Graph Matching Networks for Learning the Similarity of Graph Structured Objects

Yujia Li [1] Chenjie Gu [1] Thomas Dullien [2] Oriol Vinyals [1] Pushmeet Kohli [1]

【深度学习 56】GMN

张楚珩
清华大学 交叉信息院博士在读

19 人赞同了该文章

GMN的全称是Graph Matching Network。

原文传送门

Li, Yujia, et al. "Graph Matching Networks for Learning the Similarity of Graph Structured Objects." arXiv preprint arXiv:1904.12787 (2019).

特色

这篇文章主要提出了两种基于深度学习判断图（graph）相似性的方法。第一种方法是利用Graph Neural Network（GNN）去提取图的信息，得到一个向量，然后通过比较不同图向量之间的距离来比较图之间的相似性；第二种方法是文章提出的GMN，直接对于给定的两个图输出这两个图之间的相似性。这个工作和强化学习没啥关系，不过我最近在考虑强化学习中的迁移学习，如果能够较好的给出MDP之间的相似性度量，那么可以辅助迁移学习的进行，避免negative transfer。如果大家在这方面有什么想法的，可以和我交流。

过程

1. 图相似性学习的两种思路

文章中讲到图的相似性学习有两种途径。第一种是通过graph embedding，对于每个图得到一个该图的embedding，表示成一个向量，不同图之间的相似性比较只需要比较这些图embedding之间的距离即可。这种方法的好处是如果需要对于数据库中大量的图进行比较的时候，只需要对于数据库中图的这些embedding进行比较，可以利用已有的一些算法快速得到匹配，比如k-d trees、locality sensitive hashing。第二种是对于图进行pairwise的比较，这样对于两个图之间各个节点的相似性就能重点比较。其好处是这种比较方式结果精度会更高，但计算复杂度也更高。

本文两种方法都分别提出了一种模型，分别是Graph Embedding Model和Graph Matching Network。第一种模型如左图所示，第二种模型如右图所示。

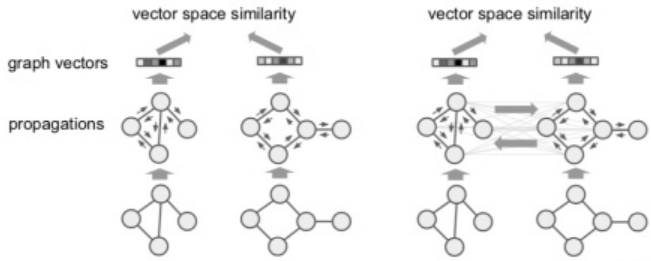

Figure 2. Illustration of the graph embedding (left) and matching models (right). 知乎 @张楚珩

2. Graph Embedding Model

主要基于GNN来提取图上的信息，通过若干轮，在图上相邻节点之间交换信息，然后再把所有节点上的信息聚合产生关于这个图的整体embedding。

第一步是对于图上每一个节点和每一条边的特征进行编码（encode），如果节点或者边没有更多额外能利用的信息的话，它们的原始特征可以设置为常数1，即下面的 $x_i = 1, x_{ij} = 1$ 。

$$\begin{aligned} \mathbf{h}_i^{(0)} &= \mathrm{MLP}_{\mathrm{node}}(\mathbf{x}_i), \quad \forall i \in V \\ \mathbf{e}_{ij} &= \mathrm{MLP}_{\mathrm{edge}}(\mathbf{x}_{ij}), \quad \forall (i,j) \in E. \end{aligned}$$

其中 x_i, x_{ij} 分别代表节点和边的特征，相应的MLP编码器就是简单的一个隐含层的神经网络。

第二步是在图上进行多轮的信息传播，对于每一轮，每一条连边都通过一个神经网络生成一个message，每个节点都接受来自相邻节点的message并且形成新的节点表示。

$$\begin{aligned} \mathbf{m}_{j \to i} &= f_{\mathrm{message}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}) \\ \mathbf{h}_i^{(t+1)} &= f_{\mathrm{node}}\left(\mathbf{h}_i^{(t)}, \textstyle\sum_{j:(j,i)\in E} \mathbf{m}_{j \to i}\right) \end{aligned}$$

第三步是把得到的节点表示都聚合起来，形成关于整个图的embedding。

$$\mathbf{h}_G = \mathrm{MLP}_G\left(\sum_{i \in V} \sigma(\mathrm{MLP}_{\mathrm{gate}}(\mathbf{h}_i^{(T)})) \odot \mathrm{MLP}(\mathbf{h}_i^{(T)})\right)$$

3. Graph Matching Network

该模型在第一步和第三步都和前一种模型相同，最主要的区别是第二步在message passing的过程中会在两个图之间传递信息，每个节点都会去尽量匹配另一个图中的相似节点，并且产生图之间的信息传递。数学表示如下

$$
\begin{aligned}
\mathbf{m}_{j\to i} &= f_{\text{message}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, \mathbf{e}_{ij}), \forall (i,j) \in E_1 \cup E_2 \\
\boldsymbol{\mu}_{j\to i} &= f_{\text{match}}(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}), \\
&\qquad \forall i \in V_1, j \in V_2, \text{or } i \in V_2, j \in V_1 \\
\mathbf{h}_i^{(t+1)} &= f_{\text{node}}\left(\mathbf{h}_i^{(t)}, \sum_j \mathbf{m}_{j\to i}, \sum_{j'} \boldsymbol{\mu}_{j'\to i}\right) \\
\mathbf{h}_{G_1} &= f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_1}) \\
\mathbf{h}_{G_2} &= f_G(\{\mathbf{h}_i^{(T)}\}_{i\in V_2}) \\
s &= f_s(\mathbf{h}_{G_1}, \mathbf{h}_{G_2}).
\end{aligned}
$$

知乎 @张楚珩

最关键的是其中的第二个式子，即如何产生两个图之间的匹配并且在图之间传递信息。这一步的具体产生形式如下

$$
\begin{aligned}
a_{j\to i} &= \frac{\exp(s_h(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}))}{\sum_{j'} \exp(s_h(\mathbf{h}_i^{(t)}, \mathbf{h}_{j'}^{(t)}))}, \\
\boldsymbol{\mu}_{j\to i} &= a_{j\to i}(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)})
\end{aligned}
$$

注意到

$$
\sum_j \boldsymbol{\mu}_{j\to i} = \sum_j a_{j\to i}(\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}) = \mathbf{h}_i^{(t)} - \sum_j a_{j\to i}\mathbf{h}_j^{(t)}.
$$

因此另一个图上所有节点传递给这个图上 i 节点的总的信息效果是该节点表示 $h_i^{(t)}$ 和另一个图上与该节点最相似节点的差别（ $a_{j\to i}$ 可以看做是softmax）

另外注意到，如果两个图是完全一样的，那么图之间的信息时时刻刻都是0，这样相当于图之间没有耦合。

相比于GNN每轮传播为 $O(|V|+|E|)$ 的复杂度，该方法复杂度为 $O(|V_1||V_2|)$ 。

4. 损失函数定义

定义了网络模型之后，整个网络都是可以求导的，因此，接下来我们只需要定义一个损失函数就可以对于网络进行训练了。损失函数的定义可以基于pair或者triplet。基于pair的方案如下，给定两个图和与之对应的二分类标签（similar/dissimilar），把两个图传过神经网络之后得到损失函数；基于triplet的方案如下，给定三个图和对应的标签，即第一个图是和第二个图更相似还是和第三个图更相似。

对于图的embedding是实向量的情况，可以定义margin-based pairwise loss

$$L_{\text{pair}} = \mathbb{E}_{(G_1,G_2,t)}[\max\{0, \gamma - t(1 - d(G_1, G_2))\}]$$

其中 $t \in \{-1, +1\}$ 是标签， $d(G_1, G_2) = \|h_{G_1} - h_{G_2}\|^2$ 是图embedding之间的欧氏距离。对于相似的对，它会鼓励其距离小于 $1-\gamma$ ；对于不相似的对，它会鼓励其距离大于 $1+\gamma$ 。

margin-based triplet loss定义如下

$$L_{\text{triplet}} = \mathbb{E}_{(G_1,G_2,G_3)}[\max\{0, d(G_1, G_2) - d(G_1, G_3) + \gamma\}].$$

它要求更为相似的对之间的距离至少比另一对之间的距离小 γ 。

对于图的embedding是binary向量的情况，即 $h_G \in \{-1, 1\}^H$ ，可以直接计算它们之间的Hamming distance并且最大或者最小化该距离。

$$\begin{aligned} L_{\text{pair}} &= \mathbb{E}_{(G_1,G_2,t)}[(t - s(G_1, G_2))^2]/4, \quad \text{and} \\ L_{\text{triplet}} &= \mathbb{E}_{(G_1,G_2,G_3)}[(s(G_1, G_2) - 1)^2 + \\ &\qquad (s(G_1, G_3) + 1)^2]/8, \end{aligned}$$

这里使用的是近似的Hamming distance

$$s(G_1, G_2) = \frac{1}{H} \sum_{i=1}^{H} \tanh(h_{G_1 i}) \cdot \tanh(h_{G_2 i})$$

文章发现这种方法更为稳定。

实验

文章主要做了两个实验。

第一个实验是人工生成的graph之间的比较，给定 n 个节点和节点之间连边的概率 p ，随机生成一个图 G_1 ，随机替换 k_p 条边生成正样本 G_2 ，随机替换 k_n 条边生成负样本 G_3 ，其中 $k_p < k_n$ 。然后在这些样本上学习（究竟需要多少个pair才能学习出来，找了半天也没找到）。实验结果显示，GNN和GMN的结果都不错，其中GMN效果更好，与之比较的是baseline方法是专门检测isomorphism的方法。

Graph Distribution	WL kernel	GNN	GMN
$n = 20, p = 0.2$	80.8 / 83.2	88.8 / 94.0	**95.0 / 95.6**
$n = 20, p = 0.5$	74.5 / 78.0	92.1 / 93.4	**96.6 / 98.0**
$n = 50, p = 0.2$	93.9 / **97.8**	95.9 / 97.2	**97.4** / 97.6
$n = 50, p = 0.5$	82.3 / 89.0	88.5 / 91.0	**93.8 / 92.6**

Table 1. Comparing the graph embedding (GNN) and matching (GMN) models trained on graphs from different distributions with the baseline, measuring pair AUC / triplet accuracy ($\times 100$).

知乎 @张楚珩

文章还给出了学到的GMN里面图之间的attention $a_{i\to j}$ ，可以看到相似的节点之间attention更强，同时度数（degree）高的节点容易产生更强的attention。

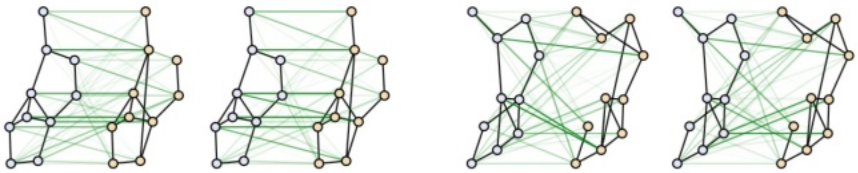

Figure 3. Visualization of cross-graph attention for GMNs after 5 propagation layers. In each pair of graphs, the left figure shows the attention from left graph to the right, the right figure shows the opposite.

知乎 @张楚珩

第二个实验是检查函数流程图之间的相似性，给定一系列函数（文章使用了开源软件ffmpg中的各个函数）不同的编译器和编译指令编译出来会得到不同的汇编命令流程图，相同函数编译出来的不同流程图应该被看做是“类似”的，而不同函数编译出来的应该被看做是“不同”的。下图展示了该实验的结果，可以看到传播的轮数 T 越大，效果越好；并且GMN（matching）效果优于GNN（embedding）。

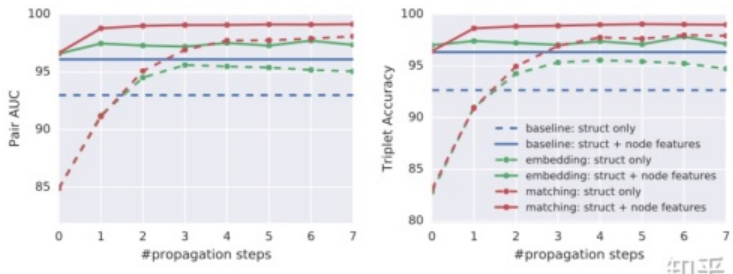

Figure 4. Performance ($\times 100$) of different models on the binary function similarity search task.

知乎 @张楚珩

发布于 2019-05-08

强化学习 (Reinforcement Learning)　深度学习（Deep Learning）

▲ 赞同 19　▼　8 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