6 GNN(Graph Neural Network)

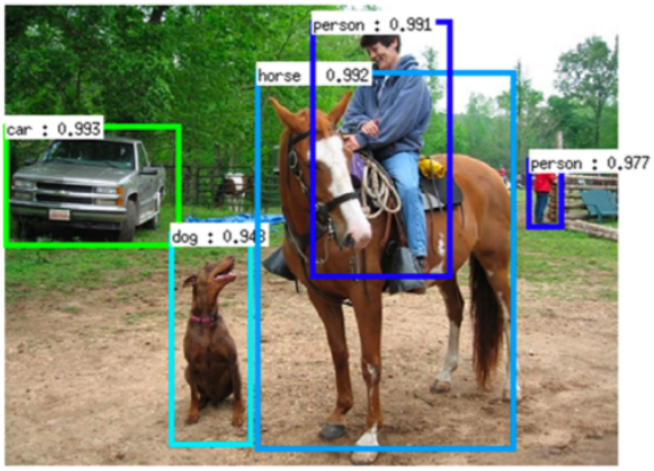
也就是经常说到的图神经网络（Graph Neural Networks，GNN）

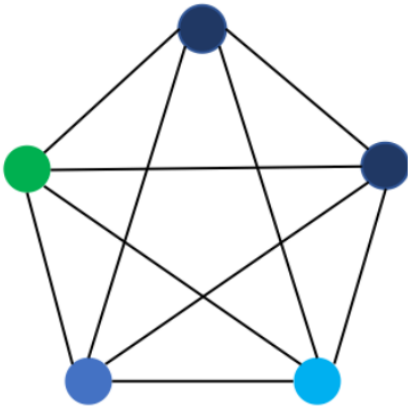
GNN能做什么?

6.1 GNN概述

6.1.1 GNN简介

图(graph)是一种数据结构，常见的图结构包含节点(node)和边(edge)，GNN是深度学习在图结构上的一个分支。





上图为图像，每一个节点表示图像中的每一个物体，边表示物体之间的联系。

GNN是一种连接模型，通过网络中节点之间的信息传递的方式来获取图中的依存关系，GNN通过从节点任意深度的邻居来更新该节点状态，这个状态能够表示状态信息。

6.1.2 GNN起源

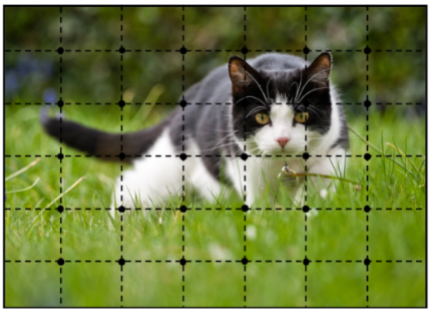
①动机一：CNN的缺陷

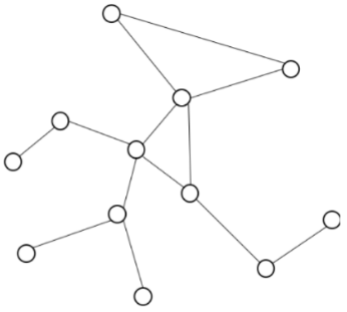
CNN的核心特点在于：局部连接(local connection)，权重共享(shared weights)和多层叠加(multi-layer)

这些同样在**图问题**中非常试用，*因为图结构是最典型的局部连接结构*，其次，共享权重可以减少计算量，另外，多层结构是处理分级模式(hierarchical patterns)的关键。对于图，在数据结构中学到，可以和树联系起来。

传统的深度学习方法被应用在提取欧氏空间数据的特征方面取得了巨大的成功，但许多实际应用场景中的数据是从非欧式空间生成的，传统的深度学习方法在处理非欧式空间数据上的表现却仍难以使人满意。

CNN只能在欧几里得数据(Euclidean data)，比如二维图片和一维文本数据上进行处理，而这些数据只是图结构的特例而已，对于一般的图结构，则很难使用

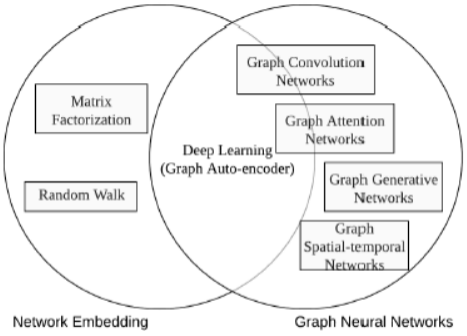




如上图，作图为欧几里得空间，右图为非欧几里得空间。

②动机二：图嵌入的缺陷

图嵌入大致可以划分为三个类别：矩阵分解、随机游走和深度学习方法



: Network Embedding v.s. Graph Neural Networks.

图嵌入和GNN的区别👆

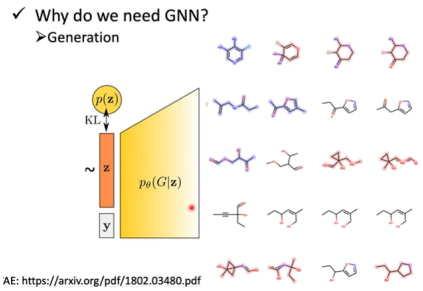
图嵌入常见模型有DeepWalk，Node2Vec等，然而，这些方法方法有两种严重的缺点，首先就是节点编码中权重未共享，导致权重数量随着节点增多而线性增大，另外就是直接嵌入方法缺乏泛化能力，意味着**无法处理动态图以及泛化到新的图**

6.1.3 GNN 使用场景

Classify分类

对于一定图结构的分子，进行分类会不会导致突变

Generation生成



比如说对于药物开发，对于相关疗效药的分子作为输入，输出一个目标疗效的药品

以及对于Social Network社会关系网络分析

6.1.4 GNN优势

与传统NN的区别（GNN优点）
①节点

- CNN和RNN等都需要节点的特征按照一定的顺序进行排列
- 但对于图结构，并没有天然的顺序。所以，GNN采用*在每个节点上分别传播(propagate)*的方式进行学习，由此忽略了节点的顺序，相当于GNN的输出会随着输入的不同而不同。

②边（图结构的边表示节点之间的依存关系）

- 传统的神经网络不是显式地表达中这种依存关系，而是通过不同节点特征来间接地表达节点之间的关系，这些依赖信息只是作为节点的特征。
- GNN 可以通过图形结构进行传播，而不是将其作为节点特征的一部分，通过邻居节点的加权求和来更新节点的隐藏状态

③推理

- 推理是高级人工智能的一个非常重要的研究课题，人脑中的推理过程几乎都是基于从日常经验中提取的图形。标准神经网络已经显示出通过学习数据分布来生成合成图像和文档的能力，同时它们仍然无法从大型实验数据中学习推理图。然而，GNN 探索从场景图片和故事文档等非结构性数据生成图形，这可以成为进一步高级 AI 的强大神经模型。

6.3 GNN原理详解

6.3.1 GNN基础框架

通俗理解GNN简单粗暴带你快速理解GNN*哔哩哔哩*bilibili，里面很好的讲到，可以把GNN抽象看成一个特征提取的过程。

给定一张图G，每个结点都有其自己的特征(feature)，用x_v表示结点v自身的特征，连接两个结点的边也有自己的特征，用$x_{v,u}$表示。

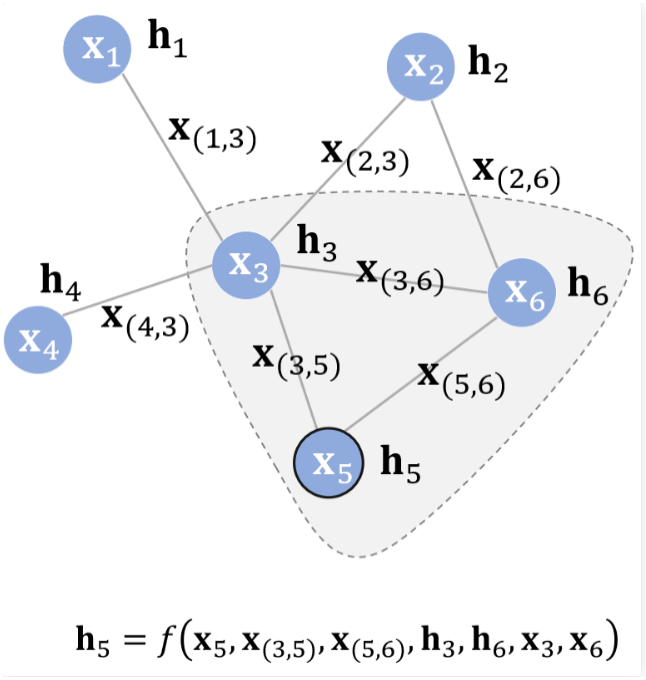
GNN的学习目标是获得每个结点的隐藏状态 $h_v \in R^s$(state embedding)，这就意味着：对于每个节点，它的隐藏状态包含了来自邻居节点的信息.h_v表示节点的状态向量，这个向量可以用于产生输出 o_v

那么，如何让每个结点都感知到图上其他的结点呢？GNN通过迭代更新所有结点的隐藏状态来实现，在$t+1$时刻，结点v的隐藏状态按照如下方式更新，一共更新4部分内容：

$$\mathrm{h_v} = \mathrm{f}(\mathrm{x_v}, \mathrm{x_{co[v]}}, \mathrm{h_{ne[v]}}, \mathrm{x_{ne[v]}})$$

- 假设$f(\cdot)$是带有参数的函数，叫做局部转移函数(local transition function)。f就是隐藏状态的状态更新函数。注意：f在所有节点中共享（全局共享）
- $x_{co[v]}$：表示与节点v关联的边的特征向量
- $x_{ne[v]}$：表示节点v的邻居节点特征向量
- $h_{ne[v]}$：表示节点v的邻居节点在t 时刻的隐藏状态

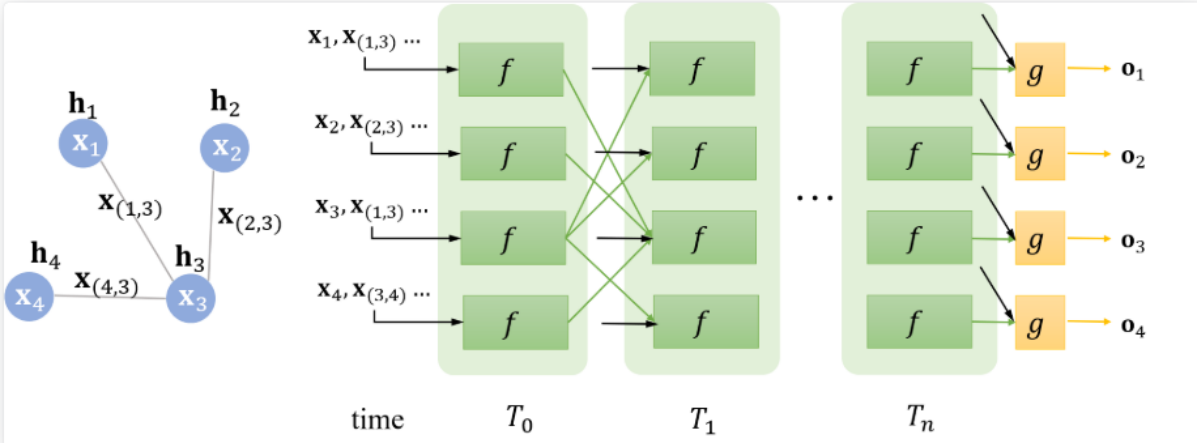
假设对于节点5，其隐藏状态的更新函数如下：



利用更新函数，不断地利用当前时刻邻居结点的隐藏状态作为部分输入来生成下一时刻中心结点的隐藏状态，直到每个结点的隐藏状态变化幅度很小。至此，每个结点都“知晓”了其邻居的信息。

除了不断的更新，最终我们需要一个输出函数，适应不同的下游任务，此时假设$g(\cdot)$为局部输出函数(local output function)，这个函数用于描述输出的产生方式。有

$$\mathrm{o_v} = \mathrm{g}(\mathrm{h_v}, \mathrm{x_v})$$



对于不同的图，收敛的时刻可能不同（因为收敛是通过两个时刻$p-$范数的差值是否小于某个阈值来判定的）

那么我们与深度学习结合呢？用神经网络来拟合复杂函数f和g

为什么会一定收敛输出

假设将所有的状态向量H，所有的输出向量O，所有的特征向量X和所有的节点特征X_N而得到的向量叠加起来，分别用H, O, X, X_N 表示，那么可以得到更加紧凑的表示：

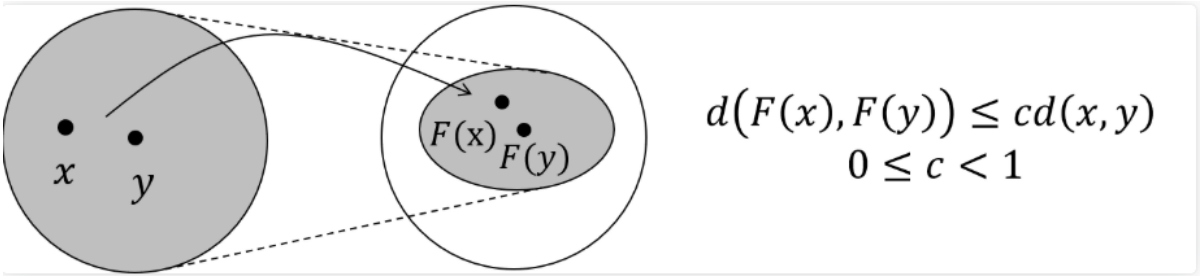
$$\mathrm{H} = \mathrm{F}(\mathrm{H}, \mathrm{X})$$
$$\mathrm{O} = \mathrm{G}(\mathrm{H}, \mathrm{X}_N)$$

根据**Banach的不动点定理**[泛函分析（二）巴纳赫（Banach）不动点，贝尔曼方程（Bellman equation）在强化学习的应用_贝尔曼不动点方程-CSDN博客]，GNN使用如下的**传统迭代方法**来计算状态参量：

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

其中,H^t表示 H的第 t个迭代周期的张量。对于任意的初始值H_0，公式$\mathrm{H}^{\mathrm{t}+1} = \mathrm{F}(\mathrm{H}^{\mathrm{t}}, \mathrm{X})$能通过快速收敛来得到$\mathrm{H} = \mathrm{F}(\mathrm{H}, \mathrm{X})$最终的固定点的解。通过更新迭代，为什么一定会收敛呢?

就是因为不动点定理，就是说不管输H^0是什么形式，如下图只要F是个压缩的映射，不断迭代压缩，最终都会收敛到某一个固定的点，称之为不动点。



怎么保证F是一个压缩函数?

f是用前馈神经网络实现，一种实现方法可以是把每个邻居结点的特征、隐藏状态、每条相连边的特征以及结点本身的特征简单拼接在一起，在经过前馈神经网络后做一次简单的加和。

$$\mathbf{h}_{\mathbf{v}}^{\mathbf{t+1}} = \mathrm{f}\left(\mathbf{x}_{\mathbf{v}}, \mathbf{x}_{\mathbf{c}}\mathrm{o}[\mathrm{v}], \mathbf{h}_{\mathbf{n}}^{\mathbf{t}}\mathrm{e}[\mathrm{v}], \mathbf{x}_{\mathbf{n}}\mathrm{e}[\mathrm{v}]\right)$$
$$= \sum_{\mathrm{u}\in\mathrm{ne}[\mathrm{v}]} \mathrm{FNN}\left(\left[\mathbf{x}_{\mathbf{v}}; \mathbf{x}_{(\mathbf{u},\mathbf{v})}; \mathbf{h}_{\mathbf{u}}^{\mathbf{t}}; \mathbf{x}_{\mathbf{u}}\right]\right)$$

f为压缩映射的等价条件是f 的梯度(导数)要小于1，我们可以通过对雅可比矩阵(Jacobian Matrix)的惩罚项(Penalty)来实现，限制f对H的偏导数矩阵的大小。

6.3.2 中间函数的参数学习

不一定所有结点都是有监督的，模型的损失只通过有监督信号的结点得到，可以将损失函数定义为如下形式：

$$\mathrm{loss} = \sum_{\mathrm{i}=1}^{\mathrm{p}} \left(\mathrm{t}_{\mathrm{i}} - \mathrm{o}_{\mathrm{i}}\right)$$

p表示监督节点的数目，t_i和o_i分别表示节点的真实值和预测值。损失函数的学习基于梯度下降策略，由以下步骤组成：

1. 状态h_v^t按照方公式f迭代更新T轮次，直到到达接近$\mathrm{H} = \mathrm{F}(\mathrm{H}, \mathrm{X})$的定点解的**时刻$T$**， 这时得到的$H$会接近不动点的解$H^T \approx H$
2. 对于有监督信号的结点，将其隐藏状态通过g得到输出，进而算出模型的损失

3. 反向传播时，权重W的梯度从loss计算得到，所以可以直接求出f和g对最终隐藏状态$h_v^{T_n}$的梯度，然后W根据上一步中计算的梯度不断更新，经过T次，得到对h_v^0的梯度，然后该梯度用于更新模型的参数。

6.4 GNN 模型局限性

- 对不动点使用迭代的方法来更新节点的隐藏状态，效率并不高。
- 原始的GNN 在迭代中使用相同的参数，而其他比较著名的模型在不同的网络层采用不同的参数来进行分层特征提取，使得模型能够学习到更加深的特征表达，而且，节点隐藏层的更新是顺序流程，可以利用 RNN 内核，如 GRU 和 LSTM，来进一步优化。
- 一些边(edges)上可能会存在某些信息特征不能被有效地考虑进去。（例如，知识图中的边具有关系类型，并且通过不同边的消息传播应根据其类型而不同）。此外，如何学习边的隐藏状态也是一个重要问题。
- **如果我们需要学习节点的向量表示而不是图的表示，则不适合使用固定点，因为固定点中的表示分布将在值上非常平滑并且用于区分每个节点的信息量较少。**

6.5 GNN变体

主要可以分为以下几个变体

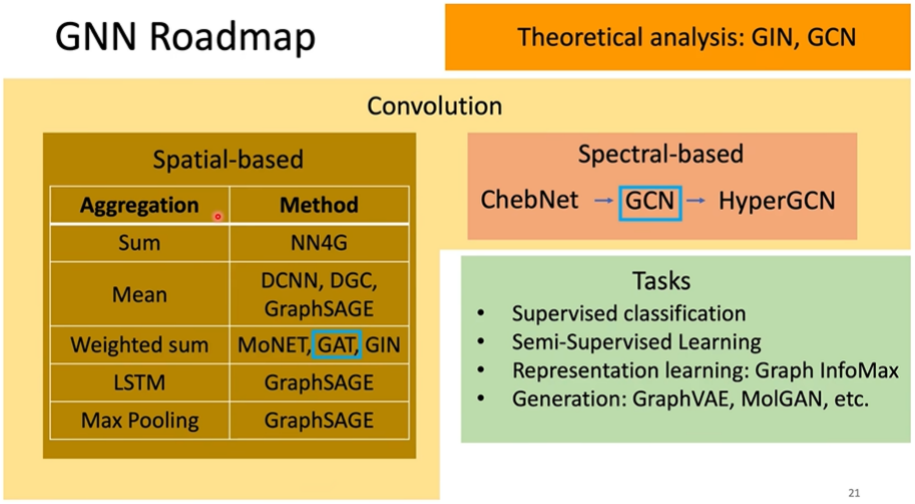
- 图卷积网络(Graph convolutional networks)和图注意力网络(graph attention networks)，因为涉及到传播步骤(propagation step)。
- 图的空域网络(spatial-temporal networks)，因为该模型通常用在动态图(dynamic graph)上。
- 图的自编码(auto-encoder)，因为该模型通常使用无监督学习(unsupervised)的方式。
- 图生成网络(generative networks)，因为是生成式网络。

6.5.1 图卷积网络

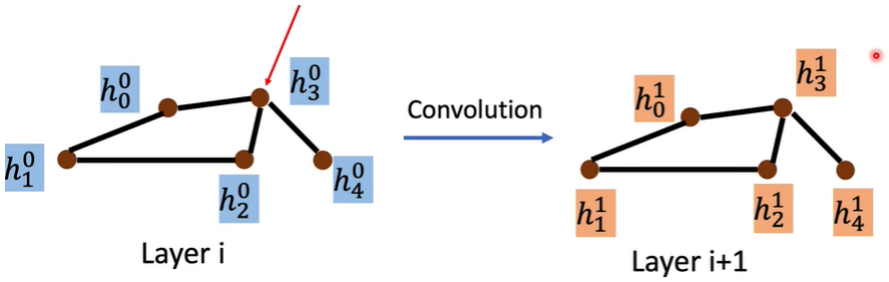
这方面的进展通常分为谱方法(Spectral Methods) 和非谱方法(Non-Spectral Methods)

Spectral Method 希望使用谱分解的方法，应用图的拉普拉斯矩阵分解进行节点的信息收集。

Non-Spectral Methods 直接使用图的拓扑结构，根据图的邻居信息进行信息收集。直接在图上定义卷积操作，也就是在空域上相邻的邻居节点上进行操作。



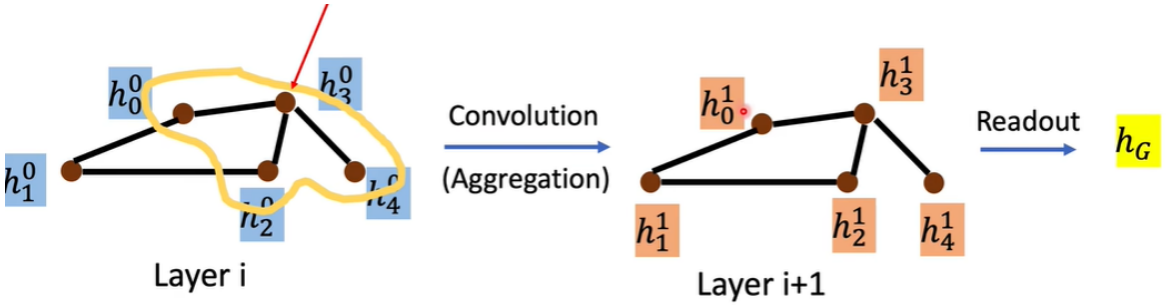
Spatial-based Convolution



上面就是首先用一个卷积更新一次节点信息

aggregate聚合

即通过图结构的点来提供整个图的特征

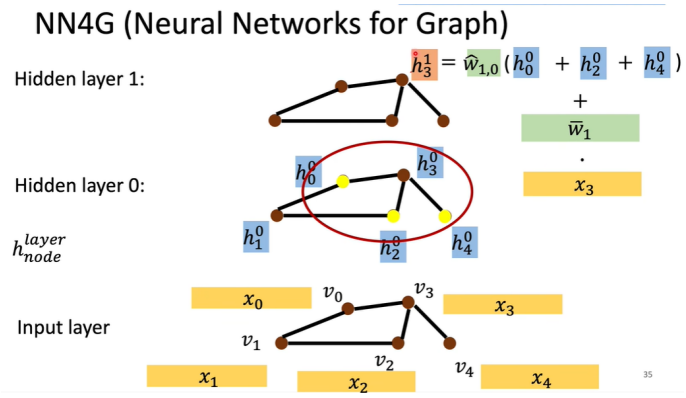


聚合后对整个图进行处理

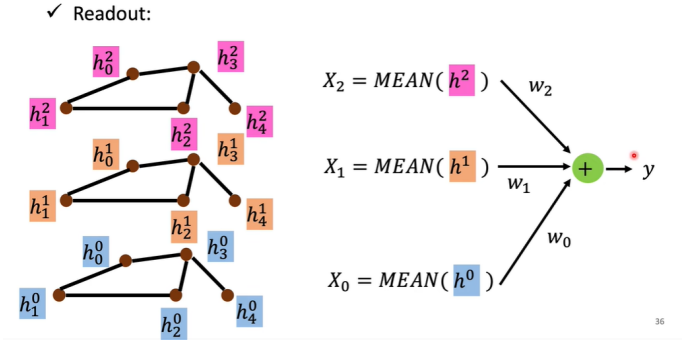
①NN4G

NN4G（Neural network for Graph)

[每次对临近节点的信息进行计算得到隐藏状态]



这个就是在上面原理讲到的，对于特征的计算需要自身信息和领近节点的信息。

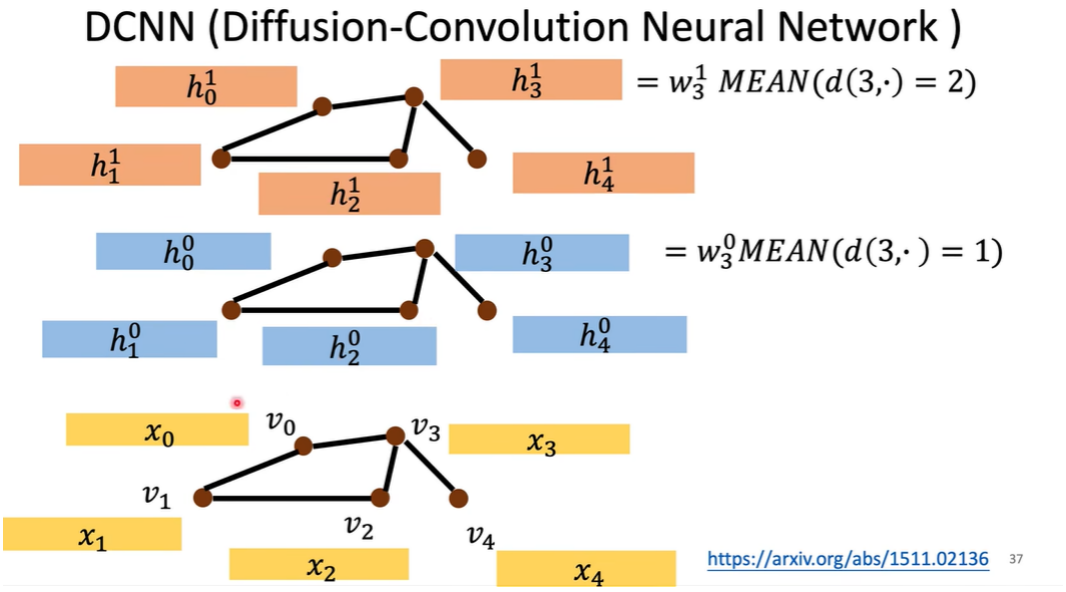


此后，对每一层的所有图节点都进行一个h_v的均值计算，然后把每一层计算得到的均值作为输入，对于输入后计算得到最终整个图的特征y

②DCNN

DCNN（Diffusion-Convolution Neural Network)

[每次按照深度对图中节点的信息进行计算得到隐藏状态]



这里主要是对每一各节点的隐藏状态h_i的计算方式进行了修改

这里引入了层数的计算，比如上面的

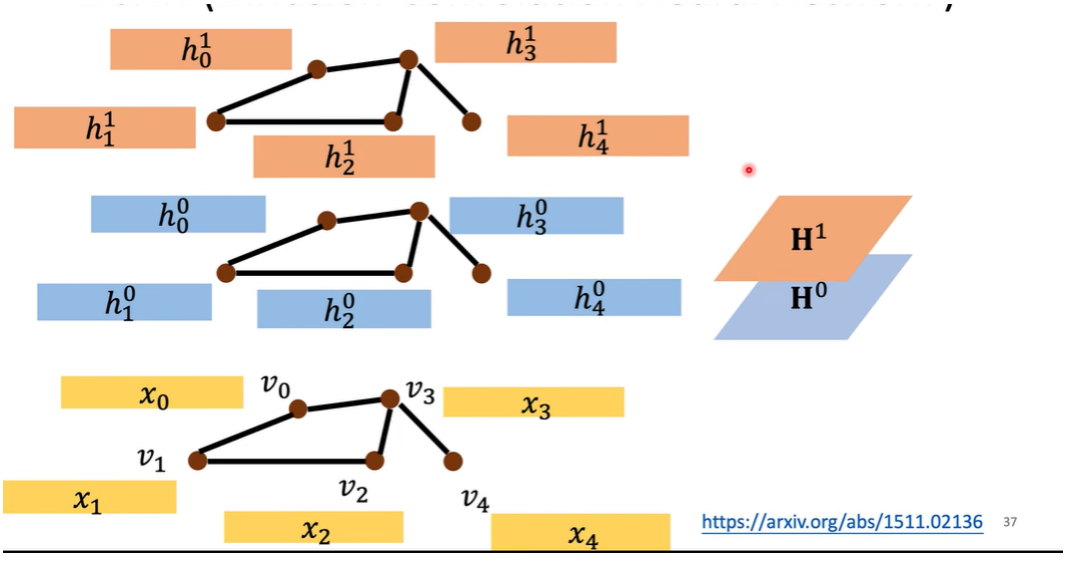
$$h_0^3 = w_0^3 MEAN(d(3,.) = 1)$$

表示这个点的隐藏状态是通过计算与自己的临近节点得到$d(3,.) = 1$表示距离x_3为1的点的信息。

同样在进行一次卷积后，这里

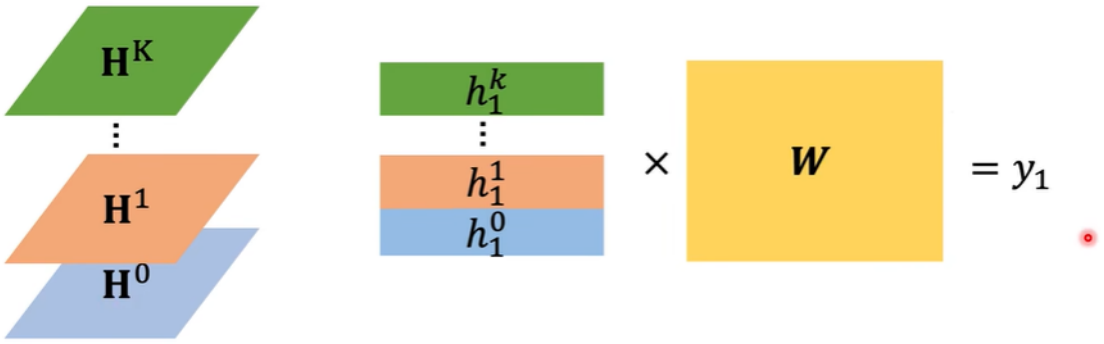
$$h_1^3 = w_1^3 MEAN(d(3,.) = 2)$$

相当于对距离为2的点进行计算。



计算可以得到一层卷积后得到一个完整的隐藏状态H^i有多少个节点，H^i就有多少维度。

✓ Node features



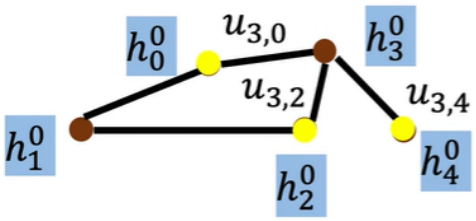
此时再通过计算这个H^K得到最终的输出

③MoNET

【CVPR-2017-paper MoNet（空间卷积方法） 混合模型网络monet-CSDN博客】

MoNET (Mixture Model Networks)

[以上都是对于点的信息计算得到隐藏状态，现在要把边的权重考虑进去计算得到隐藏状态]



$$\mathbf{u}(x,y) = (\frac{1}{\sqrt{\deg(x)}}, \frac{1}{\sqrt{\deg(y)}})^\top$$

这里距离采用

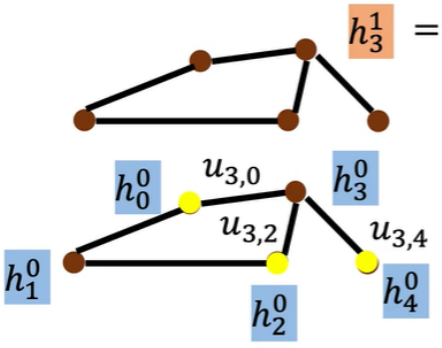
$$\mathbf{u}(x,y) = (\frac{1}{\sqrt{\deg(x)}}, \frac{1}{\sqrt{\deg(y)}})^\top$$

这里的$deg(x)$表示节点的度（数据结构，这里是无向图，也就是所连边的数目）

MoNET (Mixture Model Networks)

✓ Define a measure on node 'distances'

✓ Use weighted sum (mean) instead of simply summing up (averaging) neighbor features.



$$h_3^1 = w(\hat{u}_{3,0}) \times h_0^0 + w(\hat{u}_{3,2}) \times h_2^0 + w(\hat{u}_{3,2}) \times h_4^0$$

$w(\cdot)$ is a NN, $\hat{u}$ is a transform from u

$$\mathbf{u}(x,y) = (\frac{1}{\sqrt{\deg(x)}}, \frac{1}{\sqrt{\deg(y)}})^\top$$

https://arxiv.org/pdf/1611.08402.pdf 40

再计算隐藏状态