

A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long,
Chengqi Zhang, *Senior Member, IEEE*, Philip S. Yu, *Fellow, IEEE*

【强化学习 103】GNN综述



张楚珩

清华大学 交叉信息院博士在读

33 人赞同了该文章

去年 12 月份在 Arxiv 上更新的关于 GNN 的综述文章，应该还是比较新的。

原文传送门

Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." arXiv preprint arXiv:1901.00596 (2019).

特色

离散状态的强化学习问题中，不同的状态可以自然地表示为一个图的形式；GNN 解决的问题就包括如何学习到一个包含图信息的节点表示（graph embedding），这和强化学习里面的表示学习粗看起来还挺像的。不过感觉里面还是有很多不一样的地方，其中最大的区别在于：目前 GNN 所解决的问题中，图结构是一次性全部给出的；而在强化学习中，需要通过策略的探索来遇见相应的节点。而在强化学习中，当节点数目较多时，相应的图就会变得特别庞大；因此，如何一边探索并且记录所遇到的状态，一边对于状态（节点）做聚合（aggregation）就成为了一个十分重要的问题。带着这样的问题，我们来看看 GNN 里面有没有什么可以借鉴的做法。

这一篇文献总体上来说还是比较新的，梳理的方面也很全面，包括分类、不同 GNN 的做法、应用场景、数据集和开源代码等。不过对于我们要解决的问题帮助似乎不是特别大，因为这里面所解决的问题基本上还都是“图结构一次性全部给出”的，和我们要解决的问题关系不大。值得注意的是，有一些小的点对于我们研究的问题有一定的帮助，比如：第七节和第九节中提到 GNN 中有一些工作研究动态的图，即图的节点和连边可能有动态的删增，同时这也是一个未来的研究方向；第九节中还提到未来的研究方向之一是加强其扩展性，这就要求对图进行 sampling 或者 clustering，其实这有点像我们想要做的 aggregation（这让我想到 Wang Mengdi 做的 anchor state 啥的，瞎联想）。

过程

1. 任务和应用

先讲 GNN 适用的任务形式吧，这样更容易理解一些。先列举一下相应的任务类型分类，后面我们再看看不同的实际应用是如何对应不同的任务类型的。

- Node-level/edge-level + semi-supervised learning：这种类型的问题的训练和测试数据可以只涉及到一个图，这张图里面有一部分节点/边有标签，有一部分节点/边没有标签；我们需要利用图的拓扑结构去预测未标注节点/边的标签。对于节点的预测来说，根据节点标签的种类可以分为 node regression 和 node classification。对于连边的预测来说，也可以相应地分为 edge classification 和 link prediction。

- Graph-level + supervised learning: 这种类型的训练和测试数据包含多个图，训练数据是每个图对应一个标签。最后的目标就是训练一个有监督学习模型，模型的输入是整个图，模型的输出是相应的预测。
- Graph-level + unsupervised learning: 这种类型的训练数据是多个图，但是没有相应的标签，目标是希望能够学习到一个 encoder，使得任意给一个类似的图，能够学习到该图的表示，同时这个表示能够反映图的拓扑结构特征；或者是希望学习到一个 decoder，从而能够生成一个类似的图。除了 encoder+decoder 的处理方式之外，还可以使用 negative sampling + discriminator 来处理。

GNN 能够应用到如下领域：

- CV 领域：
 - Scene graph generation: 给定一张图片，区分图片中有语义关系的物体；或者，给定一个经过 NLP 模块解析过的物体对应关系，生成相应的图片。文章里面说，一个 scene graph 的节点是不同的物体，连边是他们的对应关系，这个问题应该对应上面的哪一种问题类型，不太清楚。但是如果把原始图片看做 graph，然后从这个 graph 到 scene graph 其实算是一个 aggregation 的过程。
 - Point cloud classification: 给定一些点云，然后需要根据它们的拓扑关系，把它们抽象成 superpoint graph，这也是一个 aggregation 的过程。这常常用于无人驾驶领域，因为无人驾驶的激光雷达返回的信号就是点云的形式。
 - Action recognition: 通过视频，可以把人的姿态抽象为一个 graph，各个骨架就是 graph 中的连边；当然，同时，这个 graph 也是随着时间变化的，因此需要对于一个随时间变化的 graph 来进行 classification。
- NLP 领域：自然语言是一个序列（sequence），但是其对应的逻辑可以表示为一个语法树或者语义树（graph），因此产生了相应的两个子问题：
 - sequence-to-graph: 把自然语言解析为相应的语法树或者语意树，该任务对于知识挖掘非常有用。
 - graph-to-sequence: 给定语法树或者语意树，生成相应的句子。
- 交通：城市的道路本身就自然形成了图的结构，各个节点或者边的属性就是某条路或者某个路口的交通状况，任务就是希望能够预测未来的道路交通状况。
- 推荐系统：把用户和商品都看做是节点，对于推荐商品的预测就可以看做是一个 link prediction 的问题，即如果该用户和某商品之间有一个连边，那么该连边的强度可能为多少。
- 化学：有机大分子或者化合物可以构成一个 molecule graph 或者 compound graph；相应的原子为节点，其化合键为连边。Node classification 任务就对应提取该化合物的特征（fingerprints）；graph classification 任务对应预测化合物的性质；graph generation 任务对应生成满足特定性质要求的化合物结构。在最近发生的疫情中，该方法在疫苗研制中应该发挥了相应的作用的（看到电视上有报道）。

2. 分类

这篇文章最大的框架就是把现有的 GNN 工作分为了四类：recurrent GNN（RecGNN）、convolutional GNN（ConvGNN）、graph autoencoder（GAE）和 spatial-temporal GNN（STGNN）；其中有两类还更加细分了一些子类别，如下图所示。

TABLE II: Taxonomy and representative publications of Graph Neural Networks (GNNs)

Category	Publications	
Recurrent Graph Neural Networks (RecGNNs)	[15], [16], [17], [18]	
Convolutional Graph Neural Networks (ConvGNNs)	Spectral methods	[19], [20], [21], [22], [23], [40], [41] [24], [25], [26], [27], [42], [43], [44]
	Spatial methods	[45], [46], [47], [48], [49], [50], [51] [52], [53], [54], [55], [56], [57], [58]
Graph Autoencoders (GAEs)	Network Embedding	[59], [60], [61], [62], [63], [64]
	Graph Generation	[65], [66], [67], [68], [69], [70]
Spatial-temporal Graph Neural Networks (STGNNs)	[71], [72], [73], [74], [75], [76], [77]	

3. RecGNN 和 ConvGNN

这两类联系比较紧密，一起说。

我对 GNN 还有一些了解，这个分类里面最不明白的是前面两类 GNN 的区别，这里先重点讲一下。GNN 的核心就是 information diffusion mechanism / message passing。其核心就是要在相互连接的节点之间交换信息，即需要迭代地更新节点的表示，每一次更新，每个节点上的信息都和相邻节点做一定的交互。这两类都通过一个参数化表示的深度学习模块来做这样的信息交换：在 RecGNN 中，每一步信息交换的变换函数都是一样的，并且目标是做很多次这样的信息交换直到每个节点上的特征都达到稳态；在 ConvGNN 中，每一步信息交换的函数都不一样，并且只经过有限步的信息交换。这一点区别如下图所示。

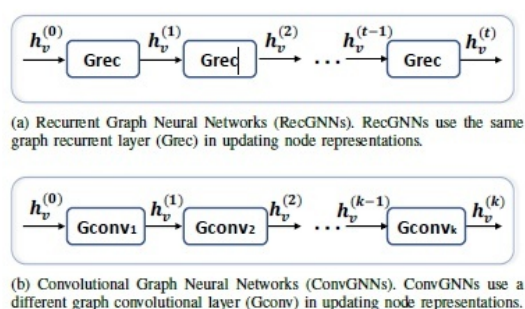


Fig. 3: RecGNNs v.s. ConvGNNs 知乎 @张楚珩

RecGNN

RecGNN 这一类方法提出的更早，由于要求经过无穷多次信息交换之后，各个节点的特征要达到稳态，因此要求每一个信息交换函数都要是一个 contractive mapping，否则各个节点表示最后的数值会发散。常见的信息交换模块的函数如下图所示（graph neural network，GNN*）

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f(\mathbf{x}_v, \mathbf{x}_{(v,u)}^a, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}), \quad (1)$$

contraction 的要求还会产生对函数 f 参数的更多要求。

当然，RecGNN 中，各个信息交换模块也可以引入 RNN，使得 contraction 的要求跟容易被满足（gated graph neural network，GGNN）

$$\mathbf{h}_v^{(t)} = GRU(\mathbf{h}_v^{(t-1)}, \sum_{u \in N(v)} \mathbf{W} \mathbf{h}_u^{(t-1)}), \quad (2)$$

对于比较大的图，可以采样一个 batch，然后分别做节点上状态的更新和梯度的计算（stochastic steady-state embedding, SSE）；在该工作中，更好地满足 contraction 的要求，其信息交换函数被定义为新状态和旧状态的加权和（和前面使用 RNN 的思路类似）：

$$\mathbf{h}_v^{(t)} = (1 - \alpha) \mathbf{h}_v^{(t-1)} + \alpha \mathbf{W}_1 \sigma(\mathbf{W}_2 [\mathbf{x}_v, \sum_{u \in N(v)} [\mathbf{h}_u^{(t-1)}, \mathbf{x}_u]]),$$

ConvGNN

ConvGNN 被分为两大类：频域方法（spectral-based method）和空间域方法（spatial-based method）。

频域方法和我的数学专栏里面讲的 spectral graph theory 有很大的联系，假设 A 是邻接矩阵，可以定义 graph Laplacian matrix 并对它做特征值分解 $L = L_n - D^{-1/2} A D^{-1/2} = U \Lambda U^T$ ，其中 D 是一个对角矩阵，每一个元素代表相应节点的度数。考虑每个节点上都有一个数值，构成一个 n 维的向量 x，如果把它和另一个 n 维的向量 g 做卷积，可以得到

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U} \mathbf{g}_\theta \mathbf{U}^T \mathbf{x}. \quad (5)$$

称 g 为 filter，不同的 filter 对应在图的频域上进行不同的操作。可以这样理解，矩阵 U 相当于一个标准正交基，其每一个向量规定了图上每个节点的一个数值，相当于一个模式；较大特征值对应的向量对应频率较低（空间上变化缓慢）的模式。上式可以看做先把 x 变换到频域上，对齐进行某个频域上的操作之后，在把它变换回空间域。g 可以由参数控制，不同的参数控制方法可以导出不同的频域 ConvGNN。比如直接把 g 用该参数表示（spectral convolutional neural network, spectral CNN）

$$\mathbf{H}_{:,j}^{(k)} = \sigma(\sum_{i=1}^{f_{k-1}} \mathbf{U} \boldsymbol{\Theta}_{i,j}^{(k)} \mathbf{U}^T \mathbf{H}_{:,i}^{(k-1)}) \quad (j = 1, 2, \dots, f_k), \quad (6)$$

用 Chebyshev 多项式或者 Cayley 多项式来近似 g（ChebNet/CayleyNet）可以得到

$$\mathbf{x} *_G \mathbf{g}_\theta = \sum_{i=0}^K \theta_i T_i(\tilde{L}) \mathbf{x}, \quad (8)$$

$$\mathbf{x} *_G \mathbf{g}_\theta = c_0 \mathbf{x} + 2 \operatorname{Re} \left\{ \sum_{j=1}^r c_j (h\mathbf{L} - i\mathbf{I})^j (h\mathbf{L} + i\mathbf{I})^{-j} \mathbf{x} \right\}, \quad (9)$$

我们注意到，spectral-based 的方法有较大的局限性：需要对于整个图的 Laplacian 矩阵做特征值分解，当图非常大的时候，该方法的时间和空间复杂度都会特别高（虽然 ChebNet 和 CayleyNet 通过近似降低了复杂度，但是它们仍然需要拿到全部的图一次性处理）；图里面任何一个位置的变动时，整个特征值系统都会变化；所学习到的 filter 只适用于这单个图，并不能泛化到别的图上；spectral-based 只适用于无向图，对于有向图或者一些更为复杂的图（比如连边带有特征的、异构的图）不能处理。而 spatial-based 的方法则能够克服以上局限性，它不需要进行特征值分解，它只需要知道某个节点及其周围的节点就能够进行计算，因此具有更高的计算效率、泛化性能和扩展性。

在 ChebNet 的基础上做一阶近似（K=1）可以得到著名的 graph convolutional network（GCN）

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta(\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}) \mathbf{x}. \quad (11)$$

同时，GCN 也可以表示为 spatial-based 的形式，

$$\mathbf{h}_v = f(\Theta^T (\sum_{u \in \{N(v) \cup v\}} \bar{A}_{v,u} \mathbf{x}_u)) \quad \forall v \in V. \quad (13)$$

即在计算中只需要知道每个节点周围的节点，而不需要知道整个图就可以计算。因此，GCN 也可以看做是从 spectral-based 到 spatial-based 的一个统一。

其他的 spatial-based 方法包括如下模型，这里就贴一下文章里面的表，具体可以看原文。（很多我也懒得看了）

TABLE III: Summary of RecGNNs and ConvGNNs. Missing values (“-”) in pooling and readout layers indicate that the method only experiments on node-level/edge-level tasks.

Approach	Category	Inputs	Pooling	Readout	Time Complexity
GNN* (2009) [15]	RecGNN	A, X, X^e	-	a dummy super node	$O(m)$
GraphESN (2010) [16]	RecGNN	A, X	-	mean	$O(m)$
GGNN (2015) [17]	RecGNN	A, X	-	attention sum	$O(m)$
SSE (2018) [18]	RecGNN	A, X	-	-	-
Spectral CNN (2014) [19]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling	max	$O(n^3)$
Henaff et al. (2015) [20]	Spectral-based ConvGNN	A, X	spectral clustering+max pooling	-	$O(n^3)$
ChebNet (2016) [21]	Spectral-based ConvGNN	A, X	efficient pooling	sum	$O(m)$
GCN (2017) [22]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
CayleyNet (2017) [23]	Spectral-based ConvGNN	A, X	mean/gracius pooling	-	$O(m)$
AGCN (2018) [40]	Spectral-based ConvGNN	A, X	max pooling	sum	$O(n^2)$
DualGCN (2018) [41]	Spectral-based ConvGNN	A, X	-	-	$O(m)$
NN4G (2009) [24]	Spatial-based ConvGNN	A, X	-	sum/mean	$O(m)$
DCNN (2016) [25]	Spatial-based ConvGNN	A, X	-	mean	$O(n^2)$
PATCHY-SAN (2016) [26]	Spatial-based ConvGNN	A, X, X^e	-	sum	-
MPNN (2017) [27]	Spatial-based ConvGNN	A, X, X^e	-	attention sum/set2set	$O(m)$
GraphSage (2017) [42]	Spatial-based ConvGNN	A, X	-	-	-
GAT (2017) [43]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
MoNet (2017) [44]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
LGCN (2018) [45]	Spatial-based ConvGNN	A, X	-	-	-
PGC-DGCNN (2018) [46]	Spatial-based ConvGNN	A, X	sort pooling	attention sum	$O(n^3)$
CGMM (2018) [47]	Spatial-based ConvGNN	A, X, X^e	-	sum	-
GAAN (2018) [48]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
FastGCN (2018) [49]	Spatial-based ConvGNN	A, X	-	-	-
StoGCN (2018) [50]	Spatial-based ConvGNN	A, X	-	-	-
Huang et al. (2018) [51]	Spatial-based ConvGNN	A, X	-	-	-
DGCNN (2018) [52]	Spatial-based ConvGNN	A, X	sort pooling	-	$O(m)$
DiffPool (2018) [54]	Spatial-based ConvGNN	A, X	differential pooling	mean	$O(n^2)$
GeniePath (2019) [55]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
DGI (2019) [56]	Spatial-based ConvGNN	A, X	-	-	$O(m)$
GIN (2019) [57]	Spatial-based ConvGNN	A, X	-	sum	$O(m)$
ClusterGCN (2019) [58]	Spatial-based ConvGNN	A, X	-	-	-

对于 graph-level 的任务来说，还需要从图上每个节点把信息聚合起来，这就涉及到很多 pooling 的技术，比如专栏前面一篇讲的。这篇综述还讲了很多面向这方面的内容。

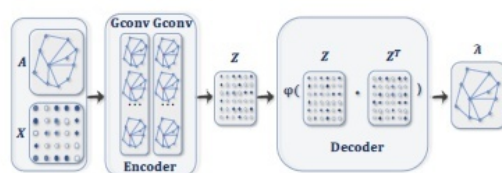
这篇综述还讲了一些理论方面的进展，主要包括以下几方面：

- Receptive field: 通过有限次 convolution 之后每个节点都能接收到整个图的信息。
- VC dimension: 模型的复杂度随着模型参数和节点数目多项式级增长。
- Graph isomorphism: GNN 提取出来的特征是否能够反映拓扑结构上的相似性。
- Equivariance and invariance: 对于图上节点的特征提取而言，要求特征变换函数 $f \in \mathbb{R}^d$ 满足 equivariance，即对于一个 permutation matrix Q ，有 $f(QAQ^T, QX) = Qf(A, X)$ 。对于图的特征提取函数 $f \in \mathbb{R}$ ，要求它满足 invariance，即 $f(QAQ^T, QX) = f(A, X)$ 。

4. GAE

文章把这一部分按照任务的目标分为了两类，一类的目标是提取整个网络的 network embedding，另一类的目标是 graph generation。

Network embedding 的目标是学习到网络的一个低维表示，使得这个低维表示能够保持节点的拓扑结构。Graph generation 按照做法上又可以分为两种不同的类型：sequential manner 和 global manner。前一种会一步一步地生成图中的节点或者连边；后一种一次性生成。这一部分跟我们要做的似乎没啥太大关系，直接贴图。



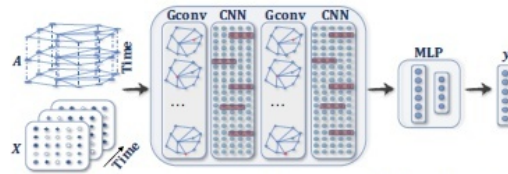
(c) A GAE for network embedding [61]. The encoder uses graph convolutional layers to get a network embedding for each node. The decoder computes the pair-wise distance given network embeddings. After applying a non-linear activation function, the decoder reconstructs the graph adjacency matrix. The network is trained by minimizing the discrepancy between the real adjacency matrix and the reconstructed adjacency matrix.

TABLE V: Main characteristics of selected GAEs

Approaches	Inputs	Encoder	Decoder	Objective
DNGR (2016) [59]	A	a multi-layer perceptron	a multi-layer perceptron	reconstruct the PPMI matrix
SDNE (2016) [60]	A	a multi-layer perceptron	a multi-layer perceptron	preserve node 1st-order and 2nd-order proximity
GAE* (2016) [61]	A, X	a ConvGNN	a similarity measure	reconstruct the adjacency matrix
VGAE (2016) [61]	A, X	a ConvGNN	a similarity measure	learn the generative distribution of data
ARVGA (2018) [62]	A, X	a ConvGNN	a similarity measure	learn the generative distribution of data adversarially
DNRE (2018) [63]	A	an LSTM network	an identity function	recover network embedding
NetRA (2018) [64]	A	an LSTM network	an LSTM network	recover network embedding with adversarial training
DeepGMG (2018) [65]	A, X, X^o	a RecGNN	a decision process	maximize the expected joint log-likelihood
GraphRNN (2018) [66]	A	a RNN	a decision process	maximize the likelihood of permutations
GraphVAE (2018) [67]	A, X, X^o	a ConvGNN	a multi-layer perceptron	optimize the reconstruction loss
RGVAE (2018) [68]	A, X, X^o	a CNN	a deconvolutional net	optimize the reconstruction loss with validity constraints
MolGAN (2018) [69]	A, X, X^o	a ConvGNN	a multi-layer perceptron	optimize the generative adversarial loss and the RL loss
NetGAN (2018) [70]	A	an LSTM network	an LSTM network	optimize the generative adversarial loss

5. STGNN

这一类最主要的应用场景还是在 traffic 上面，各个节点是动态变化的，所以同时抽取时间域上的信息和空间域上的信息。前面很多方法都在解决如何抽取图上的空间域信息，这里还会另外地抽取时间域上的信息，主要的方法有两类：RNN-based 和 CNN-based。



(d) A STGNN for spatial-temporal graph forecasting [74]. A graph convolutional layer is followed by a 1D-CNN layer. The graph convolutional layer operates on A and $X^{(t)}$ to capture the spatial dependency, while the 1D-CNN layer slides over X along the time axis to capture the temporal dependency. The output layer is a linear transformation, generating a prediction for each node, such as its future value at the next time step.

6. 其他

文章还讲了 benchmark 的数据集和 sota 的一些结果，不是很感兴趣没仔细看。不过后面列举的一些 GNN 的开源代码感觉还不错，这里记录一下。

TABLE VIII: A Summary of Open-source Implementations

Model	Framework	Github Link
GGNN (2015)	torch	https://github.com/yujiali/ggnn
SSE (2018)	c	https://github.com/HanJun-Dai/steady_state_embedding
ChebNet (2016)	tensorflow	https://github.com/mdeff/cnn_graph
GCN (2017)	tensorflow	https://github.com/tkipf/gcn
CayleyNet (2017)	tensorflow	https://github.com/amolliu/CayleyNet
DualGCN (2018)	theano	https://github.com/ZhuangCY/DGCN
GraphSage (2017)	tensorflow	https://github.com/williamleif/GraphSAGE
GAT (2017)	tensorflow	https://github.com/PetarV-/GAT
LGCN (2018)	tensorflow	https://github.com/divelab/lgcnn/
PGC-DGCNN (2018)	pytorch	https://github.com/dinhfotech/PGC-DGCNN
FastGCN (2018)	tensorflow	https://github.com/matenure/FastGCN
StoGCN (2018)	tensorflow	https://github.com/thu-ml/stochastic_gcn
DGCNN (2018)	torch	https://github.com/muhanzhang/DGCNN
DiffPool (2018)	pytorch	https://github.com/RexYing/diffpool
DGI (2019)	pytorch	https://github.com/PetarV-/DGI
GIN (2019)	pytorch	https://github.com/weihua916/powerful-gnns
Cluster-GCN (2019)	pytorch	https://github.com/benedekrozemberczki/ClusterGCN
DNGR (2016)	matlab	https://github.com/SheltonCao/DNGR
SDNE (2016)	tensorflow	https://github.com/suanrong/SDNE
GAE (2016)	tensorflow	https://github.com/limacsen/Variational-Graph-Auto-Encoders
ARVGA (2018)	tensorflow	https://github.com/Ruiqi-Hu/ARVGA
DRNE (2016)	tensorflow	https://github.com/tadpole/DRNE
GraphRNN (2018)	tensorflow	https://github.com/snap-stanford/GraphRNN
MolGAN (2018)	tensorflow	https://github.com/nicola-decao/MolGAN
NetGAN (2018)	tensorflow	https://github.com/danielzuegner/netgan
GCRN (2016)	tensorflow	https://github.com/youngjoo-epfl/gconv_RNN
DCRNN (2018)	tensorflow	https://github.com/liyaguang/DCRNN
Structural RNN (2016)	theano	https://github.com/asheshjain399/RNNexp
CGCN (2017)	tensorflow	https://github.com/VeritasYin/STGCN_UCAI-18
ST-GCN (2018)	pytorch	https://github.com/yysjke/st-gcn
GraphWaveNet (2019)	pytorch	https://github.com/nanzhan/Graph-WaveNet
ASTGCN (2019)	mxnet	https://github.com/Davidham3/ASTGCN

知乎 @张楚珩

未来的研究方向上，文章列了这么几点：

- 如何有效地提升模型复杂度：因为 convolution 层变多时，各个节点的特征将会变得越来越接近，加多层数最后会使得所有的点上的特征都变成一样的，因此不能单独靠把模型做深来提高模型复杂度。
- 如何提高模型的拓展性：当图的规模变得特别大时，就需要考虑如何来对图进行聚合并且尽量不要丢失图上的信息。有两种思路：sampling 可能会使得节点丢失一些很关键的邻居；clustering 可能会使得图丢失一些比较特别的结构模式。
- 如何融合异源数据：真实应用场景中，图可能会有不同类型的节点、连边，如何处理这些数据也将成为一个研究方向。
- 如何处理动态的图：就想 STGNN 中所做的事情一样。

编辑于 2020-02-16

arXiv

图神经网络 (GNN)

学术论文

▲ 赞同 33

● 2 条评论


🔗 分享

❤ 喜欢

★ 收藏

...

文章被以下专栏收录



强化学习前沿

读呀读paper

进入专栏