graph represention

Contents

参考 AAAI2019 的 tutorial：AAAI2019《图表示学习》Tutorial, 180 页 PPT 带你从入门到精通（下载）

ppt 下载：https://pan.baidu.com/s/1hRjm1nbMcj4_ynZ0niE2JA

传统的机器学习方法依赖于用户定义的启发式模型来提取关于图的结构信息的特征编码 (例如，degree statistics 或核函数)。然而，近年来，使用基于深度学习和非线性降维的技术，自动学习将**图结构**编码为**低维 embedding** 的方法激增。

1 Introduction

graph 的几大传统 ml 任务：

- **Node classification**：预测给定的结点的 type
- **Link prediction**：预测两个结点是否有边相连
- **Community detection**：发现联系紧密的 nodes 的 clusters
- **Network similarity**：两个（子）网是否相似

目前的深度学习：

- cnn：固定大小的图片/网格
- rnn/w2v：文本/序列

图更加复杂：

- 复杂的拓扑结构（例如，不像网格那样有 spatial locality(空间局部性，在最近的将来将用到的信息很可能与现在正在使用的信息在**空间地址上是临近的**。)）
- 没有固定的结点顺序或者参考点（reference point）(例如，isomorphism（同构）问题)
- 经常是动态的并且有 multimodal（多模态）的 features

2 part1-Node Representation Learning

2.1 Node Representation Methods

问题定义：给定 $G = (V, E, W)$，其中，V 是结点集合，E 是边的集合，W 是**边**的权重集合。所谓的 node embedding 就是对结点 i 学习一个向量 $u_i \in R^d$。

相关工作：

- 传统 graph embedding 算法：MDS, IsoMap, LLE, Laplacian Eigenmap, …。缺点：hard to scale up
- Graph factorization(Ahmed et al. 2013)：只适用于无向图，并非专门为网络表示而设计
- Neural word embeddings(Bengio et al. 2003)：Neural language model；word2vec (skipgram), paragraph vectors, etc.

2.1.1 LINE

WWW2015 上的LINE: Large-scale Information Network Embedding

LINE 代码（c++）：https://github.com/tangjianpku/LINE

特点：

- 任意类型的网络（有向图、无向图、有/无权重）
- 明确的目标函数（一阶和二阶相似性（first/second proximity））
- 可扩展性
  - 异步 sgd
  - 百万级的结点和十亿级别的边：单机数小时

2.1.1.1 一阶相似度

First-order Proximity（一阶相似度）：两个**顶点之间**的**自身**相似（不考虑其他顶点）。因为有些结点的 link 并没有被观测到，所以一阶相似度不足以保存网络结构。

分布：(定义在**无向**边 $i - j$ 上)

一阶相似度的经验分布：

$$\hat{p_1}(v_i, v_j) = \frac{w_{ij}}{\sum_{(m,n)\in E} w_{mn}}$$

一阶相似度的模型分布：

$$p_1(v_i, v_j) = \frac{\exp(\vec{u_i}^T \vec{u_j})}{\sum_{(m,n)\in V\times V} \exp(\vec{u_m}^T \vec{u_n})}$$

其中，$\vec{u_i}$ 是节点 i 的 embedding，其实就是 sigmoid：

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u_i}^T \vec{u_j})}$$

目标函数是 **KL 散度**：

$$O_1 = KL(\hat{p_1}, p_1)$$

干掉常量 $\sum_{(m,n)\in E} w_{mn}$，还有 $\sum_{(i,j)\in E} w_{ij} \log w_{ij}$ 之后：

$$O_1 = \sum_{(i,j)\in E} w_{ij} \log w_{ij} - \sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j) \approx - \sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

只考虑一阶相似度的情况下，改变同一条边的方向对于最终结果没有什么影响。因此一阶相似度只能用于**无向图**，不能用于有向图。

2.1.1.2 二阶相似度

Second-order Proximity（二阶相似度）：网络中一对顶点 (u, v) 之间的二阶相似度是它们**邻近网络结构**之间的相似性。

分布：(定义在**有向**边 $i \to j$ 上)

邻近网络的经验分布：

$$\hat{p_2}(v_j|v_i) = \frac{w_{ij}}{\sum_{k\in V} w_{ik}}$$

邻近网络的模型分布，其中，u_i 是 v_i 被视为顶点时的表示，u_i' 是 v_i 被视为 "context" 时的表示：

$$p_2(v_j|v_i) = \frac{\exp(\vec{u_j'}^T \vec{u_i})}{\sum_{k\in V} \exp(\vec{u_k'}^T \vec{u_i})}$$

目标函数是 **KL 散度**：

$$O_2 = \sum_i KL(\hat{p_2}(\cdot|v_i), p_2(\cdot|v_i)) = - \sum_{(i,j)\in E} w_{ij} \log p_2(v_j|v_i)$$

2.1.1.3 优化 trick

- sgd+negative sampling：随机 sample 一条边，以及多个 negative 的边

例如针对二阶的，对每条边 (i,j) 来说，它的目标函数就是：

$$\log \sigma(\vec{u_j'}^T \vec{u_i'}) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)}[\log \sigma(-\vec{u_n'}^T \vec{u_i'})]$$

其中 $\sigma(x) = 1/(1+\exp(-x))$，设置 $P_n(v) \propto d_v^{3/4}$，其中 d_v 是节点的出度（即 $d_i = \sum_{k \in N(i)} w_{ik}$，其中 $N(i)$ 是 v_i 的为起点的邻居的集合）。

针对一阶的，把上面式子里的第一项里的 $\vec{u_j'}^T$ 换成 $\vec{u_j}^T$ 就行啦～

- 边 (i,j) 的 embedding 的梯度：

$$\frac{\partial O_2}{\partial \vec{u_i}} = w_{ij} \frac{\partial \log \hat{p_2}(v_j|v_i)}{\partial \vec{u_i}}$$

- 当边的权重方差很大的时候，从上式可知，目标函数的梯度是 p_2 的梯度再乘以边权重，所以目标函数的梯度的方差也会很大，这样会有问题。
- 解决方法：**edge sampling**：根据边的权重来采样边，然后把采样到的边当成 binary 的，也就是把每条边的权重看成一样的！(例如一个边的权重是 w，那么拆成 w 条 binary 的边)
- 复杂度：$O(d \times K \times |E|)$：d 是 embedding 的维数，K 是负样本的个数，$|E|$ 是边的总数

2.1.1.4 讨论

- 对只有少量邻居的节点（low degree vertices）进行 embed：
  - 通过增加高阶邻居来扩展邻居
  - BFS(breadth-first search)，使用广度优先搜索策略扩展每个顶点的邻域，即递归地添加邻居的邻居
  - 在大部分场景下，只增加二阶邻居就足够了
- 对新节点进行 emb（如果新节点和已有节点有边相连，可以如下方式来搞；否则，future work…）：
  - 保持现有节点的 embedding 不变
  - 根据新节点的 embedding 求经验分布和模型分布，从而优化目标函数 w.r.t. 新 node 的 embedding

所以，对于新节点 i，直接最小化如下目标函数：

$$-\sum_{j \in N(i)} w_{ji} \log p_1(v_j, v_i)$$

或者

$$-\sum_{j \in N(i)} w_{ji} \log p_2(v_j|v_i)$$

2.1.1.5 实验

LINE(1st) 只适用于无向图，LINE(2nd) 适用于各种图。

LINE (1st+2nd)：同时考虑一阶相似度和二阶相似度。将由 LINE（1st）和 LINE（2nd）学习得到的两个向量表示，**连接成一个更长的向量**。在连接之后，对维度**重新加权**以**平衡两个表示**。因为在无监督的任务中，设定权重很困难，所以**只应用于监督学习**的场景。

更适合的方法是共同训练一阶相似度和二阶相似度的目标函数，比较复杂，文章中没有实现。

2.1.2 DeepWalk

KDD14 上的DeepWalk: Online Learning of Social Representations

使用学习 word representation 的方法来学习 node representation（例如 skip gram）

将网络上的随机游走视为句子。

分成两步：

- 通过随机游走生成结点的 context
- 预测周围的节点：

$$p(v_j|v_i) = \frac{\exp(\vec{u_i'}^T \vec{u_j})}{\sum_{k \in V} \exp(\vec{u_k'}^T \vec{u_i})}$$

2.1.3 Node2vec

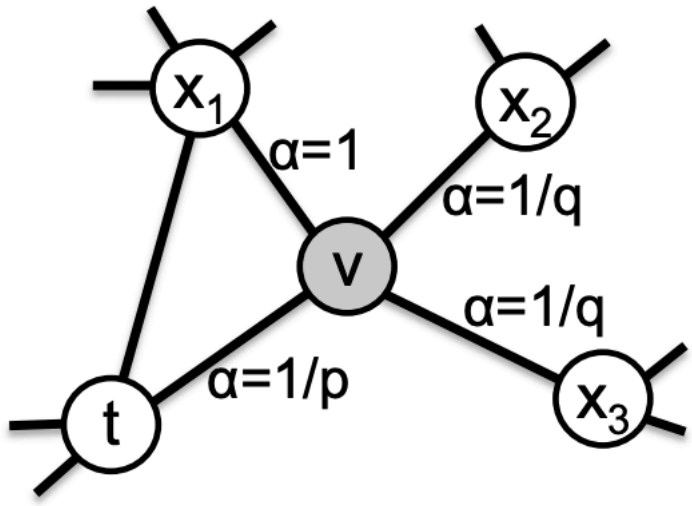
KDD16 上的node2vec: Scalable Feature Learning for Networks

通过如下混合策略去寻找一个 node 的 context：

- Breadth-firstSampling(BFS): homophily（同质性）
- Depth-firstSampling(DFS): structuralequivalence（结构等价）

使用带有参数 p 和 q 的 **Biased Random Walk** 来进行 context 的扩展，在 BFS 和 DFS 中达到一个平衡，同时考虑到微观局部 (BFS) 和宏观全局 (DFS) 的信息，并且具有很高的适应性：

- p: Return parameter，控制在 walk 的过程中，**revisit** 一个节点的概率，对应 BFS
- q: In-out parameter，控制探索 "**outward**" 节点的概率，对应 DFS
- 在有标签的数据上，用 **cross validation** 来寻找最优的 p 和 q



刚从 edge(t, v) 过来，现在在节点 v 上，要决定下一步 (v, x) 怎么走：

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{q} & if\ d_{tx} = 2 \end{cases}$$

其中的 d_{tx} 表示节点 t 到节点 x 间的最短路径：

- 为 0 表示回到节点 t 本身
- 为 1 表示节点 t 和节点 x 直接相连，但上一步却选择了节点 v
- 为 2 表示节点 t 和 x 不直接相连，但节点 v 和节点 x 直接相连

最简单的给 random walk 加上 bias 的方法就是转移概率 $\pi_{vx} = w_{vx}$，而我们的方法就是 $\pi_{vx} = \alpha_{pq}(t,x)w_{vx}$，相当于还考虑了跳到 v 之前的节点 t。

优化目标和 LINE 的**一阶相似度**类似

LINE、DeepWalk、Node2vec 的对比：

Algorithm	Neighbor Expansion	Proximity	Optimization	Validation Data
LINE	BFS	1^{st} or 2^{nd}	Negative Sampling	No
DeepWalk	Random	2^{nd}	Hierarchical Softmax	No
Node2Vec	BFS + DFS	1^{st}	Negative Sampling	Yes

node representation 的应用：

- Node **classification** (Perozzi et al. 2014, Tang et al. 2015a, Grover et al. 2015)
- Node **visualization** (Tang et al. 2015a)
- **Link** prediction (Grover et al. 2015)
- **Recommendation** (Zhao et al. 2016)
- **Text** representation (Tang et al. 2015a, Tang et al. 2015b)

node representation 的扩展：

- Leverage **global structural information** (Cao et al. 2015)
- Non-linear methods based on **autoencoders** (Wang et al. 2016) • Matrix-factorization based approaches (Qiu et al. 2018)
- **Directed** network embedding (Ou et al. 2016)
- **Signed** network embedding (Wang et al. 2017)
- **Multi-view** networks (Qu and Tang et al. 2017)
- Networks with **node attributes** (Yang et al. 2015)
- **Heterogeneous(异构)** networks (Chang et al. 2015)
- **Task-specific** network embedding (Chen et al. 2017)

2.2 Graph and High-dimensional Data Visualization

2.2.1 t-SNE

高维数据可视化的一个 state-of-the-art 的方法，tensorboard 就用的这个。

缺点：

- K-NNG(K-Nearest Neighbor Graph) construction: 复杂度是 $O(NlogN)$，假设图中有 N 个数据点
- Graph layout: 复杂度是 $O(NlogN)$
- 对参数非常敏感（Very sensitive parameters）

2.2.2 Visualizing Large-scale and High-dimensional Data

www16 的 best paper 提名Visualizing Large-scale and High-dimensional Data

largevis 代码（c++&python）：https://github.com/lferry007/LargeVis

特点：

- K-NNG construction 的高效近似：
  - 比 t-SNE 的速度快 30 倍（300w 的数据点）
  - 更好的 time-accuracy tradeoff
- graph layout 的高效的 probabilistic model
  - 从 $O(NlogN)$ 到 $O(N)$
  - 比 t-SNE 快 7 倍（300w 的数据点）
  - 更好的 visualization layouts
  - 在不同数据集间有更 stable 的参数

2.2.2.1 Learning the Layout of KNN Graph

- 保持 2D/3D 空间的节点的相似度
  - 对每个节点使用一个 2D/3D 的向量来表示
  - 保持相似的数据距离近而不相似的距离远
- 观测节点 (i, j) 间的一条 **binary** 的边的概率：

$$p(e_{ij} = 1) = \frac{1}{1 + \|\vec{y_i} - \vec{y_j}\|^2}$$

- 观测节点 (i, j) 间的一条**有权重**的边的 likelihood：

$$p(e_{ij} = w_{ij}) = p(e_{ij} = 1)^{w_{ij}}$$

2.2.2.2 A Probabilistic Model for Graph Layout

目标函数：

$$O = \prod_{(i,j)\in E} p(e_{ij} = w_{ij}) \prod_{(i,j)\in \bar{E}} (1 - p(e_{ij} = w_{ij})^{\gamma}$$

其中 γ 是给 **negative edge** 赋值的 **unified weight**

- 随机 sample 一些 negative edges
- 使用异步 sgd 来优化
- 时间复杂度：与数据点数是**线性**关系

2.3 Knowledge Graph Embedding

知识图谱是异构图，有多种类型的 relations

用 (head entity, relation, tail entity) 的三元组来表示 facts 的集合。

related works：

- 将 entities 用 embeddings 来表示
- 将 relations 用 embeddings 或者 matrices 来表示

Model	Score Function	
SE (Bordes et al., 2011)	$-\|\boldsymbol{W}_{r,1}\mathbf{h}-\boldsymbol{W}_{r,2}\mathbf{t}\|$	$\mathbf{h},\mathbf{t}\in\mathbb{R}^k,\boldsymbol{W}_{r,\cdot}\in\mathbb{R}^{k\times k}$
TransE (Bordes et al., 2013)	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
TransX	$-\|g_{r,1}(\mathbf{h})+\mathbf{r}-g_{r,2}(\mathbf{t})\|$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
DistMult (Yang et al., 2014)	$\langle\mathbf{r},\mathbf{h},\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ComplEx (Trouillon et al., 2016)	$\mathrm{Re}(\langle\mathbf{r},\mathbf{h},\bar{\mathbf{t}}\rangle)$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k$
HolE (Nickel et al., 2016)	$\langle\mathbf{r},\mathbf{h}\otimes\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ConvE (Dettmers et al., 2017)	$\langle\sigma(\mathrm{vec}(\sigma([\bar{\mathbf{r}},\bar{\mathbf{h}}]*\boldsymbol{\Omega}))\boldsymbol{W}),\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
RotatE	$-\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|^1$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k,\|r_i\|=1$

kg 的核心任务：预测 missing links

kg 的核心 idea：根据观测到的 knowledge facts，对 kg 中的 relation patterns 进行建模和 infer。也就是学习 **relations 的 relations**。

2.3.1 relation patterns

- 对称和反对称：
  - 对称 (Symmetric)：例如，marriage
  - 反对称 (Antisymmetric)：例如，Filiation(父子关系)

形式化定义：

$$\begin{array}{ll} r\ is\ Symmetric & r(x,y)\Rightarrow r(y,x)\ if\ \forall x,y \\ r\ is\ Antisymmetric & r(x,y)\Rightarrow \neg r(y,x)\ if\ \forall x,y \end{array}$$

- Inverse relations:
  - Hypernym(上位词) and hyponym(下位词)：花是鲜花的上位词，鲜花是花的下位词
  - 丈夫和妻子

形式化定义：

$$r_1\ is\ inverse\ to\ relation\ r_2:\ r_2(x,y)\Rightarrow r_1(y,x)\ if\ \forall x,y$$

- Composition Relations
  - My mother's husband is my father

形式化定义：

$$r_1\ is\ a\ composition\ of\ relation\ r_2\ and\ relation\ r_3:\quad r_2(x,y)\wedge r_3(y,z)\Rightarrow r_1(x,z)\ if\ \forall x,y,z$$

目前的方法没有一种能同时 infer 上面这所有 3 种 relation patterns，只有 RotatE 可以!!

Model	Score Function	Symmetry	Antisymmetry	Inversion	Composition
SE	$-\|\boldsymbol{W}_{r,1}\mathbf{h}-\boldsymbol{W}_{r,2}\mathbf{t}\|$	✗	✗	✗	✗
TransE	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	✗	✓	✓	✓
TransX	$-\|g_{r,1}(\mathbf{h})+\mathbf{r}-g_{r,2}(\mathbf{t})\|$	✓	✓	✗	✗
DistMult	$\langle\mathbf{h},\mathbf{r},\mathbf{t}\rangle$	✓	✗	✗	✗
ComplEx	$\mathrm{Re}(\langle\mathbf{h},\mathbf{r},\bar{\mathbf{t}}\rangle)$	✓	✓	✓	✗
RotatE	$-\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|$	✓	✓	✓	✓

2.3.2 RotatE

ICLR19 RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.

RotatE 代码（pytorch）:https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding

每一个 relation 可以看成是从 source entity 到 target entity 在 complex(复数) 向量空间上的 elementwise rotation

RotatE 可以同时建模和 infer 上面这所有 3 种 relation patterns

优化 RotatE 可以用高效的 negative sampling

在 kg 的 link prediction 的 benchmarks 中能达到 state-of-the-art 的效果

2.3.2.1 Relation as Elementwise Rotation in Complex Space

head entity: $h \in \mathbb{C}^k$; tail entity: $t \in \mathbb{C}^k$

relation r: 是一个从 h 到 t 的 elementwise rotation:

$$t = h \circ r, \ where \ |r_i| = 1$$

其中，$\circ$ 是 element-wise product，所以 $t_i = h_i r_i$，其中

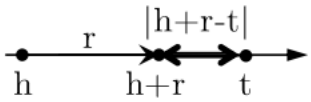
$$r_i = e^{\mathbf{i}\theta_{r,i}}$$

里面的 $\theta_{r,i}$ 是 r 的第 i 维的 phase angle，e 的 $\mathbf{i}\theta_{r,i}$ 的第一个 $\mathbf{i}$ 是虚数单位，第二个 i 是第 i 维。

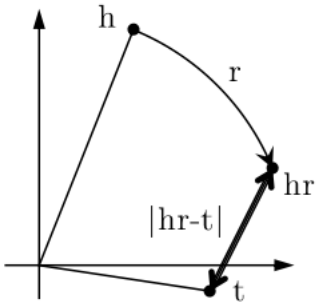
定义 distance function:

$$d_r(h,t) = \|h \circ r - t\|$$

- 如左图，transE 建模的是 $h + r$ 和 t 的距离，也就是在**实数直线**上以 **translation** 的方式建模 r;
- 如右图，RotatE 建模的是 $h \circ r$ 和 t 的距离，也就是在**复平面**上以 **rotation** 的方式建模 r。



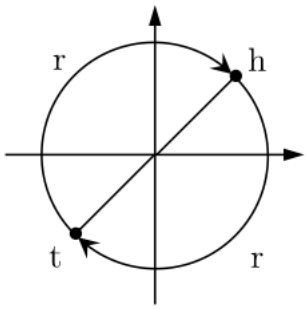
(a) TransE models **r** as translation in real line.



(b) RotatE models **r** as rotation in complex plane.

先科普一下，在复变函数中，自变量 z 可以写成 $z = r \times (\cos\theta + \mathbf{i}\sin\theta)$，$r$ 是 z 的模，即 $r = |z|$; θ 是 z 的辐角，记作 $Arg(z)$。在 $-\pi$ 到 π 间的辐角称为辐角主值，记作 $arg(z)$。指数形式 $z = r(\cos\theta + i\sin\theta) = re^{\mathbf{i}\theta}$。

- relation r 是对称的，当且仅当，$r_i = \pm 1$，也就是 $\theta_{r,i} = 0 \ or \ \pi$，例如下图，$r_i = -1$ 也就是 $\theta_{r,i} = \pi$



(c) RotatE: an example of modeling symmetric relations $\mathbf{r}$ with $r_i = -1$

- relation r 是反对称的，当且仅当，$r \circ r \neq 1$
- relation r_1 和 r_2 是 inverse，当且仅当，$r_2 = r_1^{-1}$，也就是 $\theta_{2,i} = -\theta_{1,i}$
- relation $r_3 = e^{\mathbf{i}\theta_3}$ 是两个 relation $r_1 = e^{\mathbf{i}\theta_1}$ 和 $r_2 = e^{\mathbf{i}\theta_2}$ 的 composition，当且仅当，$r_3 = r_1 \circ r_2$，也就是 $\theta_3 = \theta_1 + \theta_2$

2.3.2.2 RoteE 的优化

Negative sampling loss 如下：

$$L = -\log \sigma(\gamma - d_r(h,t)) - \sum_{i=1}^{k} \frac{1}{k} \log \sigma(d_r(h'_i, t'_i) - \gamma)$$

其中的 γ 是一个 fixed margin，σ 是 sigmoid，(h'_i, r, t'_i) 是第 i 个 negative 三元组。

然后我们要变成 self-adversarial negative sampling：

- 传统地，负样本通过 uniform 的方式（均匀分布，即等概率）来采样
  - 随着训练的继续，因为很多样本是 obviously false 了，所以这种采样是 inefficient 的
  - 没有提供有用的信息
- self-adversarial negative sampling：
  - 根据当前的 embedding model 来进行 negative 三元组的采样
  - 从更简单的 samples 开始，逐步变难
  - Curriculum Learning（递进学习，课程学习，可以参考https://blog.csdn.net/qq_25011449/article/details/82914803），从如下分布中进行采样：

$$p(h'_j, r, t'_j | \{(h_i, r_i, t_i)\}) = \frac{\exp \alpha f_r(h'_j, t'_j)}{\sum_i \exp \alpha f_r(h'_i, t'_i)}$$

其中，α 是 sampling 的 temperature，$f_r(h'_j, t'_j)$ 衡量三元组的 salience(突出程度)

但在实际应用中，从上面这个分布去 sample 的代价是很大的，所以我们把这个概率直接作为负样本的权重，所以最终的 loss 如下：

$$L = -\log \sigma(\gamma - d_r(h,t)) - \sum_{i=1}^{k} p(h'_i, r, t'_i) \log \sigma(d_r(h'_i, t'_i) - \gamma)$$

2.4 A High-performance Node Representation System

A High-Performance CPU-GPU Hybrid System for Node Embedding，投稿 www19

algorithm and system co-design 的一个 node embeddings 的系统

- CPUs: online random walk generation
- GPUs: training node embeddings
- Efficient and effective collaboration strategies between CPUs and GPUs

比现有的系统快 50 倍，一个有 100w 节点的网络只要 1min

3 part2-Graph Neural Networks

3.1 基础知识

通过一个 encoder 函数 ENC，把原始网络的结点 u 和结点 v 映射到 embedding space 的 d 维向量 z_u 和 z_v，然后希望原空间的相似度和 embedding space 的相似度（例如内积）接近：

$$similarity(u, v) \approx z_v^T z_u$$

之前的 encoder 是 shallow 的，也就是一个 Z 矩阵，使用 embedding lookup，矩阵大小是 node_num * emb_dim。缺点如下：

- **需要 $O(|V|)$ 的参数**：每个 node 有自己的 unique 的 embedding vector，**没有参数共享**！！
- **Inherently "transductive"**：固有的『直推式』。也就是说，对于**训练中没有见过的结点**，不可能生成一个 embedding
- **没有包含节点 feature**：很多图有一些我们必须要考虑和利用好的 feature。

因此需要使用 deeper 的 encoder，而这些更复杂的 encoder 也自带了 similarity 函数。

参考 2017 年的综述Representation Learning on Graphs: Methods and Applications

还有 2005 年的The Graph Neural Network Model

定义：

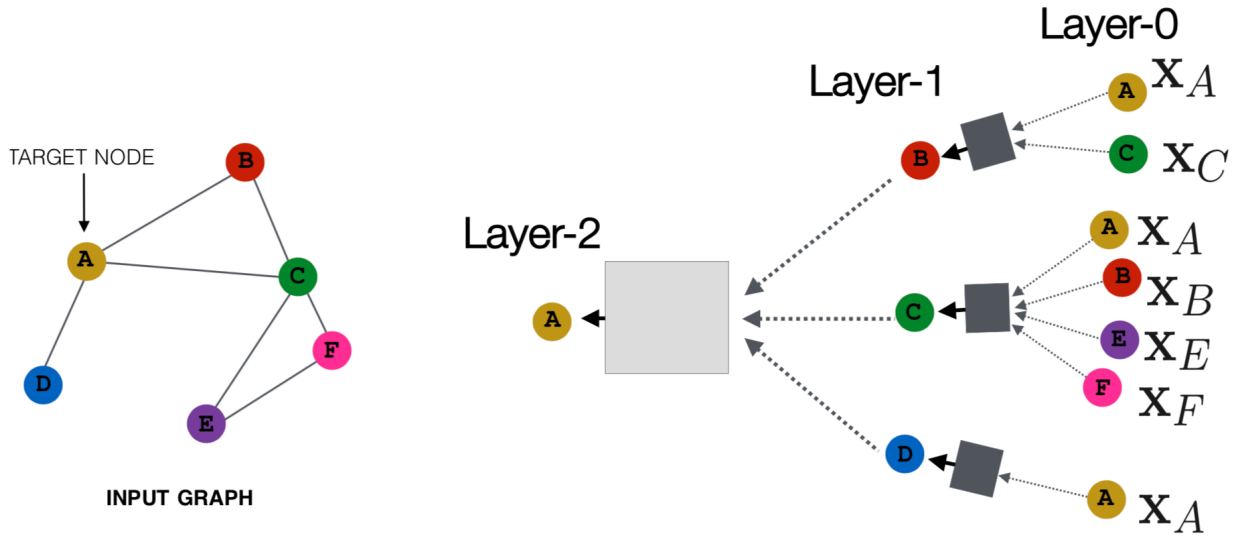
- G：图
- V：节点集合
- A：邻接矩阵 (假设是 binary 的)
- $X \in R^{m \times |V|}$：**节点 features** 的矩阵
  - 类别型的特征、文本、图像数据等
  - 节点度数、clustering coefficients(聚集系数，参考https://blog.csdn.net/pennyliang/article/details/6838956) 等
  - Indicator vectors(例如，每个节点的 one-hot vector)

3.1.0.1 Neighborhood Aggregation

核心思想：使用 nn 对节点的邻居的信息进行汇聚，生成这个节点的 embedding

如下图：

- node 在每一层都有 embedding
- 模型的 depth 可以任意
- 节点 u 在第 0 层的 embedding 是它的 input-feature x_u



neighborhood aggregateion 其实数学上和 spectral graph convolutions(参考Geometric deep learning: going beyond Euclidean data) 很像，可以看成是一种 center-surround filter。

关键在于上图的 layer1 和 layer2 用什么样的网络结构，一种 basic 的方法就是，layer2 先 average，然后再接一个神经网络：

$$
\begin{aligned}
h_v^0 &= x_v \\
h_v^k &= \sigma(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}), \forall k > 0 \\
z_v &= h_v^K
\end{aligned}
$$

- h_v^0：第 0 层的 embedding 就是 node 的特征
- h_v^k：第 k 层的 embedding，包括的两项分别是邻居节点的前一层的 emb 的平均，还有当前节点的前一层的 emb
- σ：非线性，可以是 relu/tanh 等
- W_k 和 B_k 是两个待训练的矩阵
- z_v：最终的输出结果，也就是第 K 层的输出

训练可以使用无监督的方法，loss 可以是前面讲到的任意的 node embedding 的方法：

- Random walks (node2vec, DeepWalk)
- Graph factorization
- 或者直接训练保证相似的 node 有相似的 embedding

也可以直接用监督学习的方法来训（例如是一个 node 的分类问题），其中的 θ 是 classification weights：

$$
L = \sum_{v \in V} y_v \log(\sigma(z_v^T \theta) + (1 - y_v) \log(1 - \sigma(z_v^T \theta)))
$$

归纳能力（inductive capability）：

- 所有节点共享相同的 aggretation parameters
- 模型参数是 $|V|$ 的 sublinear，而且可以对没见过的 node 生成 embed

3.2 Graph Convolutional Networks(GCN)

参考 ICLR17 的Semi-Supervised Classification with Graph Convolutional Networks

在 neighborhood aggretagtion 上有一些小改动：

$$h_v^k = \sigma(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

和普通 gnn 的区别：

- self 和 neighbor 的 embedding 共用同一个权重 W_k，而普通的 gnn 是两个权重 B_k 和 W_k，好处就是有**更多的参数共享**
- 每一个 neighbor 都有 normalization(即 $\sqrt{|N(u)||N(v)|}$)，好处就是可以**减小度数多的邻居的权重**

3.3 GraphSAGE

参考 NIPS17 的Inductive Representation Learning on Large Graphs

出发点：把上面在 aggretate 之后使用的神经网络换成任意一个可以把一堆 vectors 映射成一个单独的 vector 的可微函数（也就是下面的 $AGG(\{h_u^{k-1}, \forall u \in N(v)\})$）：

$$h_v^k = \sigma([A_k \cdot AGG(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_v^{k-1}])$$

上面的 $[A_k \cdot AGG(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_v^{k-1}]$ 是把这 self embedding 和 neighbor embedding 这两个向量 **concate** 到一起。

AGG 的变种：

- mean：

$$AGG = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$$

- pool：对 neighbor vectors 进行转换（例如下面的 Q），并进行 symmetric vector 函数变换（例如下面的 γ 就是 element-wise mean/max）

$$AGG = \gamma(\{Qh_u^{k-1}, \forall u \in N(v)\})$$

- lstm：对 neighbor 的一个**随机排列**（random permutation）使用 lstm

$$AGG = LSTM([h_u^{k-1}, \forall u \in \pi(N(v))])$$

3.4 Gated Graph Neural Networks

参考 ICLR16 的Gated Graph Sequence Neural Networks

参考 ICML17 的Neural Message Passing for Quantum Chemistry

GCNs 和 GraphSAGE 大部分情况下只有 **2-3 层**深，层数加深有如下挑战：

- 参数太多导致过拟合
- bp 的过程中出现梯度消失/爆炸

思路：

- 层间参数共享
- Recurrent state update：各层的神经网络使用 RNN。

3.4.1 Gated Graph Neural Networks 介绍

Recurrent state update 这种方法分成两步：

- 在 step k 从 neighbors 获取 “message”，**这个聚合函数与 k 无关**：

$$m_v^k = W \sum_{u \in N(v)} h_u^{k-1}$$

- 通过 gru 来更新节点的 “state”。新节点的 state 依赖 old state 以及 neighbors 的 “message”：

$$h_v^k = GRU(h_v^{k-1}, m_v^k)$$

优点：

- 可以处理 **20+ 的层数**
- 绝大部分真实世界的网络有比较小的 diameters（直径，放大倍率），**大部分小于等于 7**
- 能够将 **global 的图结构**的复杂信息传播给所有结点
- 对复杂网络的表示很有用（例如 Logical formulas，或者程序）

3.4.2 Message-Passing Neural Networks 介绍

从以下两个方面来对 gated graph neural networks 进行泛化：

- 在 step k 从 neighbors 获取 “message”：

其中的 M 可以是一个一般 (generic) 的 “message” 函数，例如 sum 或者 MLP。$e_{u,v}$ 把**边的信息**考虑进来了！

$$m_v^k = \sum_{u \in N(v)} M(h_u^{k-1}, h_v^{k-1}, e_{u,v})$$

- 更新 node 的 “state”：

其中的 U 可以是一个一般 (generic) 的 “update” 函数，例如 LSTM 或者 GRU

$$h_v^k = U(h_v^{k-1}, m_v^k)$$

所以，其实这是一个通用的 conceptual（概念性的）framework，可以归纳大部分 GNNs。

3.5 Graph Attention Networks(GAT)

参考 ICLR18 的Graph Attention Networks

key idea：某些 neighbor 更重要，所以可以使用 attention 机制来搞

$$h_v^k = \sigma(\sum_{u \in N(v) \cup \{v\}} \alpha_{v,u} W^k h_u^{k-1})$$

其中：

- σ 是非线性；
- $\sum_{u \in N(v) \cup \{v\}}$ 意味着把所有 neighbor(包括节点自己！！) 都加起来
- $\alpha_{v,u}$ 是学习到的 attention 权重

各种 attention 都是可以的，原始 GAT 用的是如下 attention 权重：

$$\alpha_{v,u} = \frac{\exp(LeakyReLU(a^T[Qh_v, Qh_u]))}{\sum_{u' \in N(v) \cup \{v\}} \exp(LeakyReLU(a^T[Qh_v, Qh_{u'}]))}$$

对照上面讲到的通用的 conceptual（概念性的）framework，其实就是**把 attention 加到获取 "message" 那步里去。**

其他新的东西：

- Generalizations based on spectral convolutions:
  - Geometric Deep Learning (Bronstein et al., 2017, Geometric deep learning: going beyond Euclidean data)
  - Mixture Model CNNs (Monti et al., 2017, Geometric deep learning on graphs and manifolds using mixture model CNNs)
- Speed improvements via subsampling:
  - FastGCNs (Chen et al., 2018, FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling)
  - Stochastic GCNs (Chen et al., 2017, Stochastic Training of Graph Convolutional Networks with Variance Reduction)

还可以参考专栏 | 深入理解图注意力机制

3.6 Subgraph Embeddings

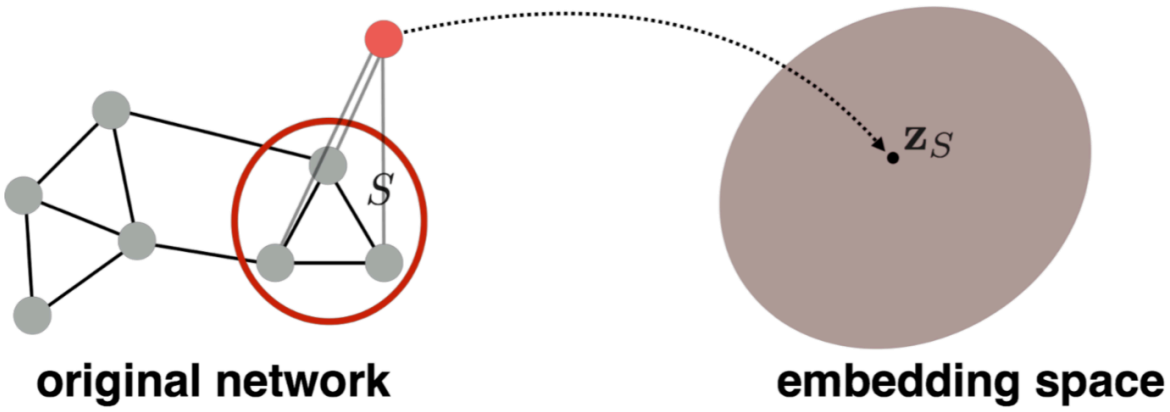
- 方法一：直接把子图中的 node 的 emb 进行 sum 或者 avg

$$z_S = \sum_{v \in S} z_v$$

见 2016 年的Convolutional Networks on Graphs for Learning Molecular Fingerprints

- 方法二：引入 "virtual node" 来表示子图，并走一个完整的 gnn

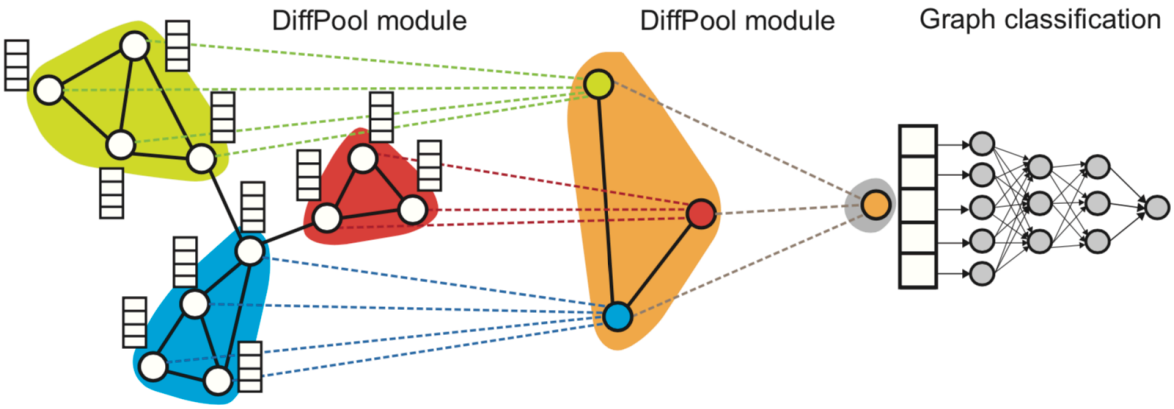
如下图



见 2016 年的Gated Graph Sequence Neural Networks

- 方法三：对节点进行层次聚类

见 2018 年的Hierarchical Graph Representation Learning with Differentiable Pooling



大致流程如下：

1. 在图上跑 GNN，得到 node 的 embeddings
2. 对 node embeddings 进行聚类，得到一个 “coarsened” graph（粗糙的）
3. 在 “coarsened” graph 上跑 GNN
4. 重复

学习 clustering 的不同方式：

- 使用 softmax weight 的 soft clustering（2018 年的Hierarchical Graph Representation Learning with Differentiable Pooling）
- 使用 hard clustering（2018 年的Towards Sparse Hierarchical Graph Classifiers和 2018 年的GRAPH U-NET）

4 part3-Deep Generative Models for Graph Generation

4.1 深度生成模型

深度生成模型的目标：为数据分布 $p(x)$ 隐式或者显式地建模，x 是一个高维随机变量

4.1.1 Variational Autoencoders (VAEs)

原始论文：2014 年 Kingma et al. 的Auto-Encoding Variational Bayes

Latent variable model:

- 一个 encoder $q_\phi(z|x)$
- 一个 decoder $q_\theta(x|z)$

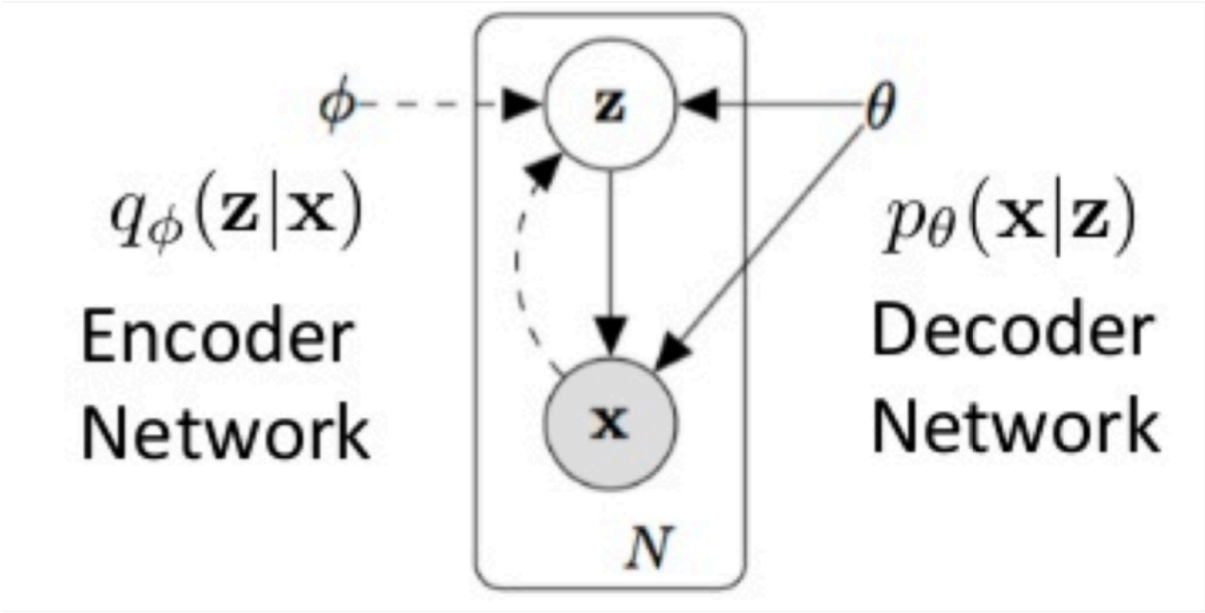
最大化 log likelihood $\log p(x)$：inference 是 intractable（棘手）的，因为 z 是连续的

最大化 variational 的下界 $L(\phi, \theta; x)$

通过 reparametrization trick 来 jointly 优化 encoder 和 decoder：

$$L(\phi, \theta; x) = E_{q_\phi(z|x)} \log p_\theta(x|z) - KL[q_\phi(z|x)||p(z)]$$

其中的 $E_{q_\phi(z|x)} \log p_\theta(x|z)$ 是 reconstruction，$KL[q_\phi(z|x)||p(z)]$ 是 regularization



小结一下，encoder 是 q_ϕ，decoder 是 p_θ，encoder 根据 x 生成 z，decoder 根据 z 生成 x。

可以参考https://blog.csdn.net/antkillerfarm/article/details/80648805

重构的过程是希望没噪声的，而 KL loss 则希望有高斯噪声的，两者是对立的。所以，VAE 跟 GAN 一样，内部其实是包含了一个对抗的过程，只不过它们两者是混合起来，共同进化的。

公式推导可以看https://blog.csdn.net/weixin_40955254/article/details/82315909

4.1.2 Generative Adversarial Networks (GANs)

原始论文：2014 年 Goodfellow et al. 的Generative Adversarial Networks

一个两个玩家的 Minimax 游戏：

- Generator $G : z \to x$。目标是迷惑 discriminator
- Discriminator $D : x \to \{0, 1\}$。目标是区分真实数据和生成的数据

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

直观地理解，这个式子包括两部分，一部分是判别真实数据是正例的概率，另一部分是判别生成的数据是负例的的概率，对于 G 来讲，期望这个式子 min，而对于 D 来讲，期望这个式子 max

4.1.3 Deep Auto-regressive Models

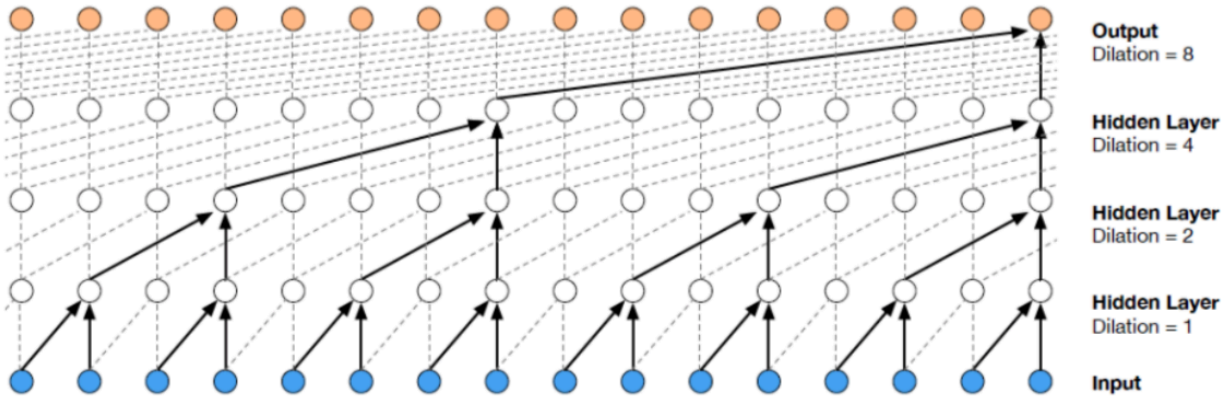
深度自回归模型：例如 RNN

例如，PixelRNN（2016 年 Oort et al. 的Pixel Recurrent Neural Networks）和 PixelCNN（2016 年也是 Oort et al. 的Conditional Image Generation with PixelCNN Decoders）：

- 一个 pixel 一个 pixel 地生成图像
- 通过一个神经网络来对条件概率分布建模

WaveNet（2017 年 Oort et al. 的WaveNet: A Generative Model for Raw Audio）

$$p(x) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1})$$



但如果要用在图上，有以下几个挑战：

- 图的 structures 和 size 是不一样的
- node 之间并没有顺序
- 离散

4.2 GraphVAE

2018 年 Simonovsky 和 Komodakis 的GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders

提出了生成图的 VAE 的框架：

- 输入 graph
- encoder：gnn+gated pooling=>graph representation，参考 Li et al. 在 2015 的
- decoder：输出一个预先定义好 max size 的 probalistic fully-connected graph
  - 对节点、边、节点和边的属性的存在性单独建模
  - graph matching 是必须的

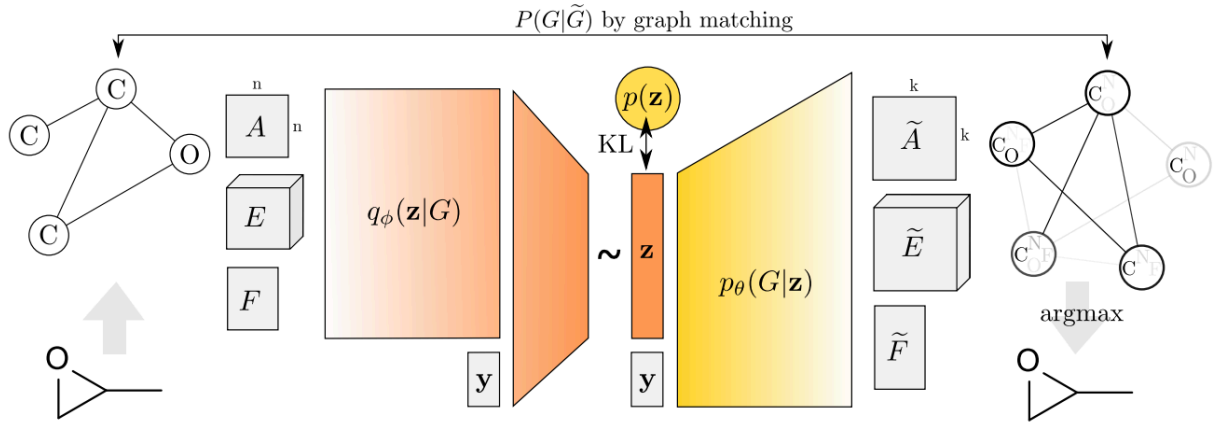


Figure 1. Illustration of the proposed variational graph autoencoder. Starting from a discrete attributed graph $G = (A, E, F)$ on n nodes (*e.g.* a representation of propylene oxide), stochastic graph encoder $q_\phi(\mathbf{z}|G)$ embeds the graph into continuous representation $\mathbf{z}$. Given a point in the latent space, our novel graph decoder $p_\theta(G|\mathbf{z})$ outputs a probabilistic fully-connected graph $\widetilde{G} = (\widetilde{A}, \widetilde{E}, \widetilde{F})$ on predefined $k \geq n$ nodes, from which discrete samples may be drawn. The process can be conditioned on label $\mathbf{y}$ for controlled sampling at test time. Reconstruction ability of the autoencoder is facilitated by approximate graph matching for aligning G with $\widetilde{G}$.

输入的 graph 是 $G = (A, E, F)$：A 是邻接矩阵；E 是边的属性的 tensor；F 是节点的属性的矩阵

decoder 的输出：

- 限制 domain 在最多 max k 个节点的所有 graphs 的集合中（k 一般是 10 左右）
- 一次输出一个 k 个节点的 probalistic fully-connected graph $\tilde{G} = (\tilde{A}, \tilde{E}, \tilde{F})$
  - 以 **bernoulli variables** 建模 nodes 和 edges 的 **existence**
  - 以 **multinomial variables** 建模 nodes 和 edges 的 **attributes**
  - $\tilde{A} \in [0, 1]^{k \times k}$：同时包括 node probabilities $\tilde{A}_{aa}$ 和 edge probabilities $\tilde{A}_{ab}$，其中 $a \neq b$
  - $\tilde{E} \in [0, 1]^{k \times k \times d_e}$：表示 edge attributes 的 probabilities
  - $\tilde{F} \in [0, 1]^{k \times d_e}$：表示 node attributes 的 probabilities
- inference：在 $\tilde{A}$，$\tilde{E}$，$\tilde{F}$ 中使用 edge-wise 和 node-wise 的 argmax
- 计算 reconstruction loss 的时候，需要使用 graph matching

缺点：

- graph 的 max size 必须是预先定义好的
- graph matching 是必须的

4.3 JTVAE

Junction Tree Variational Autoencoder for Molecular Graph Generation

- 利用了化学领域的知识
  - 每个 molecule(分子) 可以表示为化学 substructures(如环、键 (bond)) 的树状的 scaffold(骨架、支架)
- 生成一个树状结构的 object
  - 用来表示 subgraph components 的 scaffold
- 将 substructure 组装成一个 coherent(连贯的) molecular graph

4.4 MolGAN

MolGAN: An implicit generative model for small molecular graphs

- 一个 implicit, likelihood-free 的生成模型：用于分子生成

- 结合了强化学习来 encourage 生成的带有化学属性的分子
- Generator：从先验分布中生成分子
- Discriminator：区分生成的 sample 和真实的 sample
- Reward network：
  - 学习给每个分子赋值一个 reward，这个 reward 要和 external software 提供的 score 进行 match
  - invalid 的分子通常得到的 reward 是 0

整体架构图如下：

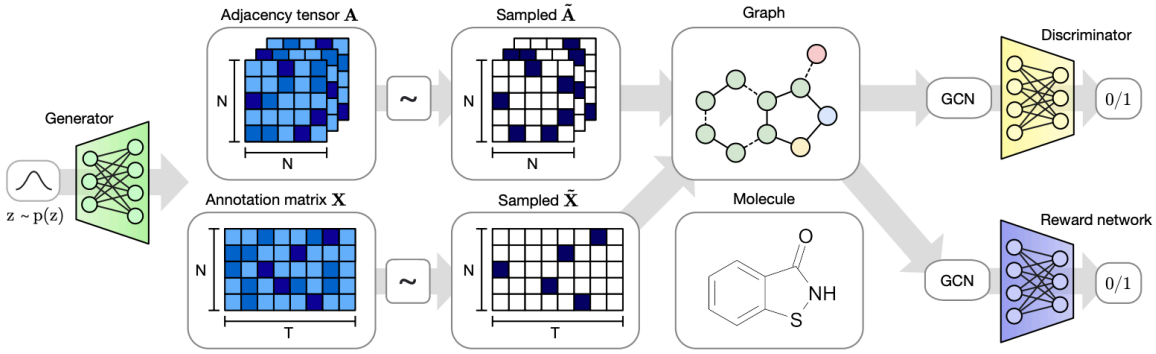


Figure 2. Outline of MolGAN. From *left*: the generator takes a sample from a prior distribution and generates a dense adjacency tensor $\boldsymbol{A}$ and an annotation matrix $\boldsymbol{X}$. Subsequently, sparse and discrete $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ are obtained from $\boldsymbol{A}$ and $\boldsymbol{X}$ respectively via categorical sampling. The combination of $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{X}}$ represents an annotated molecular graph which corresponds to a specific chemical compound. Finally, the graph is processed by both the discriminator and reward networks that are invariant to node order permutations and based on Relational-GCN (Schlichtkrull et al., 2017) layers.

Generator：

- 生成一个 probabilistic fully-connected graph：
  - $X \in R^{N \times T}$：atom types
  - $A \in R^{N \times N \times Y}$：bond types
- 目标函数：

$$L(\theta) = \lambda L_{WGAN} + (1 - \lambda) L_R L$$

Discriminator & Reward network：

- 通过 neural message passing algorithm 的一个变种-Relational-GCN，Schlichtkrull et al. 2017 的Modeling relational data with graph convolutional networks来学习分子/graph 的表示
- discriminator 和 reward network 用相同的网络结构（但参数不共享）
- reward network 用来近似 external software 的打分 (使用真实的 samples 和生成的 samples 进行训练)

优缺点：

- 不需要 graph matching
- graphs/分子的 max size 仍然需要预先定义

4.5 GCPN

You et al. 在 2018 的Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation

- 将分子的生成看成序列决策问题
  - 增加节点和边
  - 一个马尔可夫决策过程
- 目标：发现分子式，能优化融入了 chemical rules 的特定的 properties
- GCPN：一个结合了 RL 的面向目标（goal-directed）的通用的 model

  - 使用 policy gradients 来优化 adversarial loss 和 domain-specific rewards
  - 能够在一个融入了 domain-specific rules 的 environment 中生效

整体架构图如下：

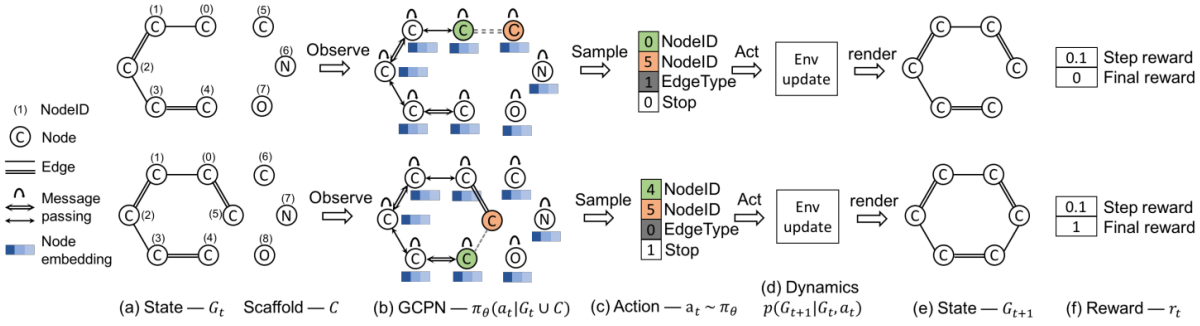


Figure 1: An overview of the proposed iterative graph generation method. Each row corresponds to one step in the generation process. **(a)** The state is defined as the intermediate graph G_t, and the set of scaffold subgraphs defined as C is appended for GCPN calculation. **(b)** GCPN conducts message passing to encode the state as node embeddings then produce a policy π_θ. **(c)** An action a_t with 4 components is sampled from the policy. **(d)** The environment performs a chemical valency check on the intermediate state, and then returns **(e)** the next state G_{t+1} and **(f)** the associated reward r_t.

- $M = (S, A, P, R, \gamma)$:
  - states $S = \{s_i\}$：包括所有 intermediat 和 final graphs
  - actions $A = \{a_i\}$：每一个 step 对当前 graph 进行的修改
  - 状态转移概率 P
  - reward 函数 R
  - discount factor γ
- 状态空间：
  - s_t 是中间生成的图 G_t
  - G_0 包括一个 single node，表示一个 carbon atom(碳原子)
- 动作空间：
  - 每个 step 将要添加的一个 atoms 的集合：$C = \cup_{i=1}^{S} C_i$
  - 具体的 actions:
    * 把一个新的 atom C_i 连接到现有的 G_t 中的一个节点上去
    * 连接 G_t 内退出 (exiting) 的节点
- state transition dynamics:
  - 在 state transition dynamics 中融入了 domain-specific rules，只执行遵守规则的 actions
  - policy network 产生的 infeasible(不可实行的) 动作会被 rejected，而 state 保持不变
- Reward 设计
  - final rewards：domain-specific rewards 之和（例如，最终的 property scores，对不真实的分子的惩罚，adversirial rewards）
  - intermediate rewards：step-wise validity(有效性) rewards 和 adversirial rewards
- GCPN
  - 使用 neural message passing 算法计算节点的 embeddings
  - 预测 action：
    * 挑选两个节点
    * 预测边的类型
    * 预测是否结束（termination）

整体公式如下：

$$a_t = CONCAT(a_{first}, a_{second}, a_{edge}, a_{stop})$$

其中

$$
\begin{array}{ll}
f_{first}(s_t) = SOFTMAX(m_f(X)), & a_{first} \sim f_{first}(s_t) \in \{0,1\}^n \\
f_{second}(s_t) = SOFTMAX(m_s(X_{a_{first}}, X)), & a_{second} \sim f_{second}(s_t) \in \{0,1\}^{n+c} \\
f_{edge}(s_t) = SOFTMAX(m_e(X_{a_{first}}, X_{a_{second}})), & a_{edge} \sim f_{edge}(s_t) \in \{0,1\}^b \\
f_{stop}(s_t) = SOFTMAX(m_t(AGG(X))), & a_{stop} \sim f_{stop}(s_t) \in \{0,1\}
\end{array}
$$

5 未来方向

参考https://zhuanlan.zhihu.com/p/38142339

主要想法：将 relational 的关系转化成 attention，利用 attention 来代表两个 entity 的关系。隐式地将 relational 引入 NN 结构中

Zambaldi et al. 在 2018 的Relational deep reinforcement learning