PR Structured Ⅰ：Graph Neural Network An Introduction

最近，我逐渐将研究重心从MDP转向了结构化建模。无论是我的专栏写作进度，还是科研中的项目要求，都同样指向了结构化建模，这么巧肯定说明了结构化建模是大势所趋。在这个19年都是老研究的AI领域，2020年年中的我还对已经火到巅峰的**图神经网络**一知半解就说不过去了。

本文根据以下三篇综述，旨在入门 **Graph Neural Networks**：

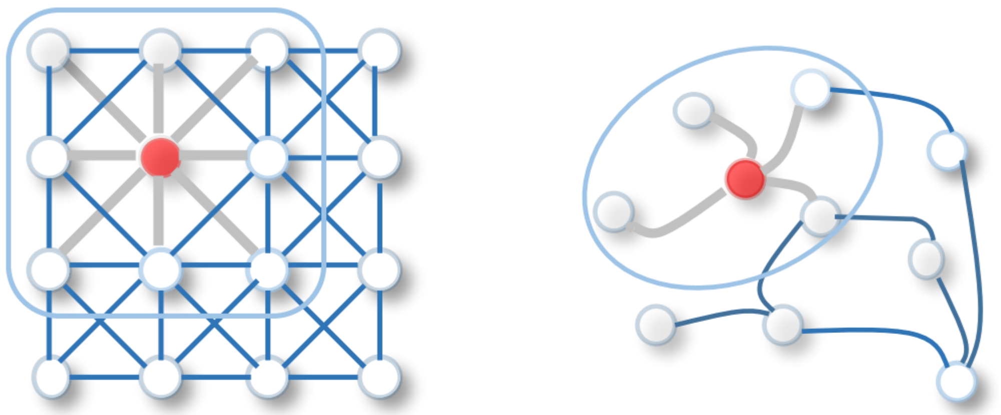
- A Comprehensive Survey on Graph Neural Networks | 2019 Jan
- Graph Neural Networks: A Review of Methods and Applications | 2018 Dec
- How Powerful are Graph Neural Networks? | 2018 Oct

1 Introduction

我们知道，CNNs、RNNs以及 autoencoders 等深度学习方法，可以取代手工的特征提取，有效地捕获**欧氏**数据的隐含特征。但现实生活中，数据更普遍的形式是可以被构建为图的非欧数据。例如，化学分子结构、知识图谱、电子商务等。

由于图可能是不规则的，节点大小、邻居数量不同，从而传统深度学习难以应用于图域。此外，现有机器学习算法的核心假设是实例彼此独立。这种假设不再适用于图数据，因为每个实例（节点）通过各种类型的连接与其他节点相关联。

图神经网络主要解决 **表示对象之间复杂关系的非欧氏域数据处理问题**。传统的 Deep Leanring methods 可以视为图神经网络在欧氏情况下的**子集**。



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Fig. 1: 2D Convolution vs. Graph Convolution.

现阶段的图神经网络可以分为以下四种：

- **recurrent graph neural networks**
- **convolutional graph neural networks**
- **graph autoen- coders**
- **spatial-temporal graph neural networks**

1.1 GNN 与 network embedding

Network embedding 旨在将网络的节点表示为低维的向量。

GNN 可以以端到端的方式抽取 high-level 的表征。

不同在于：GNN是一组针对各种任务而设计的神经网络模型，而网络嵌入涵盖了针对同一任务的各种方法。

1.2 GNN 与 graph kernel methods

graph kernel methods 是用于解决图分类问题，使SVM这种基于 kernel 的方法可以用于图数据的监督学习。此外，graph kernel methods 也可以用于 embed graphs or nodes。区别在于，这种 embedding mapping 是确定性的 func，而非 learnable。GNN直接提取图信息的表征来做分类比 graph kernel methods 更有效。

1.3 Definition of GNN

图 (Graph) 常被表示为 $G = (V, E)$：

- V 表示**节点 (nodes) 或顶点 (vertices)**，E 表示连接节点的**边 (edges)**；
- $e_{ij} = (v_i, v_j) \in E$ 就代表了连接 i, j 两个节点的边；
- 节点 v 的**邻居**表示为：$N(v) = \{u \in V | (v, u) \in E\}$；
- 图可以写成**邻接矩阵**的形式，$A_{ij} = 1 \text{ if } e_{ij} \in E, A_{ij} = 0 \text{ if } e_{ij} \notin E$；
- 节点属性可以写作节点的特征矩阵 $\boldsymbol{X}$，边的属性可以写作边的特征矩阵 $\boldsymbol{X}^e$

有向图 (Directed Graph) 指所有边都从一个节点指向另一个节点的图，无向图是邻接矩阵对称的有向图。

时空图 (Spatial-Temporal Graph) 是一个属性图，其中节点属性随**时间动态变化**。$G^{(t)} = \left(\mathbf{V}, \mathbf{E}, \mathbf{X}^{(t)}\right)$ with $\mathbf{X}^{(t)} \in \mathbf{R}^{n \times d}$

2 Categorization & Frameworks

2.1 Categorization

Recurrent graph neural networks (RecGNNs)

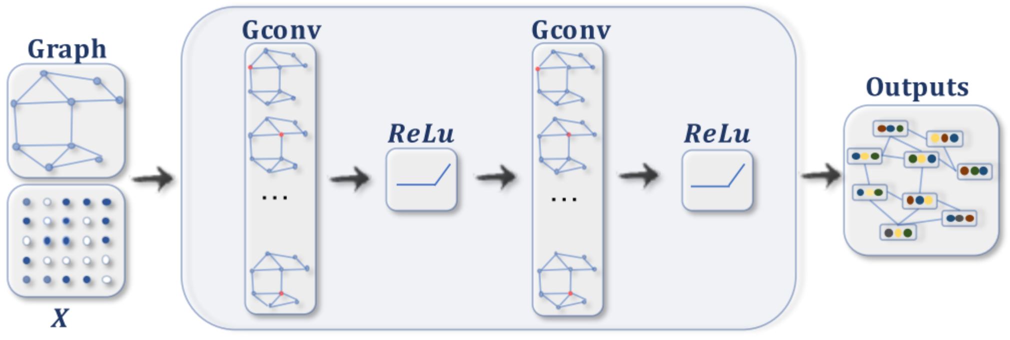
GNN 的 先驱，假设图中的一个节点不断与其邻居交换信息/消息，直到达到稳定的平衡。其消息传递的思想被**空域卷积图神经网络 (spatial- based convolutional graph neural networks)**所继承。

Convolutional graph neural networks (ConvGNNs)

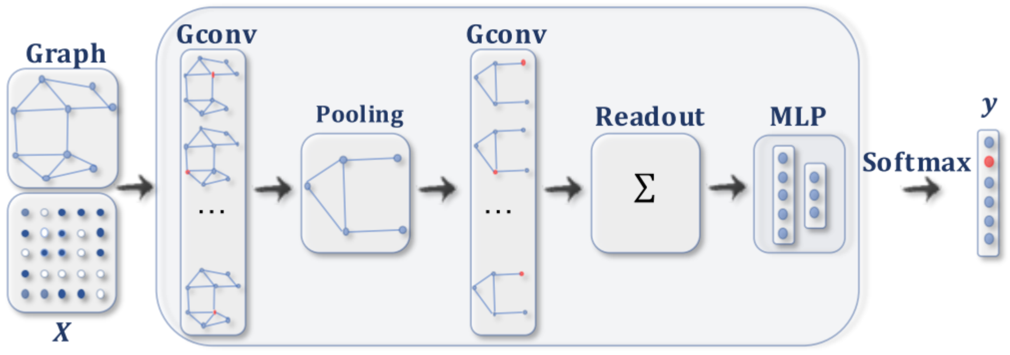
主要分两种：

- Spectral methods
- Spatial methods

将卷积的思想从 grid data 拓展到 graph data。主要思想是通过汇总节点自身的特征 x_v 和邻居的特征 x_u 来生成节点 v 的表征形式，其中 $u \in N(v)$。通过堆叠多个图卷积层可以提取高级节点表征。下图 a 表示节点分类，b 表示图分类过程。



(a) A ConvGNN with multiple graph convolutional layers. A graph convolutional layer encapsulates each node's hidden representation by aggregating feature information from its neighbors. After feature aggregation, a non-linear transformation is applied to the resulted outputs. By stacking multiple layers, the final hidden representation of each node receives messages from a further neighborhood.



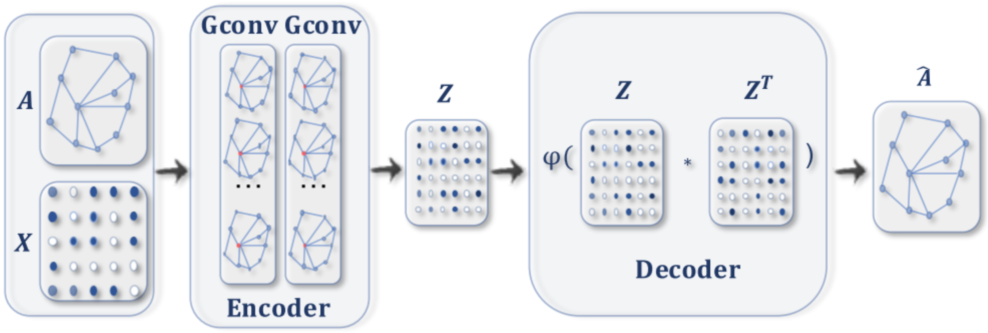
(b) A ConvGNN with pooling and readout layers for graph classification [21]. A graph convolutional layer is followed by a pooling layer to coarsen a graph into sub-graphs so that node representations on coarsened graphs represent higher graph-level representations. A readout layer summarizes the final graph representation by taking the sum/mean of hidden representations of sub-graphs.

Graph autoencoders (GAEs)

是一种无监督的学习框架，可将节点/图编码到隐向量空间中，并从编码后的信息中重建图数据。用于学习网络嵌入和图生成分布。

- Network Embedding，GAE通过重建图结构信息（例如图邻接矩阵）来学习潜在节点表示。
- Graph Generation，某些方法逐步生成图的节点和边，而其他方法则一次全部输出图。

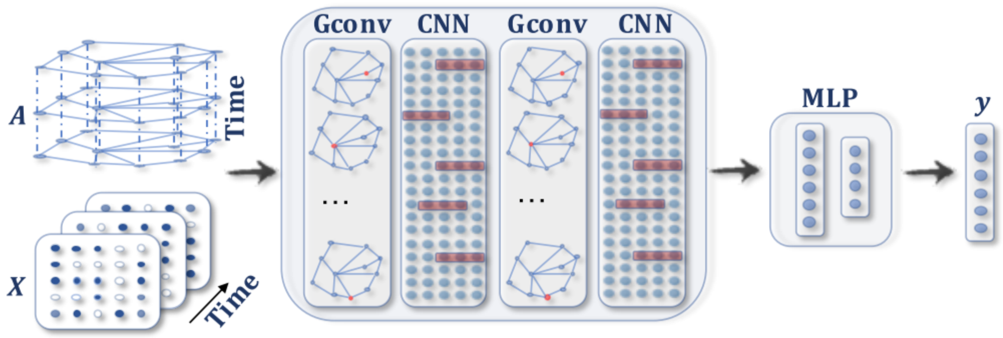
下图展示了用于网络嵌入的GAE。



(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.

Spatial-temporal graph neural networks (STGNNs)

STGNNs 同时考虑空间依赖性和时间依赖性。当前许多方法将图卷积与RNN或CNN集成在一起以捕获空间依赖性，从而对时间依赖性进行建模。旨在从时空图学习隐式特征。下图是用于时空图预测的STGNN。



(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.

2.2 Frameworks

使用图结构和节点内容信息作为输入，GNN的输出可以通过以下机制之一专注于不同的图分析任务：

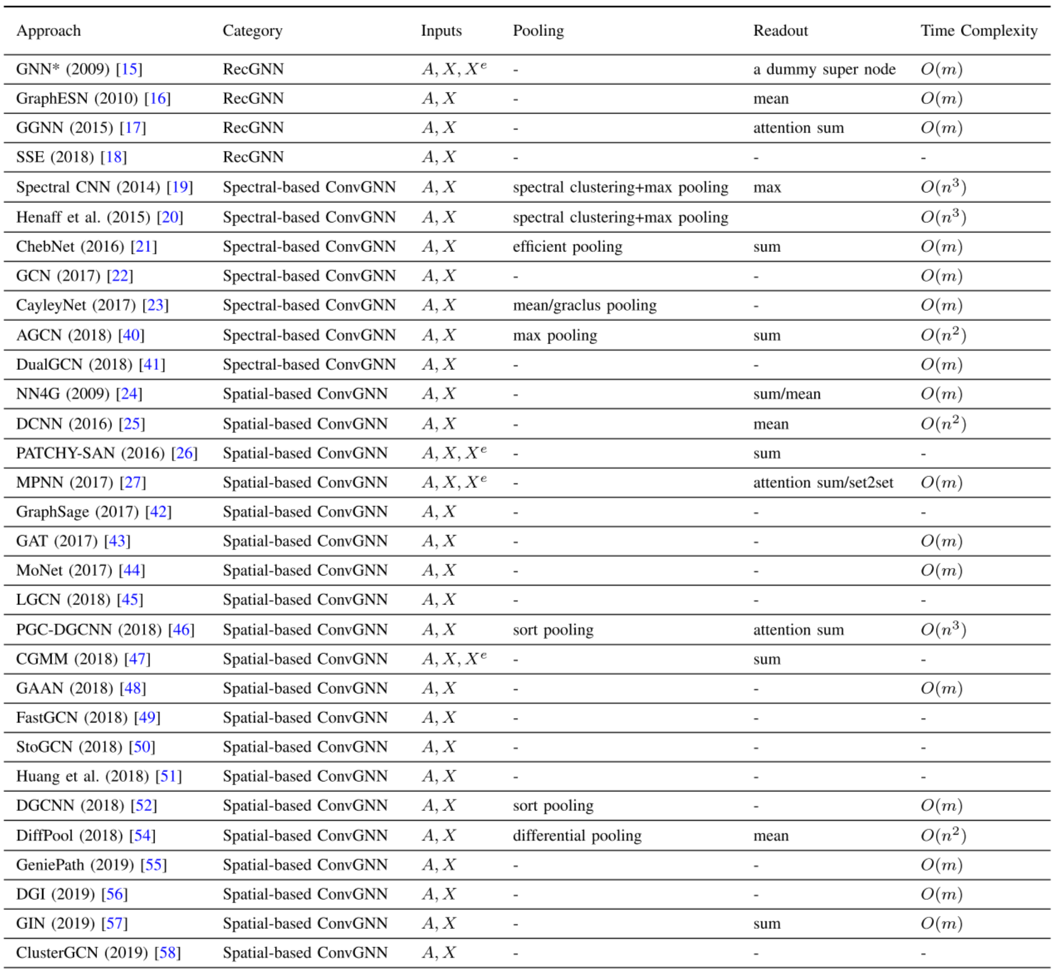
- Node-level：输出与节点回归和节点分类任务有关；
- Edge-level：输出与边缘分类和连接预测任务有关；
- Graph-level：输出与图分类任务有关。

训练方法：

- Semi-supervised learning for node-level classification：部分节点有label，通过堆叠 ConvGNN + softmax 可以实现半监督的多分类；
- Supervised learning for graph-level classification：图级别的监督分类；
- Unsupervised learning for graph embedding：以两种方式利用边的信息
    - 采用自动编码器框架，其中编码器采用图卷积层将图嵌入到潜在表示中，在该表示中使用解码器来重构图结构；
    - 利用负采样方法，该方法将一部分节点对采样为负对，而图中具有链接的现有节点对为正对。然后应用逻辑回归层来区分正对和负对。

2.3 近年GNN算法总览

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/graclus 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	-	-	-

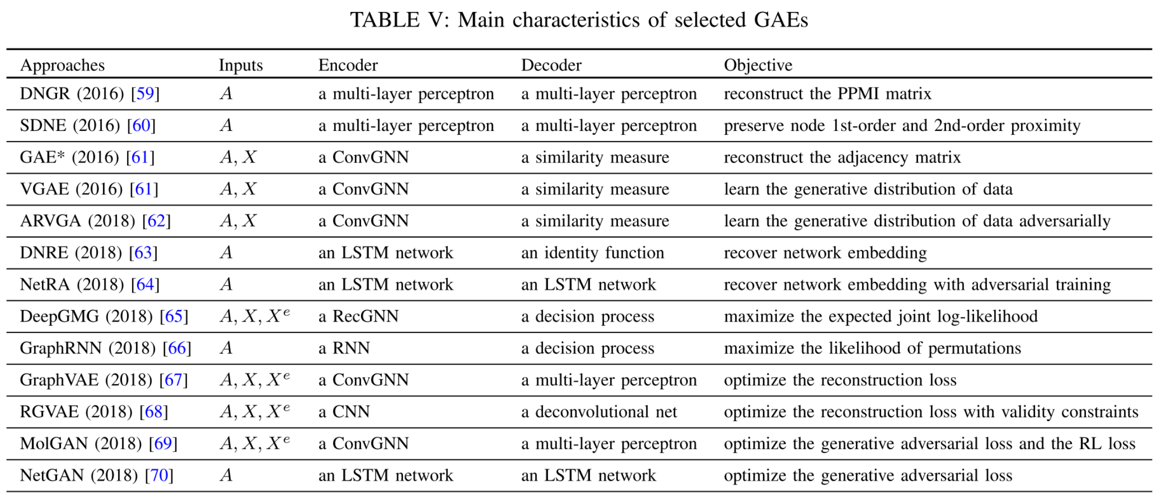


TABLE V: Main characteristics of selected GAEs

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^e	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^e	a ConvGNN	a multi-layer perceptron	optimize the reconstruction loss
RGVAE (2018) [68]	A, X, X^e	a CNN	a deconvolutional net	optimize the reconstruction loss with validity constraints
MolGAN (2018) [69]	A, X, X^e	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

下面详细介绍这四种类别 GNN 的理论与架构。

3 Recurrent graph neural networks

Vanilla GNN (Scarselli et al.) 基于信息传播机制，根据邻居节点之间信息的循环交换来达到稳定的平衡态。节点的隐状态可表示为：

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

Graph Echo State Network (GraphESN) 提高了 Vanilla GNN 的训练效率。GraphESN 由一个 encoder 和一个 output layer 构成。它实现了 contractive状态转移方程来循环更新节点状态，直到全局图状态达到收敛为止。之后，通过将固定的节点状态作为输入来训练输出层。

Gated Graph Neural Network (GGNN) 利用GRU作为循环方程，将重复执行减少到固定的步骤数。好处是不再需要约束参数以确保收敛。节点隐状态由其先前的隐状态及其相邻的隐状态更新：

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

其使用 BPTT 训练参数，所有节点的中间状态要存储在内存中，可能不易于大型图的计算。

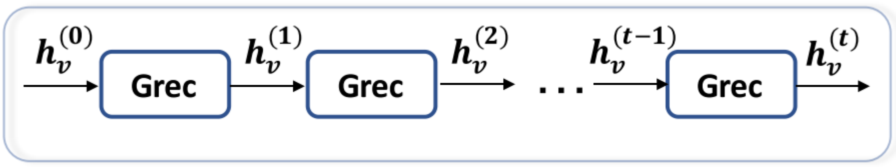
Stochastic Steady-state Embedding (SSE) 以随机和异步方式周期性地更新节点隐状态，解决了扩展到大型图的问题。它交替的采样 batch 用于节点的状态更新与节点的梯度计算。为保证稳定性，SSE 取了过去状态与现在状态的加权平均。

$$\mathbf{h}_v^{(t)} = (1-\alpha)\mathbf{h}_v^{(t-1)} + \alpha \mathbf{W}_1 \sigma\left(\mathbf{W}_2\left[\mathbf{x}_v, \sum_{u\in N(v)}\left[\mathbf{h}_u^{(t-1)}, \mathbf{x}_u\right]\right]\right) \quad (3)$$

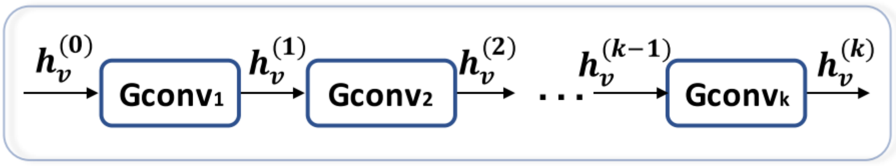
SSE 并没有收敛的理论保证。

4 Convolutional graph neural networks

区别于递归图神经网络，ConvGNN不是使用收缩约束来迭代节点状态，而是使用固定数量的具有不同权重的层在体系结构上解决循环的相互依赖性。



(a) Recurrent Graph Neural Networks (RecGNNs). RecGNNs use the same graph recurrent layer (Grec) in updating node representations.



(b) Convolutional Graph Neural Networks (ConvGNNs). ConvGNNs use a different graph convolutional layer (Gconv) in updating node representations.

Fig. 3: RecGNNs v.s. ConvGNNs

- Spectral- based：通过从图信号处理的角度引入滤波器来定义图卷积，其中图卷积运算被解释为从图信号中去除噪声。
- Spatial-based：继承了RecGNN的思想，以通过信息传播来定义图卷积。自从GCN 弥合了基于频谱的方法与基于空间的方法之间的差距以来，基于空间的方法由于其引人注目的效率，灵活性和通用性而迅速发展。

4.1 Spectral-based ConvGNNs

4.1.1 基础

卷积操作本身就是从数字信号处理中引申出来的，所以先有了基于频谱的方法并不奇怪。该方法假设图是无向的。归一化图拉普拉斯矩阵是无向图的数学表示，定义为：

$$\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \tag{4}$$

D 是表示 node degrees 的对角矩阵，$\mathbf{D}_{ii} = \sum_j (\mathbf{A}_{i,j})$。

归一化图拉普拉斯矩阵具有实对称半正定性质，可被分解成特征矩阵 $\mathbf{L} = \mathbf{U\Lambda U}^T$。

在**图信号处理**中，一个图信号 $\mathbf{X}$ 表示图节点的 feature vector，其 **图傅里叶变换** 表示为 $\mathscr{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$，逆变换为 $\mathscr{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$。图傅立叶变换将输入图信号投影到正交空间，正交基由归一化图拉普拉斯算子的特征向量形成。现在图信号就被映射到频谱空间。**基于频谱的图卷积**定义为：

$$\begin{aligned} \mathbf{x} *_G \mathbf{g} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})) \\ &= \mathbf{U}\left(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}\right) \end{aligned}$$

可将其中的 $\mathbf{g}_\theta = \mathrm{diag}\left(\mathbf{U}^T\mathbf{g}\right)$ 定义为**滤波器 (filter)**。定义简化为

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \tag{5}$$

4.1.2 方法

Spectral Convolutional Neural Network (Spectral CNN) 将滤波器视作可学的参数并考虑具有多个通道的图信号。图卷积层定义为：

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

4.2 Spatial-based ConvGNNs

4.3 Graph Pooling Modules

5 Graph Autoencoders

5.1 Network Embedding

5.2 Graph Generation

6 Spatial-temporal graph neural networks

7 Applications

8 Future Directions