《EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs》

《EvolveGCN：针对动态图的演化图卷积网络》

摘要：

1. **本文的背景：**随着图神经网络(GNN)在静态场景中的成功，研究者们开始进一步讨论由动态图演化的实际场景。然而，大多数现有的方法是先通过 GNN 来做每个时刻图节点的嵌入，然后再使用递归神经网络(RNN)进一步调节嵌入和学习动态时间关联的。这些方法需要了解任意节点的全部时间跨度，且并不能很好地适应频繁变化的节点集。因此，为了解决上述缺陷，本文试图通过使用 RNN 来演化模型 GCN 的权重参数而非其节点的嵌入表示，以便更好地捕获动态信息。
2. **本文的贡献：**提出了一种新型的图神经网络架构——演化图卷积网络（$EvolveGCN$），通过使用RNN来演化GCN参数，以捕获图序列的动态特征。
3. **主要创新点：**从关注于节点的嵌入转为关注于模型本身，解除了节点的变化对于训练模型的限制；通过使用RNN来训练GCN参数，使得模型的参数数量（模型大小）不会随着时间步数的增加而增加。
4. **实验结果：**研究结果表明，在链接预测、边缘分类和节点分类等任务上，与相关基线方法相比，$EvolveGCN$的性能普遍较高。

符号规范：

在本文中，我们使用下标t来表示时间索引，上标l来表示GCN层的索引。此外，为了避免术语在表示上的混乱，我们假设所有时刻的图都有N个节点。**但特别注意：实际中的节点集和节点数都可能会随时间而发生变化。**

接着，在时间步t时刻，我们假设输入数据都是由一对数据（$A_t \in \mathbb{R}^{N\times N}$，$X_t \in \mathbb{R}^{N\times d}$）组成的。其中，$A_t$是图的（加权）邻接矩阵，$X_t$是输入节点集的特征矩阵。具体来说，$X_t$的每一行都是相应节点的$d$维特征向量。

图卷积网络（GCN）：

GCN的计算过程可以表示为：

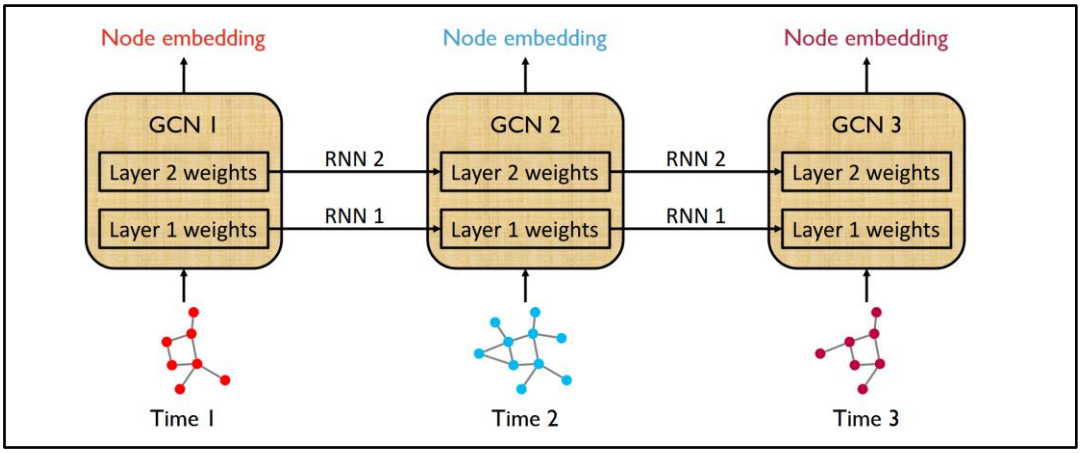
$$H_t^{(l+1)} = GCONV\left(A_t, H_t^{(l)}, W_t^{(l)}\right)$$

$$:= \sigma(\hat{A}_t, H_t^{(l)}, W_t^{(l)})$$

其中，$\hat{A}_t$是A_t的标准化定义，具体表示如下：

$$\hat{A}_t = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}, \qquad \tilde{A} = A + I, \qquad \tilde{D} = diag(\sum_j \tilde{A}_{ij})$$

σ是除输出层外所有层的激活函数（通常为$ReLU$）。此外，初始嵌入矩阵来自节点特征，即$H_t^{(0)} = X_t$。



上图是$EvolveGCN$的整体示意图，其中每个时间步包含一个以时间为索引的GCN。而GCN中的参数是不同时间步长t和层数l的权重矩阵$W_t^{(l)}$。**注意：**在$EvolveGCN$框架中，GCN模型本身并不训练权重矩阵，权重矩阵$W_t^{(l)}$的训练是由RNN来负责的。

权重演化（Weight Evolution）：

实际上，$EvolveGCN$框架的核心在于权重矩阵$W_t^{(l)}$的更新。而根据当前和历史信息，这种需求可以很自然的由递归结构(RNN)来实现。所以，本文提出了两种方案。

方案一：将$W_t^{(l)}$作为动态系统的隐藏状态，并使用GRU来根据系统在t时刻下的输入更新隐藏层状态，相关公式表达如下：

$$\overbrace{W_t^{(l)}}^{\text{GCN weights}} = \text{GRU}(\overbrace{H_t^{(l)}}^{\text{node embeddings}}, \overbrace{W_{t-1}^{(l)}}^{\text{GCN weights}}),$$

$W_t^{(l)}$: hidden state; $H_t^{(l)}$: input; $W_{t-1}^{(l)}$: hidden state

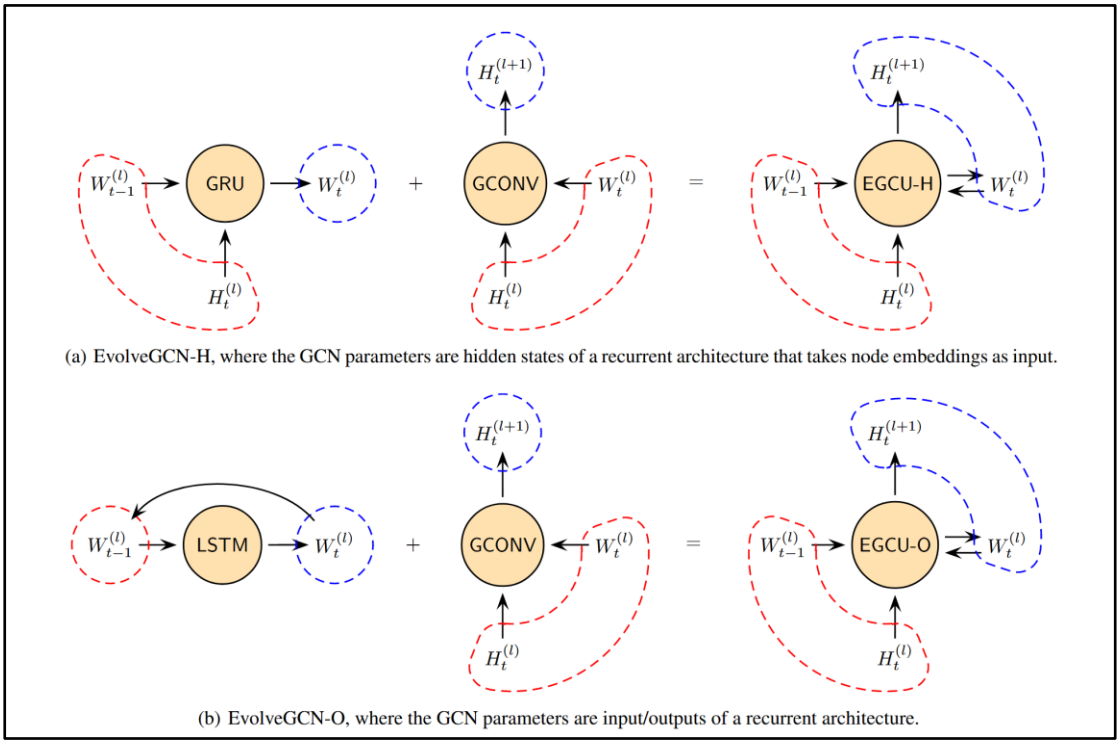
论文中用“$EvolveGCN-H$”来表示上述方案。

方案二：将$W_t^{(l)}$作为动态系统的输出（之后会成为下个时间点的输入），使用$LSTM$单元来建模输入和输出的关系，$LSTM$通过单元上下文来获得系统信息。此外，在该方案中节点的嵌入表示$H_t^{(l)}$也不会被使用为输入。相关公式表达如下：

$$\overbrace{W_t^{(l)}}^{\text{GCN weights}} = \text{LSTM}(\overbrace{W_{t-1}^{(l)}}^{\text{GCN weights}}),$$

$W_t^{(l)}$: output; $W_{t-1}^{(l)}$: input

论文中用“$EvolveGCN-O$”来表示上述方案。



(a) EvolveGCN-H, where the GCN parameters are hidden states of a recurrent architecture that takes node embeddings as input.

(b) EvolveGCN-O, where the GCN parameters are input/outputs of a recurrent architecture.

上图是两个不同方案（$EvolveGCN-H$、$EvolveGCN-O$）下的$EvolveGCN$结构。在每个方案中，最左边都是递归架构RNN；中间是图卷积单元$GCONV$；最右边是演化的图卷积单元$EGCU$。其中，红色区域表示输入单元的信息，蓝色区域表示输出的信息。符号W表示GCN的参数，H表示节点的嵌入。时间t从左至右，而神经网络层l则从下至上。

演化图卷积单元（*EGCU*）：

根据*GCN*权重$W_t^{(l)}$两种不同的演化方式，我们也会有两个不同版本的*EGCU*。它们的算法伪代码实现如下：

1: **function** $[H_t^{(l+1)}, W_t^{(l)}] = \text{EGCU-H}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
2: $\quad W_t^{(l)} = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)})$
3: $\quad H_t^{(l+1)} = \text{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$
4: **end function**

1: **function** $[H_t^{(l+1)}, W_t^{(l)}] = \text{EGCU-O}(A_t, H_t^{(l)}, W_{t-1}^{(l)})$
2: $\quad W_t^{(l)} = \text{LSTM}(W_{t-1}^{(l)})$
3: $\quad H_t^{(l+1)} = \text{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$
4: **end function**

可以看到，在$-H$版本中，*GCN*权重被视为递归结构的隐藏状态；而在$-O$版本中，这些权重被视为输入/输出。不过，在这两个版本中，*EGCU*都会沿层执行图卷积，同时随着时间的推移不断演化权重矩阵。

而如果我们自底向上连接这些*EGCU*，就会获得一个多时间步下的多层的*GCN*。然后，在水平方向上展开时间，这些单元就会形成一个架构，其中信息$H_t^{(l)}$和$W_t^{(l)}$会在其中流动。因此，这样的一个模型我们称之为**演化图卷积网络(*EvolveGCN*)**。

$-H$版本的实现：

相较于标准*GRU*，$EGCU-H$的实现需要进行以下改造：

1）将输入和隐藏状态从向量扩展到矩阵(因为隐藏状态现在是*GCN*权重矩阵)；

实现：使用相同的*GRU*来处理*GCN*权值矩阵的每一列。

*GRU*的函数形式：

1: **function** $H_t = g(X_t, H_{t-1})$
2: $\quad Z_t = \text{sigmoid}(W_Z X_t + U_Z H_{t-1} + B_Z)$
3: $\quad R_t = \text{sigmoid}(W_R X_t + U_R H_{t-1} + B_R)$
4: $\quad \widetilde{H}_t = \tanh(W_H X_t + U_H (R_t \circ H_{t-1}) + B_H)$
5: $\quad H_t = (1 - Z_t) \circ H_{t-1} + Z_t \circ \widetilde{H}_t$
6: **end function**

注意：这里的X与H只表示输入矩阵与隐藏状态，与之前的定义不同。

2）匹配输入的列维与隐藏状态的列维。

实现：将所有节点的嵌入向量汇总为k个具有代表性的向量，并且通过转置操作，将节点的嵌入向量作为输入矩阵的列向量。

具体操作过程如下：

1: **function** $Z_t = summarize(X_t, k)$
2: $y_t = X_t p / \|p\|$
3: $i_t = \text{top-indices}(y_t, k)$
4: $Z_t = [X_t \circ \tanh(y_t)]_{i_t}$
5: **end function**

因此，经过定义之后的循环神经网络的计算过程为：

$$W_t^{(l)} = \text{GRU}(H_t^{(l)}, W_{t-1}^{(l)}) := g(summarize(H_t^{(l)}, \#col(W_{t-1}^{(l)}))^T, W_{t-1}^{(l)}),$$

其中，$\#col$表示矩阵的列数，上标T表示矩阵的转置。实际上，它是将节点嵌入矩阵$H_t^{(l)}$总结为一个具有适当维度的矩阵。然后，将过去时间步$t-1$的权重矩阵$W_{t-1}^{(l)}$演化为当前时间t的权重矩阵$W_t^{(l)}$。

$-O$版本的实现：

实现$-O$版本只需要将标准$LSTM$从向量版本直接扩展到矩阵版本，因此算法伪代码如下：

1: **function** $H_t = f(X_t)$
2: Current input X_t is the same as the past output H_{t-1}
3: $F_t = \text{sigmoid}(W_F X_t + U_F H_{t-1} + B_F)$
4: $I_t = \text{sigmoid}(W_I X_t + U_I H_{t-1} + B_I)$
5: $O_t = \text{sigmoid}(W_O X_t + U_O H_{t-1} + B_O)$
6: $\widetilde{C}_t = \tanh(W_C X_t + U_C H_{t-1} + B_C)$
7: $C_t = F_t \circ C_{t-1} + I_t \circ \widetilde{C}_t$
8: $H_t = O_t \circ \tanh(C_t)$
9: **end function**

因此，经过定义之后的循环神经网络的计算过程为：

$$W_t^{(l)} = \text{LSTM}(W_{t-1}^{(l)}) := f(W_{t-1}^{(l)}).$$

$-H$和$-O$版本的选择：

选择正确的版本取决于数据集。当节点特征是信息性的，$-H$版本可能更有效，因为它合并了额外的节点嵌入到递归网络中。另一方面，如果节点特征的信息量不大，但图结构起着更重要的作用，那么$-O$版本因为更关注结构的变化，可能会更有效。

实验及结果：

1）数据集

本文的实验一共使用到了 7 个数据集，分别是 1）Stochastic Block Model（SBM）：一种广泛用于模拟群落结构和演化的随机图模型；2）Bitcoin OTC (BC-OTC)和 Bitcoin Alpha (BC-Alpha)：可以看做是比特币上的社交网络，对比特币地址间的可信任程度进行打分；3）UC Irvine messages（UCI）和 Autonomous systems（AS）：社交网络。4）Reddit Hyperlink Network（Reddit）：超链接网络；5）Elliptic：比特币交易网络，节点为交易，边为比特币流动。

各数据集的情况如下表所示：

Table 1: Data sets.

	# Nodes	# Edges	# Time Steps (Train / Val / Test)
SBM	1,000	4,870,863	35 / 5 / 10
BC-OTC	5,881	35,588	95 / 14 / 28
BC-Alpha	3,777	24,173	95 / 13 / 28
UCI	1,899	59,835	62 / 9 / 17
AS	6,474	13,895	70 / 10 / 20
Reddit	55,863	858,490	122 / 18 / 34
Elliptic	203,769	234,355	31 / 5 / 13

2）验证任务

- **链路预测：**指标为 mean average precision (MAP)和 mean reciprocal rank (MRR)
- **边分类：**预测边(u, v)在时间t的标签，指标为 precision、recall 和 F1；
- **节点分类：**预测节点u在时间t的标签，指标同上。

3）对比算法

• **GCN**：没有任何时态建模的GCN，损失是沿时间轴累积的；

• **GCN-GRU**：也是一个单一的GCN模型，但是它在节点嵌入上与一个递归模型（GRU）协同训练；

• **DynGEM**：基于图数据自动编码器的无监督节点嵌入方法；

• **dyngraph2vec**：也是一种无监督的方法，且它有两种变体 dyngraph2vecAE 和 dyngraph2vecAERNN。

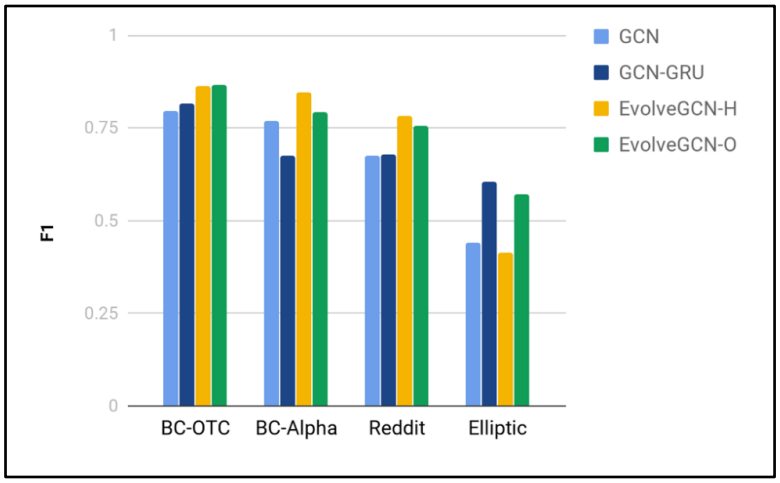
4）链路预测结果

Table 2: Performance of link prediction. Each column is one data set.

	mean average precision					mean reciprocal rank				
	SBM	BC-OTC	BC-Alpha	UCI	AS	SBM	BC-OTC	BC-Alpha	UCI	AS
GCN	0.1987	0.0003	0.0003	0.0251	0.0003	0.0138	0.0025	0.0031	0.1141	0.0555
GCN-GRU	0.1898	0.0001	0.0001	0.0114	0.0713	0.0119	0.0003	0.0004	0.0985	0.3388
DynGEM	0.1680	0.0134	0.0525	0.0209	0.0529	0.0139	0.0921	0.1287	0.1055	0.1028
dyngraph2vecAE	0.0983	0.0090	0.0507	0.0044	0.0331	0.0079	0.0916	0.1478	0.0540	0.0698
dyngraph2vecAERNN	0.1593	**0.0220**	**0.1100**	0.0205	0.0711	0.0120	**0.1268**	**0.1945**	0.0713	0.0493
EvolveGCN-H	0.1947	0.0026	0.0049	0.0126	**0.1534**	**0.0141**	0.0690	0.1104	0.0899	**0.3632**
EvolveGCN-O	**0.1989**	0.0028	0.0036	**0.0270**	0.1139	0.0138	0.0968	0.1185	**0.1379**	0.2746

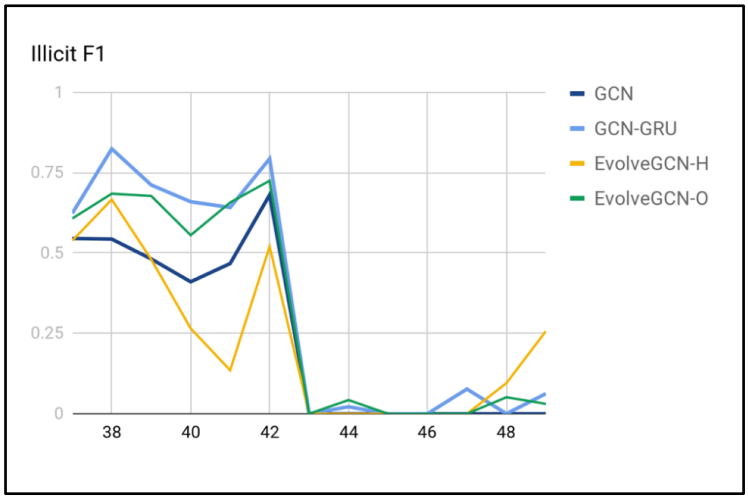
可以看到，$EvolveGCN$模型在 3 个数据集上的效果达到了最佳，但在剩余两个数据集上其好于GCN但仍然弱于Dyn等图自编码器模型，一定原因来自随机的初始化节点。因此，虽然$EvolveGCN$在很大程度上优于GCN，但它仍然没有达到图自动编码所达到的效果。

5）边分类结果



可以看到，在三个数据集中，两种版本的$EvolveGCN$都比$baseline$的GCN和$GCN-GRU$要好。

6）节点分类结果



节点分类只做了一个实验（Elliptic 数据集），在比特币交易网络上对非法类（illicit）和合法类（licit）交易进行分类，如上图所示。而作者发现各种方法在 average micro-F1 都高于 0.95，没有区分度。而对于金融犯罪取证来说，非法类是主要关注，所以展示的是非法类的$F1$结果。而我们可以看到，$EvolveGCN-O$比静态的方法GCN表现更好，但不如$GCN-GRU$那么好。

结论：

$EvolveGCN$将RNN和GCN更加有效地结合了起来，通过利用RNN来演化GCN的权重参数，使得整个模型能够应对频繁变化的图，且不需要提前预知节点的所有变化。但是，它也有一定的局限性：1）虽然不需要提前预知节点的所有变化，但是需要预知图中的所有结点，不能应对节点的变化；2）一个训练好的$EvolveGCN$模型往往只能捕捉一种动态性。

对本文的感悟：

简单来说，本文的研究者做出的最大创新点在于将以往研究的重点从用RNN模型关注于节点的嵌入表示转变为利用RNN模型来重点关注于图模型本身。而这样的方式虽然在一定程度上使模型获得了提升，但效果却一般。但我认为它这样做的最突出的优势是使得模型的参数数量（模型大小）不会随着时间步数的增加而增加。因此，总的来说，这也是一个很好的待研究方向。