# 【Embedding】Metapath2vec：异构网络表征

原创 阿泽crz 阿泽的学习笔记 2020-06-02

今天学习的是微软的一篇论文《metapath2vec: Scalable Representation Learning for Heterogeneous Networks》，发表于 KDD 2017，目前引用次数超 500 次。

很多网络表征学习主要是针对同构网络的，而本文提出的一种专门用于异构网络表征学习的方法——Metapath2Vec，其能够同时捕捉不同类型节点之间的**「结构关系」**和**「语义关系」**。

Metapath2Vec 使用基于元路径的随机游走方法来捕捉节点的异构邻居，然后使用异构 Skip-Gram 模型进行训练，同时建模结构上和语义上相近的节点。

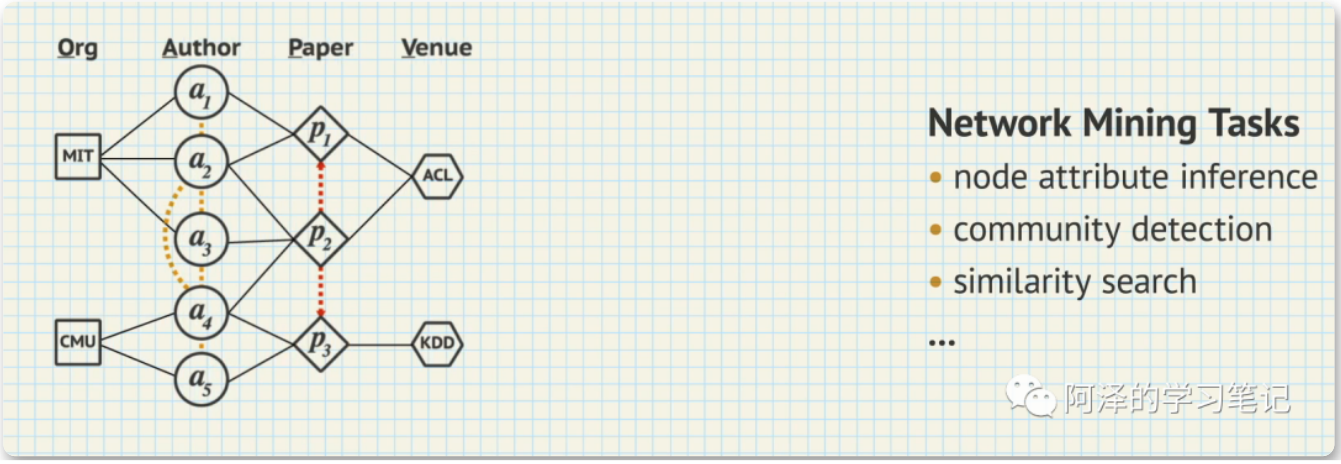
此外，作者还提出了 Metapath2Vec++ 方法，针对每种类型的节点进行单独归一化，即把异构网络分解成不同的同构网络。

最终实验表明，这两种 Metapath2Vec 方法不仅在异构网络挖掘任务中取得了 SOTA 的成绩，而且还能够识别不同网络对象之间的结构和语义关系。

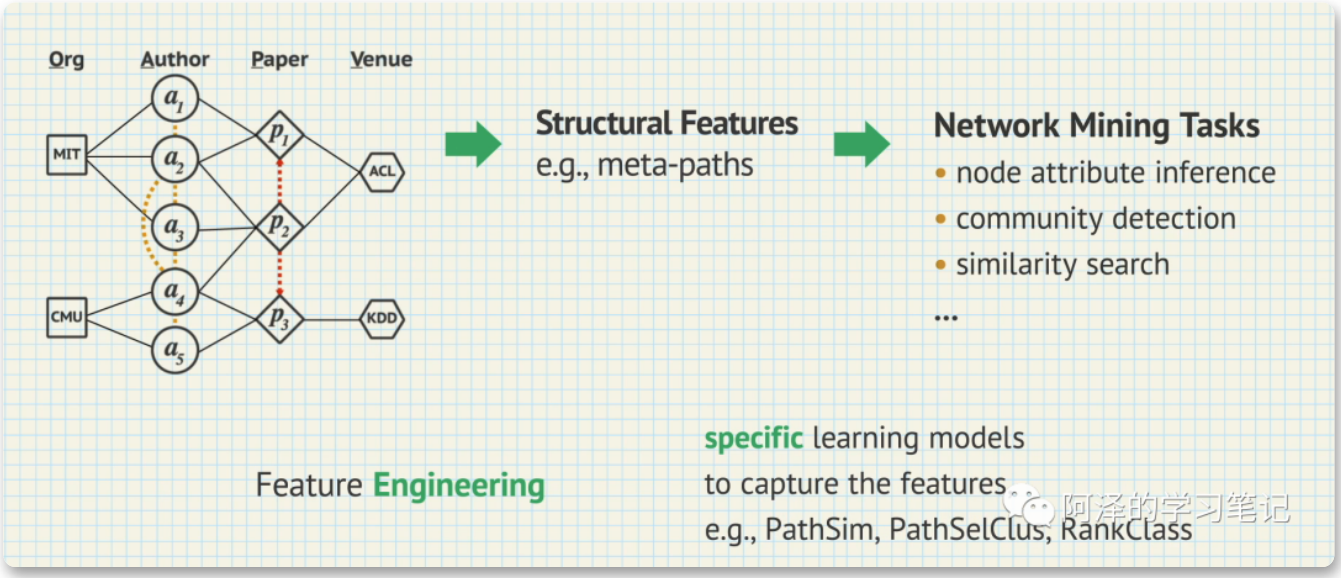
## 1.Introduction

目前大部分的工作都集中在同构网络中，但真实场景下异构网络才是最常见的。针对同构网络设计的模型很多都没法应用于异构网络，比如说，对于一个学术网络而言：如何高效根据上下文信息表征不同类型的节点？能否用 Deepwalk 或者 Node2Vec 来学习网络中的节点？能否直接将应用于同构网络的 Embedding 模型直接应用于异构网络？

解决诸如此类的挑战，有利于更好的在异构网络中应用多种网络挖掘任务：

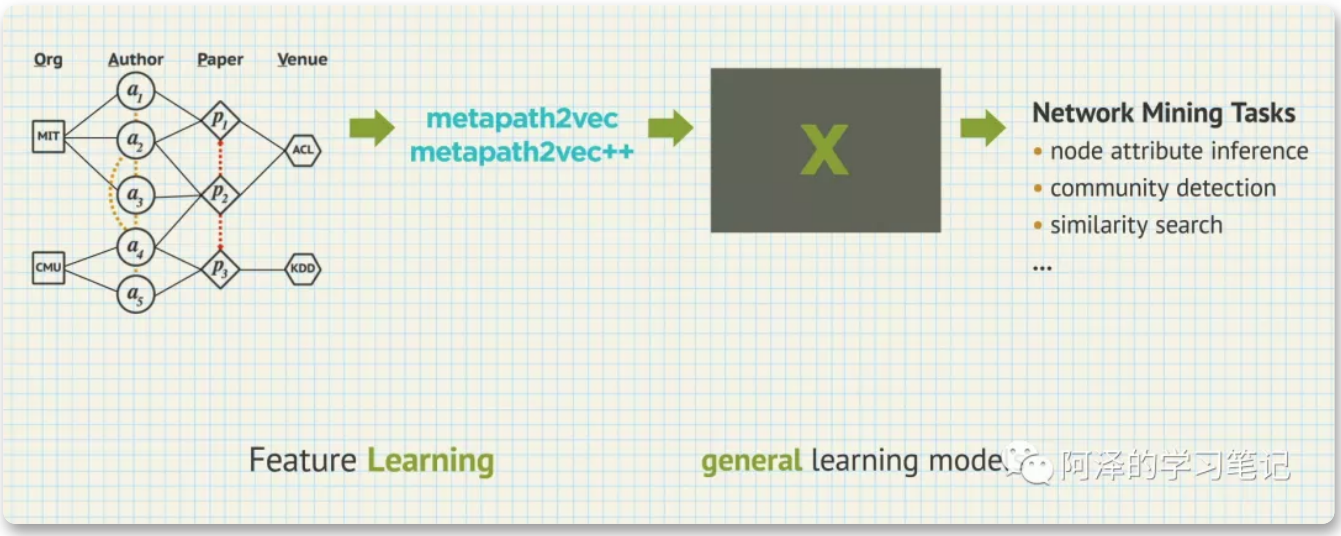


传统的方法都是基于结构特征（如元路径 meta-path）来求相似性，类似的方法有 PathSim、PathSelClus、RankClass 等：




但这种方式挖掘出来的元路径（如 “APCPA”）经常会出现相似度为 0 的情况。如果我们能够将 Embedding 的思想应用于异构网络，则不会再出现这种情况。

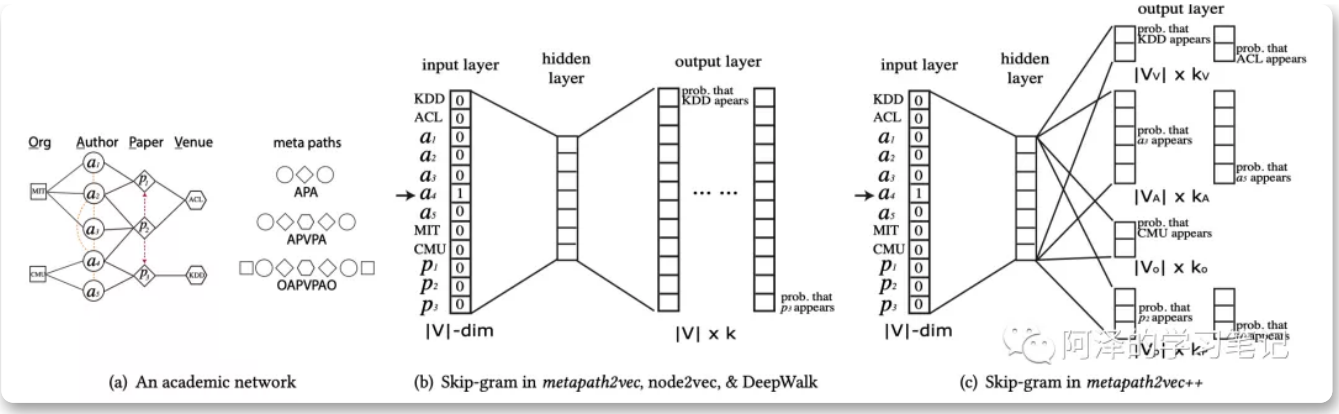
基于这种观察，作者提出了两个可以应用于异构网络的 Graph Embedding 的算法模型——metapath2vec 以及 metapath2vec++。




## 2.Metapath2Vec

为了对异构网络节点中的邻居进行建模，metapath2vec 引入了异构 skip-gram 模型。此外，为了捕获异构网络的结构，作者还提出了基于元路径的随机游走策略。

先给出流程图：




## 2.1 Meta-Path-Based Random Walks

Metapath2vec 同构 metapath 来指导随机游走的节点跳转。给出元路径模式 $\mathcal{P}$:

$$\mathcal{P} = V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$$

其中，节点类型间的关系表示:

$$R = R_1 \circ R_2 \circ \cdots \circ R_{l-1}$$

“APA” 关系表示两位作者（A）在一篇论文（P）上的合著关系；“APVPA” 表示两位作者(A)在同一会议（V）发表过论文(P)。这种元路径有利于异构网络的数据挖掘。

基于元路径模式 $\mathcal{P}$，我们给出转移概率:

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

其中 $v_t^i \in V_t$，$N_{t+1}(v_t^i)$ 表示节点 $v_t^i$ 的邻居中属于 $V_{t+1}$ 类型的节点集合。

也就是说，游走是在预先设定的 meta-path $\mathcal{P}$ 的条件上。通常 meta-path 一般用在对称的路径上，第一个节点类型与最后一个节点类型相同，例如 OAPVPAO。

$$p\left(v^{i+1}|v_t^i\right) = p\left(v^{i+1}|v_1^i\right) \quad if\ t = l$$

## 2.2 Heterogeneous skip-gram

对于每个节点 v，根据其不同类型的上下文最大化其上下文:

$$\arg\max_{\theta} \sum_{v \in V} \sum_{t \in T_V} \sum_{c_t \in N_t(v)} \log p(c_t|v;\theta) p(c_t|v;\theta) = \frac{e^{X_{c_t} X_v}}{\sum_{u \in V} e^{X_u X_v}}$$

其中，V 表示网络的节点集合；$T_V$ 表示节点类型的集合；$N_t(v)$ 表示节点 v 的类型为 t 的邻居集合。$X_v$ 表示节点 v 的 Embedding 向量。

考虑负采样的目标函数：

$$log\ \sigma(X_{c_t} \cdot X_v) + \sum_{m=1}^{M} \mathbb{E}_{u^m \sim P(u)}[log\ \sigma(-Xu^m \cdot X_v)]$$

其中，$P(u)$ 是负采样中样本的预定义分布；metapath2vec 通过均匀地观察不同类型的节点并绘制(负)节点来维护一个节点频率分布。

相比于考虑负采样的 Skip-gram 的目标函数而言并无本质区别，唯一的区别在于采样的策略上发生了变换。

##2.3 Metapath2Vec++

Metapath2Vec 在计算 Softmax 时不考虑节点的类型。Metapath2Vec++ 在采集负样本时，考虑样本与正样本属于同一个节点类型。也就是**「异构负采样（Heterogeneous negative sampling)」**。

考虑条件概率 p 在特定的节点类型 t 上做标准化：

$$p(c_t|v;\theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u_t \in V_t} e^{X_{u_t} \cdot X_v}}$$

此时，目标函数为：

$$log\ \sigma(X_{c_t} \cdot X_v) + \sum_{m=1}^{M} \mathbb{E}_{u_t^m \sim P_t(u_t)}[log\ \sigma(-X_{u_t^m} \cdot X_v)]$$

与 Skip-gram 没有本质区别，但异构网络的**「异构」**信息不仅仅在采样中体现出来，也在目标函数中被体现出来。

来看下伪代码：

**Input:** The heterogeneous information network $G = (V, E, T)$, a meta-path scheme $\mathcal{P}$, #walks per node $w$, walk length $l$, embedding dimension $d$, neighborhood size $k$

**Output:** The latent node embeddings $\mathbf{X} \in \mathbb{R}^{|V| \times d}$

initialize $\mathbf{X}$ ;
**for** $i = 1 \rightarrow w$ **do**
  **for** $v \in V$ **do**
    $MP$ = MetaPathRandomWalk($G$, $\mathcal{P}$, $v$, $l$) ;
    $\mathbf{X}$ = HeterogeneousSkipGram($\mathbf{X}$, $k$, $MP$) ;
  **end**
**end**
return $\mathbf{X}$ ;

**MetaPathRandomWalk**($G$, $\mathcal{P}$, $v$, $l$)
$MP[1] = v$ ;
**for** $i = 1 \rightarrow l-1$ **do**
  draw u according to Eq. 3 ;
  $MP$[i+1] = $u$ ;
**end**
return $MP$ ;

**HeterogeneousSkipGram**($\mathbf{X}$, $k$, $MP$)
**for** $i = 1 \rightarrow l$ **do**
  $v = MP$[i] ;
  **for** $j = max(0, i\text{-}k) \rightarrow min(i+k, l)$ & $j \neq i$ **do**
    $c_t = MP$[j] ;
    $X^{new} = X^{old} - \eta \cdot \frac{\partial O(\mathbf{X})}{\partial X}$ (Eq. 7) ;
  **end**
**end**

**ALGORITHM 1:** The *metapath2vec++* Algorithm.



## 3.Experiment

简单看一下实验。

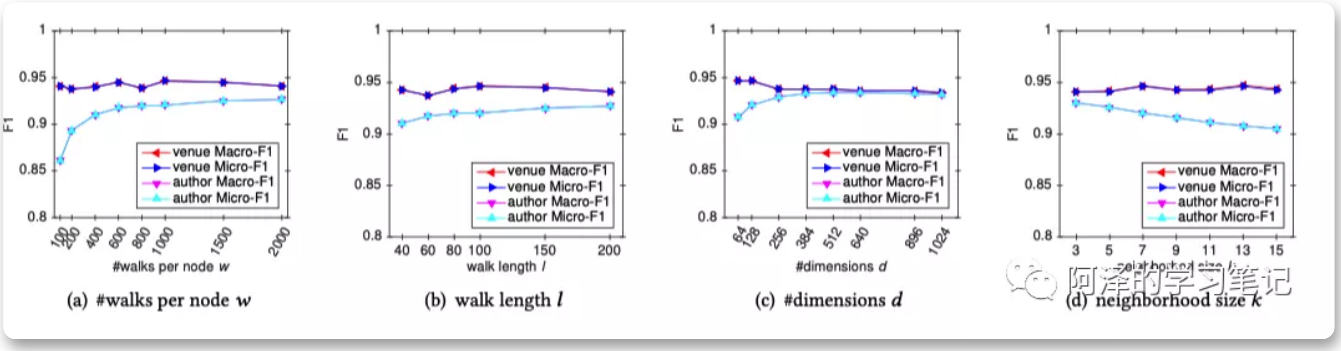
以 Aminer 数据集为例，“会议”节点节点分类的结果：（百分号为训练的数据集的占比）

| 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 |

“作者”节点分类的结果：

| 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 |

参数敏感性实验：



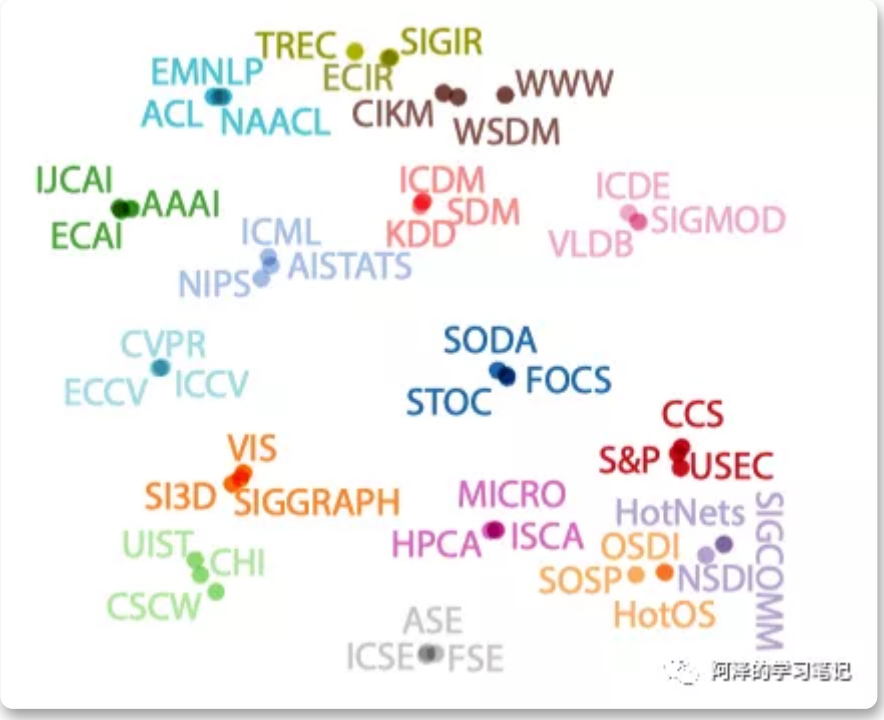

(a) #walks per node $w$ (b) walk length $l$ (c) #dimensions $d$ (d) neighborhood size $k$

节点分类结果：

| methods | venue | author |
|---|---|---|
| DeepWalk/node2vec | 0.1952 | 0.2941 |
| LINE (1st+2nd) | 0.8967 | 0.6423 |
| PTE | 0.9060 | 0.6483 |
| *metapath2vec* | 0.9274 | 0.7470 |
| *metapath2vec++* | 0.9261 | 0.7354 |



metapath2vec++ 聚类结果的可视化：




## 4.Conclusion

总结：本文定义了异构网络中表征学习问题，其存在不同类型的节点和边。为了应对异构网络所带来的挑战，作者提出了 Metapath2Vec 和 Metapath2Vec++ 两种算法。Metapath2Vec 首先**「基于元路径的引导进行随机游走」**并采集到相关序列，该能够捕捉到不同类型节点的关系结构和语义相关性。虽然，作者利用异构 Skip-gram 和异构负采样技术来学习节点的表征。Metapath2Vec++ 算法则是在计算 Softmax 时不考虑节点的类型。最终实验表明，这两种算法在异构网络中取得了不错的成绩。