Graph Neural Network

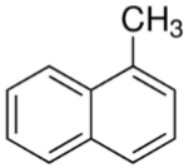
slides: http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML2020/GNN.pdf

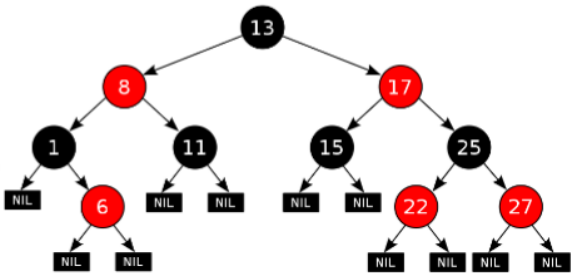
Introduction

定义

GNN 图神经网络 就是把图作为神经网络的输入，识别图结构，提取图信息，或生成特定结构的图的神经网络模型。

这里的“图”是指图论中的图，即由边和节点构成的图，比如下图中化学分子结构图，二叉树、台湾省地铁线路图：





8

训练GNN将遇到的问题

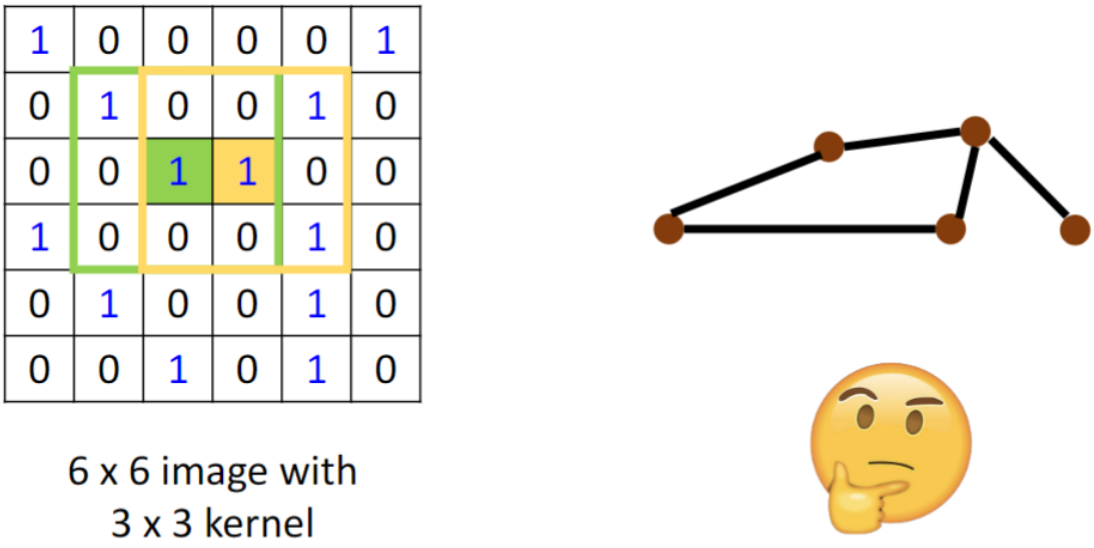
如何利用图的结构和节点之间的关系信息训练模型（即模型如何吃Graph）？

如果图非常大怎么办?

图非常大的时候通常我们没有所有node的label information，怎么办?

首先第一个问题，model如何吃graph的问题：

convolution?

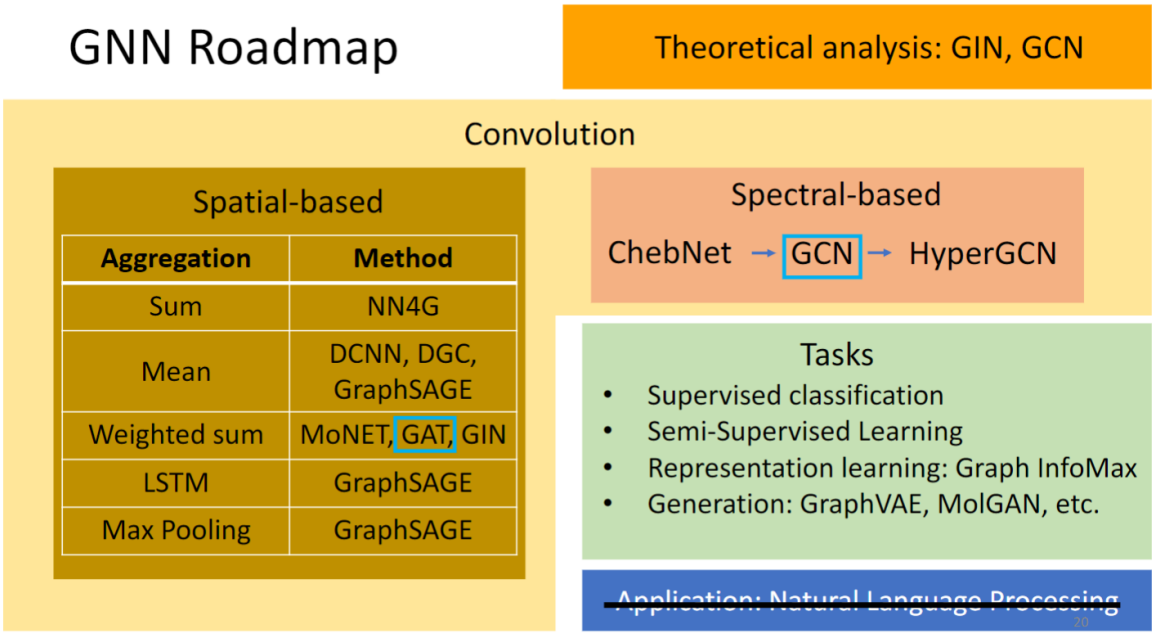


如何把CNN里卷积的方法移植到图神经网络?

所以如何使用convolution将node 嵌入到一个feature space?

- Solution 1: Generalize the concept of convolution (corelation) to graph >> Spatial-based convolution
- Solution 2: Back to the definition of convolution in signal processing >> Spectral-based convolution

Roadmap



本节会讲的是：GAT和GCN.

Tasks, Dataset, and Benchmark

Tasks

- Semi-supervised node classification

- Regression
- Graph classification
- Graph representation learning
- Link prediction

Common dataset

- CORA: citation network. 2.7k nodes and 5. 4k links
- TU-MUTAG: 188 molecules with 18 nodes on average

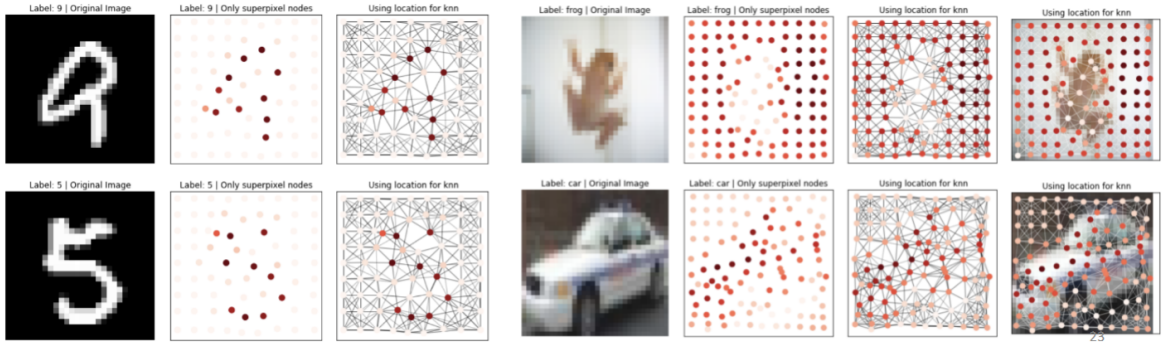
这些benchmark的task和dataset用来评估GNN模型好坏，建议先看完下一节Spatial-based GNN再回来看这些任务

https://arxiv.org/pdf/2003.00982.pdf

Benchmark tasks

✓ Graph Classification: SuperPixel MNIST and CIFAR10

Domain/Construction	Dataset	# graphs	# nodes
Computer Vision/ Graphs constructed with super-pixels	MNIST CIFAR10	70K 60K	40-75 85-150



这个数据集通过某种算法将原始MNIST和CIFAR10的图片转成graph，GNNs的任务就是graph classification

https://arxiv.org/pdf/1711.07553.pdf

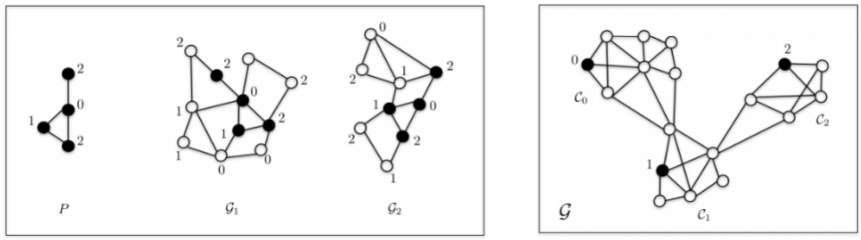
Benchmark tasks

✓ Regression: ZINC molecule graphs dataset

Domain/Construction	Dataset	# graphs	# nodes
Chemistry/ Real-world molecular graphs	ZINC	12K	9-37
Artificial/ Graphs generated from Stochastic Block Model	PATTERN CLUSTER	14K 12K	50-180 40-190

✓ Node classification:Stochastic Block Model dataset

➢ graph pattern recognition and semi-supervised graph clustering



ZINC 是通过分子graph 计算分子溶解度，属于回归任务

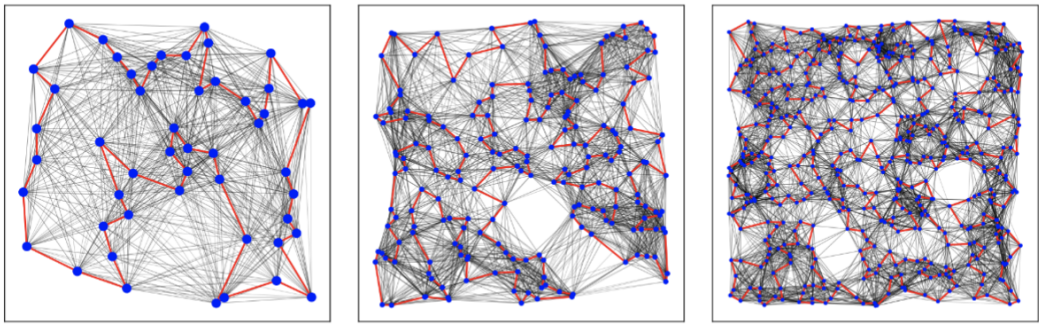
Stochastic Block Model dataset 是给出一个pattern，model要是别这个pattern是否出现在一个graph中

这个数据集还可以做另一个任务：每个图由不同的community或者说cluster，model要判断一个node属于哪个cluster

Benchmark tasks

✓ Edge classification: Traveling Salesman Problem

Domain/Construction	Dataset	# graphs	# nodes
Artificial/ Graphs generated from uniform distribution	TSP	12K	50-500



(a) TSP50 (b) TSP200 (c) TSP500

25

TSP 路径规划问题，就不用赘述了.

Results

✓ SuperPixel

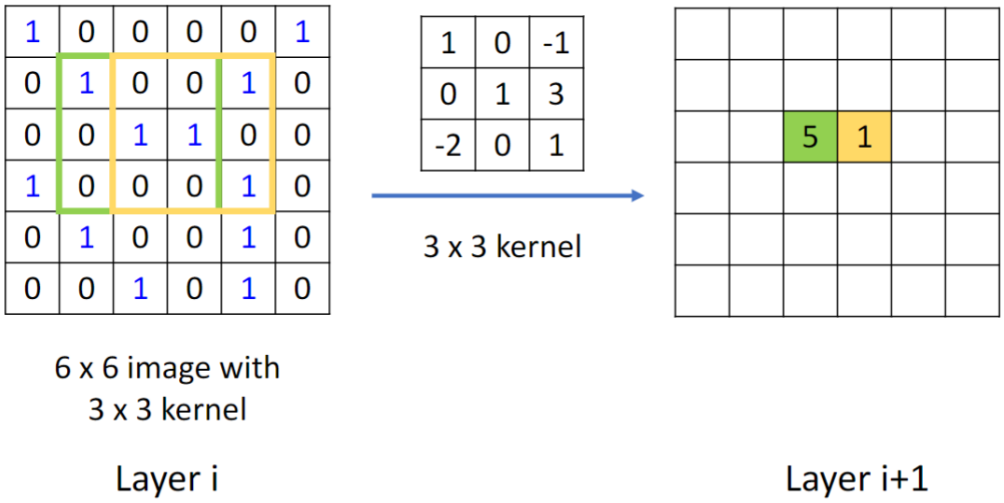
Dataset	Model	#Param	No Residual	
			Acc	Epoch/Total
MNIST	MLP	104044	94.46±0.28	21.82s/1.02hr
	MLP (Gated)	105717	95.18±0.18	22.43s/0.73hr
	GCN	101365	**89.05±0.21**	79.18s/1.76hr
	GraphSage	102691	**97.20±0.17**	76.80s/1.42hr
	GIN	105434	**93.96±1.30**	34.61s/0.74hr
	DiffPool	106538	**94.66±0.48**	171.38s/4.45hr
	GAT	110400	**95.56±0.16**	377.06s/6.35hr
	MoNet	104049	**89.73±0.48**	567.12s/12.05hr
	GatedGCN	104217	**97.36±0.12**	127.15s/2.13hr
	GatedGCN-E*	104217	**97.47±0.13**	127.86s/2.15hr
CIFAR10	MLP	104044	56.01±0.90	21.82s/1.02hr
	MLP (Gated)	106017	56.78±0.12	27.85s/0.68hr
	GCN	101657	51.64±0.45	100.30s/2.44hr
	GraphSage	102907	**66.08±0.24**	96.00s/1.79hr
	GIN	105654	47.66±0.47	44.30s/0.93hr
	DiffPool	108042	56.84±0.37	299.64s/10.42hr
	GAT	110704	**65.48±0.33**	386.14s/7.75hr
	MoNet	104229	50.99±0.17	869.90s/21.79hr
	GatedGCN	104357	**68.92±0.38**	145.14s/2.49hr
	GatedGCN-E*	104357	**69.37±0.48**	145.66s/2.43hr

26

Spatial-based GNN

复习一下CNN的Convolution：

Review: Convolution



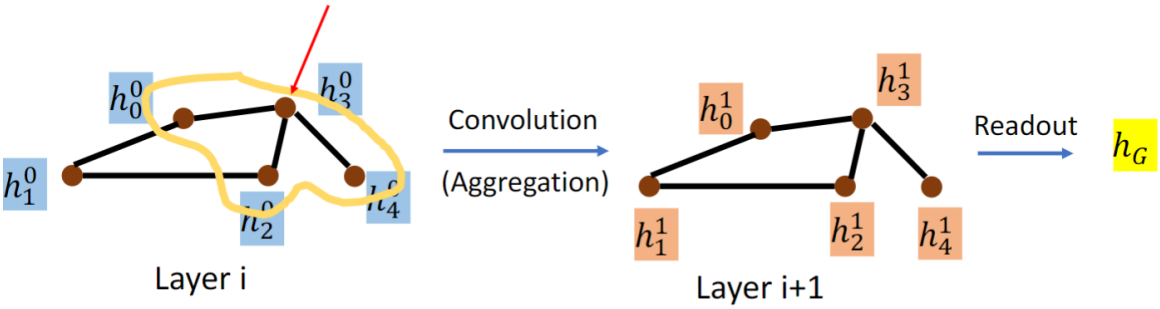
32

CNN在layer i上通过kernel计算feature得到layer i+1层的feature，类比到Graph上：

Spatial-based Convolution

✓ Terminology:

- ➢Aggregate: 用 neighbor feature update 下一層的 hidden state
- ➢Readout: 把所有 nodes 的 feature 集合起來代表整個 graph

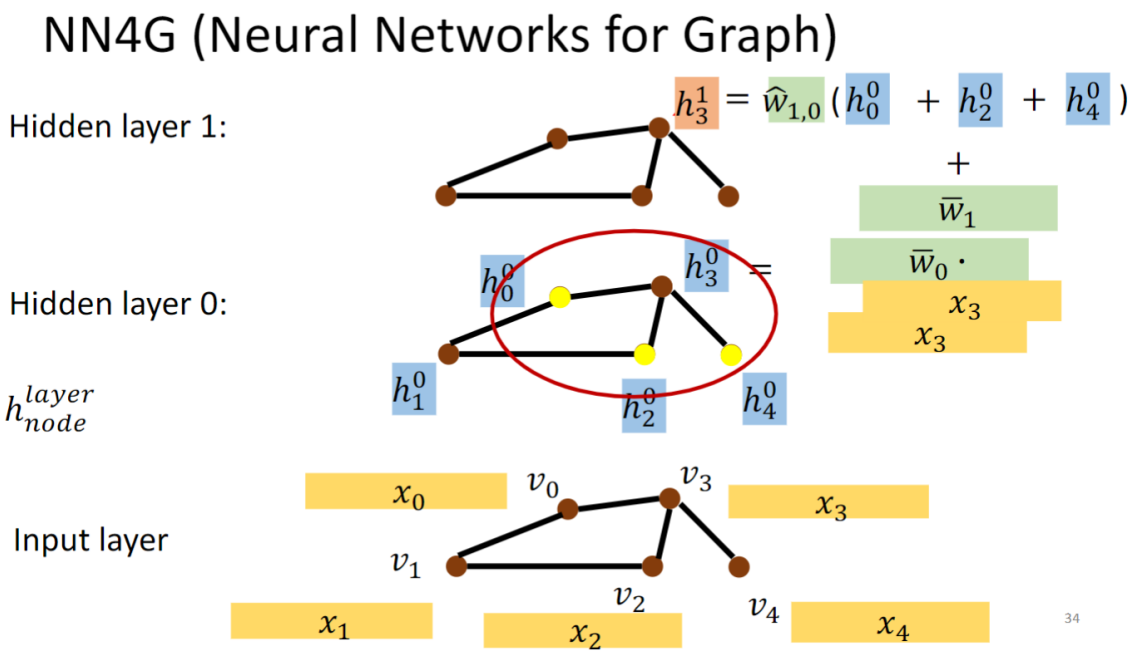


33

以 h_3^0 为例，0表示第0 layer，3表示第3个node，它的邻居是黄色圈起来的三个几点，我们就用这三个邻居的hidden feature 来算出下一层的hidden layer，这一招叫做**Aggregation**.

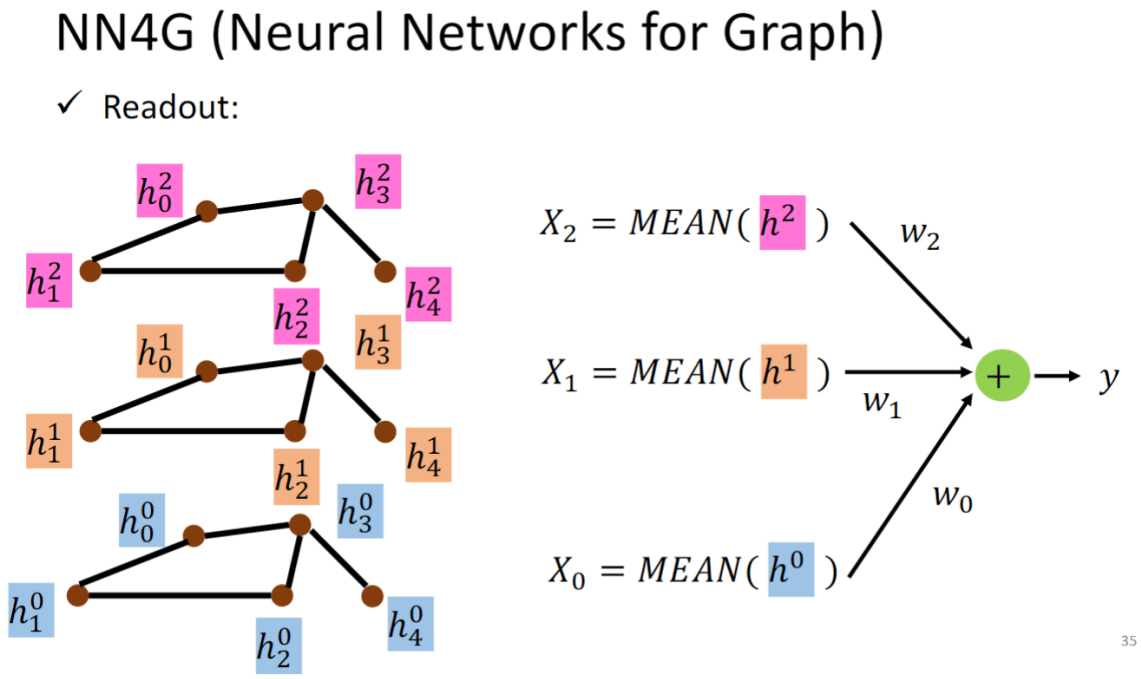
如果我们还想同时计算出某一层整个graph的representation，预测一个分子是否会变异，就把所有node的feature集合起来，这个操作叫做**readout**.

NN4G(Neural Networks for Graph)



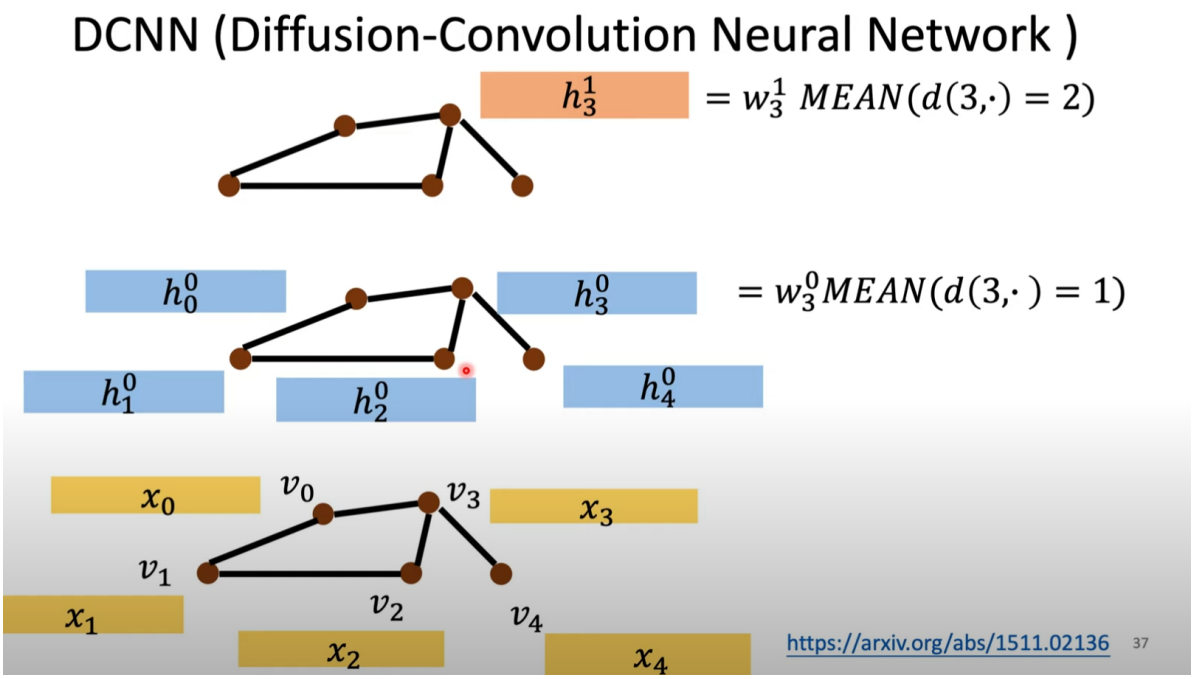
ref: https://ieeexplore.ieee.org/document/4773279

NN4G这个模型的做法如上图所示，每个节点v都用一个特征向量x表示，对每个节点做embedding，即乘一个矩阵w，得到h，然后从一个hidden layer到下一个hidden layer做Aggregation，具体做法就是将一个节点的邻居的h加起来乘上一个w，就得到这个节点在下一个hidden layer的h.



NN4G这个模型Readout的做法如上图所示，无需多言.

DCNN(Diffusion-Convolution Neural Network)



ref: https://arxiv.org/abs/1511.02136

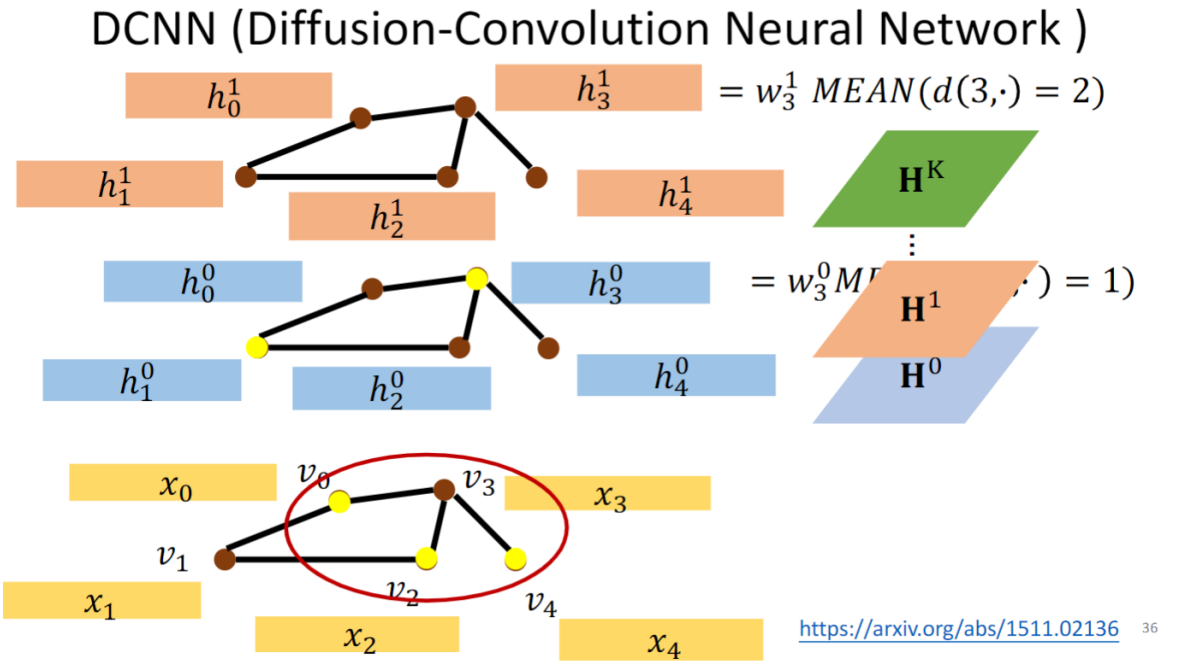
这个模型的Aggregation的做法是：每一层都从第一层计算得来，算是这样算的：

$$h_3^0 = w_3^0 MEAN(d(3,\cdot) = 1)$$

这是说和v_3距离为1的节点取平均，乘上一个权重矩阵w_3^0，即节点v0 v2 v4。算完每个节点以后得到第一层的h。

$$h_3^1 = w_3^1 MEAN(d(3,\cdot) = 2)$$

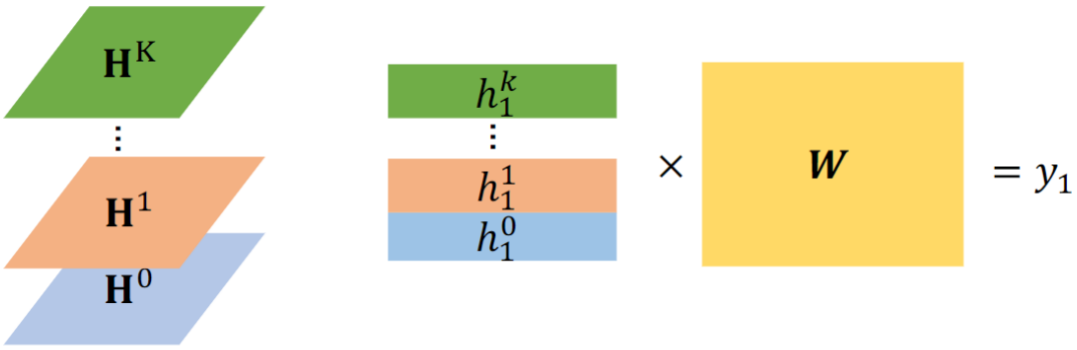
这是说和v_3距离为2的节点取平均，乘上一个权重矩阵w_3^1，即节点v1 v3。算完每个节点以后得到第二层的h。



H^k 是将每层的h一行一行放在一起，叠成一个矩阵.

DCNN (Diffusion-Convolution Neural Network)

✓ Node features



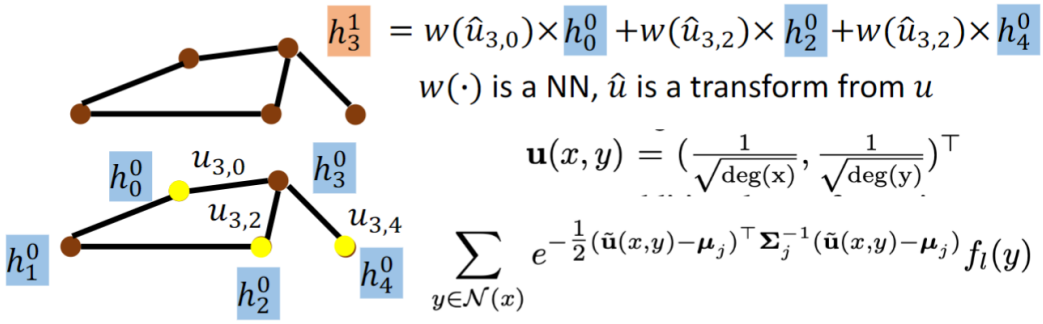
readout的做法如上图所示，无需多言.

MoNET (Mixture Model Networks)

MoNET (Mixture Model Networks)

✓ Define a measure on node 'distances'

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



https://arxiv.org/pdf/1611.08402.pdf 39

> ref: https://arxiv.org/pdf/1611.08402.pdf

MoNET 对图中的边定义了距离权重，这篇文章中距离是由公式定义的，是可以直接计算的，也有的模型（GAT ）是通过graph data学出的.

MoNET 将weighted sum neighbor features取代了NN4G和DCNN简单的相加求平均.

GAT (Graph Attention Networks)

GAT (Graph Attention Networks)

- ✓ Input: node features $\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$
- ✓ Calculate energy: $e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$
- ✓ Attention score (over the neighbors)

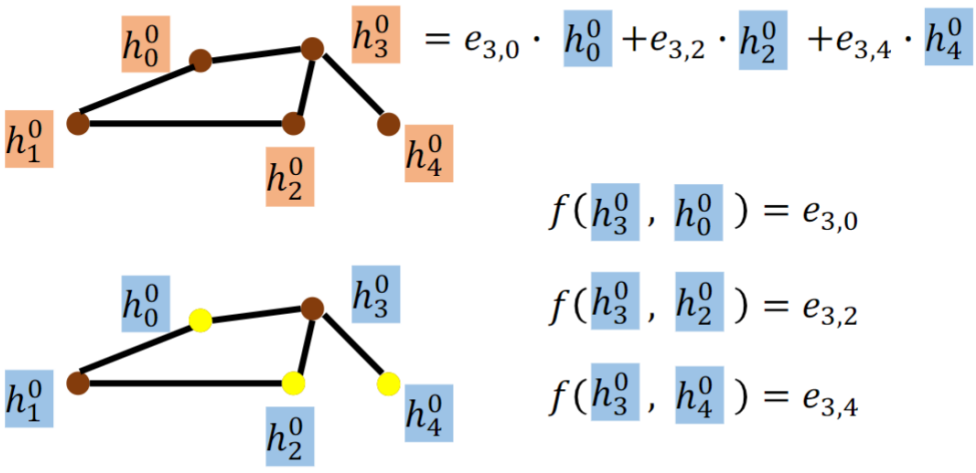
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

https://arxiv.org/pdf/1710.10903.pdf

Published as a conference paper at ICLR 2018

> ref: https://arxiv.org/pdf/1710.10903.pdf

GAT (Graph Attention Networks)



GAT计算节点在下一层上的hidden representation的时候，先算一个节点对它所有邻接节点的energy（即可变的weight），然后把这个energy作为权重乘上节点在当前层上的hidden representation，再求和（如上图所示），作为最终的在下一层的表示.

GIN (Graph Isomorphism Network)

前面的方法我们就直接用了，也没有问为什么它们会work，GIN这篇paper给出了一些证明，告诉你什么样的GNN model会work

> ref: https://openreview.net/forum%3Fid=ryGs6iA5Km

- A GNN can be at most as powerful as WL isomorphic test
- Theoretical proofs were provided

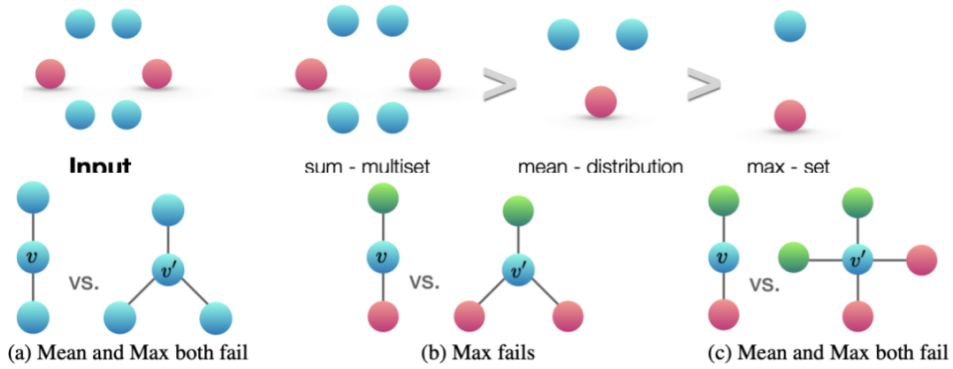
结论：更新节点 representation 的时候最好使用下图中的公式所示的方式更新.

GIN (Graph Isomorphism Network)

✓ $h_v^{(k)} = \text{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$

✓ Sum instead of mean or max

✓ MLP instead of 1-layer



48

上图公式：

$$h_v^{(k)} = MLP^{(k)}((1+\epsilon^{(k)})\cdot h_v^{(k-1)} + \sum_{u\in N(v)} h_v^{(k-1)})$$

k是layer，v是node id，h update的方式应该要先将neighbor 全都加起来，而不能用max pooling也不能用mean pooling，然后加上某个constant 乘以 自己的feature，这个constant就是 $1+\epsilon^{(k)}$ 这里的 ϵ 是可以学出来的，但是paper中也说了这里设为0也没太大差别.

为什么不能用max pooling也不能用mean pooling，看上图下面一排：

a告诉我们，max or mean 无法区分a中的两个graph；

max无法分辨b中的两个graph；

max和mean也无法分辨c中的两个graph；

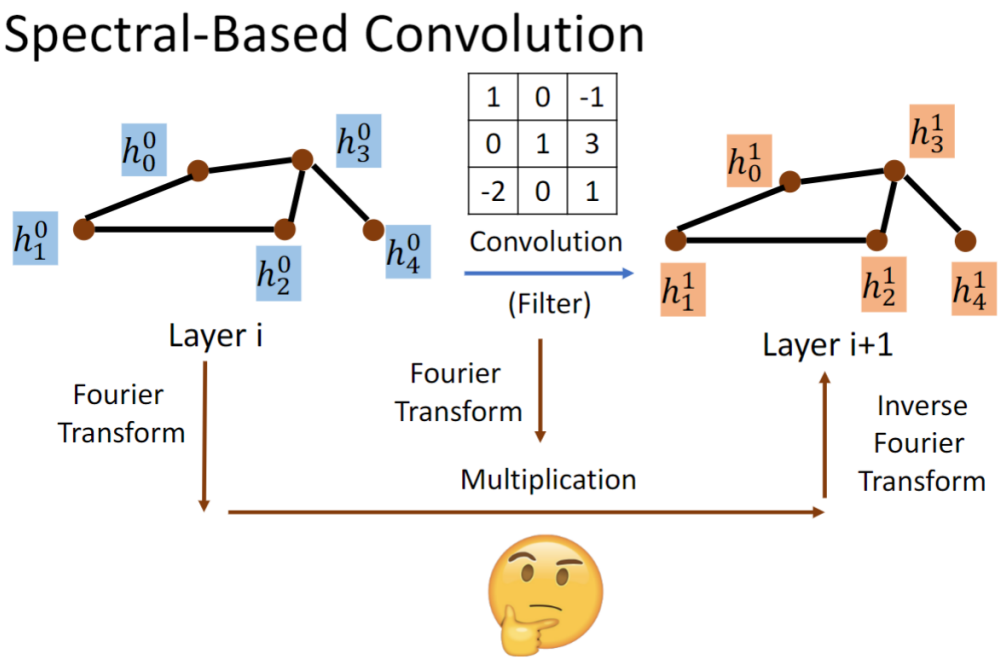
MLP是multi layer perceptron

GIN (Graph Isomorphism Network)

✓ Experiment

	Datasets	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB	MUTAG	PROTEINS	PTC	NCI1
Datasets	# graphs	1000	1500	2000	5000	5000	188	1113	344	4110
	# classes	2	3	2	5	3	2	2	2	2
	Avg # nodes	19.8	13.0	429.6	508.5	74.5	17.9	39.1	25.5	29.8
GNN variants	SUM-MLP (**GIN-0**)	**75.1** ± **5.1**	**52.3** ± **2.8**	**92.4** ± **2.5**	**57.5** ± **1.5**	**80.2** ± **1.9**	**89.4** ± **5.6**	**76.2** ± **2.8**	**64.6** ± **7.0**	**82.7** ± **1.7**
	SUM-MLP (**GIN-ϵ**)	**74.3** ± **5.1**	**52.1** ± **3.6**	**92.2** ± **2.3**	**57.0** ± **1.7**	**80.1** ± **1.9**	**89.0** ± **6.0**	**75.9** ± **3.8**	63.7 ± 8.2	**82.7** ± **1.6**
	SUM-1-LAYER	74.1 ± 5.0	**52.2** ± **2.4**	90.0 ± 2.7	55.1 ± 1.6	**80.6** ± **1.9**	**90.0** ± **8.8**	**76.2** ± **2.6**	63.1 ± 5.7	82.0 ± 1.5
	MEAN-MLP	73.7 ± 3.7	**52.3** ± **3.1**	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	**66.6** ± **6.9**	80.9 ± 1.8
	MEAN-1-LAYER (GCN)	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0	20.0 ± 0.0	79.0 ± 1.8	85.6 ± 5.8	76.0 ± 3.2	64.2 ± 4.3	80.2 ± 2.0
	MAX-MLP	73.2 ± 5.8	51.1 ± 3.6	–	–	–	84.0 ± 6.1	76.0 ± 3.2	64.6 ± 10.2	77.8 ± 1.3
	MAX-1-LAYER (GraphSAGE)	72.3 ± 5.3	50.9 ± 2.2	–	–	–	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5

Graph Signal Processing and Spectral-based GNN



Spectral-based GNN 要做的事情就是将graph 和convolution kernel 都转换到傅里叶域中，在傅里叶域中做multiplication，再转换回去，就是得到下一个layer.

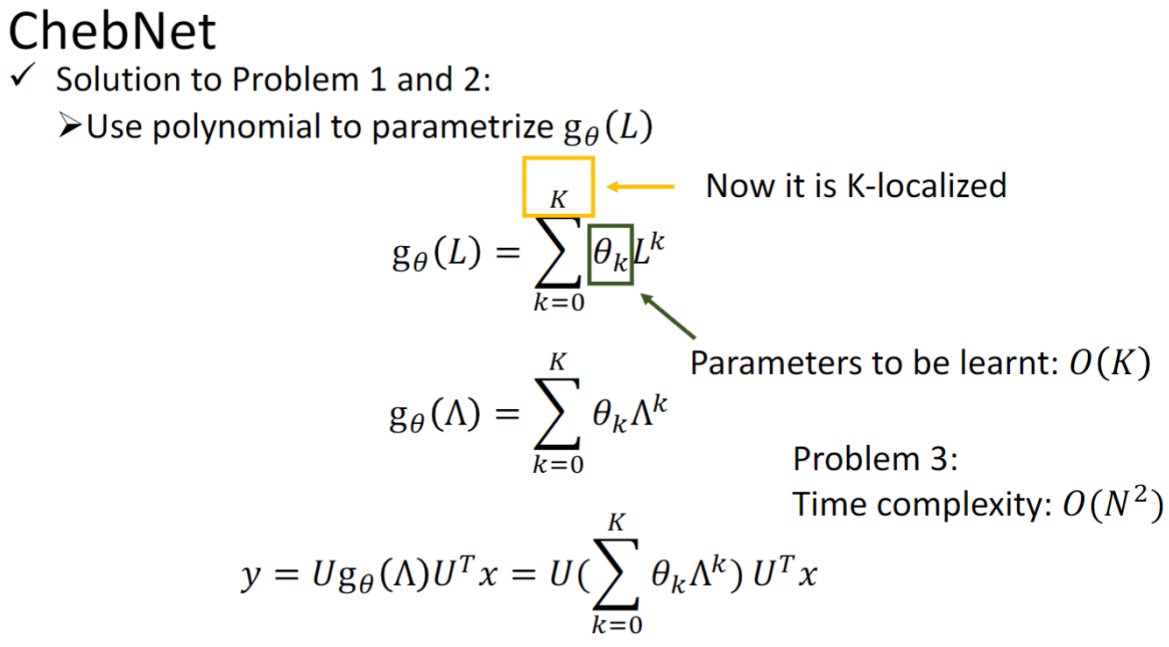
问题是这个傅里叶变换要怎么做呢，要回答这个问题要引入很多信号与系统的东西.

Warning of ~~Math~~ Signal And System

没学过信号与系统，听不懂，但不影响对整个GNN的理解（大概

如果想很好的理解下面讲的Spectral-based GNN的话，建议还是好好理解一下 Warning of Signal And System 这一部分.

ChebNet



https://arxiv.org/pdf/1606.09375.pdf

主打特性是快，且localize

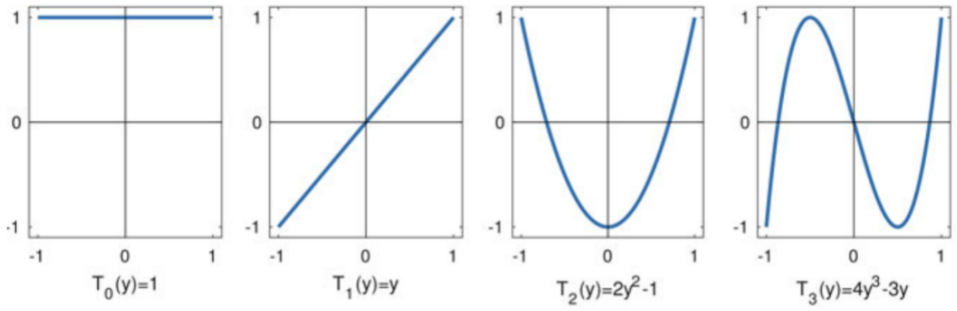
> ref: https://arxiv.org/pdf/1606.09375.pdf

L^k 是拉普拉斯算子Laplacian的一个多项式函数，通过你选择让 $g_\theta(L)$ 是多项式函数的方式，你就可以让他是K-localized，因为根据上一节原理中讲的，如果你让g函数只到k次方，它就只能看到k-neighbor.

用Chebyshev 多项式解决时间复杂度太高的问题：

ChebNet

- ✓ Solution to Problem 3:
  - ➢Use a polynomial function that can be computed recursively from L
- ✓ Chebyshev polynomial
  - ➢$T_0(x) = 1, T_1(x) = x, T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), x \in [-1, 1]$



84

ChebNet

$$T_0(\tilde{\Lambda}) = \mathrm{I}, T_1(\tilde{\Lambda}) = \tilde{\Lambda}, T_k(\tilde{\Lambda}) = 2\tilde{\Lambda}T_{k-1}(\tilde{\Lambda}) - T_{k-2}(\tilde{\Lambda})$$

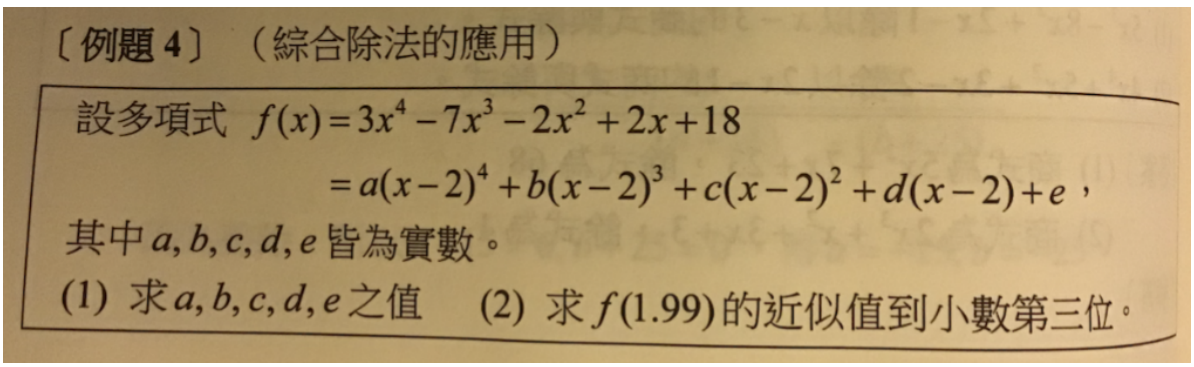
$$\text{where } \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - \mathrm{I}, \quad \tilde{\lambda} \in [-1, 1]$$

$$\mathrm{g}_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k \Lambda^k \longrightarrow \mathrm{g}_{\theta'}(\tilde{\Lambda}) = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda})$$

$$y = \mathrm{g}_{\theta'}(\tilde{L})x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x$$

85

所以原本的 $g_\theta(\Lambda)$ 变为 $g_{\theta'}(\tilde{\Lambda})$，怎么理解这个转换的作用，为什么要把一个多项式组合换成另一个多项式组合呢？你可以参考下面这个高中数学题：


〔例題 4〕（綜合除法的應用）

設多項式 $f(x)=3x^4-7x^3-2x^2+2x+18$

$=a(x-2)^4+b(x-2)^3+c(x-2)^2+d(x-2)+e$，

其中 a,b,c,d,e 皆為實數。

(1) 求 a,b,c,d,e 之值　(2) 求 $f(1.99)$ 的近似值到小數第三位。

你可以把 $g_\theta(\Lambda)$ 看成$3x^4-7x^3-2x^2+2x+18$ 把 $g_{\theta'}(\widetilde{\Lambda})$ 看作后面的形式，这个转换使得 $f(1.99)$ 更容易计算了.

所以最后模型要学的东西就是 $g_{\theta'}(\widetilde{\Lambda})$ 中的 θ'_k .

ChebNet

$$y = g_{\theta'}(L)x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x$$

$$= \theta'_0 T_0(\tilde{L})x + \theta'_1 T_1(\tilde{L})x + \theta'_2 T_2(\tilde{L})x + \cdots + \theta'_K T_K(\tilde{L})x$$

$$T_0(\tilde{L}) = \mathrm{I}, T_1(\tilde{L}) = \tilde{L}, T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L})$$

$$T_0(\tilde{L})x = x, T_1(\tilde{L})x = \tilde{L}x, T_k(\tilde{L})x = 2\tilde{L}T_{k-1}(\tilde{L})x - T_{k-2}(\tilde{L})\,x$$

Define $T_k(\tilde{L})x = \bar{x}_k$

$$\bar{x}_0 = x, \bar{x}_1 = \tilde{L}x, \bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$$

87

ChebNet

$$\bar{x}_0 = x, \bar{x}_1 = \tilde{L}x, \bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$$

$$y = g_{\theta'}(L)x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x$$

$$= \theta'_0 T_0(\tilde{L})x + \theta'_1 T_1(\tilde{L})x + \theta'_2 T_2(\tilde{L})x + \cdots + \theta'_K T_K(\tilde{L})x$$

$$= \theta'_0 \bar{x}_0 + \theta'_1 \bar{x}_1 + \theta'_2 \bar{x}_2 + \cdots + \theta'_K \bar{x}_K$$

$$= [\bar{x}_0 \;\; \bar{x}_1 \;\; \ldots \;\; \bar{x}_K][\theta'_0 \;\; \theta'_1 \;\; \ldots \theta'_K]$$

Calculating $\bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ cost $O(E)$

Total complexity: $O(KE)$

88

GCN

这是比较受喜欢的，被大家广泛使用的模型.

GCN

https://openreview.net/pdf?id=SJU4ayYgl

✓ $y = g_{\theta'}(L)x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x \ , K = 1$

$$
\begin{aligned}
y = g_{\theta'}(L)x &= \theta'_0 x + \theta'_1 \tilde{L} x && \because \tilde{L} = \frac{2L}{\lambda_{max}} - I \\
&= \theta'_0 x + \theta'_1 (\frac{2L}{\lambda_{max}} - I)x && \because \lambda_{max} \approx 2 \\
&= \theta'_0 x + \theta'_1 (L - I)x && \because L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \\
&= \theta'_0 x - \theta'_1 (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x && \because \theta = \theta'_0 = -\theta'_1 \\
&= \theta (I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x
\end{aligned}
$$

renormalization trick: $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

90

> ref: https://openreview.net/pdf%3Fid=SJU4ayYgl

GCN

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right)$$

Can be rewritten as:

$$h_v = f\left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} W x_u + b\right), \quad \forall v \in \mathcal{V}.$$

GCN对每个layer中的node h_v 计算就是将所有neighbor包括他自己乘一个weight，再加起来，，再加上一个bias，再经过一个nonlinear activation，就结束.

GCN

✓ Experiment: Semi-supervised classification

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

✓ Node features:

➢ Citation network: Bag of Word of the document

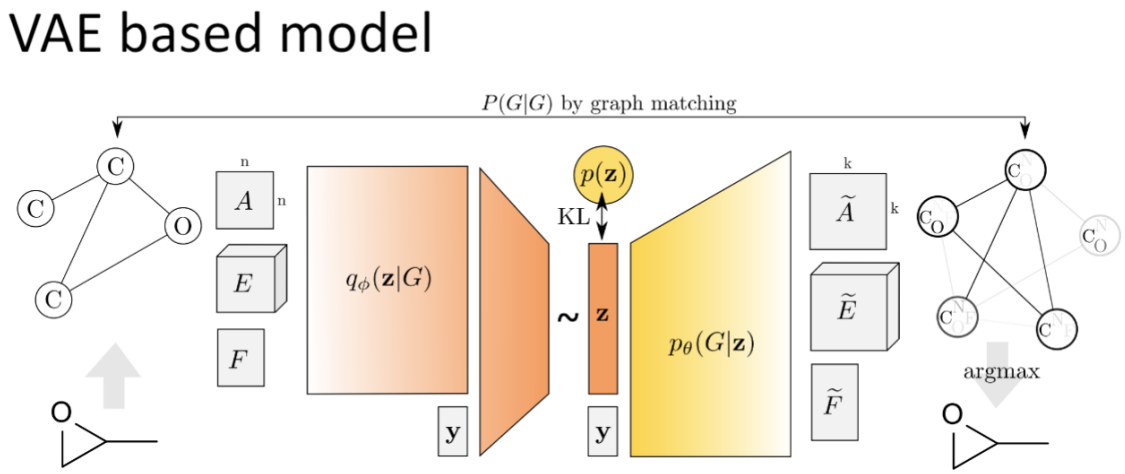
➢ In the knowledge base, each relation is denoted as a triplet (e_1, r, e_2)

Graph Generation

- VAE-based model: Generate a whole graph in one step
- GAN-based model: Generate a whole graph in one step

- Auto-regressive- based model: Generate a node or an edge in one step

VAE-based model

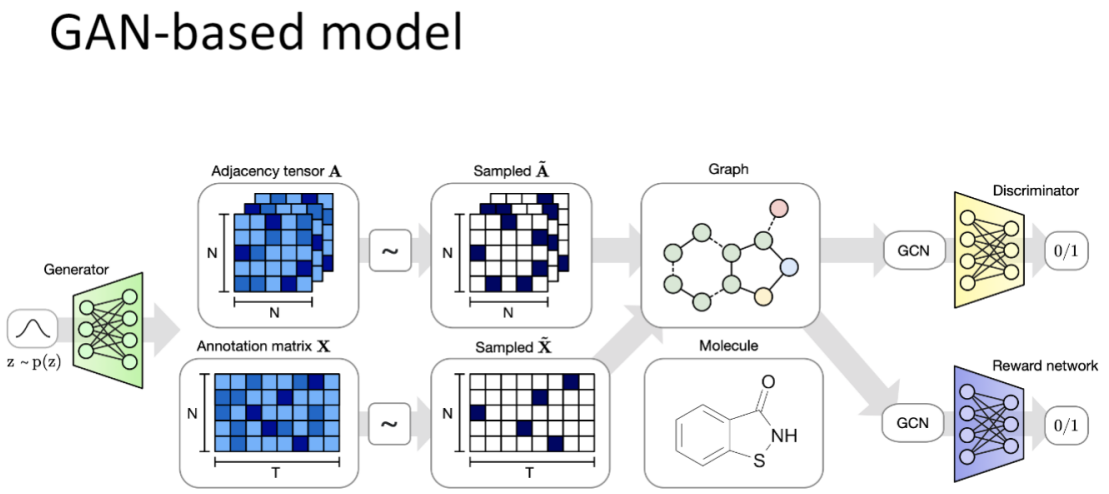


https://arxiv.org/pdf/1802.03480.pdf

106

> ref: https://arxiv.org/pdf/1802.03480.pdf

左边是encoder，右边是decoder，input是一个adjacency matrix，Edge Feature 和node Feature，生成一样是一个adjacency matrix，Edge Feature 和node Feature.



https://arxiv.org/pdf/1901.00596.pdf

107

> ref: https://arxiv.org/pdf/1901.00596.pdf

GAN的输入是adjacency matrix和feature matrix

AR-based model

https://arxiv.org/pdf/1803.03324.pdf

$$s_u = f_s(\mathbf{h}_u^{(T)}, \mathbf{h}_v^{(T)}), \quad \forall u \in V$$

$f_{addnode}(G) = \mathrm{s}$ $f_{addedge}(G, v) = f_{addnode}(G) =$ $f_{addedge}(G, v) = f_{nodes}(G, v) = \mathrm{softmax}(\mathbf{s})$

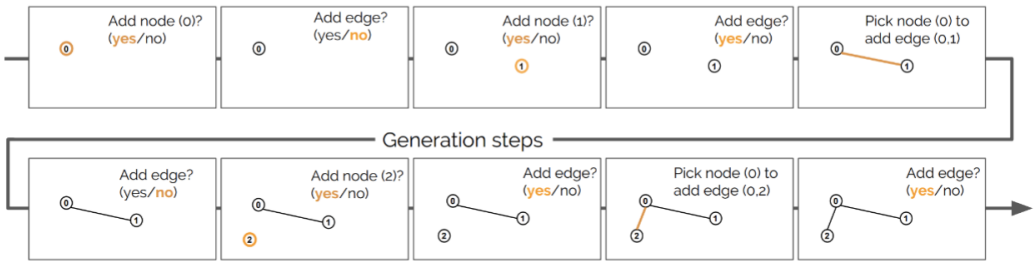


Figure 1. Depiction of the steps taken during the generation process.

$$\mathbf{a}_v = \sum_{u:(u,v)\in E} f_e(\mathbf{h}_u, \mathbf{h}_v, \mathbf{x}_{u,v}) \quad \forall v \in V \qquad \mathbf{h}_V^{(T)} = \mathrm{prop}^{(T)}(\mathbf{h}_V, G)$$

$$\mathbf{h}_v' = f_n(\mathbf{a}_v, \mathbf{h}_v) \quad \forall v \in V, \qquad \mathbf{h}_G = R(\mathbf{h}_V^{(T)}, G)$$

108

> ref: https://arxiv.org/pdf/1803.03324.pdf

Online Resources

PyTorch Geometric: https://github.com/rusty1s/pytorch_geometric

Deep Graph Library: http://dgl.ai/