gated GCN 的模型设计如图 3-44 所示。

从图 3-44 中可以看到节点特征向量 $\boldsymbol{h}_i$ 和边特征向量 $\boldsymbol{e}_{ij}$ 的具体数据更新流程。首先每个节点 i 在每层 l 有

一个特征表示 $\boldsymbol{h}_i^l$，节点 i 邻居节点的特征向量集合为 $\{\boldsymbol{h}_j^l\}$。每条边 (i,j) 在每层 l 也有一个特征表示 $\boldsymbol{e}_{ij}^l$。节点特征 $\boldsymbol{h}_i^l$ 通过矩阵 $\boldsymbol{U}^l$ 和 $\boldsymbol{A}^l$ 转换。相邻节点的特征 $\{\boldsymbol{h}_j^l\}$ 通过矩阵 $\boldsymbol{V}^l$ 和 $\boldsymbol{B}^l$ 转换。边特征 $\{\boldsymbol{e}_{ij}^l\}$ 通过矩阵 $\boldsymbol{C}^l$ 转换，这里的矩阵实际上是神经网络层中的参数矩阵。节点特征 $\boldsymbol{h}_i^l$、邻居节点的特征 $\{\boldsymbol{h}_j^l\}$ 和边特征 $\{\boldsymbol{e}_{ij}^l\}$ 分别经过 $\boldsymbol{A}^l$、$\boldsymbol{B}^l$ 和 $\boldsymbol{C}^l$ 映射后的新特征表示将进行求和(Sum)操作，这一步的含义是进行边特征与节点特征的信息交互和融合，求和后的结果会经过激活函数 ReLU 处理。处理后的结果一方面作为更新后的边特征 $\{\boldsymbol{e}_{ij}^{l+1}\}$；另一方面将经过 Sigmoid 函数 σ 的处理产生一个门控信号，这个门控信号与相邻节点特征的加权(经过 $\boldsymbol{V}^l$ 变换后的 $\boldsymbol{h}_j^l$)和相乘，实际上是对每个邻居节点的贡献进行缩放。这就是门控机制，它控制着信息的流向，与 LSTM 中的门控机制类似。最终，门控后的信息被累加到当前节点特征上(经 $\boldsymbol{U}^l$ 变换后的 $\boldsymbol{h}_i^l$)，通过一个求和(Sum)操作。结果再次通过 ReLU 激活函数得到当前层的输出结果。

经过上述处理后得到的结果 $\boldsymbol{h}_i^{l+1}$ 是节点 i 在下一层 $l+1$ 的新特征表示。同时，边的特征也更新为 $\boldsymbol{e}_{ij}^{l+1}$。这个过程体现了 Gated GCN 在图卷积中加入门控机制的重要性。通过调节门控信号，模型能够控制每个节点在其邻居之间传递的信息量，这样就可以捕捉到图中更加复杂的结构模式，同时减少不必要的信息传递，提高了模型的泛化能力和处理大规模图数据的效率。

3.4.3 Relational Graph Convolutional Neural Network

对比 GCN 讲解 RGCN 的基本思想。GCN 通过将图的邻接矩阵与节点特征结合，利用统一的权重矩阵 $\boldsymbol{W}$ 来更新节点的特征，从而捕获图中的拓扑结构信息。这种方法在处理同质图时非常有效，即所有节点和边被假定具有相同的类型和属性，然而，GCN 不直接处理图中的异质性或复杂的关系类型。

与 GCN 不同，RGCN(Relational Graph Convolutional Neural Network)被设计用来处理具有多种关系的异构图。此时，节点或边的种类有很多种情况，每种类型可能有其特定的属性和语义，例如，在生物医学的异构图中，不同类型的节点(例如药物分子、疾病名称、蛋白质名称、生物通路等)和边(例如促进、抑制、治疗、包含等)包含丰富的语义信息。RGNN 能够对这些不同类型的实体和关系分别学习表示，这样能够更好地理解它们各自及它们之间的复杂关系。用公式表示为

$$G=(V,E,R,T) \tag{3-28}$$

其中，V 为节点集，每个节点在下文表示为 v_i；E 为边集，边 $(v_i,r,v_j)\in E$；$T(v_i)$ 为节点类型；R 为边类型集合，r_i 在下文表示每条边的属性。

【例 3-3】 边有 3 种不同的类型 (r_1,r_2,r_3)，分别用 3 种不同的线段格式表示，如图 3-45 所示，例如，节点 A 通过类型为 r_1 的边与节点 B 和 D 相连，通过类型为 r_2 的边与节点 C 相连等。图 3-45 中也展示了关系图卷积网络(RGCN)的一个关键特点：不同类型的边使用不同的权重矩阵 $(\boldsymbol{W}_{r1},\boldsymbol{W}_{r2},\boldsymbol{W}_{r3})$ 来处理信息。这意味着，当更新一个节点的特征时，

RGCN 会根据边的类型使用不同的权重矩阵。值得注意的是，在这个例子中，r_i 有 3 种类型，但这并不意味着要维护 3 个独立的神经网络。实际上，$\boldsymbol{W}_{r1}$，$\boldsymbol{W}_{r2}$，$\boldsymbol{W}_{r3}$ 可以共同组成一层神经网络的参数。假设用于更新边信息的神经网络输入/输出都是由 12 个神经元组成的网络层，那么第 1～3 个神经元可以用于更新第 1 种边类型；第 4～6 个神经元用于更新第 2 种边类型；第 7～9 个神经元用于更新第 3 种边类型。

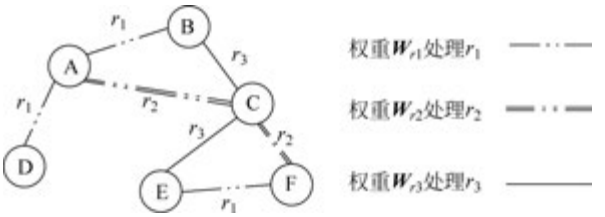


图 3-45　节点映射

RGCN 为每种类型的边引入了不同的权重矩阵，以此来捕获不同关系类型之间的细微差异。每种关系类型的权重矩阵负责编码特定类型的边对节点特征更新的贡献。这样，RGCN 能够综合节点的特征及与其相连的各种类型边的信息，生成更丰富的节点表征。实际上，RGNN 更像是一种思想，而不是固定的模型，它可以扩展到其他 GNN 模型上，例如 GraphSAGE、RGAT 等。节点 i 的特征更新规则在 RGCN 中可以表示为

$$\boldsymbol{h}_i^{(l+1)}=\sigma\left(\sum_{r\in R}\sum_{j\in N_r(i)}\frac{1}{c_{i,r}}\boldsymbol{W}_r^{(l)}\boldsymbol{h}_j^{(l)}\right) \tag{3-29}$$

其中，

- $\boldsymbol{h}_j^{(l)}$ 是节点 j 在第 l 层的特征向量。
- $\boldsymbol{W}_r^{(l)}$ 是关系类型 r 对应的权重矩阵，在第 l 层中用于转换特征。
- $N_r(i)$ 表示与节点 i 相连的特定关系类型 r 的邻居节点集合。
- $c_{i,r}$ 是一个归一化项，通常是节点 i 的类型 r 关系的邻居数量，用于平衡不同节点之间不同数量邻居的影响。
- σ 是一个非线性激活函数，例如 ReLU。

在每层中，对于每种类型的关系 r，首先使用权重矩阵 $\boldsymbol{W}_r^{(l)}$ 来转换与节点 i 相连的每个邻居节点 j 的特征，然后我们对所有这些转换后的特征进行加权求和，其中每个邻居的特征向量除以归一化常数 $c_{i,r}$，以避免不同数量的邻居对聚合结果的不均衡影响。最终，我们对所有关系类型的累加结果应用非线性激活函数，以得到节点 i 在下一层的新特征向量 $\boldsymbol{h}_i^{(l+1)}$。这种机制允许 RGCN 分别学习不同类型的关系如何影响节点特征，而不是像 GCN 那样对所有边一视同仁，或者像 GraphSAGE 那样只是简单地聚合邻居节点的特征。

3.4.4　Deep Graph Infomax

Deep Graph Infomax(DGI)是一种基于图的无监督学习方法，其主要目标是学习图中节点的有效表示，即图数据的 Embedding。DGI 由 Petar Veličković 等在 2019 年的论文中

提出，这种方法通过最大化局部图结构（节点）和全局图结构（整幅图）之间的互信息来学习节点表示，DGI 整体的模型计算流程如图 3-46 所示。

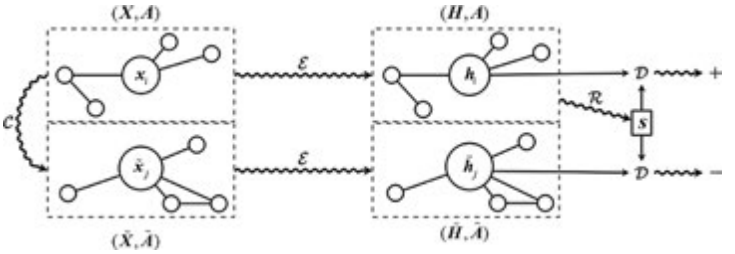


图 3-46 DGI 的模型架构

从图 3-46 可以看出，第 1 步，算法需要一个正常图$(\boldsymbol{X},\boldsymbol{A})$的扰动版本$(\widetilde{\boldsymbol{X}},\widetilde{\boldsymbol{A}})$，扰动版本通过负样本操作$C$来实现。负样本操作可以通过多种方式实现，本质是一个函数。这个函数会接受图的节点特征 $\boldsymbol{X}$ 和邻接矩阵 $\boldsymbol{A}$ 作为输入，然后对原图结构进行变换，例如随机打乱节点的特征，或者在邻接矩阵中随机重排列边，其目的是创建一个与原始图在统计特性上有显著区别的样本，这个负样本在后续步骤中用来帮助判别器学习如何区分图的正样本表示和与图不相关的表示。

第 2 步使用编码器函数 $\varepsilon(\boldsymbol{X},\boldsymbol{A})$处理原始图的每个节点，从而得到每个节点的向量表示。这里的编码器 ε 通常是一个图神经网络（例如 GCN），它能够考虑节点的特征及节点之间的连接（通过邻接矩阵 $\boldsymbol{A}$）来产生一个低维而信息丰富的特征向量 $\boldsymbol{h}_i$。编码器的目的是捕获节点的局部图结构和节点特征，生成的集合$(\boldsymbol{H},\boldsymbol{A})=\boldsymbol{h}_1,\boldsymbol{h}_2,\cdots,\boldsymbol{h}_n$ 包含了输入图所有节点的这些表示。同理，在上一步中采样的负样本也需要经过编码器 ε 的处理，从而得到对应的$(\widetilde{\boldsymbol{H}},\widetilde{\boldsymbol{A}})$。

第 3 步，算法需要从整个输入图的节点表示中提取一个单一的图级别的表示 $\boldsymbol{s}$。这通常通过一个汇总函数 $R(\boldsymbol{H},\boldsymbol{A})$完成，它可以是简单的平均节点表示，或者一个更复杂的结构，例如基于注意力的汇总方法。这个图级别的表示 $\boldsymbol{s}$ 应该捕获整幅图的全局性质，它将在损失函数中用于比较正样本和负样本的节点表示。

最后一步是模型学习的关键。在这一步中，通过梯度下降来更新编码器 ε、汇总函数 R 及判别器 D 的参数。判别器也是一个图神经网络，其任务是根据节点表示（$\boldsymbol{h}_i$ 或 $\boldsymbol{h}_j$）和图表示来判断节点是否属于该图。损失函数 L 在正样本的情况下鼓励判别器给出高的得分，而在负样本的情况下给出低的得分。通过最大化损失函数中定义的互信息量，模型能够训练出区分原始图节点和负样本节点表示的能力。

这个训练过程的目标是调整模型的参数，使从正样本图中提取的节点表示与图的全局表示尽可能地接近，而从负样本图中提取的节点表示则与图的全局表示尽可能远，即最大化节点表示和对应图级表示的互信息。这种方法认为，如果节点表示能够捕捉到与整幅图表示高度相关的信息，那么这些节点表示就是有用的。这便完成了 DGI 的主要目标：学习图中节点的有效表示，即图数据的 Embedding。

3.1.2 节介绍的随机游走算法的目的也是学习图中节点的有效表示，并且与 DGI 同属无监督学习方法，但在方法和理念上存在显著差异。随机游走侧重于通过模拟从一个节点开始的随机路径来捕捉图中节点的局部连接模式，以探索和利用图的局部结构特性。这种方法通过观察节点间的“游走”行为，直接反映了节点的邻近关系。相比之下，DGI 采用基于互信息最大化的策略，通过学习节点表示和全局图表示之间的互信息来提取特征，不仅考虑了局部结构，还尝试捕捉整幅图的全局上下文。DGI 通常使用图神经网络（例如图卷积网络（GCN））作为编码器，更侧重于深度学习技术。总体而言，随机游走方法更加直观和简单，适用于图的局部特征学习，而 DGI 则是一种更复杂的深度学习方法，能够综合局部与全局信息，适合于需要深层特征提取的复杂图形任务。

此外，DGI（Deep Graph Infomax）和生成对抗网络虽然都涉及一个判别器组件，但它们的目标和训练过程有本质的不同。GAN 由两个模型组成：生成器和判别器。生成器的目标是创建尽可能地接近真实数据的新数据实例，而判别器则试图区分真实数据和生成器创建的伪造数据。这两个模型在训练过程中相互竞争，生成器不断提高其生成的数据质量以试图“欺骗”判别器，而判别器也不断提高其识别真伪数据的能力。这种对抗性的训练过程导致了生成器学会生成高质量的数据。DGI 的目的不是生成新的数据，而是从图数据中学习有用的节点表示。在 DGI 中，判别器的作用不是与编码器对抗，而是帮助编码器学习生成高质量的节点嵌入。DGI 使用一个非对抗的信息最大化原则，这个原则指导编码器生成的节点表示要包含足够的信息以便判别器可以准确区分节点是来自原始图还是扰动后的图（正样本或负样本）。换句话说，判别器用于指导编码器捕捉到有意义的特征——使这些特征对于区分不同图结构是有信息的。DGI 的目标是通过这种方式提升表示的质量，而不是在编码器和判别器之间创建对抗。

3.4.5 GraphGAN

GraphGAN 是一种结合了图神经网络和生成对抗网络概念的机器学习模型。它旨在通过对抗学习框架解决图数据中的节点分类和链接预测问题，特别是在缺少标签数据的情况下。GraphGAN 通过结合生成器和判别器这两个主要组件来学习图中节点的有效表示。

GraphGAN 的框架：设 $G=(V,\mathcal{E})$ 为一幅给定的图，其中 $V=\{v_1,v_2,\cdots,v_V\}$ 代表顶点的集合，而 $\mathcal{E}=\{e_{ij}\}_{i,j=1}^{V}$ 代表边的集合。对于给定的顶点 v_c，将 $\mathcal{N}(v_c)$ 定义为直接连接到 v_c 的顶点集，也就是顶点 v_c 的 1-hop 邻居。我们将顶点 v_c 的真实连通性分布表示为条件概率 $p_{\text{true}}(v|v_c)$，它反映了 v_c 的连通性偏好和它所连接顶点的类型。从这个视角来看，$\mathcal{N}(v_c)$ 可以被看作从 $p_{\text{true}}(v|v_c)$ 抽取的一组观察样本。给定图 G，GraphGAN 旨在学习以下两个模型。

（1）生成器 $G(v|v_c;\theta_G)$：尝试近似顶点 v_c 的真实连通性分布 $p_{\text{true}}(v|v_c)$，并生成（或选择）最有可能与 v_c 相连接的顶点集。

（2）判别器 $D(v,v_c;\theta_D)$：旨在鉴别顶点对 (v,v_c) 的连通性。$D(v,v_c;\theta_D)$ 输出一个标量值，表示边 (v,v_c) 存在的概率。

生成器 G 和判别器 D 作为两个对手：生成器 G 将试图完美地匹配 $p_{\text{true}}(v|v_c)$ 并生成类似于 v_c 的真实直接邻居的相关顶点以欺骗判别器，而判别器 D 则相反，它会尝试检测这些顶点是 v_c 的真实邻居还是由其对手 G 生成的。形式上，G 和 D 在一个有以下价值函数 $V(G,D)$ 的双人博弈中进行对抗：

$$\min_{\theta_G}\max_{\theta_D}V(G,D)=\sum_{c=1}^{V}(E_{v\sim p_{\text{true}}(v|v_c)}[\log D(v,v_c;\theta_D)]+E_{v\sim p_G(v|v_c;\theta_G)}[\log(1-D(v,v_c;\theta_D))]) \tag{3-30}$$

式(3-30)描述的最小和最大(Minimax)博弈公式是生成对抗网络的核心。在图生成对抗网络(GraphGAN)的上下文中，生成器 G 的目标是生成看起来像真实数据的样本。在 GraphGAN 的情境下，生成器试图生成看似是 v_c 的 1-hop 邻居的顶点，而鉴别器 D 的目标是区分输入样本是来自真实数据还是生成器生成的假数据。在 GraphGAN 中，它试图区分一个顶点对 (v,v_c) 是否是真实的 1-hop 邻点对。这个公式包含了两个期望(Expectations)：

第一部分为真实数据的期望 $E_{v\sim p_{\text{true}}(v|v_c)}[\log D(v,v_c;\theta_D)]$。这个期望表示对所有真实的顶点对 (v,v_c)，鉴别器 D 输出它们是真实邻居的概率的对数。v 是从顶点 v_c 的真实邻居分布 $p_{\text{true}}(v|v_c)$ 中采样的。在理想情况下：如果 v 确实是 v_c 的真实邻居，则希望鉴别器 D 的输出 $D(v,v_c;\theta_D)$ 接近于 1(最大的可能性)。此时 $\log D(v,v_c;\theta_D)$ 会接近 0(因为 $\log(1)=0$)。

第二部分是关于生成数据的期望 $E_{v\sim p_G(v|v_c;\theta_G)}[\log(1-D(v,v_c;\theta_D))]$。这个期望表示对所有生成器 G 生成的顶点对 (v,v_c)，鉴别器 D 输出它们不是真实邻居的概率的对数。v 是从生成器 G 的分布 $p_G(v|v_c;\theta_G)$ 中采样的。在理想情况下：如果 v 是由生成器 G 生成的，并不是 v_c 的真实邻居，希望判别器 D 的输出 $D(v,v_c;\theta_D)$ 接近于 0。此时，$\log(1-D(v,v_c;\theta_D))$ 也会接近 0(因为 $\log(1-0)=0$)。

可以看出这两部分期望的目标都是希望经过训练后，其值越来越小，在理想情况下为 0，因此可以看作 GraphGAN 的损失函数，GraphGAN 的训练状态如图 3-47 所示。

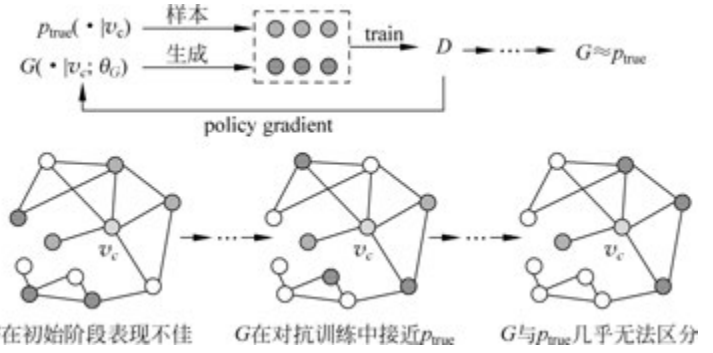


图 3-47 GraphGAN 训练状态

阴影的圆代表从真实图结构中采样的真实与顶点 v_c 连接的邻居，横条纹的圆代表生成器认为的应该与 v_c 连接的邻居。在训练的初始阶段(左图)，生成器 G 的表现并不佳，其生成的顶点与实际的邻居相比，差异较大，很容易被鉴别器 D 区分。随着对抗训练的进行(中

图),生成器 G 逐渐学习到更好的生成策略,使生成的邻居和实际的邻居更难以区分。在经过充分训练之后,生成器 G 生成的邻居与实际的邻居非常接近,即右图中,阴影的圆和横条纹的圆存在大量重叠。鉴别器 D 几乎无法区分。在这个博弈中,鉴别器尽量提高识别真实和生成数据对的准确性,而生成器尽量生成看起来更真实的数据。通过这种对立的训练过程,两个模型都在不断改进,直到达到一个平衡点,即生成器生成的数据非常逼真,鉴别器很难区分真假数据对。

下面具体来讲解 GraphGAN 的优化过程,对于判别器而言,定义鉴别器 D 的输出为两个输入顶点(v,v_c)内积的 Sigmoid 函数:

$$D(v,v_c)=\sigma(\boldsymbol{d}_v^{\mathrm{T}}\boldsymbol{d}_{v_c})=\frac{1}{1+\exp(-\boldsymbol{d}_v^{\mathrm{T}}\boldsymbol{d}_{v_c})} \tag{3-31}$$

定义 θ_D 是判别器的可学习参数。任何具有区分能力的模型都可以在此作为 D,例如一个神经网络模型,对顶点 v 和 v_c 的特征向量进行学习映射后得到的 k 维新的向量表示为 $\boldsymbol{d}_v,\boldsymbol{d}_{v_c}$,然后进行内积和 Sigmoid 处理。这个内积表示两个顶点之间的相似性或关系强度。内积的结果通过 Sigmoid 函数进行处理,得到一个在 0 到 1 的值,可以解释为顶点对 v 和 v_c 是图中真实连接的概率,其中 v 的采样有两种情况,换句话说,判别器随机接收两类输入数据($v\sim p_{\text{true}}$ 或 $v\sim G$)。根据损失函数式(3-30),判别器的梯度$\nabla_{\theta_D}V(G,D)$计算如下:

$$\nabla_{\theta_D}V(G,D)=\begin{cases}\nabla_{\theta_D}\log D(v,v_c), & \text{如果 } v\sim p_{\text{true}}\\ \nabla_{\theta_D}(1-\log D(v,v_c)), & \text{如果 } v\sim G\end{cases} \tag{3-32}$$

在训练过程中,GraphGAN 交替地更新生成器 G 和判别器 D 的参数,以便最小化生成器的损失函数和最大化判别器的损失函数,这个过程通常被称为对抗训练。式(3-32)便是固定生成器 G 更新判别器 D 时判别器的梯度计算方法。

类似地,生成器 G 具体的模型类型也没有限制,GCN、GAT 甚至可以是简单的神经网络等,只要能具备预测某顶点与其他顶点间连接概率的能力即可。当固定判别器 D 并更新生成器 G 的参数时,式(3-30)中的 $E_{v\sim p_{\text{true}}(\cdot|v_c)}[\log D(v,v_c;\theta_D)]$部分可以看作常数,只需关注另一部分,即 $E_{v\sim G(\cdot|v_c;\theta_G)}[\log(1-D(v,v_c;\theta_D))]$。此时,生成器的梯度$\nabla_{\theta_G}V(G,D)$为

$$\begin{aligned}\nabla_{\theta_G}V(G,D)&=\nabla_{\theta_G}\sum_{c=1}^{V}E_{v\sim G(\cdot|v_c)}[\log(1-D(v,v_c))]\\&=\sum_{c=1}^{V}\sum_{i=1}^{N}\nabla_{\theta_G}G(v_i\mid v_c)\log(1-D(v_i,v_c))\\&=\sum_{c=1}^{V}\sum_{i=1}^{N}G(v_i\mid v_c)\nabla_{\theta_G}\log G(v_i\mid v_c)\log(1-D(v_i,v_c))\\&=\sum_{c=1}^{V}E_{v\sim G(\cdot|v_c)}[\nabla_{\theta_G}\log G(v\mid v_c)\log(1-D(v,v_c))]\end{aligned} \tag{3-33}$$

在式(3-33)的推导中，$\sum_{c=1}^{V} E_{v\sim G(\cdot|v_c)}[\log(1-D(v,v_c))]$用于对所有顶点$v_c$的期望求和，其中每个期望是从生成器$G$生成的顶点$v$对应的$\log(1-D(v,v_c))$的平均值。这表明我们在考虑所有生成器产生的假顶点与真实顶点v_c之间关系的平均对数概率。期望的实现可以转化成对每个生成样本求和，即$\sum_{i=1}^{N} \nabla_{\theta_G} G(v_i \mid v_c)\log(1-D(v_i,v_c))$，这里对每个生成样本$v_i$求和，计算生成概率相对于$\theta_G$的梯度与$\log(1-D(v_i,v_c))$的乘积。又因为概率的梯度可以写作概率乘以似然比例的梯度，因此可以使用$G(v_i \mid v_c)\nabla_{\theta_G}\log G(v_i \mid v_c)\log(1-D(v_i,v_c))$代替$\nabla_{\theta_G} G(v_i \mid v_c)\log(1-D(v_i,v_c))$。最后，我们用期望值来代替求和，因为期望值是随机变量的平均值，这里的随机变量是v的生成概率，得到推导后生成器的梯度计算公式：$E_{v\sim G(\cdot|v_c)}[\nabla_{\theta_G}\log G(v \mid v_c)\log(1-D(v,v_c))]$。整体上，这个梯度表达式说明：为了更新生成器$G$的参数，需要考虑由$G$生成的每个顶点$v$对损失函数的贡献，并根据该贡献来调整$\theta_G$。梯度的方向指示了如何更新参数$\theta_G$以减少生成的顶点$v$被判别器$D$识别出的概率，从而使生成器产生更真实的样本。

值得注意的是，在给定顶点v_c时，生成器G计算其他顶点与v_c的连通性概率$G(v|v_c)$需要使用Softmax函数计算出v_c外的所有节点。公式如下。

$$G(v \mid v_c) = \frac{\exp(\boldsymbol{g}_v^{\mathrm{T}}\boldsymbol{g}_{v_c})}{\sum_{v\neq v_c}\exp(\boldsymbol{g}_v^{\mathrm{T}}\boldsymbol{g}_{v_c})} \tag{3-34}$$

当图数据体量庞大时，上述公式的计算复杂度是非常高的，因此在GraphGAN框架中，提出了一种称为Graph Softmax的新方法，它的核心思想主要有3点。

(1) 规范化(Normalized)：生成器应该产生一个有效的概率分布，这意味着对于一个给定的顶点v_c，所有可能与之相连的顶点v的生成概率之和应该等于1，这也是Softmax函数的基本思想。

(2) 图结构意识(Graph-structure-aware)：生成器在确定顶点之间的连通性概率时，应该利用图的结构信息。具体来讲，如果两个顶点之间的最短路径距离增加，则它们被认为是相连的概率应该下降。

(3) 计算效率(Computationally Efficient)：与Softmax不同，Graph Softmax在计算时只考虑图中少数的特定顶点，从而提高计算效率。

为了实现Graph Softmax，首先需要执行一个从顶点v_c出发的广度优先搜索(Breadth-First Search，BFS)，以此来构建一棵以v_c为根的BFS树T_c。使用这棵树，可以计算每个邻居顶点v_i相对于v_c的相关性概率$p_c(v_i)$，这是一个标准的Softmax函数，它在v_c的直接邻居集$N_c(v)$上运算。接着，为了计算任意顶点v与v_c之间的连通性概率$G(v|v_c;\theta_G)$，我们使用T_c中从v_c到v的唯一路径。路径上每对相邻顶点的相关性概率的乘积定义了v和v_c之间的连通性概率。通过这种方式，图Softmax只关注v_c通过BFS可达的部分图，而不是整幅图，这极大地提高了计算效率。同时，它也自然地考虑到了图的结构信息，因为它直接在BFS

树上进行操作，这棵树反映了图中顶点之间的真实距离关系，因此这种方法是规范化的，具有图结构意识，并且计算效率高，具体示例如图 3-48 所示。

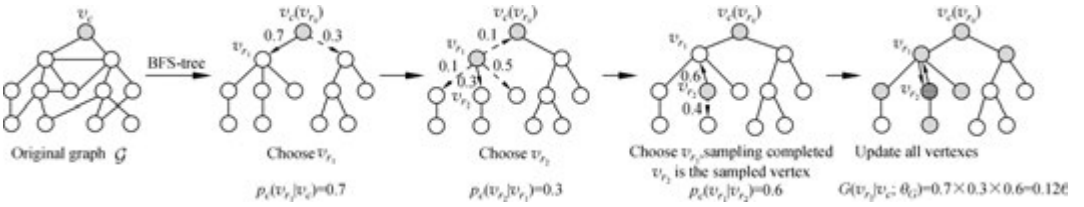


图 3-48　BFS 树

如图 3-48 所示，从顶点 v_c 开始，我们有原始的图 G 和一棵以 v_c 为根节点的 BFS 树。第 1 步是从 v_c 的直接邻居中选择下一个顶点。在这个例子中，顶点 v_{r_1} 被选中，与它相关联的概率是 0.7。接下来，从 v_{r_1} 的邻居中选择下一个顶点。顶点 v_{r_2} 被选中，与它相关联的概率是 0.3，然后从 v_{r_2} 的邻居中选择下一个顶点。可能的选择只有两种，与它们相关联的概率是 0.6 和 0.4，假设此时以 0.6 的概率选择了 v_{r_1}，那么 v_{r_2} 的路径采样过程完成，否则路径采样继续延续。最后一步是更新采样路径上所有顶点的概率。计算路径 $v_c \rightarrow v_{r_1} \rightarrow v_{r_2}$ 上所有顶点的相关性概率的乘积，从而得到 $0.7\times0.3\times0.6=0.126$，这个数字代表从 v_c 到 v_{r_2} 的连通性概率。