H-GCN 学习笔记

Linyxus

May 2019

目录

1 背景

1.1 问题与动机

H-GCN 的目的是解决图上的半监督分类任务。

图，作为一种表现力很强的数据结构，近来在数据科学领域备受青睐。他能够表示许多许多复杂的系统，如社交关系、生物分子和出版物的引用。在很多领域中，对图结构的数据进行分类是意义重大的。

不仅如此，由于自然中有大量未标记的图结构数据，而对数据进行标记经常非常昂贵并浪费时间。因而，对图结构的数据进行半监督学习很多时候是至关重要的。

而在图上的半监督学习是富有挑战性的。比如，对于文献引用网络的半监督分类任务，需要用极少的已标记数据预测所有文章的标签。

在很多相关的尝试中，图嵌入作为一种高效而有效的方式吸引了很多研究者的注意。在图嵌入方向，已有很多模型被提出。比如 Deepwalk 与 node2vec，以及多种不同形式的图卷积网络 *(GNNs)*。他们都在图嵌入领域取得了不错的表现，也各有进展。但他们都具有不容忽视的不足。

1. 一些模型，如 Deepwalk，是无监督的学习算法。他不能端到端 *(end-to-end)* 地进行节点分类任务。

2. 图卷积网络大多非常浅，也缺乏图池化 *(graph pooling)* 的手段，因此，图卷积网络的接受野一般不够大。

H-GCN 受深度网络结构与池化操作在图像识别领域的巨大成功启发，是一个具有粗化层 *(coarsening layer)* 的分级 *(hierarchical)* 深度模型。他拓宽了图卷积的感知野，能够更好的提取图的全局信息。

1.2 已有工作分析

在图模型领域，已有工作可以大概地分为三个方向：应用于半监督学习的图卷积网络、图上的分级表示学习 *(hierarchical representation learning)* 与图缩减 *(graph reduction)*。

图卷积网络 近年来，图卷积网络方向上的模型非常多。总的来说，这些模型都基于利用代理函数从邻居节点提取信息的策略。而这一方向上的模型可以更进一步地分为两种类别：基于谱图 *(spectral)* 的与基于空间 *(spatial)* 的。

基于谱图的模型 这一类模型利用谱图理论来定义卷积操作。这一类模型的缺陷在于，由于利用了矩阵分解等较为复杂的运算，他们带来了高昂的计算成本，使得他们很难被应用在大规模的图上。

基于空间的模型 这一类模型在节点空间内综合邻居节点的信息来生成节点的嵌入表达。这一方面也有很多工作，如 GraphSAGE 与 GAT。

不管是哪一类的 GCN 网络，大多有一个无法回避的不足：他们都属于较浅的网络，导致他们无法从全局中获得足够的信息来更好地处理节点的表达。而且，**不能通过简单地加深网络层数来解决他们的弊端**：Kipf 的研究中显示，简单地增加残差连接，加深他们的 GCN 模型并不能显著地提升模型性能——甚至会降低模型的表现。有理论分析表明，GCN 的每一层本质上是某种形式的拉普拉斯平滑 *(Laplacian smoothing)*。这会使相邻节点的嵌入表达趋同。因此，添加过多的图卷积层会导致输出的嵌入向量过于平滑而难以用来分辨不同节点。

图上的分级表示学习 在这一领域，也已经有了许多研究工作。但这些模型都或多或少地存在着不足，不能很好地解决半监督图节点分类的问题。

比如 Harp 与 MILE，他们都利用一些图粗化 *(coarsening)* 的方法把图转化为一张更小的图，随后在上面应用无监督的学习方法，如 Deepwalk 和 node2vec。最后，再进行图细化 *(refining)* 来得到原始节点的嵌入表达。因而，他们也继承了 Deepwalk 与 node2vec 的缺点：不能端到端地应用到节点分类任务中。

又如 JK-Nets，他们给出了一种图代理机制 *layer aggregation mechanism* 来综合不同 GCN 层的输出。然而，他们只能在图的边上传播信息，并不能有层次地传递信息。所以，JK-Net 不能学习图的层次结构。

为了解决 JK-Nets 的不足，DiffPool 提出了一种池化层，利用可微的网络减小图的大小。然而，DiffPool 是为图分类任务设计的，不能生成所有节点的嵌入表达，所以并不能被直接应用在节点分类任务中。

图缩减 图缩减的主要目的是在不损失太多信息的前提下缩减图的大小，为许多下游任务提供了便利，社群发现 *(community discovery)* 与数据总括 *(data summerization)*。图缩减方向的模型有两个主要类别：图采样与图粗化。

图采样基于一些图采样的策略。但可能会在采样的过程中损失一些关键信息。

而典型的图粗化有两个主要过程：分组 *(grouping)* 和展开 *(collapsing)*。分组过程将图节点分配到不同的组中，形成一个超节点 *(hyper-node)*。这些超节点便形成了一张更小，更粗 *(coarser)* 的图。而展开的过程，便是将超节点还原为组成节点的构成。图的原始结构也在这一过程中复原。

然而，这些图缩减的方法也有着显著的不足。首先，他们大多被应用在无监督学习任务中，不能直接应用到半监督的节点分类任务中。而且，他们也不具备学习图的复杂属性与结构信息的能力。

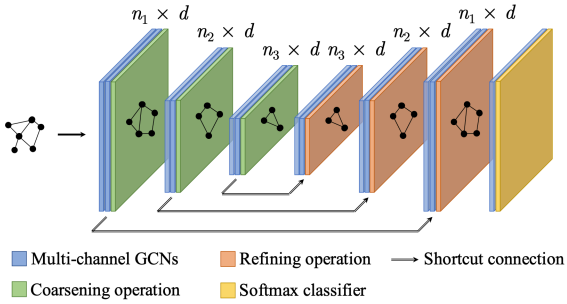
1.3 模型引入

H-GCN 的全称是 Hierarchical Graph Convolutional Network。H-GCN 是一种利用图粗化机制进行图节点分类任务的深层层次模型 *(deep hierarchical model)*。

H-GCN 的灵感源于近来将深层网络与池化操作应用于图像识别任务的热潮。他提高了图卷积的感知野 *(receptive field)*，可以更好地提取图的全局信息。

总的来说，H-GCN 由一些图粗化层与图细化层组成。

图 1: H-GCN 结构示意图



在图粗化层中，首先应用一次图卷积操作，来学习节点的嵌入表达。随后，应用图粗化操作，将结构上相似的节点整合为节点组，形成一个超节点。

在粗化操作完成后，由超节点组成的新图中，每一个超节点都表示着原图中的某种局部结构。这便于图全局结构特征的发掘。

在一系列粗化层之后，H-GCN 接着应用图细化层。图细化层与之前的图粗化层是对称的，他们能够从由超节点组成的粗化图中复原原图的结构，并得到各节点的嵌入表达。通过这种方式，H-GCN 可以全面地提取图节点的特征信息，从局部到全局，从而生成更好的嵌入表达。

H-GCN 的贡献主要有两方面。一方面，H-GCN 是第一个应用于半监督图分类任务的深度层次网络。他利用粗化与细化操作拓宽了图卷积的感知野，可以更好的提取图的全局信息。另一方面，在实验中，H-GCN 的表现稳定地超过了其他 state-of-art 模型。值得一提的是，H-GCN 相较于其他的模型，具有客观的性能提升，就算每一类别的数据都只有极少的部分被标记了。

由于 H-GCN 利用图粗化与细化操作可以整合节点不同层次的特征信息，故可以避免多次图卷积带来的过平滑问题。

2 模型介绍

2.1 模型概述

对于一个共有 l 层的 H-GCN 网络，他的总体结构是，最开始有 $\frac{l-1}{2}$ 层图粗化层，接着是 $\frac{l-1}{2}$ 层的图细化层，再加上最后的一层分类器。

H-GCN 中，图粗化层与图细化层是对称的。也即，第 i 层与第 $l+1-i$ 层输出的图结构完全相同。比如示意图中的七层 H-GCN，第三层的输入 $\mathcal{G}_3$ 与第五层的输入 $\mathcal{G}_5$ 在经过一次对称的粗化操作与细化操作后，他们的图结构显然是完全一致的。

图粗化层的主要流程是：首先，对输入的图与特征相邻进行图卷积，以提取图中的特征。随后，将相似节点聚合为超节点，得到一张更粗且有更少节点的图 $\mathcal{G}_i$ 以及其中节点的嵌入特征表达 U_{i+1}。

对称地，图细化层的主要流程也与之类似。首先对输入的图进行图卷积，随后，对图进行细化 *(refine)*，得到更为精细的图结构与展开出来的更多节点的嵌入特征表达。

H-GCN 也有许多细节值得一提。比如，由于网络层数较深，为了加快模型的训练过程，可以利用图粗化层与图细化层的对称性向模型中添加残差连接。

而由于图的结构在模型的不同层之间发生了变化，H-GCN 定义了节点权重 *(node weight)* 为超节点中包含节点的个数。进一步的，为了利用节点权重中所包含的图结构信息，H-GCN 中引入了节点权重嵌入，将节点权重转化为一个实值向量，与节点的嵌入表达拼接在一起作为层的输出。

也为了增强模型的表达能力，H-GCN 中应用了多通道图卷积，来从不同角度更全面地提取节点特征。

模型的最后一层是分类器，基于节点的嵌入特征向量预测节点属于不同类的概率。分类器的结构是一层 GCN 再加上一个 softmax 分类层。

2.2 GCN 结构

H-GCN 是基于 GCN 的，所以，H-GCN 中大量地使用到了 GCN 中的图卷积操作：

$$f(H^{(i)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}^{(i)}\hat{D}_{(i)}^{-\frac{1}{2}}H^{(i)}W^{(i)}\right),$$

其中，i 为层编号；σ 为激活函数，H-GCN 中使用了 ReLU；$\hat{A}^{(i)}$ 为输入图 $\mathcal{G}_i$ 带自环的邻接矩阵，而 $\hat{D}_{(i)}$ 为相应的度数矩阵；$H^{(i)}$ 即为输入的节点特征向量矩阵。

2.3 Multi-Channel GCN 结构

多通道图卷积可以在不同的子空间中获取特征，H-GCN 应用了多通道的图卷积以在一层网络中获取足够的信息。多通道图卷积的主要流程如下。

首先计算 c 个不同通道获取的嵌入特征表达 $[G_i^1, G_i^2, \cdots, G_i^c]$，随后，计算他们的加权和

$$G_i = \sum_{j=1}^{c} = w_j \cdot G_i^j,$$

其中，w_j 为通道 j 可训练的权重。

2.4 节点权重嵌入

在图的粗化过程中，结构相似的节点形成一个超节点，故而不同的超节点中可能含有不同数量的节点。H-GCN 定义超节点中含有节点数量为节点权重 *(node weight)*。

而根据设想，节点权重中也蕴含了粗化后图的层级信息。为了能够提取利用这些信息，H-GCN 采用了节点权重嵌入的方法，将节点权重转化为实数值向量，来将节点权重信息添加到节点的隐藏表示中。节点权重嵌入被应用到了每一层粗化与细化层中。

节点权重嵌入的具体实现方法为，生成一个随机嵌入向量 $V \in \mathbb{R}^{|T|\times p}$。其中，$T$ 为节点权重的集合，p 为节点权重嵌入向量的维度。为了得到节点权重的嵌入向量，只需要在嵌入矩阵 V 中查找即可。如若某个节点的节点权重在排序后是第三个，则 V 中的第三行会被选中作为该节点的节点权重 S_i。

最后，将 H_i 与 S_i 拼接得到的 $(n+p)$ 维向量即为层的输出。

2.5 图粗化层

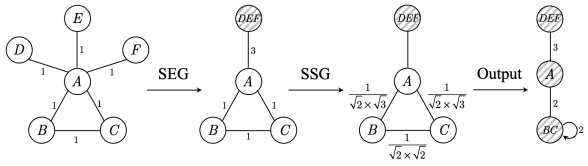
暂时不考虑在上一小节提到的节点权重嵌入，图粗化层输入为一张图 $\mathcal{G}_i$ 与相应的节点嵌入表达 H_i；输出为粗化后的图 $\mathcal{G}_{i+1}$ 与新图中各节点的嵌入表达 H_{i+1}。

图粗化层的主要过程分为两部分：图卷积与图粗化。

图卷积便是之前提到的 GCN 中的卷积操作。目的是从邻居节点提取特征与结构信息。

图粗化也主要有两个策略：首先是结构等同分组 *(structural equivalence grouping)*，随后是结构相似分组 *(structural similarity grouping)*。

图 2: 图粗化操作



SEG：结构等同分组 在 SEG 策略中，结构完全等同的节点会被划分到一组，形成一个超节点。怎样的节点会被认为是 “结构等同” 呢？如果两个节点的邻居节点完全相同，则认为他们属于结构等同的节点。

比如示意图中的 D, E, F 节点，便是结构等同的，他们形成了一个超节点。

SSG：结构相似分组 两个节点之间结构的相似度是这样定义的：

$$s(v_j, v_k) = \frac{A_{jk}}{\sqrt{D(v_j) \cdot D(v_k)}},$$

其中，A 为邻接矩阵，$D(\cdot)$ 代表节点权重。

在这一步中，相似度大的节点会被分到一组，形成超节点。比如示意图中的 B, C 具有最大的结构相似度，故而他们会一起形成超节点。

图粗化层的算法描述如图 3 所示。

值得一提的是，图粗化操作中，所有处理过（已分组并形成超节点）的节点会被标记，被标记的节点在接下来的操作中不再会被选中。也就是说，在图粗化操作中，每一个节点只会参与一次分组，并且产生的超节点不会继续参与分组。

可以知道，图粗化层的主要流程如下：

图卷积与初始化 利用图卷积计算出嵌入向量 G_i；将所有节点初始化为未标记，准备开始粗化操作。

执行 SEG 执行 SEG 策略，处理结构等同节点，并将处理过的节点打上标记。

图 3: 图粗化层

Algorithm 1: The graph coarsening operation

Input: Graph $\mathcal{G}_i$ and node representation H_i
Output: Coarsened graph $\mathcal{G}_{i+1}$ and node representation H_{i+1}

Calculate GCN output G_i according to Eq. (1)
Initialize all nodes as unmarked
`/* Structural equivalence grouping */`
Group and mark node pairs having the same neighbors
`/* Structural similarity grouping */`
Sort all unmarked nodes in ascending order according to the number of neighbors
repeat
  for *each unmarked node* v_j **do**
    for *each unmark node* v_k *adjacent to* v_j **do**
      Calculate $s(v_j, v_k)$ according to Eq. (2)
    Group and mark the node pair (v_j, v_k) having the largest $s(v_j, v_k)$
until *all nodes are marked*
Update node weights and edge weights
Construct grouping matrix M_i according to Eq. (3)
Calculate node representation H_{i+1} according to Eq. (4)
Construct coarsened graph $\mathcal{G}_{i+1}$ according to Eq. (5)
return $\mathcal{G}_{i+1}, H_{i+1}$

执行 SSG 在执行 SSG 的过程中，H-GCN 以度数从小到大的次序先后处理节点。这是考虑到度数小，也即拥有更少邻居的节点，更可能因为邻居节点都已被标记而无法分组（在这种情况下，直接将该节点标记），故优先处理这些节点，以获取更好的粗化效果。

进行 SSG 的具体过程在算法中已有很清晰的体现：按照上述度数递增的顺序选取未标记的节点 v_j，计算他每个相邻节点的结构相似度，选取结构相似度最大的邻居节点 v_k，归为一组并打上标记。重复这一过程直到所有节点都已被标记。

图 4: 邻接矩阵与分组矩阵的计算

$$A_1 = \begin{array}{c} \begin{matrix} A & B & C & D & E & F \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad M_1 = \begin{array}{c} \begin{matrix} A & BC & DEF \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix} \end{array} \quad A_2 = M_1^T A_1 M_1 = \begin{array}{c} \begin{matrix} A & BC & DEF \end{matrix} \\ \begin{pmatrix} 0 & 2 & 3 \\ 2 & 2 & 0 \\ 3 & 0 & 0 \end{pmatrix} \end{array}$$

构造分组矩阵 分组矩阵 *(grouping matrix)*M 包含了粗化操作产生的节点分组信息，他可以按如下方式构造：

$$m_{jk} = \begin{cases} 1, & \text{if } v_j \text{ in } \mathcal{G}_i \text{ is grouped into } v_k \text{ in } \mathcal{G}_{i+1}; \\ 0, & \text{otherwise.} \end{cases}$$

分组矩阵的行代表了原图中的节点，列代表了粗化后图中的节点，也就是超节点。若原图中的节点 i 被分组形成了超节点 j，则 $M_{ij} = 1$，反之,$M_{ij} = 0$。

计算嵌入特征表达 嵌入向量的计算由下面的公式实现：

$$H_{i+1} = M_i^\top \cdot G_i.$$

这个公式的本质其实是，超节点的嵌入向量为其组成节点的嵌入向量之和。

计算粗化后的图 粗化后的图的邻接矩阵可以由下面的公式计算：

$$A_{i+1} = M_i^\top \cdot A_i \cdot M_i$$

事实上，根据这个公式计算出的新图的邻接矩阵具有这样的特性：超节点之间的边权为其组成节点之间所有的边权之和；超节点自环的权重为超节点组成节点之间边权之和的两倍。

$$(A_{i+1})_{jk} = \begin{cases} \sum_{l,m\in\mathcal{V}_i} M_{lj} M_{mk} A_{lm}, & j \neq k; \\ 2\sum_{l,m\in\mathcal{V}_i} M_{lj} M_{mj} A_{lm}, & j = k. \end{cases}$$

其中，$\mathcal{V}_i$ 为第 i 层的输入 $\mathcal{G}_i$ 的节点集。

至此，H-GCN 的图粗化层就完成了他的计算，他的输出 $\mathcal{G}_{i+1}$（用邻接矩阵 A_{i+1} 表示）与嵌入向量 H_{i+1} 会被输入到下一层中。

2.6 图细化层

为了复原图原本的拓扑结构，在图粗化层后，H-GCN 放置了相同个数且对称的图细化层。图细化层的主要步骤有两个：应用图卷积与计算图节点嵌入向量。下面的公式很清晰的展现了这个过程：

$$H_{i+1} = M_{l-i} \cdot G_i + G_{l-i}.$$

其中，M_{l-i} 的作用便是利用节点分组信息复原图的结构。而 $M_{l-i} \cdot G_i$ 所做的事情本质上是将超节点的嵌入表达复制给组成他的每一个节点。

2.7 输出层

输出层的结构是一层 GCN 再加上 Softmax 分类器。这一层可以很清晰地表示成：

$$H_{l+1} = \text{softmax}(\text{ReLU}(\hat{D}_l^{-\frac{1}{2}} \hat{A}_l \hat{D}_l^{-\frac{1}{2}} H_l W_l)),$$

其中，$H_{l+1} \in \mathbb{R}^{n_1 \times |\mathcal{Y}|}$ 代表了节点属于各个分类的概率。

在模型的训练中，可以直接采用交叉熵作为损失函数。

3 模型评价

3.1 表现

图 5: 节点分类准确率

Method	Cora	Citeseer	Pubmed	NELL
DeepWalk	67.2%	43.2%	65.3%	58.1%
Planetoid	75.7%	64.7%	77.2%	61.9%
GCN	81.5%	70.3%	79.0%	73.0%
GAT	83.0 ± 0.7%	72.5 ± 0.7%	79.0 ± 0.3%	–
DGCN	83.5%	72.6%	79.3%	74.2%
H-GCN	**84.5 ± 0.5%**	**72.8 ± 0.5%**	**79.8 ± 0.4%**	**80.1 ± 0.4%**

论文中将 H-GCN 在四个常见数据及上进行了实验，包括三个引用网络与一个知识网络，并与 state-of-art 模型进行比较。

H-GCN 在上述实验中都取得了优异的成绩，相比已有方法有了非常可观的准确率提升。

图 6: 减少训练样本后各模型的准确率

Method	20	15	10	5
GCN	79.0%	76.9%	72.2%	69.0%
GAT	79.0%	77.3%	75.4%	70.3%
DGCN	79.3%	77.4%	76.7%	70.1%
H-GCN	**79.8%**	**79.3%**	**78.6%**	**76.5%**

不仅如此，在减少训练样本的实验中，如果将每一类别标记的节点从 20 个逐渐减少为 5 个可以发现随着训练样本的减少，H-GCN 的性能虽有下降，但相较于其他模型的优势却越发显著。

3.2 分析

论文中从两个角度分析了 H-GCN 的效果与表现。

模型简化测试 为了验证 H-GCN 中的图粗化、图细化操作与节点权重嵌入真的发挥了作用，论文中进行了模型简化测试，来验证他们对模型性能表现的影响。

可以看到，在取出图粗化与图细化操作后，模型的性能有了显著的降低，这反映了图粗化与图细化操作对于模型性能的显著提升。

同样的，通过对比试验，可以发现节点权重嵌入也能为模型带来可观的性能提升。

这证明 H-GCN 中应用的图粗化、图细化操作与节点权重嵌入是有效的。

图 7: 简化测试的结果

Method	Cora	Citeseer	Pubmed	NELL
H-GCN without coarsening and refining layers	80.3%	70.5%	76.8%	75.9%
H-GCN without node weight embeddings	84.2%	72.4%	79.5%	79.6%
H-GCN	84.5%	72.8%	79.8%	80.1%

3.3 优点

作为总结，相较于已有的很多模型，H-GCN 具有一些显著而优势：他拓展了图卷积的感知野，并利用图粗化与细化操作解决了过平滑的问题，使更深的模型成为可能。因而，H-GCN 可以更为全面地提取图的全局信息，综合局部与全局的特征，生成更好的嵌入表达。

H-GCN 在实验中的性能表现超过了已有的 state-of-art 模型。不仅如此，在已标记节点数量非常稀少的情况下，H-GCN 的优势更加明显。这意味着 H-GCN 将具有更高的实用性，可以很好地应用到现实中标记数量很少的图数据上。

4 感想

H-GCN 在实验中优异的表现已经证明，他是一个优秀的模型。

H-GCN 可以说是对利用卷积与池化在图像识别任务中大放异彩的 CNN 更为深入且成功的借鉴与类比。GCN 中引入的图卷积操作很好地将图像识别中的卷积迁移到了图上，通过代理函数提取邻居节点的信息来生成节点的嵌入表达，取得了很好的表现。然而，CNN 中的池化操作在应用图卷积的模型中是缺失的。池化能够很好地帮助模型提取重要信息，加深模型深度，拓展卷积的感知野，与池化对应的操作在图模型中的缺失，意味着图卷积模型的局限性。

而 H-GCN 的图粗化操作很好的解决了这个问题，他就是某种形式上的 “池化”。他将结构相似的节点聚合为一个超节点，用来表示局部区域的特征信息。随着粗化操作的叠加，超节点能够表示更加广泛的区域的特征信息，因而对超节点进行的图卷积操作虽然仍然只能提取超节点的邻居信息，但由于超节点所表示的范围的扩大，图卷积操作的感知野也随之自然而然地扩张了。图中的全局信息也得以被提取。

而对称的图细化操作从某种意义上来说是必要的——唯有如此才能复原图的拓扑结构，从超节点的嵌入表达得到原图中各节点的嵌入表达。在这一过程中，超节点被逐步拆分为更小的超节点，直到得到原始节点。超节点中包含的图局部区域信息也借助细化与图卷积操作被分配、综合到了每一个节点上。

可以这样理解 H-GCN 模型的数据处理过程：各节点中的特征信息在聚合为超节点的信息的过程中，事实上是被提取到了更为抽象、更高而广泛的层次中；在更高的层次中，超节点之间信息的传递与提取完成了图中区域之间全局结构的信息传递；在这一过程之后，高层次的信息再通过细化操作逐渐拆分综合为原始节点的信息，生成原始节点的嵌入特征表达。

而在接下来的研究中，或许可以通过分析生成的节点嵌入表达，与 GCN 所生成的相比较，进一步说明 H-GCN 的确能够减少过平滑的出现。

同时，由于图细化操作从某种角度来说，类似于一种 upsampling。而在 H-GCN 中图细化的实现事实上是把超节点的嵌入表达直接复制给了其组成节点（由于分组策略，一个节点在一次粗化操作中只会分配给一个超节点）。H-GCN 的优异表现证明了这么做是可行的，直观来说，在细化操作后紧接着的图卷积操作能够综合超节点传递的信息。**未来更深入的研究中，能否找到更好的方法来优化模型的表现呢?** 比如，修改分组策略使得一个节点能够被分配到多个超节点中，使得在图细化操作中，细化出的节点能够接受来自多个超节点的信息。或是采用某种可训练的方法来分配超节点的信息，以获取更好的表达。

不仅如此，在未来的研究中，我们也能对其他分组策略进行探索，比如将节点嵌入特征的相似度也考虑在内。

而在 H-GCN 的实现中，可以发现两种分组策略都是基于图的结构、节点之间的边权的，并不依赖于节点的特征表达。所以，我们可以在开始训练之前就基于给出的图数据完成粗化与细化的操作，来加速模型的训练过程。

4.1 疑惑与建议

在笔记的最后，我打算列出在阅读论文中产生的疑惑与一些微小的建议。

图细化层的计算公式 在论文中，给出了图细化层的计算公式:

$$H_i = M_{l-i} \cdot G_i + G_{l-i}.$$

由于第 $i(i \geq \frac{l+1}{2})$ 层的输入图结构与第 $l-i$ 层的输出图结构是一致的，所以第 $l-i$ 层的分组矩阵 M_{l-i} 能够将第 i 层的输入图结构转化为想要的输出图结构。然而，第 i 层的输出根据符号约定应当用 H_{i+1} 表示，所以，此处的公式是否应该为:

$$H_{i+1} = M_{l-i} \cdot G_i + G_{l-i}.$$

才更加符合约定?

输出层公式 类似地，在输出层公式中:

$$H_l = \text{softmax}(\text{ReLU}(\hat{D}_l^{-\frac{1}{2}} \hat{A}_l \hat{D}_l^{-\frac{1}{2}} H_{l-1} W_l)),$$

是否应当用 H_l 来表示第 l 层的输入，H_{l+1} 来表示其输出?

$$H_{l+1} = \text{softmax}(\text{ReLU}(\hat{D}_l^{-\frac{1}{2}} \hat{A}_l \hat{D}_l^{-\frac{1}{2}} H_l W_l)).$$

多通道 GCN 公式 在多通道 GCN 的表述公式中:

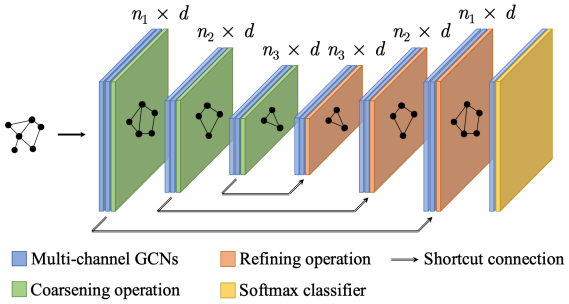
$$G_i = \sum_{j=1}^{c} = w_i \cdot G_i^j,$$

其中应当是 j 代表了通道的序号，因而是否应当用 w_j 来表示该通道的权重呢?

节点权重嵌入的细节 为了将节点权重转化为实向量，需要生成随机的节点权重矩阵 $V \in \mathbb{R}^{|T| \times p}$，其中 T 代表的是所有节点权重的集合。

而节点权重是否可能是不连续的呢？也即不存在节点权重为 1 和 2 的节点，那么，对于节点权重为 3 的节点，是否应当选取 V 中的第一行而不是第三行呢？或者，为了简化实现，生成行数为节点权重最大值的矩阵而不是行数为 $|T|$ 的矩阵是否更加便利？

图 8: H-GCN 结构示意图



对于 H-GCN 结构示意图的小建议 关于论文中的 H-GCN 结构示意图，似乎对于前三层，图粗化层而言，层上面绘制的图简略示意图代表的是输出的图的大概形状；而对于接下来三层，图细化层而言，层上绘制的图简略示意图却代表了输入的图的大概形状。这可能会带来歧义与误解，为了保持一致性，是否将图细化层上绘制的简略示意图也改为输出的图的大概形状更为合适？