论文阅读笔记 FW2021

(GIN) How Powerful Are Graph Neural Networks

https://arxiv.org/abs/1810.00826

研究背景

模型

Weisfeiler-Lehman Isomorphism Test （图的同构测试）

首先将图中每个节点染同一种颜色。然后每一轮根据节点邻居颜色及数量染不同颜色。例如某一轮中，若A、B两个节点邻居同样有三个红色节点和两个蓝色节点，A、B两个节点颜色就保持一致。如果B的邻居有三个红色和三个蓝色，就把B染成与A不同的颜色。经过多轮后图中每个节点都染色且下一轮没法再改变颜色（收敛了）。若两图经过上述染色后每种颜色的节点数量相同，则两图同构。

文章总结了两种常见的图神经网络结构（GCN、GraphSAGE）并论证他们的表达能力与同构测试相同

图神经网络每层中先将当前节点邻居特征聚合为 $a_v^{(k)}$ ，再将邻居特征与上一层当前节点特征组合。

$a_v^{(k)} = AGGREGATE^{(k)}(h_u^{(k-1)} : u \in N(v))$ 邻居节点聚合

$h_v^{(k)} = COMBINE^{(k)}(h_v^{(k-1)}, a_v^{(k)})$ 当前节点输出

在GraphSAGE中第一条公式具体为：将所有输出经过一层全连接后，使用最大值池化。通俗来讲就是对于邻居节点的每一个特征，选取经过全连接层后最大值。

$a_v^{(k)} = MAX(ReLU(Wh_u^{(k-1)}) : u \in N(v))$

第二条公式具体为：将邻居特征本层输出与当前节点特征上一层输出拼接，一层全连接层后输出。

$h_v^{(k)} = W[h_v^{(k-1)} : a_v^{(k)}]$

在GCN中两条公式合并为：

$h_v^{(k)} = ReLU(W \cdot Mean(h_u^{(k-1)} : \forall u \in N(v) \cup v))$

文章提出了一个新模型GIN，更新公式如下。说白了就是把所有上一层输入加起来，经过多层感知机后输出。简单的模型可以对抗图的过平滑问题。

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

Graph WaveNet for Deep Spatial-Temporal Graph Modeling

https://arxiv.org/abs/1906.00121

研究背景：

- 只考虑到了节点与节点间的空间结构图。忽略了图中两个节点的相关可以不是空间的。在推荐系统中，A与B可能互相关注，但其喜好完全不同。在路况预测中，路段A和B路段都是高速出口，虽然空间图中没有边，但其拥堵的规律会相似。
- 过去的时序图神经网络用RNN和1-D CNN来捕捉时序特征。但RNN迭代缓慢且会有梯度消失的问题。而1-D CNN 虽然运算快，但需要叠加很多层才能捕捉到长时间的依赖特征（例如这周三对上个周三的依赖）。

本文提出了新模型Graph WaveNet：一种基于GCN与空洞卷积的时序图神经网络模型。对提出的第一个问题，模型训练一种自适应（self adaptive）的邻接矩阵用于刻画非空间上的依赖。对提出的第二个问题，文章采取了空洞卷积（dilated convolution），既保留了CNN运算时间快的优势，又不需要多层迭代就能捕捉长时间的依赖关系。

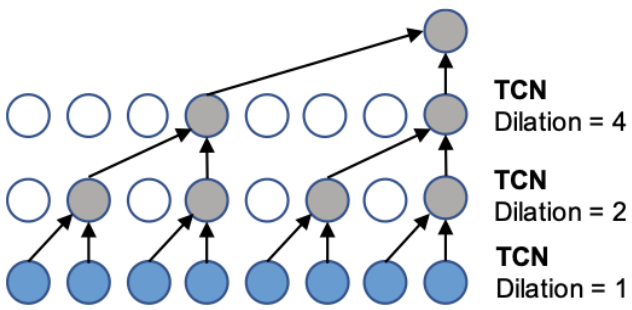
模型

GCN部分：

$$Z = \sum_{k=0}^{K} P_f^k X W_{k1} + P_b^k X W_{k2} + \tilde{A}_{apt}^k X W_{k3}$$

Dilated CNN部分：d为空洞卷积参数、控制每隔多少个输入进入卷积

$$x * f(t) = \sum_{s=0}^{K-1} f(s) x(t - d * s)$$



(STFGNN) Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting AAAI2021

https://arxiv.org/abs/2012.09641

研究背景：

过去基于图神经网络的时空序列预测模型缺少空间相关性的考虑：忽略了两个节点即使是地理上不相邻，在空间上也有强相关性（比如在早晚高峰时间，科技园附近拥堵严重）。其中STSGCN与Graph WaveNet两个模型提出了改变本来的图邻接矩阵来刻画空间相关度，但这些自适应的邻接矩阵无法描述复杂的时序特征（不仅仅周期变化相似）；而STNNs https://arxiv.org/abs/2001.02908 与STGNN https://dl.acm.org/doi/abs/10.1145/3366423.3380186提出了注意力机制应用到捕捉时空序列特征的模型。注意力机制虽然能捕捉到预训练出的时空序列矩阵捕捉不到的序列信息，但其容易受到噪音影响有过拟合的问题。

本文针对以上问题提出了基于Dynamic Time Warping算法构建时间序列相似度矩阵，用于捕捉地理上不相关的节点相似信息。并将模型输入并行输入到两个模块STFGN模块与Gated CNN模块，用于捕捉地理上的特征。

DTW算法：计算时间序列X与时间序列Y的相似度。

首先构建时间序列x与时间序列y之间差值矩阵M，其中

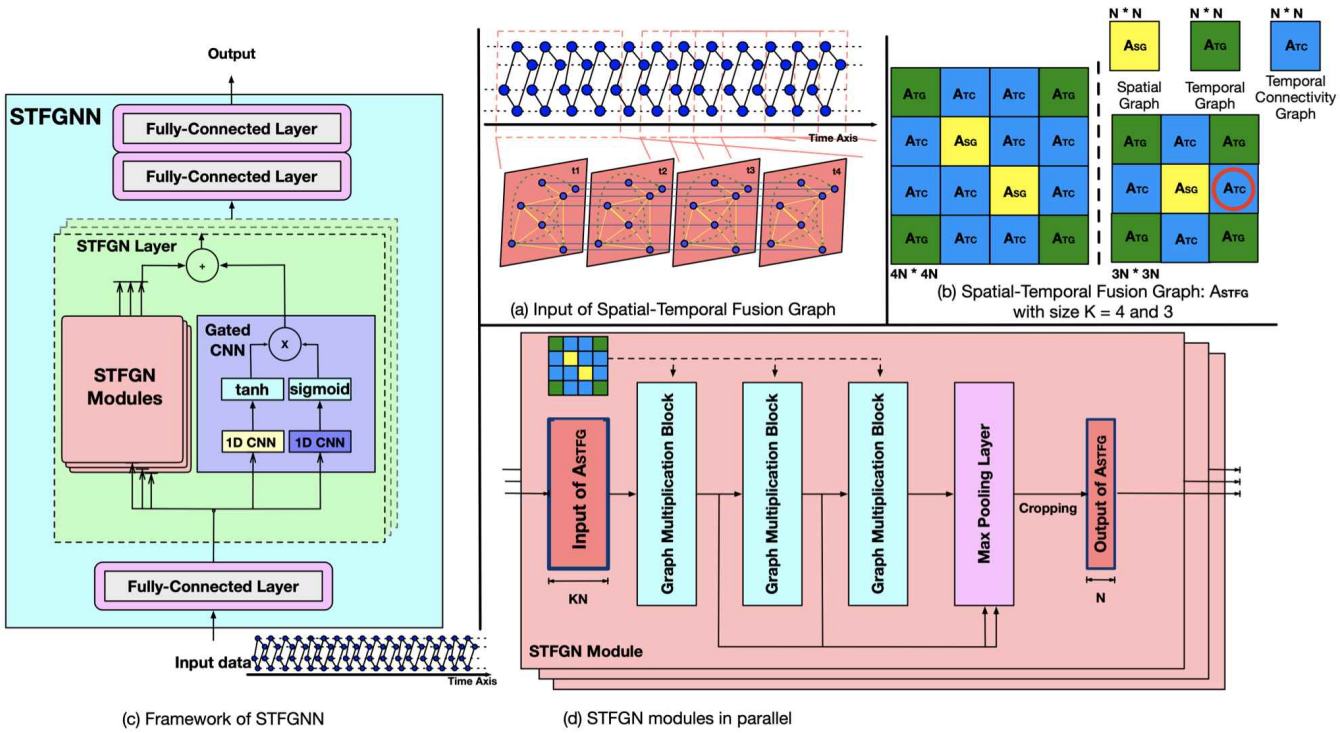
$$M_{i,j} = |x_i - y_j|$$

然后基于动态规划算法构建相似度矩阵F，其中

$$F_{i,j} = M_{i,j} + min(F_{i-1,j}, F_{i,j-1}, F_{i-1,j-1})$$

原文中约束了一个时间点最多能匹配多长的时间区间，增强了时序特征的相似度同时也加快了训练速度。

模型



左边是整个模型的示意图。右上角为构造拼接而成的矩阵，其中黄色的为空间邻接矩阵，绿色的为由DTW构造的时间序列相似度矩阵。

实验结果

Datasets	Metric	FC-LSTM	DCRNN	STGCN	ASTGCN(r)	Graph WaveNet	STSGCN	STFGNN
PEMS03	MAE	21.33 ± 0.24	18.18 ± 0.15	17.49 ± 0.46	17.69 ± 1.43	*19.85 ± 0.03*	17.48 ± 0.15	**16.77 ± 0.09**
	MAPE(%)	23.33 ± 4.23	18.91 ± 0.82	17.15 ± 0.45	19.40 ± 2.24	*19.31 ± 0.49*	16.78 ± 0.20	**16.30± 0.09**
	RMSE	35.11 ± 0.50	30.31 ± 0.25	30.12 ± 0.70	29.66 ± 1.68	*32.94 ± 0.18*	29.21 ± 0.56	**28.34± 0.46**
PEMS04	MAE	27.14 ± 0.20	24.70 ± 0.22	22.70 ± 0.64	22.93 ± 1.29	*25.45 ± 0.03*	21.19 ± 0.10	**19.83± 0.06**
	MAPE(%)	18.20 ± 0.40	17.12 ± 0.37	14.59 ± 0.21	16.56 ± 1.36	*17.29 ± 0.24*	13.90 ± 0.05	**13.02± 0.05**
	RMSE	41.59 ± 0.21	38.12 ± 0.26	35.55 ± 0.75	35.22 ± 1.90	*39.70 ± 0.04*	33.65 ± 0.20	**31.88± 0.14**
PEMS07	MAE	29.98 ± 0.42	25.30 ± 0.52	25.38 ± 0.49	28.05 ± 2.34	*26.85 ± 0.05*	24.26 ± 0.14	**22.07± 0.11**
	MAPE(%)	13.20 ± 0.53	11.66 ± 0.33	11.08 ± 0.18	13.92 ± 1.65	*12.12 ± 0.41*	10.21 ± 1.65	**9.21± 0.07**
	RMSE	45.94 ± 0.57	38.58 ± 0.70	38.78 ± 0.58	42.57 ± 3.31	*42.78 ± 0.07*	39.03 ± 0.27	**35.80± 0.18**
PEMS08	MAE	22.20 ± 0.18	17.86 ± 0.03	18.02 ± 0.14	18.61 ± 0.40	*19.13 ± 0.08*	17.13 ± 0.09	**16.64± 0.09**
	MAPE(%)	14.20 ± 0.59	11.45 ± 0.03	11.40 ± 0.10	13.08 ± 1.00	*12.68 ± 0.57*	10.96 ± 0.07	**10.60± 0.06**
	RMSE	34.06 ± 0.32	27.83 ± 0.05	27.83 ± 0.20	28.16 ± 0.48	*31.05 ± 0.07*	26.80 ± 0.18	**26.22± 0.15**

Dataset	Model Elements	MAE	MAPE%	RMSE
PEMS04	STSGCN	21.19	13.90	33.65
	$[ST_3, T_{sp5}]$	20.74	13.77	33.44
	$[ST_3, T_{sp1}]$	20.09	13.24	32.44
	$[ST_4, T_{sp1}]$	19.92	13.03	31.93
	$[T_4, T_{sp1}, \Theta]$	20.02	13.17	31.98
	$[T_4, T_{sp5}, \Theta]$	19.91	13.11	32.19
	$[ST_4, T_{sp1}, \Theta]$	**19.83**	**13.02**	**31.88**
PEMS08	STSGCN	17.13	10.96	26.80
	$[ST_3, T_{sp5}]$	19.47	12.27	29.59
	$[ST_3, T_{sp1}]$	16.84	10.80	26.58
	$[ST_4, T_{sp1}]$	16.70	10.63	26.24
	$[T_4, T_{sp1}, \Theta]$	18.23	11.52	29.05
	$[T_4, T_{sp5}, \Theta]$	**16.02**	**10.07**	**25.39**
	$[ST_4, T_{sp1}, \Theta]$	16.64	10.60	26.22

(DGI) Deep Graph Infomax ICLR 2019

https://arxiv.org/abs/1809.10341

研究背景：

基于随机游走（Random Walk）的图嵌入方式问题是他们已经预设了相邻节点间包含的信息类似，而不相邻节点包含的信息不同。

基于无监督学习（Unsupervised Learning）的一些方法、作者提出了一种图上的无监督学习预训练模型DGI。无监督学习一般基于样本间对比的方法：构造正样本与负样本并给正样本打上更高的分数。并且无监督学习一般预先整合了人的经验知识在训练过程中：例如我们倾向于认为相邻节点的embedding应该相近。本文模型同样整合了人的经验知识：但是同时考虑到了节点间和整个图模型两方面。

模型：

本文意图训练一个F编码器，将N*F特征矩阵（N为节点数量，F为特征数量）与N*N的邻接矩阵作为输入，映射到N*F'的矩阵。其中每个节点有F'维度的embedding。

$$E(X,A)=H=\{h_1,h_2,...,h_N\}$$

其中h为节点的embedding向量。

训练方式：

1. 通过腐化方程生成根据输入的（X，A）生成负样本（X', A'）
2. 使用编码器生成输入X，A的图嵌入
3. 使用同样的编码器生成负样本X', A'的图嵌入
4. 生成全局图嵌入向量s = R（H）
5. 计算损失，使用梯度下降训练。

损失函数：目的是让D（正样本）尽量大而D（负样本）尽量小。

$$\mathcal{L}=\frac{1}{N+M}\left(\sum_{i=1}^{N}\mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log\mathcal{D}\left(\vec{h}_i,\vec{s}\right)\right]+\sum_{j=1}^{M}\mathbb{E}_{(\widetilde{\mathbf{X}},\widetilde{\mathbf{A}})}\left[\log\left(1-\mathcal{D}\left(\vec{\widetilde{h}}_j,\vec{s}\right)\right)\right]\right)$$

对于样本较少的直推式学习（Transductive Learning），编码器为一个层数为1的GCN模型：

$$\mathcal{E}(\mathbf{X},\mathbf{A})=\sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}\right)$$

其中A为邻接矩阵，D为节点度数矩阵（对角矩阵）。激活函数为PReLU。

腐化函数为矩阵X的随机线性变换，几何意义为同样的节点在同一张图中，只将连通关系打乱。

对于图较大的归纳式学习（Inductive Learning），编码器为一个GraphSAGE与GCN的混合变种：

$$\mathrm{MP}(\mathbf{X},\mathbf{A})=\hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{X}\mathbf{\Theta}$$

其中MP为Mean Pooling。对于Reddit数据集，本文采用三层Mean Pooling并整合了skip-connection的编码器。具体为

$$\widetilde{\mathrm{MP}}(\mathbf{X},\mathbf{A})=\sigma\left(\mathbf{X}\mathbf{\Theta}'\|\mathrm{MP}(\mathbf{X},\mathbf{A})\right)\qquad\mathcal{E}(\mathbf{X},\mathbf{A})=\widetilde{\mathrm{MP}}_3(\widetilde{\mathrm{MP}}_2(\widetilde{\mathrm{MP}}_1(\mathbf{X},\mathbf{A}),\mathbf{A}),\mathbf{A})$$

第一个部分为skip-connection，将输入直接与编码器结果拼接后激活。

对于所有的实验，Readout函数R为将所有节点的特征取均值后输入sigmoid。为hi与s打分函数D为如下的bilinear scoring function。

$$R = \sigma(\frac{1}{N}\sum_{i=1}^{N} h_i) \quad D = \sigma(h_i^T W s)$$

实验结果：

Transductive

Available data	Method	Cora	Citeseer	Pubmed
X	Raw features	47.9 ± 0.4%	49.3 ± 0.2%	69.1 ± 0.3%
A, **Y**	LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
A	DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
X, **A**	DeepWalk + features	70.7 ± 0.6%	51.4 ± 0.5%	74.3 ± 0.9%
X, **A**	Random-Init (ours)	69.3 ± 1.4%	61.9 ± 1.6%	69.6 ± 1.9%
X, **A**	**DGI** (ours)	**82.3** ± 0.6%	**71.8** ± 0.7%	**76.8** ± 0.6%
X, **A**, **Y**	GCN (Kipf & Welling, 2016a)	81.5%	70.3%	79.0%
X, **A**, **Y**	Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%

Inductive

Available data	Method	Reddit	PPI
X	Raw features	0.585	0.422
A	DeepWalk (Perozzi et al., 2014)	0.324	—
X, **A**	DeepWalk + features	0.691	—
X, **A**	GraphSAGE-GCN (Hamilton et al., 2017a)	0.908	0.465
X, **A**	GraphSAGE-mean (Hamilton et al., 2017a)	0.897	0.486
X, **A**	GraphSAGE-LSTM (Hamilton et al., 2017a)	0.907	0.482
X, **A**	GraphSAGE-pool (Hamilton et al., 2017a)	0.892	0.502
X, **A**	Random-Init (ours)	0.933 ± 0.001	0.626 ± 0.002
X, **A**	**DGI** (ours)	**0.940** ± 0.001	**0.638** ± 0.002
X, **A**, **Y**	FastGCN (Chen et al., 2018)	0.937	—
X, **A**, **Y**	Avg. pooling (Zhang et al., 2018)	0.958 ± 0.001	0.969 ± 0.002

(TrGNN) Traffic Flow Prediction with Vehicle Trajectories AAAI2021

https://ojs.aaai.org/index.php/AAAI/article/view/16104

研究背景：

作者认为交通数据有两种：轨迹数据和流量数据。然而业界的流量预测模型很少将轨迹数据整合进去。轨迹数据又与流量数据有强相关性。作者认为将轨迹数据应用到流量预测中主要有以下难点：

- 轨迹数据都是历史数据
- 轨迹数据需要一个很『properly』的方式聚合才能够反应流量数据
- 未来的流量受到很多因素影响：天气、交通事故、重大活动.....

模型：

作者应用贝叶斯模型，整合所有的轨迹数据计算出从某一节点转移到下个节点的概率，并将概率应用到图神经网络中。

在空间特征处理上，作者拼接每一层的GCN输出，作为某一时间的信号输出。并应用了注意力机制交通状况输出与交通需求输出之间。注意力的结果为某一节点在某一时刻的embedding，将节点在所有时刻的embedding同时输入一层全连接后得到下一个时刻的预测流量值。作者提到这里如果要预测多个时刻的流量值可以使用RNN编码解码的方式（GRU、LSTM）。

实验结果：

Method	Overall			Peak hours			Non-peak hours			MRT breakdown		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
HA	33.74	0.34	52.58	36.83	0.25	55.02	32.53	0.28	48.67	**40.07**	0.27	**59.34**
MA	31.55	0.35	47.69	36.14	0.26	53.18	28.18	0.27	39.41	44.85	0.30	71.43
VAR	29.27	0.33	43.22	34.23	0.24	49.71	28.10	0.26	39.28	40.68	**0.27**	64.41
RF	29.26	0.33	43.38	34.13	**0.24**	49.75	**27.53**	**0.26**	**38.53**	42.28	0.28	66.53
T-GCN	31.12	0.35	45.69	36.57	0.27	52.91	30.03	0.29	41.53	42.38	0.30	67.39
STGCN	29.88	0.33	44.51	34.86	0.24	50.86	27.94	0.27	39.05	42.19	0.28	66.40
DCRNN	**29.01**	**0.31**	**43.12**	**33.74**	0.25	**48.88**	27.75	0.27	38.74	40.39	0.28	64.28
TrGNN-	27.34	0.31	40.05	31.35	0.23	45.11	26.61	0.26	37.20	38.57	0.27	59.53
TrGNN	**26.43**	**0.30**	**38.65**	**29.81**	**0.23**	**42.62**	**25.65**	**0.25**	**35.68**	**34.56**	**0.25**	**54.31**
%diff	-9%	-5%	-10%	-12%	-6%	-13%	-7%	-4%	-7%	-14%	-8%	-8%

(Metapath2vec) Scalable Representation Learning For Heterogeneous Networks

https://ericdongyx.github.io/papers/KDD17-dong-chawla-swami-metapath2vec.pdf

研究背景

过去研究图和图中节点的表示方式都是基于同构图（Homogeneous Graph）的，而真正的信息网络是异构了，拥有不同类型的节点与边。比如一个学术研究的网络，节点可以是人、论文、会议...而边的构成又有人和人之间的Co-author，人和论文之间所属关系，论文和论文之间的引用关系...作者提出了将图中基于随机游走的嵌入应用到异构图中的方式。

模型

首先同构图的skip-gram模型旨在最大化对于某个节点、其邻居节点的概率：

$argmax_\theta \ \ p(c|v;\theta)$ 其中c为v节点基于随机游走生成的邻居。

而异构图的skip-gram模型训练目标与同构图类似，同样是『给中心词，最大化上下文出现概率』这里的上下文指的是在异构图中随机游走得来的邻域。

$$p(c_t|v;\theta) = \frac{e^{X_{c_t}X_v}}{\sum e^{X_u X_v}}$$

如果继续在异构图中使用同构图中随机游走的方式，模型将会『biased to highly visible type of nodes』。如果学术图中作者数很多且论文数量少，会议数量更少，那么同构图随机游走的方式只能学到作者和作者之间的关系。为了解决这样的问题，并整合人工知识，作者先预设了路径走法：比如

作者 - 论文 - 会议 - 论文 - 作者 这样的路径就能用于学习不同作者在同一会议发表的不同论文，进而学习到这些作者和论文在同一领域。具体随机游走的转移函数如下：

$$p(v^{i+1}|v_t^i,\mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1},v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1},v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1},v_t^i) \notin E \end{cases}$$

metapath2vec算法中，所有的种类都是拥有同样权重的：具体提现为对于某个在节点随机游走而来的诶邻域的点，我们不考虑他和原节点是否为同样种类直接计算softmax。作者提出metapath2vec++，对于每一种类的节点计算一次softmax。比如对于作者Yoshua Bengio作为『中心词』，计算一次其他作者的softmax，其他论文的softmax还有会议的softmax。

实验结果

Table 2: Multi-class venue node classification results in AMiner data.

Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.0723	0.1396	0.1905	0.2795	0.3427	0.3911	0.4424	0.4774	0.4955	0.4457
	LINE (1st+2nd)	0.2245	0.4629	0.7011	0.8473	0.8953	0.9203	0.9308	0.9466	0.9410	0.9466
	PTE	0.1702	0.3388	0.6535	0.8304	0.8936	0.9210	0.9352	0.9505	0.9525	0.9489
	metapath2vec	0.3033	0.5247	0.8033	0.8971	0.9406	0.9532	0.9529	0.9701	0.9683	0.9670
	metapath2vec++	0.3090	0.5444	0.8049	0.8995	0.9468	0.9580	0.9561	0.9675	0.9533	0.9503
Micro-F1	DeepWalk/node2vec	0.1701	0.2142	0.2486	0.3266	0.3788	0.4090	0.4630	0.4975	0.5259	0.5286
	LINE (1st+2nd)	0.3000	0.5167	0.7159	0.8457	0.8950	0.9209	0.9333	0.9500	0.9556	0.9571
	PTE	0.2512	0.4267	0.6879	0.8372	0.8950	0.9239	0.9352	0.9550	0.9667	0.9571
	metapath2vec	0.4173	0.5975	0.8327	0.9011	0.9400	0.9522	0.9537	0.9725	0.9815	0.9857
	metapath2vec++	0.4331	0.6192	0.8336	0.9032	0.9463	0.9582	0.9574	0.9700	0.9741	0.9786

Table 3: Multi-class author node classification results in AMiner data.

Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
Macro-F1	DeepWalk/node2vec	0.7153	0.7222	0.7256	0.7270	0.7273	0.7274	0.7273	0.7271	0.7275	0.7275
	LINE (1st+2nd)	0.8849	0.8886	0.8911	0.8921	0.8926	0.8929	0.8934	0.8936	0.8938	0.8934
	PTE	0.8898	0.8940	0.897	0.8982	0.8987	0.8990	0.8997	0.8999	0.9002	0.9005
	metapath2vec	0.9216	0.9262	0.9292	0.9303	0.9309	0.9314	0.9315	0.9316	0.9319	0.9320
	metapath2vec++	0.9107	0.9156	0.9186	0.9199	0.9204	0.9207	0.9207	0.9208	0.9211	0.9212
Micro-F1	DeepWalk/node2vec	0.7312	0.7372	0.7402	0.7414	0.7418	0.7420	0.7419	0.7420	0.7425	0.7425
	LINE (1st+2nd)	0.8936	0.8969	0.8993	0.9002	0.9007	0.9010	0.9015	0.9016	0.9018	0.9017
	PTE	0.8986	0.9023	0.9051	0.9061	0.9066	0.9068	0.9075	0.9077	0.9079	0.9082
	metapath2vec	0.9279	0.9319	0.9346	0.9356	0.9361	0.9365	0.9365	0.9365	0.9367	0.9369
	metapath2vec++	0.9173	0.9217	0.9243	0.9254	0.9259	0.9261	0.9261	0.9262	0.9264	0.9266

Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting

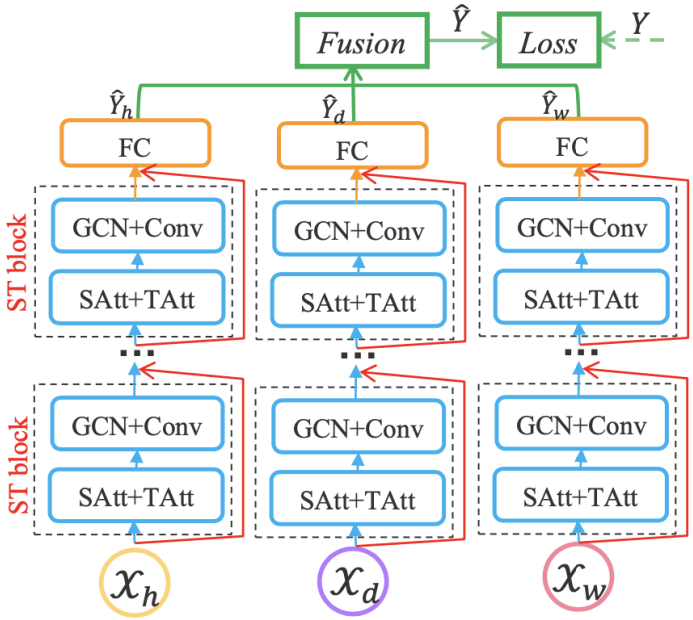
https://ojs.aaai.org//index.php/AAAI/article/view/3881 AAAI2019

https://github.com/guoshnBJTU/ASTGCN-r-pytorch Code

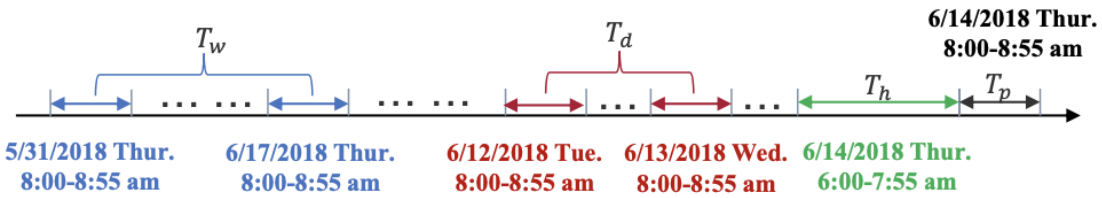
研究背景

过去使用卷积（CNN、GCN）的方式提取时空序列特征的尝试无法同时捕捉到时间和空间信息。使用GCN时无法将时间维度考虑进去。STGCN模型的作者提出了一个新的时序图神经网络。在时间上使用CNN、在空间上应用GCN并将注意力机制融入。

模型



如下图所示，首先将时间序列信息分为三种：小时（h）、天（d）和周（w）。三个输入矩阵维度为（N * C * T）。其中T为时间序列的时间点个数。



注意力机制：

对于空间注意力，作者构造矩阵S，其中第i行j列为第i个节点对第j个节点的注意力。注意力的计算方式如下：其中标黄的部分为输入的序列矩阵（N * C * T），V、W与b为可训练的参数。

$$\mathbf{S} = \mathbf{V}_s \cdot \sigma((\boldsymbol{\mathcal{X}}_h^{(r-1)}\mathbf{W}_1)\mathbf{W}_2(\mathbf{W}_3\boldsymbol{\mathcal{X}}_h^{(r-1)})^T + \mathbf{b}_s$$

$$\mathbf{S}'_{i,j} = \frac{\exp(\mathbf{S}_{i,j})}{\sum_{j=1}^{N}\exp(\mathbf{S}_{i,j})}$$

相当于对于每个时间点，把图直接输入进一层标准的带注意力的GCN。不同时间点的GCN参数不同。

时间注意力上，作者同样构造矩阵E，第i行j列为时间i对时间j的注意力。直接讲输入的（N* C * T）矩阵与矩阵E相乘得到最终输入的（N * C * T)矩阵。

最终一个ASTGCN module的更新公式如下：先经过GCN处理每一层的空间信息后再在时间上做CNN。

$$\boldsymbol{\mathcal{X}}_h^{(r)} = ReLU(\Phi * (ReLU(g_\theta *_G \hat{\boldsymbol{\mathcal{X}}}_h^{(r-1)}))) \in \mathbb{R}^{C_r \times N \times T_r}$$

实验结果

Model	PeMSD4		PeMSD8	
	RMSE	MAE	RMSE	MAE
HA	54.14	36.76	44.03	29.52
ARIMA	68.13	32.11	43.30	24.04
VAR	51.73	33.76	31.21	21.41
LSTM	45.82	29.45	36.96	23.18
GRU	45.11	28.65	35.95	22.20
STGCN	38.29	25.15	27.87	18.88
GLU-STGCN	38.41	27.28	30.78	20.99
GeoMAN	37.84	23.64	28.91	17.84
MSTGCN (ours)	**35.64**	**22.73**	**26.47**	**17.47**
ASTGCN (ours)	**32.82**	**21.80**	**25.27**	**16.63**

Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting

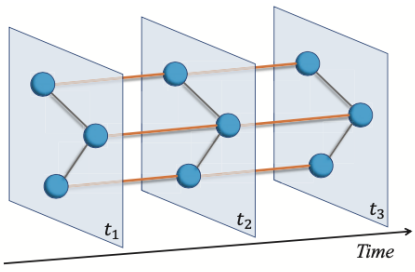
https://ojs.aaai.org//index.php/AAAI/article/view/5438

研究背景：

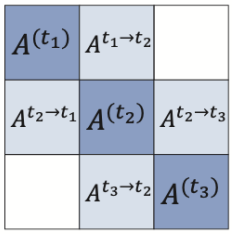
基于时序的图神经网络能够捕获到时间、空间两个维度的依赖关系。但过去的研究忽略了时空依赖关系(Spatial - Temporal Correlation)。也就是这个时刻节点A的状态会影响到下个时刻节点B、C等其他节点的状态。这种时间和空间中复杂的依赖关系很难通过图神经网络去学习。同时，不同时间点的依赖关系是不一样(Heterogeneity)的。通俗来解释就是早上八点钟堵点与堵点之间的依赖关系和晚上六点的依赖关系是不同的。作者提出了STSGCN模型，使用一个神经网络同时能够捕获时间、空间以及时空依赖关系。同时使用多模块叠加用于捕获不同时间点时空序列图的差异。

模型：

为了捕获当前时间与前一时刻、后一时刻的依赖关系，作者将邻接矩阵从N*N扩大为3N*3N，将每个节点与邻居连接的同时，与上一时刻的自己与下一时刻的自己相连。



(a) Localized Spatial-Temporal Graph



(b) Adjacency matrix of Localized Spatial-Temporal Graph

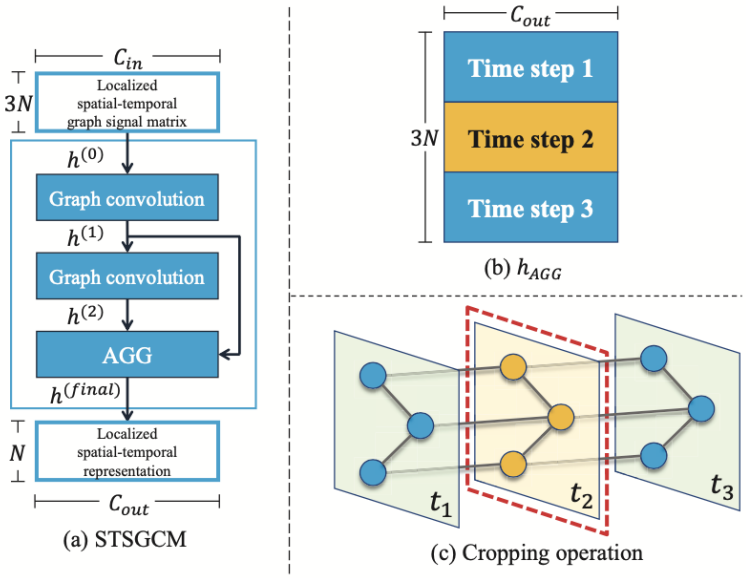
在图(b)中，对角线上的N*N子矩阵为邻接矩阵。而非对角线上的子矩阵为不同时刻的同一节点之间的边。比如 $A_{1,N+1}$ 这个节点值为1，位于最上面一排中间的子矩阵中。刻画的是第一个时刻节点1和第二个时刻节点1之间的这条边。

然而，这样构造出来的邻接矩阵将同一时刻的不同节点和不同时刻的同一节点一视同仁，没有很好的体现他们的区别。作者受启发于ConvS2S https://arxiv.org/abs/1705.03122，训练节点的位置embedding矩阵 $T_{emb} \in \mathbb{R}^{C \times T}$ 和空间embedding矩阵 $S_{emb} \in \mathbb{R}^{N \times C}$ ，将二者广播并与输入的时空序列矩阵 $X \in \mathbb{R}^{N \times C \times T}$ 相加得到最终的输入特征矩阵。

图卷积部分、GCN的公式如下

$$GCN(h^{(l-1)}) = h^{(l)} = \sigma(A' h^{(l-1)} W + b) \in \mathbb{R}^{3N \times C_{out}}$$

输入的矩阵的维度为(3N, C_in)，激活函数为ReLU或者GLU，经过图卷积层后经过最大值池化聚合，并剪裁只保留中心点那层的信息，最终输出维度(N，C_out)。



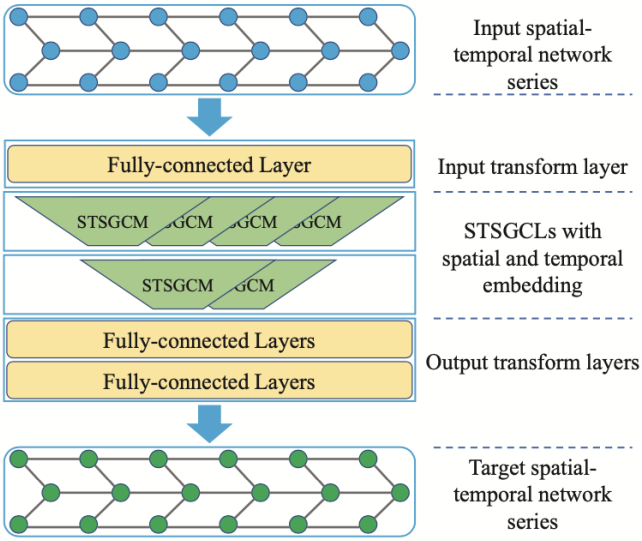
空间卷积部分，作者使用了一种类似于滑动窗口的思想，输入的T个时间节点可以作为中心点的去掉首位有T-2个，分别输入进T-2个不同的前面所说的模块，拼接后得到（T-2, N, C_out）维度的输出。

几个优化点：

Mask Matrix：在邻接矩阵输入到GCN里之前，先与Mask相乘用于过滤掉两个相邻节点之间没有依赖关系的。有点类似注意力机制。

损失函数为Huber Loss：对于异常值比平均差(squared error loss)更不敏感。

模型结构：



实验结果：

Table 2: Performance comparison of different approaches for traffic flow forecasting.

Baseline methods / Datasets	Metrics	VAR	SVR	LSTM	DCRNN	STGCN	ASTGCN(r)	STG2Seq	Graph WaveNet	**STSGCN**
PEMS03	MAE	23.65	21.97 ± 0.00	21.33 ± 0.24	18.18 ± 0.15	17.49 ± 0.46	17.69 ± 1.43	19.03 ± 0.51	19.85 ± 0.03	**17.48 ± 0.15**
	MAPE (%)	24.51	21.51 ± 0.46	23.33 ± 4.23	18.91 ± 0.82	17.15 ± 0.45	19.40 ± 2.24	21.55 ± 1.68	19.31 ± 0.49	**16.78 ± 0.20**
	RMSE	38.26	35.29 ± 0.02	35.11 ± 0.50	30.31 ± 0.25	30.12 ± 0.70	29.66 ± 1.68	29.73 ± 0.52	32.94 ± 0.18	**29.21 ± 0.56**
PEMS04	MAE	23.75	28.70 ± 0.01	27.14 ± 0.20	24.70 ± 0.22	22.70 ± 0.64	22.93 ± 1.29	25.20 ± 0.45	25.45 ± 0.03	**21.19 ± 0.10**
	MAPE (%)	18.09	19.20 ± 0.01	18.20 ± 0.40	17.12 ± 0.37	14.59 ± 0.21	16.56 ± 1.36	18.77 ± 0.85	17.29 ± 0.24	**13.90 ± 0.05**
	RMSE	36.66	44.56 ± 0.01	41.59 ± 0.21	38.12 ± 0.26	35.55 ± 0.75	35.22 ± 1.90	38.48 ± 0.50	39.70 ± 0.04	**33.65 ± 0.20**
PEMS07	MAE	75.63	32.49 ± 0.00	29.98 ± 0.42	25.30 ± 0.52	25.38 ± 0.49	28.05 ± 2.34	32.77 ± 3.21	26.85 ± 0.05	**24.26 ± 0.14**
	MAPE (%)	32.22	14.26 ± 0.03	13.20 ± 0.53	11.66 ± 0.33	11.08 ± 0.18	13.92 ± 1.65	20.16 ± 4.36	12.12 ± 0.41	**10.21 ± 0.05**
	RMSE	115.24	50.22 ± 0.01	45.84 ± 0.57	**38.58 ± 0.70**	38.78 ± 0.58	42.57 ± 3.31	47.16 ± 3.66	42.78 ± 0.07	39.03 ± 0.27
PEMS08	MAE	23.46	23.25 ± 0.01	22.20 ± 0.18	17.86 ± 0.03	18.02 ± 0.14	18.61 ± 0.40	20.17 ± 0.49	19.13 ± 0.08	**17.13 ± 0.09**
	MAPE (%)	15.42	14.64 ± 0.11	14.20 ± 0.59	11.45 ± 0.03	11.40 ± 0.10	13.08 ± 1.00	17.32 ± 1.14	12.68 ± 0.57	**10.96 ± 0.07**
	RMSE	36.33	36.16 ± 0.02	34.06 ± 0.32	27.83 ± 0.05	27.83 ± 0.20	28.16 ± 0.48	30.71 ± 0.61	31.05 ± 0.07	**26.80 ± 0.18**

Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks

https://arxiv.org/pdf/1905.07953.pdf

研究背景：

作者研究了各个GCN模型训练的效率。影响模型效率的因素有三

- 内存需求（参数数量，反向传播时休要记住多少东西）
- 每个epoch的训练时间
- 收敛速度

在GCN首次提出的论文Semi-Supervised Classification with Graph Convolutional Networks中，训练方法是Full-batch gradient descent。计算模型梯度的时候，需要存储每一层中间的node embedding，空间复杂度为O(N * F * L)。其中N为节点数量，F为特征数量，L为GCN层数。尽管每个epoch的训练速度很快，但是模型收敛很慢因为每个参数在每个epoch只被更新一次。

内存需求：不好 每个epoch时间：好 收敛速度：不好

GraphSAGE使用的训练方式是Mini-batch gradient descent。然而BGD又会衍生新的计算成本（Overhead）：在计算某节点在L层的梯度，需要知道其所有邻居在L-1层的embedding，从而需要这些邻居的邻居在L-2层的embedding。这就让计算成本指数级增加。GraphSAGE为了对抗这种高额计算成本，使用固定大小的邻居数量在反向传播中。FastGCN提出了重要程度采样。但随着GCN层数增加，这些计算成本依然很高。

内存需求：好 每个epoch时间：不好 收敛速度：好

VR-GCN采取了一种方差缩小（Variance Reduction）的方式来收缩邻居大小。尽管成功减少了需要计算的邻居，模型需要存贮所有的中间层输出。这样空间复杂度成为了O(NFL)。在节点数N很大时无法使用内存低的GPU计算。

内存需求：不好 每个epoch时间：好 收敛速度：好

本文模型核心在将大的、稀疏的图分解成很多个小的、稠密的图。是每个子图内部的边数尽量多而连接不同子图之间的边尽量少。

模型：Cluster - GCN

Embedding Utilization - Motivation

计算节点i的梯度时，需要其邻居在上一层的embedding，而这些embedding又需要在上两层这些邻居的邻居的embedding。这样一在一个平均度数为d的图上运行L+1层的GCN需要聚合O（d^L)这么多节点的信息。计算每个embedding需要矩阵乘法，复杂度为O（F^2)。这样计算**某一个**节点的总时间复杂度O(F^2 * d^L)。

如若我们要计算所有的梯度，复杂度会比O(N * F^2 * d^L)低。其原因是某个节点的embedding在下一层被用到的次数很多。就引出了我们的算法Cluster - GCN，希望能够每个batch内增大低层某个节点的embedding在高层的使用次数。

Vanilla Cluster-GCN 朴素聚合GCN

$$\bar{G} = [G_1, \cdots, G_c] = [\{\mathcal{V}_1, \mathcal{E}_1\}, \cdots, \{\mathcal{V}_c, \mathcal{E}_c\}],$$

where each $\mathcal{E}_t$ only consists of the links between nodes in $\mathcal{V}_t$. After reorganizing nodes, the adjacency matrix is partitioned into c^2 submatrices as

$$A = \bar{A} + \Delta = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix} \tag{4}$$

and

$$\bar{A} = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix}, \tag{5}$$

将图划分为c个子图。把邻接矩阵划分为c^2个子矩阵。对角线上的矩阵为每个子图的邻接矩阵。非对角线上的矩阵为每个子图与子图之间连接的邻接矩阵。对角线上的值比较稠密而非对角线上的比较稀疏。这样GCN的公式和损失函数就为

$$Z^{(L)} = \bar{A}'\sigma(\bar{A}'\sigma(\cdots\sigma(\bar{A}'XW^{(0)})W^{(1)})\cdots)W^{(L-1)} \quad (6)$$
$$= \begin{bmatrix} \bar{A}'_{11}\sigma(\bar{A}'_{11}\sigma(\cdots\sigma(\bar{A}'_{11}X_1W^{(0)})W^{(1)})\cdots)W^{(L-1)} \\ \vdots \\ \bar{A}'_{cc}\sigma(\bar{A}'_{cc}\sigma(\cdots\sigma(\bar{A}'_{cc}X_cW^{(0)})W^{(1)})\cdots)W^{(L-1)} \end{bmatrix}$$

due to the block-diagonal form of $\bar{A}$ (note that $\bar{A}'_{tt}$ is the corresponding diagonal block of $\bar{A}'$). The loss function can also be decomposed into

$$\mathcal{L}_{\bar{A}'} = \sum_t \frac{|\mathcal{V}_t|}{N}\mathcal{L}_{\bar{A}'_{tt}} \quad \text{and} \quad \mathcal{L}_{\bar{A}'_{tt}} = \frac{1}{|\mathcal{V}_t|}\sum_{i\in\mathcal{V}_t}\text{loss}(y_i, z_i^{(L)}). \quad (7)$$

每个epoch的时间复杂度为：

O(AF+ NF^2) 其中A为A矩阵中非0数。

总空间复杂度为

O(bLF）batch * level * feature_channels

大图、中图、小图：

由于在图被划分后，非对角线上的数字（部分与部分之间的边）被忽略了。同时划分方式会使子图的节点分布和整个图的节点分布不同。这样整个梯度就会有偏差。所以我们先将图划分成很小很细的微型子图。然后每个batch更新的时候选取q个子图，将他们合成一个中型子图。

一些优化：

直觉来看，相邻比较近的节点肯定对中心节点影响大于离的比较远的节点（当然这个不一定），作者提出了将邻接矩阵加上一个单位矩阵。（这个优化在GCN中也有）。

然而作者认为每个节点的权值应该跟他们的邻居数量有关。如果邻居数量很多那自己的权值应该更高，这样我们对加了单位矩阵的A进行归一化（当然这个优化在GCN中也有）。然后再加上带权的度数矩阵。

$$X^{(l+1)} = \sigma((\tilde{A} + \lambda\text{diag}(\tilde{A}))X^{(l)}W^{(l)}).$$

实验结果：

空间复杂度：

Table 5: Comparisons of memory usages on different datasets. Numbers in the brackets indicate the size of hidden units used in the model.

	2-layer			3-layer			4-layer		
	VRGCN	Cluster-GCN	GraphSAGE	VRGCN	Cluster-GCN	GraphSAGE	VRGCN	Cluster-GCN	GraphSAGE
PPI (512)	258 MB	39 MB	51 MB	373 MB	46 MB	71 MB	522 MB	55 MB	85 MB
Reddit (128)	259 MB	284 MB	1074 MB	372 MB	285 MB	1075 MB	515 MB	285 MB	1076 MB
Reddit (512)	1031 MB	292 MB	1099 MB	1491 MB	300 MB	1115 MB	2064 MB	308 MB	1131 MB
Amazon (128)	1188 MB	703 MB	N/A	1351 MB	704 MB	N/A	1515 MB	705 MB	N/A

时间复杂度、空间复杂度和准确率：

Table 8: Comparisons of running time, memory and testing accuracy (F1 score) for Amazon2M.

	Time		Memory		Test F1 score	
	VRGCN	Cluster-GCN	VRGCN	Cluster-GCN	VRGCN	Cluster-GCN
Amazon2M (2-layer)	337s	1223s	7476 MB	2228 MB	89.03	89.00
Amazon2M (3-layer)	1961s	1523s	11218 MB	2235 MB	90.21	90.21
Amazon2M (4-layer)	N/A	2289s	OOM	2241 MB	N/A	90.41

Papers Read

Bag of Tricks for Node Classification with Graph Neural Networks

https://arxiv.org/pdf/2103.13355.pdf

COMBINING LABEL PROPAGATION AND SIMPLE MODELS OUT-PERFORMS GRAPH NEURAL NETWORKS

https://arxiv.org/pdf/2010.13993.pdf

Why Propagate Alone? Parallel Use of Labels and Features on Graphs

https://arxiv.org/abs/2110.07190

STGCN Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting https://arxiv.org/abs/1709.04875

Semi-Supervised Classification with Graph Convolutional Networks https://arxiv.org/abs/1609.02907（GCN）

Inductive Representation Learning on Large Graphs https://arxiv.org/abs/1706.02216（GraphSAGE）

Graph Attention Networks https://arxiv.org/abs/1710.10903 （GAT）

DCRNN Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting https://arxiv.org/abs/1707.01926

STRATEGIES FOR PRE-TRAINING GRAPH NEURAL NETWORKS https://arxiv.org/pdf/1905.12265

Papers to Read

Graph Bert

https://arxiv.org/pdf/2001.05140.pdf

（Baidu UniMP）Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification

https://arxiv.org/pdf/2009.03509.pdf

APPNP

https://arxiv.org/abs/1810.05997

Meta-Gradient Reinforcement Learning

https://arxiv.org/abs/1805.09801

Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting（stemgnn）

https://proceedings.neurips.cc/paper/2020/file/cdf6581cb7aca4b7e19ef136c6e601a5-Paper.pdf

- [x] ~~ETA Prediction with Graph Neural Networks in Google Maps~~

https://arxiv.org/abs/2108.11482

https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks

- [ ] GPT-GNN: Generative Pre-Training of Graph Neural Networks https://arxiv.org/abs/2006.15437
- [ ] GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training https://arxiv.org/abs/2006.09963
- [ ] GATNE: General Attributed Multiplex HeTerogeneous Network Embedding https://arxiv.org/pdf/1905.01669.pdf

(ConvS2S) Convolutional Sequence to Sequence Learning https://arxiv.org/abs/1705.03122

- [x] ~~PairNorm: Tackling Oversmoothing in GNNs https://openreview.net/forum?id=rkecl1rtwB~~
- [ ] https://arxiv.org/abs/1706.06978 DIN
- [ ] https://static.googleusercontent.com/media/research.google.com/zh-CN//pubs/archive/45530.pdf Youtube推荐
- [ ] Learning to Embed Categorical Features without Embedding Tables for Recommendation https://arxiv.org/pdf/2010.10784.pdf KDD2021
- [ ] Xavier Initialization https://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf
- [ ] Generative Adversial Networks https://arxiv.org/abs/1406.2661（GAN）