论文阅读报告

目录

1. 基本信息

标题: Co-Regularized Deep Multi-Network Embedding
链接: https://dl.acm.org/citation.cfm?id=3186113
作者: Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, Xiang Zhang
作者单位: College of Information Sciences and Technology, Pennsylvania State University, IBM T. J. Watson Research Center Department of Biomedical Engineering, Pennsylvania State University, NEC Laboratories America
文章来源: WWW 2018
关键词: Multi-network; Network embedding; Representation learning
问题: 论文提出了一种多网络的网络嵌入的方法。

2. 论文分析

2.1 论文模型

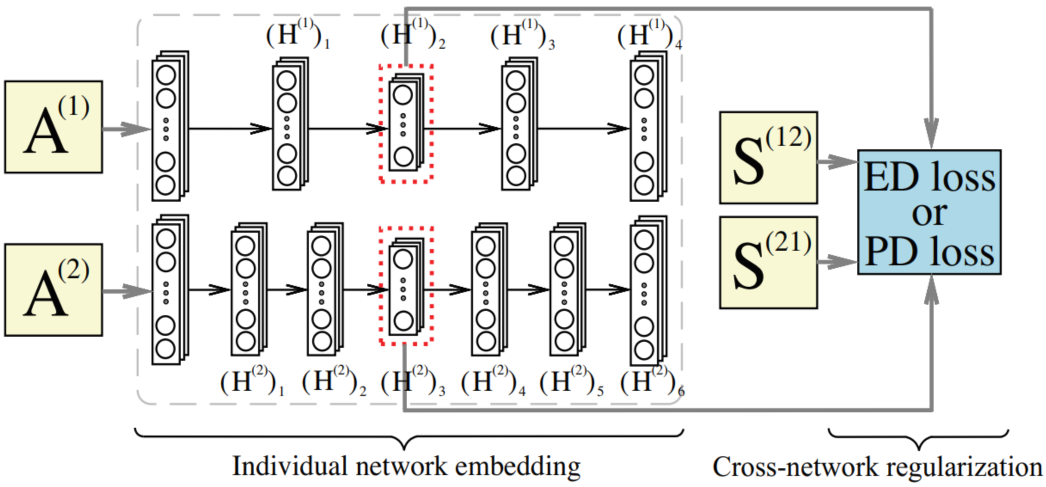

图 1 模型

上图显示的是两个网络时的情况，首先计算单个网络的嵌入，$(H^{(1)})_2$和$(H^{(2)})_3$分别是网络嵌入后的向量集，然后根据网络与网络的边集合 $S^{(12)}$和 $S^{(21)}$计算网络与网络的嵌入，根据 ED loss 和 PD loss 来修正求得的网络中点的向量。

2.2 相关研究与工作

1. Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. 15, 6 (2003), 1373–1396.
2. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In KDD. ACM, 701–710.
3. Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In WWW. International World Wide Web Conferences Steering Committee, 1067–1077.

4. Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In KDD. ACM, 855–864.

总结：上述的多种方案，都是对于单个网络的解决方案，例如 2 中的 deepwalk 方法是通过沿边随机取点来取“邻居”，然后通过“邻居”来嵌入结点。3 中的 LINE 模型，前期报告我做过详细解释了，4 中的 node2vec 其实是对 deepwalk 的衍生，方式在于取“邻居”这个过程发生了变化，通过深度优先和广度优先的遍历方式来取得“邻居”，但是这几种模型都缺乏对于多个网络的解决方案。

3. 论文系统分析

论文做的工作是处理多个连接的网络的 embedding，将多个网络切分成单个网络（通过结点类型切分），对于单个网络首先进行 embedding，求得所有结点的向量，然后处理网络与网络间的边，通过给出定义的损失函数，对已求得的结点的向量进行修正。

3.1 创新点

论文的创新点在于能够处理边类型复杂的多网络（包括是否有向，是否有权重），网络之间的边的关系可以是多对多的，有权重的，论文也提出了一种有效的提升方法，能够处理大规模的数据，论文实验表明，论文的嵌入方法相比于其他方法，准确率有提高。

3.2 实验过程

3.2.1 结点“邻居”选取

Symbol	Meaning
g	The number of networks
n_i	The number of nodes in the i-th network
d_i	The dimensionality of the i-th embedding space
L_i	The number of neural net. layers for the i-th net. data
$\mathbf{G}^{(i)}$	The adjacency matrix of the i-th network
$\mathbf{A}^{(i)}$	The structural context matrix of the i-th network
$\mathbf{S}^{(ij)}$	The relationship matrix between nodes in $\mathbf{G}^{(i)}$ and $\mathbf{G}^{(j)}$
$\tilde{\mathbf{S}}^{(ij)}$	The row-normalized version of $\mathbf{S}^{(ij)}$
$\mathbf{U}^{(i)}$	The embedding matrix of the i-th network
$\{\mathbf{W}_l^{(i)}\}_{l=1}^{L_i}$	The weight matrices for the i-th network data
$\{\mathbf{b}_l^{(i)}\}_{l=1}^{L_i}$	The bias vectors for the i-th network data
$\boldsymbol{\theta}^{(i)}$	The model parameters $\boldsymbol{\theta}^{(i)} = \{\mathbf{W}_l^{(i)}, \mathbf{b}_l^{(i)}\}_{l=1}^{L_i}$
$\mathcal{I}$	The set of cross-network relationships

图 2 记号图

上图是会使用到的一些标记

结点的“邻居”选取方式为 random walk with restart（RWR）。在网络 G 中，假设初始结点为 x，我们选取 RWR 的步长为 3，$P_y^{(k)}$表示经过 k 步到达 y 结点的概率。$P^{(0)}$向量为 one-hot 形式，$P_x^{(0)}$为 1，其他都为 0。逐步推导的公式为：

$$\mathbf{p}^{(k)} = c\mathbf{p}^{(k-1)}[(\mathbf{D})^{-1}\mathbf{G}] + (1-c)\mathbf{p}^{(0)}$$

在其中，矩阵 D 为一个对角矩阵，$D_{xx} = \sum_{y=1}^{n} G_{xy}$，c 表示从结点 x 重新出发的概率。对于结点 x 的“邻居”结点，我们定义为：

$$\mathbf{a} = \sum_{k=1}^{K} \mathbf{p}^{(k)}$$

其中 K 的值是个小数值的整型，一般选取为 3。把所有结点的 a，即“邻居”值计算出来，保存在矩阵 A 中，每一行代表一个结点的 a。$A^{(i)}$代表网络 i 的所有结点的 a 值。

3.2.2 单个网络嵌入

对于单个网络的嵌入，输入单个网络的 $A^{(i)}$矩阵，整个神经网络包括 L_i+1 层，

前 L_i/2 层编码，得到一个压缩过的向量，后 L_i/2 解压，得到一个和压缩前相同

维度的向量。输入 $(\mathbf{h}_x^{(i)})_0 = \mathbf{A}_{x*}^{(i)} \in \mathbb{R}^{1\times n_i}$作为第一层，每一层的压缩和解压公式为：

$$(\mathbf{h}_x^{(i)})_l = \sigma((\mathbf{h}_x^{(i)})_{l-1}\mathbf{W}_l^{(i)} + \mathbf{b}_l^{(i)}) \in \mathbb{R}^{1\times k_l}$$

K_l 表示每一层输出的维度，$\mathbf{W}_l^{(i)} \in \mathbb{R}^{k_{l-1}\times k_l}$和$\mathbf{b}_l \in \mathbb{R}^{1\times k_l}$是 l 层的参数。计算输入向量和重构后的向量的差值，调整参数，公式如下：

$$\min_{\theta^{(i)}} \mathscr{L}_{ae}^{(i)} = \|\mathbf{A}^{(i)} - \hat{\mathbf{A}}^{(i)}\|_F^2 + \lambda \sum_{l=1}^{L_i} \|\mathbf{W}_l^{(i)}\|_F^2$$

$\hat{\mathbf{A}}^{(i)}$同$\mathbf{A}^{(i)}$类似，是由$(\mathbf{h}_x^{(i)})_{L_i}$组成的矩阵。

最终取$(\mathbf{h}_x^{(i)})_{L_i/2}$为我们想要的 i 网络的 x 结点的向量。

3.2.3 多个网络嵌入

现已获得单个网络嵌入的向量，再利用网络与网络之间的关系去修正结点的向量，有两种损失函数去修正，一种是 ED loss，另一种是 PD loss，ED loss 要求不同网络之间的向量维度是一致的，而 ED loss 就没有这种要求。

ED loss:

我们用 $h_x^{(i)}$来代表网络 $G^{(i)}$中 x 结点的向量，即 $\mathbf{h}_x^{(i)} = (\mathbf{h}_x^{(i)})_{L_i/2}$，如果网络 $G^{(i)}$中 x 结点与网络 $G^{(j)}$中 y 结点有边相连，那么它们的向量应该更加接近，记$\mathcal{N}^{(i\to j)}(x)$为网络 $G^{(j)}$中与网络 $G^{(i)}$中的 x 结点相连的所有结点的集合，引出下面的损失函数：

$$\mathscr{L}_x = \|\mathbf{h}_x^{(i)} - \mathbf{h}_x^{(i\to j)}\|_F^2$$

其中:

$$\mathbf{h}_x^{(i\to j)} = \frac{1}{\sum_{y\in\mathcal{N}^{(i\to j)}(x)} \mathbf{S}_{xy}^{(ij)}} \sum_{y\in\mathcal{N}^{(i\to j)}(x)} \mathbf{S}_{xy}^{(ij)}\mathbf{h}_y^{(j)}$$

实质是对所有在网络 $G^{(j)}$中与网络 $G^{(i)}$中 x 结点相连的结点做了一个加权平均，$\mathbf{S}_{xy}^{(ij)}$代表网络 $G^{(i)}$中 x 结点与网络 $G^{(j)}$中 y 结点的相连的边的权重。

这是与网络 $G^{(i)}$中一个结点 x 的损失函数，下面我们计算整个网络 $G^{(i)}$与整个网络 $G^{(j)}$的损失函数：

$$\mathscr{L}_{ed}^{(ij)} = \|\mathbf{O}^{(ij)}\mathbf{H}^{(i)} - \tilde{\mathbf{S}}^{(ij)}\mathbf{H}^{(j)}\|_F^2$$

$\tilde{\mathbf{S}}_{xy}^{(ij)}$ 定义如下：

$$\tilde{\mathbf{S}}_{xy}^{(ij)} = \frac{\mathbf{S}_{xy}^{(ij)}}{\sum_{z=1}^{n_j} \mathbf{S}_{xz}^{(ij)}}$$

$\tilde{\mathbf{S}}^{(ij)}$ 是 $\tilde{\mathbf{S}}_{xy}^{(ij)}$ 构成的矩阵，就是将 $\mathbf{S}^{(ij)}$ 按照行标准化了。$\mathbf{H}^{(i)} = [(\mathbf{h}_1^{(i)})^T, \ldots, (\mathbf{h}_{n_i}^{(i)})^T]^T \in \mathbb{R}^{n_i\times d_i}$代表网络 $G^{(i)}$，矩阵 O 是一个对角指示矩阵，如果$\tilde{\mathbf{S}}^{(ij)}$x 行全为 0，即代表 x 结点与网络 $G^{(j)}$没有相连的边，所以$\mathbf{O}_{xx}^{(ij)} = 0$，否则$\mathbf{O}_{xx}^{(ij)} = 1$。

PD loss:

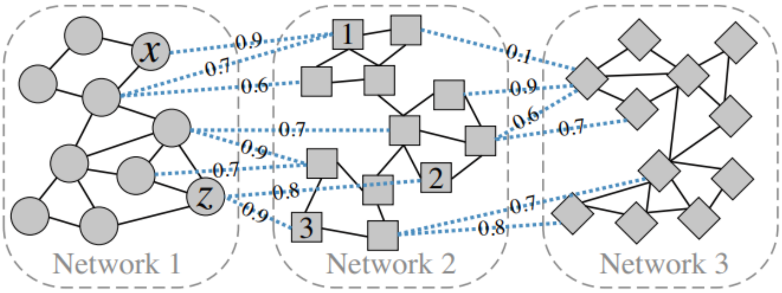

图 3 示例图

在 ED loss 中直接将网络 $G^{(i)}$ 中 x 结点的向量 $h_x^{(i)}$ 与结点 x 在网络 $G^{(j)}$ 中相连的结点的向量加权平均值去求相似度，这样做的问题是当在网络 $G^{(j)}$ 中与 x 相连的结点并不靠近时，效果不好。所以我们不再直接比较网络 $G^{(i)}$ 中 x 结点与在其他网络中与 x 相连的结点的平均值。我们比较在同一网络中的两个点的相似值和这两个点在其他网络中(单个)所连的结点的平均值的相似值。例如在上图 3-示例图中，如果 network2 中结点 1 和结点 2，3 的平均值很接近，那么 network1 中的 x 结点和 z 结点的就应该很接近。所以可得损失函数为：

$$\mathcal{L}_{pd}^{(ij)} = \sum_{x=1}^{n_i}\sum_{z=1}^{n_i}[\mathbf{h}_x^{(i)}(\mathbf{h}_z^{(i)})^T - \mathbf{h}_x^{(i\to j)}(\mathbf{h}_z^{(i\to j)})^T]^2$$
$$= \|\mathbf{O}^{(ij)}\mathbf{H}^{(i)}(\mathbf{O}^{(ij)}\mathbf{H}^{(i)})^T - \tilde{\mathbf{S}}^{(ij)}\mathbf{H}^{(j)}(\tilde{\mathbf{S}}^{(ij)}\mathbf{H}^{(j)})^T\|_F^2$$

3.2.4 联合训练

我们已经得到求单个网络的方法和处理网络与网络之间的边的方法，作者又提出了一种加速训练的方式，用$\{\mathbf{U}^{(i)}\}_{i=1}^g$来代替$\{\mathbf{H}^{(i)}\}_{i=1}^g$，那么 ED loss 和 PD loss 就会变为如下形式：

$$\mathcal{L}_{ed}^{(ij)} = \|\mathbf{O}^{(ij)}\mathbf{U}^{(i)} - \tilde{\mathbf{S}}^{(ij)}\mathbf{U}^{(j)}\|_F^2$$
$$\mathcal{L}_{pd}^{(ij)} = \|\mathbf{O}^{(ij)}\mathbf{U}^{(i)}(\mathbf{O}^{(ij)}\mathbf{U}^{(i)})^T - \tilde{\mathbf{S}}^{(ij)}\mathbf{U}^{(j)}(\tilde{\mathbf{S}}^{(ij)}\mathbf{U}^{(j)})^T\|_F^2$$

添加一项正则化项，确保$\{\mathbf{U}^{(i)}\}_{i=1}^g$和$\{\mathbf{H}^{(i)}\}_{i=1}^g$相似，正则项如下：

$$\mathcal{L}_{hu}^{(i)} = \|\mathbf{U}^{(i)} - \mathbf{H}^{(i)}\|_F^2$$

损失函数总计为：

$$\min_{\{\theta^{(i)},\mathbf{U}^{(i)}\}_{i=1}^{g}} \mathscr{L} = \sum_{i=1}^{g} \mathscr{L}_{ae}^{(i)} + \alpha \sum_{(i,j)\in\mathcal{I}} \mathscr{L}_{R}^{(ij)} + \beta \sum_{i=1}^{g} \mathscr{L}_{hu}^{(i)}$$

其中$\mathscr{L}_{ae}^{(i)}$代表的单个网络的损失函数，在单个网络嵌入那一部分已经定义过了，$\mathscr{L}_{R}^{(ij)}$可以为$\mathscr{L}_{ed}^{(ij)}$或者是$\mathscr{L}_{pd}^{(ij)}$。算法如下图所示：

Algorithm 1: Deep multi-network embedding (DMNE)

Input: Networks $\{\mathbf{A}^{(i)}\}_{i=1}^{g}$, cross-network relationships $\{\mathbf{S}^{(ij)}\}_{(i,j)\in\mathcal{I}}$, parameters α, β and λ

Output: $\{\mathbf{U}^{(i)}\}_{i=1}^{g}$ and $\{\theta^{(i)}\}_{i=1}^{g}$

```
Pretrain neural network for each A^(i) to obtain initial {θ^(i)}_{i=1}^g and
{H^(i)}_{i=1}^g, initialize each U^(i) = H^(i);
repeat
    for i ← 1 to g do
        Back-propagation to update θ^(i) based on Eq. (15);
    end
    for i ← 1 to g do
        Forward propagation to obtain H^(i);
        Update U^(i) by Eq. (13) or Eq. (14);
    end
until Convergence
return {U^(i)}_{i=1}^g and {θ^(i)}_{i=1}^g.
```

图 4 算法图

3.3 实验结果分析

实验在多个数据集采用多种方式比较，数据集有如下图 5 所示的 5 类多网络数据集，对比方法有：LE，Spectral，DeepWalk，LINE，GraRep，node2vec，DNRG，AE，DMNE(ED)，DMNE(PD)。

Dataset	# Networks	# Nodes	# Links	# CrossLinks	LabeledNet.	# LabeledNodes	# Classes
6-NG	5	4,500	16,447	66,756	All	4,500	6
9-NG	5	6,750	24,778	100,585	All	6,750	9
DP-NET	2	13,583	51,918	2,107	Disease	675	18
DBIS	2	24,535	85,184	38,035	Collaboration	2,890	4
CiteSeer-M10	3	15,533	56,548	11,828	Collaboration	3,284	10

图 5 数据集图

实验将多种方式求得的带标签的结点的向量，放入一个 SVM 分类器中，进行训练和测试，实验结果如下图 6 所示：

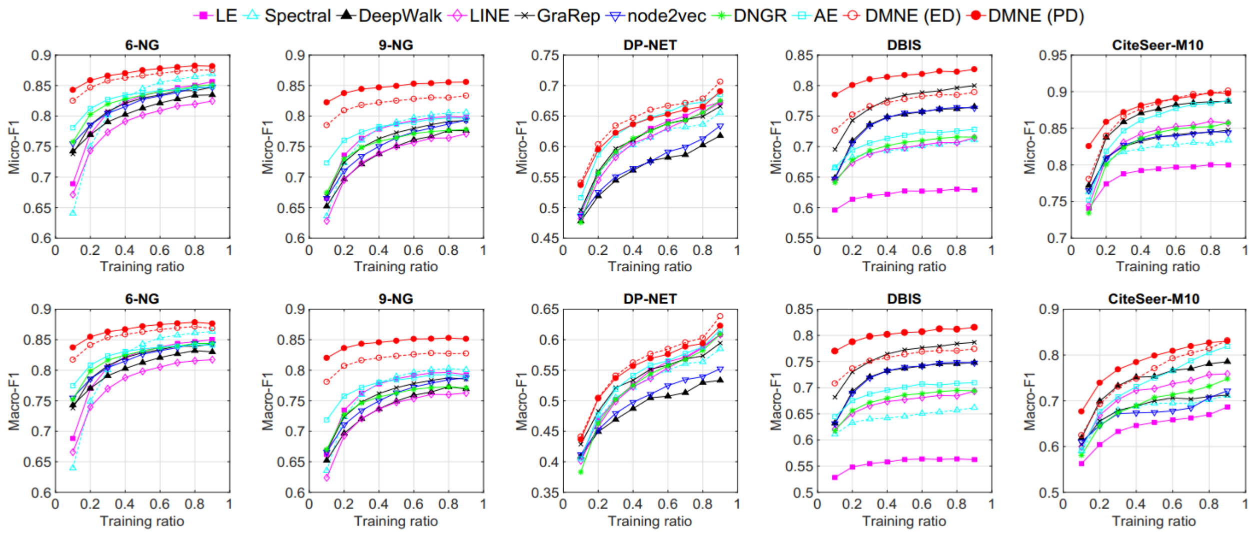

图 6 实验结果图

实验结果分析如下：

1）DMNE(ED)和 DMNE(PD)在绝大多数数据集上的表现要远超其他方法，因为利用了网络与网络间的边来修正结点的向量，证明了论文中提出的方法的优越性。

2）当切分较小的训练集时，DMNE(ED)和 DMNE(PD)表现得更加优秀，说明论文中的方法更加具有实用性，因为实际应用中未打标签的数据占大多数

3）DMNE(PD)在绝大多数情况下表现的比 DMNE(ED)优秀。因为 PD 可以让多个网络的向量维度不同，而不是像 ED 一样强制要求相同。

4. 总结与展望

应用领域：

深度多网络嵌入

不足：

1.实验只进行到 10 万结点这个量级，但大多数现实情况下，结点数都高达百万，尚未知晓对于百万这个量级，论文方法的效果和效率。

2.计算相对复杂，时间耗费较多，相比于 LINE 等模型，鲁棒性更差。

后续工作：

多数 network embedding 的方法实质都只利用到了边的信息，结点并未承载任何信息，而边承载的信息也很少(权重，有向无向)，现在的 network embedding 方法对于图信息的嵌入是不足够的，丢失很多信息，应用面不多，

后期工作看一下是不是能够利用起结点的信息(即属性)，这一块的前途比较广阔。