

# 数据挖掘与应用

## 图数据挖掘

授课教师：周晟

浙江大学 软件学院

2021.11



## ① 图数据挖掘简介

## ② 图上的分类

## ③ 图上的聚类

## ④ 图上的异常检测



# 认识图数据

## 图的定义

图是一种描述样本间**关系**的通用语言。许多复杂的模式往往需要通过数据之间的结构关系进行分析。

## 图的数学定义

图一般定义为:  $G = V, E, X$ , 其中  $V$  是节点的集合,  $E$  是边的集合,  $X$  是节点属性的集合。

常见的复杂图:

- ① 有向图
- ② 异构图
- ③ 动态图
- ④ ...



# 认识图数据



计算机网络



社交网络



交通网络



神经元网络



卫星网络



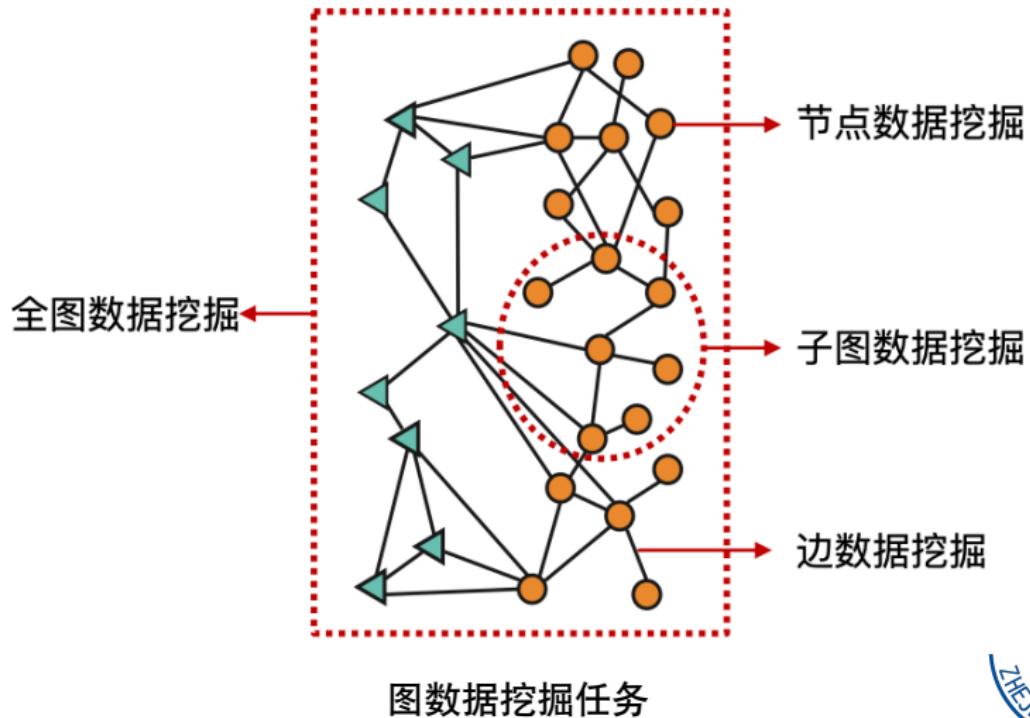
# 图数据挖掘的特点

## 图数据挖掘的主要特点

- ① 没有固定的大小和格式（输入数据多样化）
- ② 图上的节点没有固定的顺序或编号
- ③ 图上的结构特征难以显式定义
- ④ 图数据动态变化
- ⑤ 图数据可包含其他类型特征



# 图数据挖掘的经典任务



## 1 图数据挖掘简介

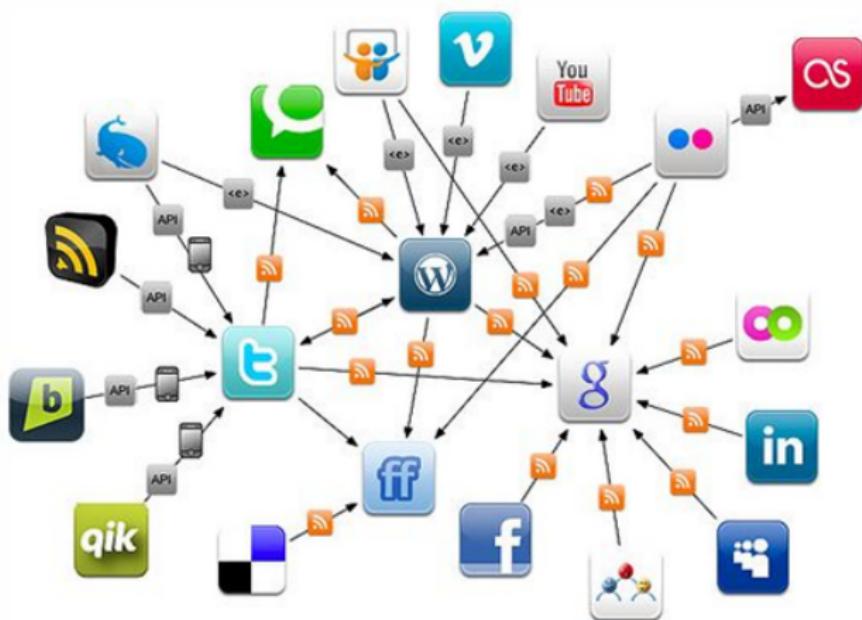
## 2 图上的分类

## 3 图上的聚类

## 4 图上的异常检测

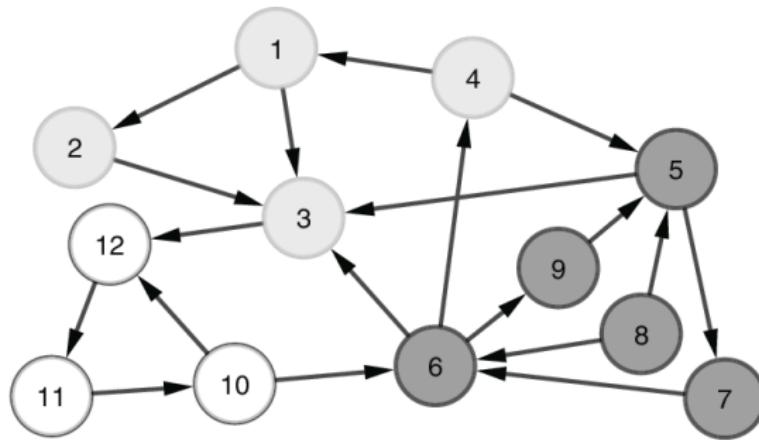


# PageRank 背景



如何分析网页的相关性和重要性？

# PageRank 背景



由页面构建的有向图

节点表示页面，边表示页面之间的跳转关系



# PageRank 基本假设

## PageRank 基本假设

网页中的每一次超链接（跳转）表示源页面（Source Page）到目标页面（Target Page）的一次投票（Vote），一个重要的页面往往能收到更多的投票。

直接计算每个页面的输入链接？



# PageRank 数学原理

## PageRank 数学原理

PageRank 本质上是一种稳态传播模型，它假设有向图上的每一条边，都表示节点重要性的流动，最终的节点重要性由稳态决定。

### 边的重要性

如果一个节点的重要性是  $r_i$ ，它有  $d_i$  条出度边，则每一条边上传播的重要性为  $\frac{r_i}{d_i}$ ，可以用矩阵  $M \in \mathbb{R}^{N \times N}$  表示，其中  $M_{ij} = \frac{1}{d_j}$

### 节点的重要性

每个节点的重要性  $r_i$  是所有入度边上传播的重要性的和，归一化后满足  $\sum_i r_i = 1$



# PageRank 数学原理

## 网络的稳态

当网络满足稳态时，节点的重要性满足：

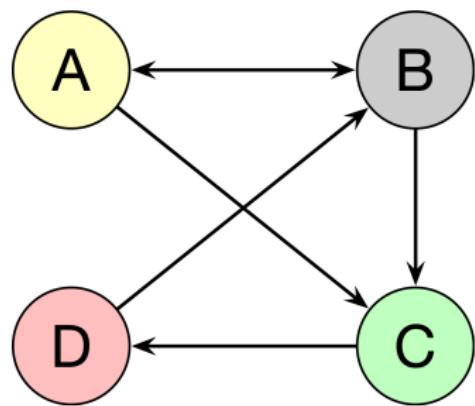
$$r = M \cdot r$$

排序向量  $r$  是节点之间转移概率矩阵  $M$  的特征向量（对应的特征值为 1）。

$M$  矩阵也成为随机邻接矩阵（Stochastic Adjacency Matrix）



## PageRank 数学原理



$$r_A = r_B/2$$

$$r_B = r_A/2 + r_D$$

$$r_C = r_A/2 + r_B/2$$

$$r_D = r_C$$

	A	B	C	D
A	0	$r_A/2$	$r_A/2$	0
B	$r_B/2$	0	$r_B/2$	0
C	0	0	0	$r_C$
D	0	$r_D$	0	0



# PageRank 数学原理

随机游走的网络爬虫：

- ①  $t$  时刻，爬虫处于第  $i$  个页面
- ②  $t+1$  时刻，爬虫从第  $i$  个页面的所有出边中随机选择一条边
- ③  $t+1$  时刻，爬虫从选择的这条边进行跳转，达到第  $j$  个页面
- ④ 爬虫进行无限次的上述操作

## 页面停留概率

给定随机游走的网络爬虫，定义  $t$  时刻  $N$  维页面停留概率  $p(t)$ ，第  $i$  维表示  $t$  时刻停留在第  $i$  个页面的概率。



# PageRank 数学原理

在稳定状态下，概率分布  $p(t)$  需满足：

$$p(t+1) = M \cdot p(t) = p(t)$$

$p(t)$  也称为随机游走的稳态分布 (Stationary Distribution of random walk)



# PageRank 缺陷

PageRank 是否一定会收敛？

PageRank 存在的缺陷：

- ① 部分节点没有出边
- ② 部分节点形成局部子图，且与网络的其他部分没有连接

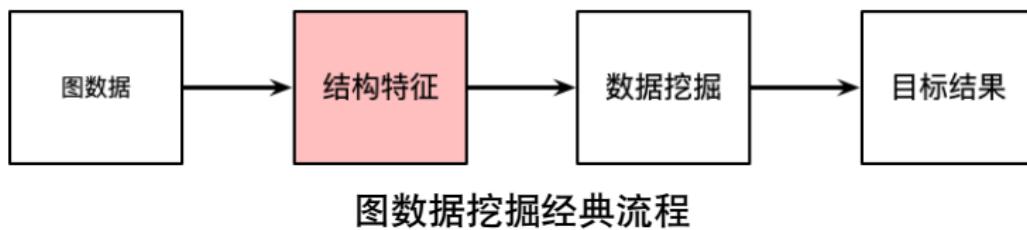
## 解决方案

在每一次随机游走的过程中，有一定的概率  $\beta$  继续从当前节点的出边中选择一条进行跳转，还有  $1 - \beta$  的概率随机跳转到任意页面。这种条件下节点的重要性定义为：

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

# 节点分类

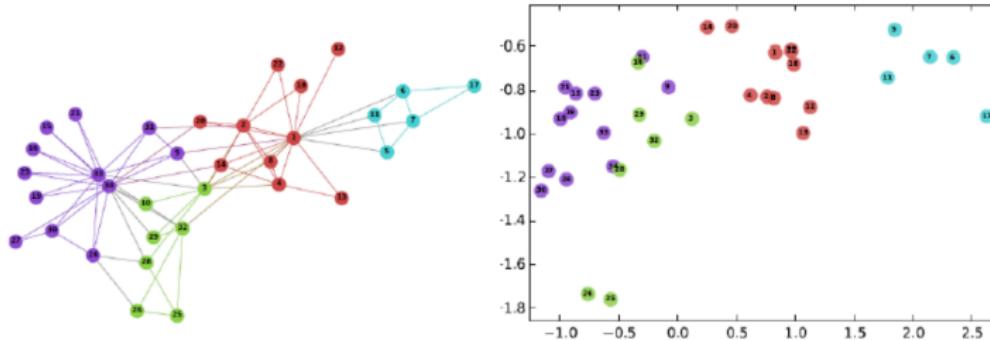
传统的分类方法需要数据的特征输入，但是图数据中节点仅有邻居信息，无法直接用于分类等下游任务。



如何构造有效的结构特征是图数据挖掘任务的关键



# 节点表征学习

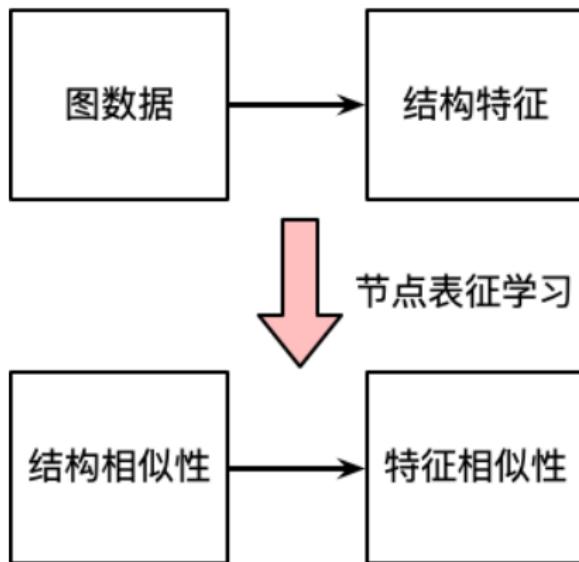


Zachary's Karate Club 网络节点表征学习

## 节点表征学习

节点级数据挖掘任务的核心是学习节点的有效低维表征，使得输入网络的结构信息能被保留到低维节点向量中。

# 节点表征学习



## 结构相似性

- ① 邻居关系
- ② 共享邻居关系 (Jaccard)
- ③ 最短路径
- ④ ...

## 特征相似性

- ① 向量点积
- ② 余弦距离
- ③ 分布相似性
- ④ ...



# 基于随机游走的节点表征学习

## 核心思想

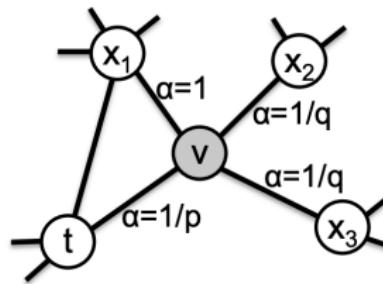
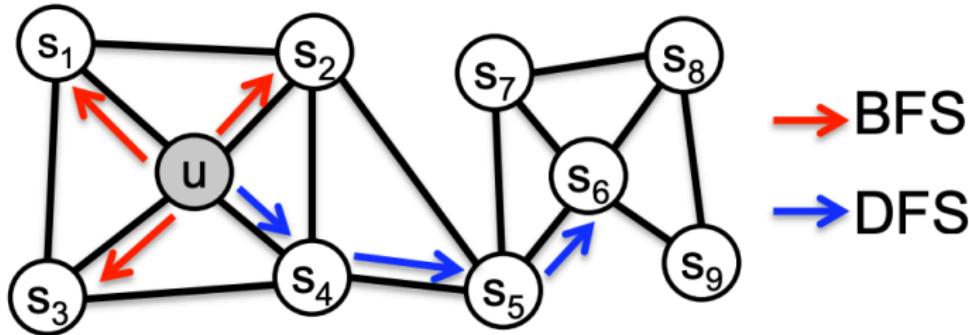
使用随机游走估计节点之间的结构相似性，进而指导节点的表征学习。

## 随机游走的优势

- ① 同时包含节点之间的低阶临近关系和高阶临近关系
- ② 支持批量训练，适用于大规模场景
- ③ 方便对不同类型的网络进行适应



# 基于随机游走的节点表征学习



Node2Vec 随机游走



# 基于随机游走的节点表征学习

给定网络上的随机游走结果，节点表征学习的目标是最大化如下的 log 似然：

$$\max_f \sum_{u \in V} \log P(N_R(u) | \mathbf{z}_u)$$

该目标函数等价于：

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v | \mathbf{z}_u))$$

$$P(v | \mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^T \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^T \mathbf{z}_n)}$$



# 基于随机游走的节点表征学习

估计节点和邻居的共现概率，需要考虑所有的节点：

$$P(v | \mathbf{z}_u) = \frac{\exp(\mathbf{z}_u^T \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^T \mathbf{z}_n)}$$

在实际计算中计算开销过大，一般采用负采样策略，即按照节点的度分布进行采样，使用  $K$  个负样本估计似然函数。

$$\begin{aligned} & \log \left( \frac{\exp(\mathbf{z}_u^T \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^T \mathbf{z}_n)} \right) \\ & \approx \log(\sigma(\mathbf{z}_u^T \mathbf{z}_v)) - \sum_{i=1}^k \log(\sigma(\mathbf{z}_u^T \mathbf{z}_{n_i})), n_i \sim P_V \end{aligned}$$



## ① 图数据挖掘简介

## ② 图上的分类

## ③ 图上的聚类

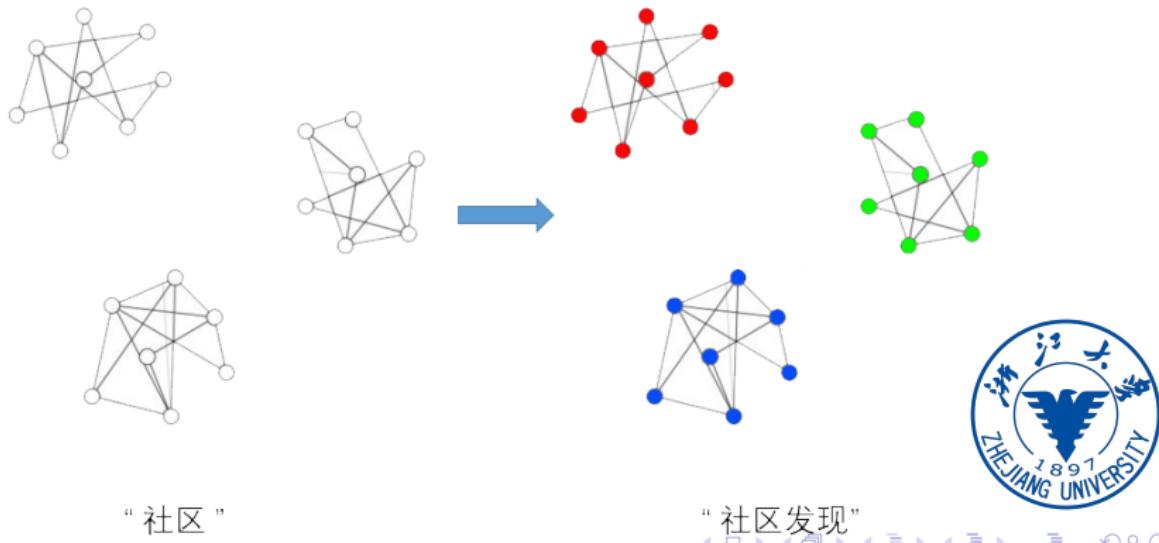
## ④ 图上的异常检测



# 社区发现

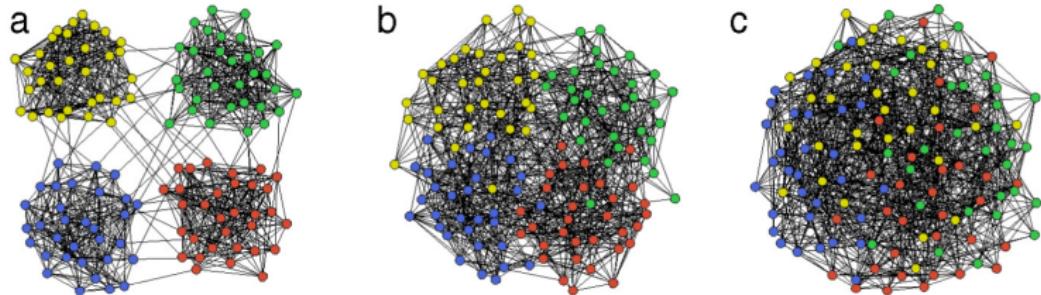
## 社区发现定义

社区发现是经典的子图数据挖掘任务之一，它的目标是**自动发现**数据中的社区，使得社区内的节点稠密连接，而社区间稀疏连接。



# 社区发现-图聚类

- 社区发现定义和谱聚类的定义并没有本质的区别，因此图聚类属于社区发现的一种方法
- 如图 (Figure) 所示，图 (Graph) 上的'社区'的结构是人为附加的概念，能够一眼看出'社区'结构完全是人为的操作。不像其他分类、识别、分割问题，一般来讲，社区发现几乎无法人工完成。



# 社区发现-Newman 算法 [Girvan and Newman, 2002]



把节点想象成一个地点，边想象成连接地点的道路，此时社区发现任务可以理解为将地区划分为城市行政区，城市内部道路连接紧密，城市间则仅有几条高速公路连接。



# 社区发现-Newman 算法 [Girvan and Newman, 2002]

## 介数

两个可达的点之间至少有一条最短路径，两两节点之间产生了众多最短路径。通过边的最短路径的条数，是**边的属性**

## Newman 算法基本假设

假设两个城市之间仅有一条高速公路，那么该高速公路是城市互通的**必经之路**，也就是说该高速路的介数较高，通过切断必经之路就可以把城市有效分离开



# 社区发现-Newman 算法 [Girvan and Newman, 2002]

Newman 算法的基本步骤：

- ① 计算图中所有边的介数
- ② 移除介数最高的边
- ③ 重新计算剩下图中的边的介数
- ④ 重复第二步直到达到社区发现的要求

## Newman 算法的缺陷

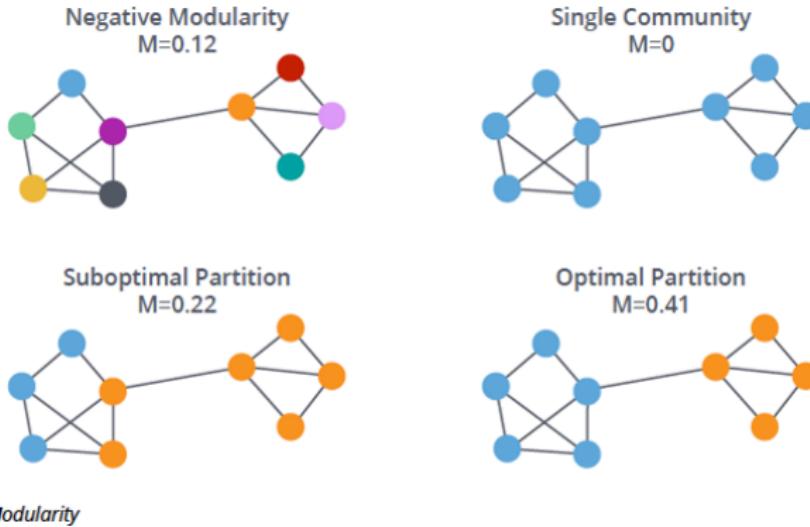
这是一个 Top-Down 的算法，但是时间复杂度似乎有一点高（每次迭代都需要重新计算所有的最短路径）

两年后，Newman 又提出了另一个 Bottom-Up 社区发现算法  
[Newman, 2004]



# 社区发现-Fast-Newman 算法 [Newman, 2004]

- 上一代算法仅仅提出了解决方案，但是没有提出如何衡量解决方案的好坏
- 定义模块度（Modularity Q）用于衡量社区发现的结果好不好：



社区发现-Fast-Newman 算法 [Newman, 2004]

- 如何计算模块度 (Modularity Q):

$$Q = \sum_{i=1}^c \left( \frac{I}{E} - \left( \frac{2I + O}{2E} \right)^2 \right)$$

- ①  $E$  为图中边的数目
  - ②  $I$  表示两个端点均在同一社区中的边的数目
  - ③  $O$  表示其中一个端点在社区中，而另一个端点不在相同社区中的边的数目

● 在一个社区网络结构中，**社区内连边数与随机期望的差值**。当实际的连边越高于随机的期望时，这个社区的结构就比较合理。



# 社区发现-Fast-Newman 算法 [Newman, 2004]

- 上面的式子并不是它的定义式，而是它的计算式。定义式长这样：

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

- 模块度越大则表明社区划分效果越好。Q 值的范围在  $[-0.5, 1]$ ，论文表示当 Q 值在 0.3~0.7 之间时，说明聚类的效果很好。
- 假设有  $x$  个节点，将这些输入划分为  $N$  个社区，节点彼此之间共有  $m$  个连接。 $v$  和  $w$  是  $x$  中的任意两个节点，当两个节点直接相连时  $A_{vw} = 1$ ，否则  $A_{vw} = 0$ 。 $k_v$  代表的是节点  $v$  的度。
- $\delta(c_v, c_w)$  是用来判断节点  $v$  和  $w$  是否在同一个社区内，在同一个社区内  $\delta(c_v, c_w) = 1$ ，否则  $\delta(c_v, c_w) = 0$



# 社区发现-Fast-Newman 算法 [Newman, 2004]

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) = \frac{1}{2m} \sum_{vw} A_{vw} \delta_{vw} - \frac{1}{2m} \sum_{vw} \frac{k_v k_w}{2m} \delta_{vw}$$

- 后一项意思就是，对网络的边进行随机分配，需要将每条边切断一分为二，切断的点称作末梢点 (stub)，这样  $m$  条边就会产生  $l_n = 2m = \sum_{v=1}^n k_v$  个末梢点
- 随机的将这  $l_n$  个末梢点进行连接，包括同一节点拥有的末梢点的自接。这样可以保持每个节点原有的度不变的条件下，可以得到一个完全随机的网络。
- 在该随机网络下，任意两点  $v w$  连接边数的期望值是：

$$\exp_{vw} = \frac{k_v k_w}{l_n} = \frac{k_v k_w}{2m}$$



# 社区发现-Fast-Newman 算法 [Newman, 2004]

- 两个 node  $v$  和  $w$  的度数分别是  $K_v$  和  $K_w$ 。在随机网络中，两个节点间有一条边的概率是  $(k_v * k_w) / 2m$
- 前一项则表示实际上社区内的连接数
- 那么模块度可以表示为**社区内的实际连接数 - 社区内连接数的期望**:

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) = \frac{1}{2m} \sum_{vw} A_{vw} \delta_{vw} - \frac{1}{2m} \sum_{vw} \frac{k_v k_w}{2m} \delta_{vw}$$

- 由于边和度是可以相互转化计算的，转化后的式子就是：

$$Q = \sum_c \left( \frac{\sum_{\text{in}}}{2m} - \frac{1}{2m} \sum_{v,w \in c} \frac{k_v k_w}{2m} \right)$$

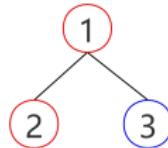
$$Q = \sum_c \left( \frac{I}{m} - \left( \frac{2I + O}{2m} \right)^2 \right)$$



# 社区发现-Fast-Newman 算法 [Newman, 2004]

$$Q = \sum_c \left( \frac{I}{m} - \left( \frac{2I + O}{2m} \right)^2 \right)$$

- 以下图为例，用计算式实际计算一遍：



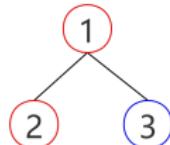
$$Q = \left( \frac{1}{2} - \left( \frac{3}{4} \right)^2 \right) + \left( \frac{1}{2} - \left( \frac{3}{4} \right)^2 \right) = -\frac{1}{8}$$



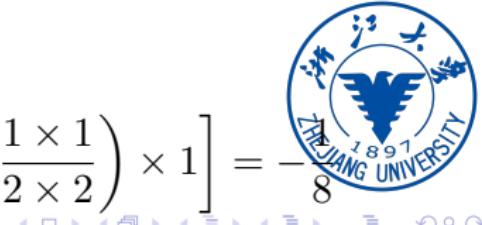
# 社区发现-Fast-Newman 算法 [Newman, 2004]

$$Q = \frac{1}{2m} \sum_{vw} \left[ A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

- 以下图为例，用定义式再实际计算一遍：



$$\begin{aligned}
 Q &= \frac{1}{2 \times 2} \left[ \left( 0 - \frac{2 \times 2}{2 \times 2} \right) \times 1 + 0 + \left( 1 - \frac{2 \times 1}{2 \times 2} \right) \times 1 \right] \\
 &\quad + \frac{1}{2 \times 2} \left[ 0 + \left( 0 - \frac{1 \times 1}{2 \times 2} \right) \times 1 + 0 \right] \\
 &\quad + \frac{1}{2 \times 2} \left[ \left( 0 - \frac{1 \times 2}{2 \times 2} \right) \times 1 + 0 + \left( 1 - \frac{1 \times 1}{2 \times 2} \right) \times 1 \right] = -\frac{1}{8}
 \end{aligned}$$



# 社区发现-Fast-Newman 算法 [Newman, 2004]

给定模块度的定义，Fast-Newman 算法的流程可以直观理解为：

- ① 将图中的每个节点看成一个独立的社区
- ② 计算每种合并方案后的  $Q$  值，采用最大的  $Q$  值方案，将合并后的社区视为一个节点
- ③ 重复第二步直到达到社区发现的要求

该算法的核心是划时代的度量衡：模块度  $Q$ 。将社区发现问题转化为  
了模块度  $Q$  的优化问题，Fast-Newman 算法采用了遍历的优化策略。



# 社区发现-Fast-Newman 算法 [Newman, 2004]

用树结构表示层级社区发现的结果，在每一层计算模块度 Q，结果如下：

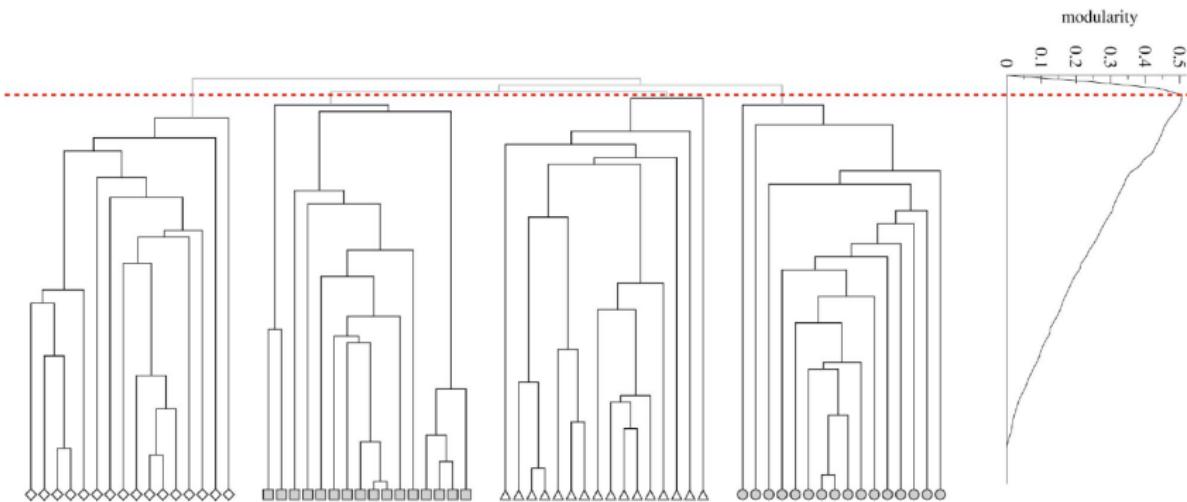


FIG. 6. Plot of the modularity and dendrogram for a 64-vertex random community-structured graph generated as described in the text with, in this case,  $z_{in}=6$  and  $z_{out}=2$ . The shapes at the bottom denote the four communities in the graph and, as we can see, the peak in the modularity (dotted line) corresponds to a perfect identification of the communities.

# 社区发现-LOUVAIN[Blondel et al., 2008]

理解了模块度  $Q$ , LOUVAIN 算法就可以一秒理解。LOUVAIN 算法流程如下:

- ① 把所有节点都视为单独的社区
- ② 对每个点, 尝试将其分配到其邻居节点所在的社区, 合并后的社区视为一个节点。计算并选择最优  $Q$  值方案。若  $Q$  值变化  $\leq 0$ , 则不分配。
- ③ 把每个社区视为一个节点, 重新构图
- ④ 重复 2-3



# 社区发现-LOUVAIN[Blondel et al., 2008]

- Newman 算法、Fast-Newman 算法、Louvian 算法是一脉相承的，Louvian 算法是贪心算法，而 Newman 算法和 Fast-Newman 算法是遍历算法。可以看出每一版的算法流程都只有不太大的改动。但是 Louvian 算法的时间复杂度比起前代算法低了一整个量级。
- 但是我们来看看 LOUVAIN 算法面对大数据的卓越表现：11 亿 8 千万个结点进行层级社区结构发现仅需要 152 分钟
- ps: 这张图的完整邻接矩阵，如果仅用一个 bit 表示连接信息，需要 50655GB 存储空间，用稀疏邻接表存储需要 50GB



# 社区发现-特征图社区发现

- 以上算法全部定义在无向无权图上，随着时代的发展，这种图结构并不足以完全表达图结构的数据。
- 比如社交网络，人和人存在好友关系，而且人作为节点，自身存在特征，比如身高、体重、兴趣爱好、乃至购物记录，这些信息是简单的图无法表达的。于是特征图应运而生：

$$G = (V, E) \rightarrow G_a(V, E, \mathcal{A}, F)$$

拓扑结构信息 → 拓扑结构信息 + 节点属性特征信息



## 社区发现-特征图社区发现

$$G = (\mathbf{V}, \mathbf{E}) \quad \rightarrow \quad G_a(V, E, \mathcal{A}, F)$$

- 特征图的解决方案也很简单，结合图拓扑和节点特征，重新计算节点之间的权重，重新构图

$$\text{dis}(u, v) = \alpha \text{ dist}_T(d_u, d_v) + (1 - \alpha) \text{ diss}_S(d_u, d_v)$$

- $Dist_T$ : 拓扑学距离 (最短路径长度、Neighborhood random walk distance)
  - $Diss$ : 属性距离, 简单的欧氏距离, 复杂的如核函数



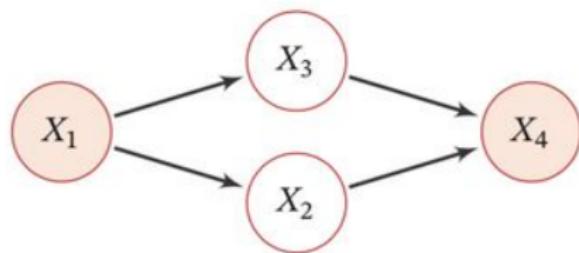
# 社区发现-SBM

- 随机块模型 SBM(stochastic block model) 是由社会学领域角色分析模型——块模型 (block model) 发展而来的, 根据概率对等性 (stochastic equivalent) 将具有相似角色的节点分类
- 该模型不对网络结构做任何假设, 可较好地发现未知先验的网络的结构

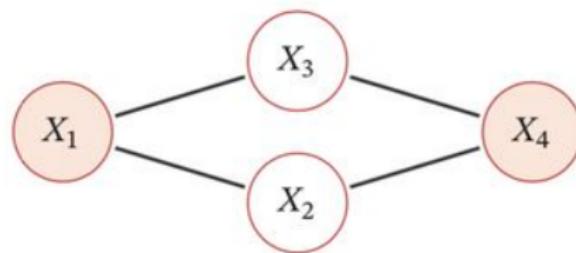


## 概率图模型

- 概率图模型 (Probabilistic Graphical Model, PGM), 简称图模型 (Graphical Model, GM), 是指一种用图结构来描述多元随机变量之间条件独立性的概率模型, 从而给研究高维空间的概率模型带来了很大的便捷性。



(a) 有向图: 贝叶斯网络



(b) 无向图: 马尔可夫随机场



# 社区发现-SBM

- SBM 是一个网络生成模型, 分两步生成网络:
  - ① 先为所有节点指派隶属类
  - ② 然后根据类及类间链接概率决定任意两个节点间是否产生链接
- SBM 倾向于产生描绘以连通性 (connectedness) 为特征的社区子集的图。如果连通性用边密度来描述, 那么该问题与我们之前提到的标准社区发现完全一致。



# 概率社区发现-SBM

## SBM 模型假设

SBM 假设存在一个对称矩阵  $B \in \mathbb{R}^{k \times k}$ ,  $k \ll n$ , 和一个映射函数  $C : \{1, \dots, n\} \rightarrow \{1, \dots, k\}$ , 使得

$$p_{ij} = B_{C(i), C(j)}$$

对于观测到的图  $G$ ,  $A = [A_{ij}] \in \mathbb{R}^{n \times n}$  是邻接矩阵, 在无向图中  $A$  是对称阵。假设  $A_{ij}$  是独立的 Bernoulli 变量。

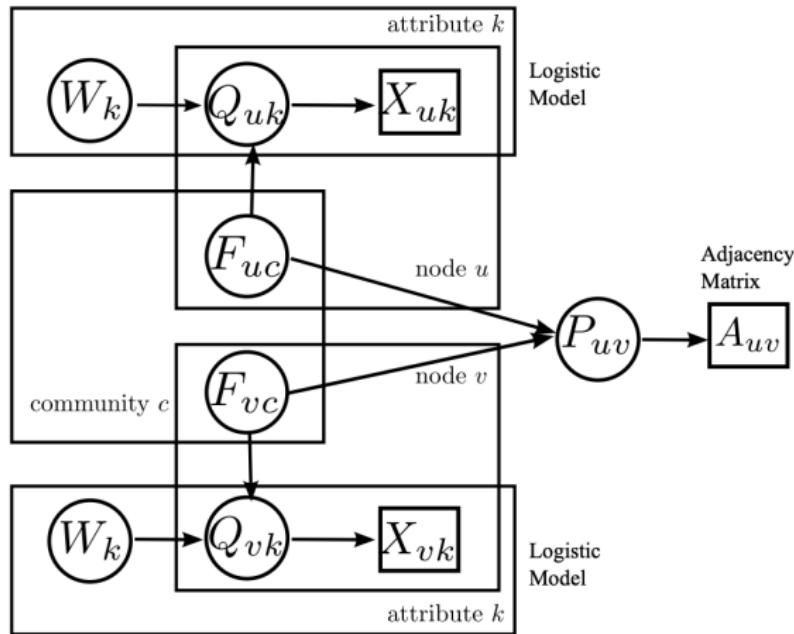
$$A_{ij} \sim \text{Bernoulli}(p_{ij}), A_{ji} = A_{ij}, p_{ij} \in (0, 1), i < j, A_{ii} = 0$$

根据 SBM 模型的定义, 在给定的图  $G = \{V, E\}$  中学习社区结构, 需要估计  $k(k + 1)/2$  个参数。



# SBM 拓展

## 带属性网络的社区发现



## 带属性网络社区发现算法

# 社区发现的应用

- 网络运维
  - 服务器交换机路由器组成的拓扑
- 传染病控制
  - 通过对接触人群的社区结构的发现，做到及时隔离有效隔离
- 流量的社区结构分析
  - 分布式云服务数据中心管理
- 被访问行为聚类
  - 网站热点缓存管理
- 微博粉丝群体聚类
  - 广告推广



## 1 图数据挖掘简介

## 2 图上的分类

## 3 图上的聚类

## 4 图上的异常检测



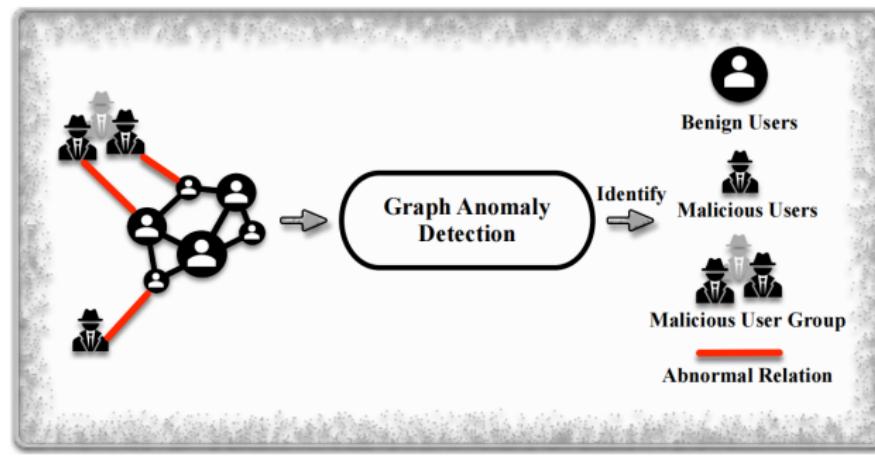
# 图异常检测

- 传统的异常检测技术倾向于将真实世界中的对象表示为特征向量，然后在特征空间中找到异常点。
- 然而，数据对象并不能总是被视为独立位于多维空间中的点——它们之间可能存在某种关系，表现出相互依赖性，可以为异常检测提供非常有价值的补充信息。
- 因此，传统的异常检测方法在 Graph 上无法直接应用，如何检测 Graph 中的异常值是一个热门的问题。



# 异常类型

- 在 Graph 中，异常可以分为三类：
  - nodes：异常对象
  - edges：异常关系
  - sub-graphs：异常群体



# 应用场景

- 金融交易网络中的异常现象
  - 一组交易员在一段时间内对特定的股票进行大量相互交易，拉高股价，再将其向公众出售，以此牟利。
- 基于子结构的异常检测：两种不同的行为可以用图的术语来清晰表示，用节点表示交易者，边表示交易量。
  - 第一阶段，交易员组内大量相互交易，对应的子结构中边权非常高。
  - 而销售阶段，子结构对外部的权重远大于内部权重。
- 由此，可以基于这些突出表现在大规模且动态变化的金融交易网络中快速检测这种异常行为。



# 应用场景

- 真·网络（Web）中的异常——垃圾网页
- 核心思想：置信度传播。
  - 每个页面都有其置信度，页面间有链接跳转（存在单向边）表示源页面信任目标页面。此时，若目标页面为垃圾网页，则源页面的置信度也会受到惩罚。



# 应用场景

- 计算机网络中的异常：攻击与入侵
- 节点代表了服务器和客户端，而边表示的是它们在网络上的通信。
- 大多数基于 Graph 的网络入侵检测方法都关注于图的动态增长和变化，在这里是基于一个假设：一个计算节点的通信行为会在受到攻击时改变。



# 评价指标

- 异常检测常用的性能评价指标如下：
  - 精度 precision
  - 召回率 recall
  - F-score
  - ROC&AUC
- 这些指标也适用于图数据挖掘，但需要根据具体应用场景进行选择。
  - 网络入侵检测对 false positive 更敏感，而 false positive 则较少考虑——宁可错杀，不可放过。



# 经典图异常检测方法

根据算法模型的检测级别，图异常检测任务大体上分为三类：

- Node-Level：异常点检测
  - OddBall
- Edge-Level：异常边检测
  - SedanSpot
- (Sub)Graph-Level：异常（子）图检测
  - LockInfer



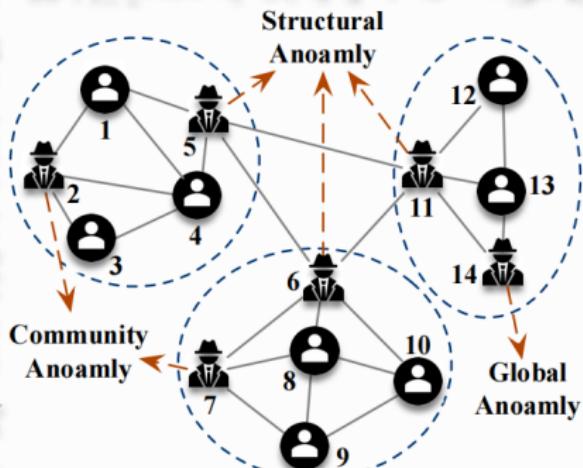
# 图异常节点类型

针对静态图，主要从节点或者边属性进行区分。异常节点主要可以分为以下三种类型：

- 全局异常点 (Global anomalies)：考虑节点属性，属性与图中其他节点明显不同；
- 结构异常点 (Structural anomalies)：考虑图结构信息，连接模式明显与其他节点不同；
- 社团异常点 (Community anomalies)：考虑节点属性与结构，在相同社区中节点属性与其他节点不同；



# 图异常节点类型



Community	Node	Node Features			
Community 1	1	1	0	0	0
	2	1	1	0	0
	3	1	0	0	0
	4	1	0	0	0
	5	1	0	0	0
Community 2	6	0	1	0	0
	7	0	1	1	0
	8	0	1	0	0
	9	0	1	0	0
	10	0	1	0	0
Community 3	11	0	0	1	0
	12	0	0	1	0
	13	0	0	1	0
	14	0	0	1	1

异常节点类型



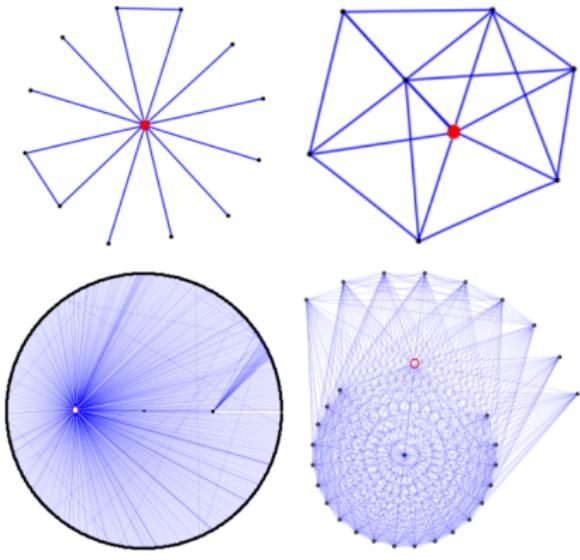
# 传统图异常点检测——OddBall[Akoglu et al., 2010]

- OddBall 是一个在无向带权图上检测异常节点的经典方法。
- 将我们关注的节点定义为中心节点 ego，由 ego 及其邻居组成的子图称为 egonet。这里只考虑一阶邻居，这是为了减少计算量并提高可解释性。
- 作者观察并归纳了几种 egonet 中存在的异常模式。
  - Near-star 和 Near-clique
  - Heavy vicinity
  - Dominant edge



# OddBall

- Near-star: 中心节点与大量节点存在关联，但是邻居之间联系很少。
  - 在社交网络中，我们通常认为朋友之间可能会相互认识，因此一阶 ego-net 中的邻居之间关联极少是非常可疑的
- Near-clique: 中心节点与大量节点存在关联，邻居之间的联系非常密集。
  - 相反，邻居节点之间存在大量关联也是非常可疑的。

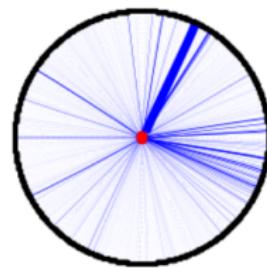
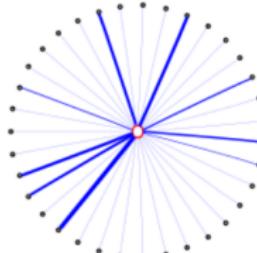
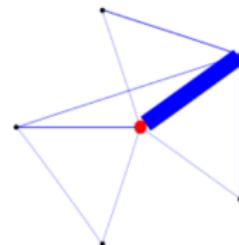
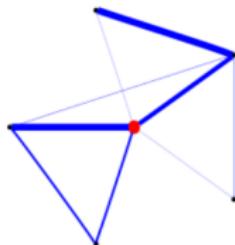


(上面是示例，下面是  
真实数据集中情况。)



## OddBall

- Heavy vicinity: ego 与邻居之间边的总权重过大。
    - 若节点  $i$  与  $n$  个不同的邻居相连，我们希望与节点  $i$  有关的边的总权重是与  $n$  有关的函数。若总权重过大，则认为节点  $i$  是异常的。
  - Dominant edge: ego 与某个邻居之间存在超大权重边。



(c) Heavy vicinity (d) Dominant edge



# OddBall

异常模式的形式化描述：

- 定义以下“特征”：

- $N_i$ : ego  $i$  的邻居数量
- $E_i$ : egonet  $i$  中边的数量
- $W_i$ : egonet  $i$  的总边权
- $\lambda_{w,i}$ : egonet  $i$  的带权邻接矩阵的主特征值（绝对值最大的特征值）

- 不同的组合适合于不同的场景：

- $N$  和  $E$ : Near-star 和 Near-clique
- $E$  和  $W$ : Heavy vicinity
- $\lambda_w$  和  $W$ : Dominant edge



# OddBall

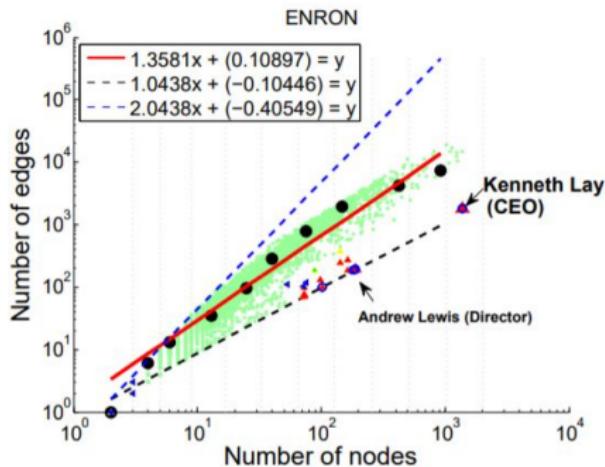
- 如何通过这些“特征”来区分 egonet 的正常与否呢？
- 作者进行了观察并总结了如下定律：
  - ① EDPL: Egonet Density Power Law
  - ② EWPL: Egonet Weight Power Law
  - ③ ELWPL: Egonet  $\lambda_w$  Power Law
- 定义参数：对于一个给定的图  $\mathcal{G}$ ，节点  $i \in \mathcal{V}(\mathcal{G})$ ， $\mathcal{G}_i$  为节点  $i$  的 egonet。



# OddBall

- EDPL: Egonet Density Power Law
- $\mathcal{G}_i$  的节点数  $N_i$  和边数  $E_i$  满足幂定律：

$$E_i \propto N_i^\alpha, \quad 1 \leq \alpha \leq 2$$

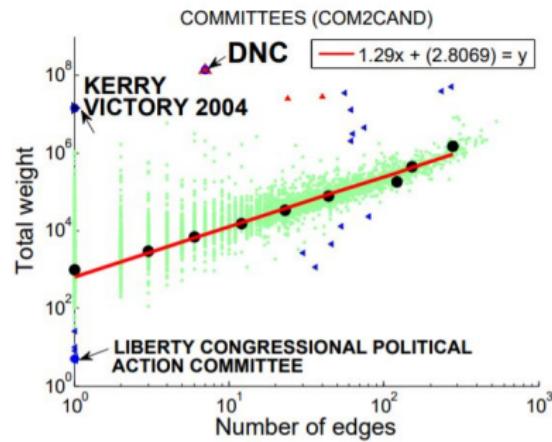


- $\alpha$  越大，即绿色的点越接近蓝线，则 egonet 越倾向于 clique 形态
- $\alpha$  越小，即绿色的点越接近黑线，则 egonet 越倾向于 star 形态
- 红线为中位数拟合的结果，表示正常的情况。



# OddBall

- EWPL: Egonet Weight Power Law
- $\mathcal{G}_i$  边的总权重  $W_i$  和边数  $E_i$  满足幂定律:
$$W_i \propto E_i^\beta, \quad \beta \geq 1$$
- $\beta$  越大, egonet 越倾向于 Heavy vicinity 形态

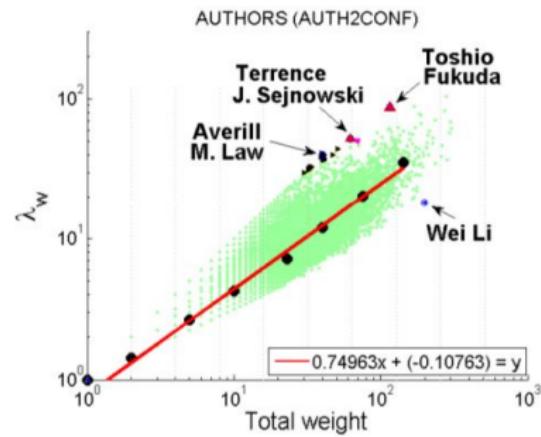


# OddBall

- ELWPL: Egonet  $\lambda_w$  Power Law
- $\mathcal{G}_i$  加权邻接矩阵主特征值  $\lambda_{w,i}$  和边的总权重  $W_i$  满足幂定律:

$$\lambda_{w,i} \propto W_i^\gamma, \quad 0.5 \leq \gamma \leq 1$$

- $\gamma$  越大, egonet 越接近 Dominant edge 的形态



# OddBall

- 由上述观察到的模式以及检测依据为基础，OddBall 给出了一个评估指标 out-score，值越大则对应的节点越可疑。

$$\text{out-score}(i) = \text{out-line}(i) + \text{out-lof}(i)$$

- out-line：点与拟合直线（红线）的偏离程度。
- out-lof：考虑到 out-line 无法识别离正常点很远，但与拟合直线很近的异常点，因此结合传统基于密度的方法 LOF 加入补偿因子。



# OddBall

- out-line: 各检测指标下相对于均值（红线）的偏移程度。

$$\text{out-line}(i) = \frac{\max(y_i, Cx_i^\theta)}{\min(y_i, Cx_i^\theta)} * \log(|y_i - Cx_i^\theta| + 1)$$

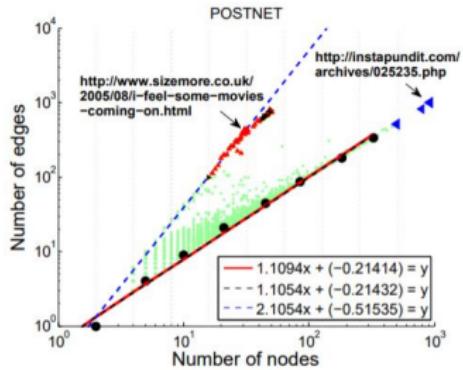
其中,  $y_i$  为实际值,  $Cx_i^\theta$  为在拟合直线上的预测值, 二者相减为偏离程度。

$\frac{\max(\cdot)}{\min(\cdot)}$  为惩罚系数: 实际值偏离正常的倍数。



# OddBall

- out-lof: out-line 计算中的补偿因子。
- 如下图所示, 红线和黑线高度重合, 此时若仅仅使用 out-line, 则图中蓝色三角形标注的异常点就无法被识别。



- OddBall 选择了基于密度的方法 LOF 来解决这个问题, 具体的 LOF 请见课程 6 PPT。



# 深度图异常点检测—— DONE[Bandyopadhyay et al., 2020]

- DONE 利用自动编码器 AE 来检测属性图中的异常点。
- 作者定义了三种异常点：
  - ① 结构异常点 (Structural Outlier)：节点在结构上连接了不同社区的节点。
  - ② 属性异常点 (Attribute Outlier)：节点的属性与来自不同社区的节点相似。
  - ③ 组合异常点 (Combined Outlier)：节点在结构上属于一个社区，但在属性方面与另一社团模式相似。



DONE

我们先进行以下定义：

- $\mathcal{G} = (V, E, C)$ : 图
- $V = \{v_1, v_2, \dots, v_N\}$ : 节点集合
- $E \subset \{(v_i, v_j) | v_i, v_j \in V\}$ : 边的集合
- $\mathcal{N}(v_i) = \{v_j \in V | (v_i, v_j) \in E\}$ : 节点的一阶邻域
- $A \in \mathbb{R}^{N \times N}$ :  $\mathcal{G}$  的邻接矩阵。
- $C \in \mathbb{R}^{N \times D}$ : 节点属性矩阵，其中  $c_i \in \mathbb{R}^D$  表示节点  $v_i$  的属性。



## DONE

- DONE 的目标是学习一个嵌入函数，将每个节点映射到一个  $K$  维的向量 ( $K < \min(N, D)$ )。

$$f : v_i \mapsto h_i \in \mathbb{R}^K$$

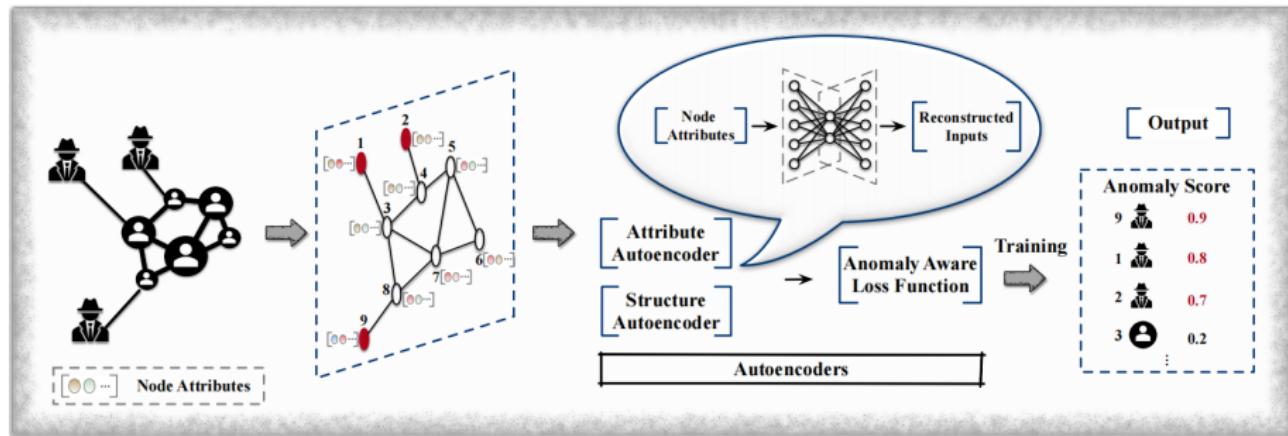
- 同时，学习每个节点的离群值  $o_i^s$ ,  $o_i^a$  和  $o_i^{com}$ ，对应于结构 (Structural)、属性 (attribute) 和组合 (combined) 异常点。所有离群值的集合用  $O$  表示。

$$\sum_{i=1}^N o_i^s = 1, \quad \sum_{i=1}^N o_i^a = 1, \quad \sum_{i=1}^N o_i^{com} = 1, \quad o_i^s, o_i^a, o_i^{com} > 0$$

为  $o$  设置上界，防止其在优化过程中趋向于  $+\infty$ 。

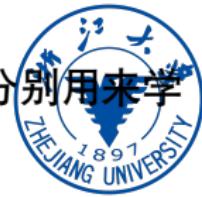


DONE



DONE 模型

DONE 的主要网络结构包括两个并联的 AE  $Enc^s$  和  $Enc^a$ ，分别用来学习结构的表示和属性的表示。



DONE

- 输入两个 AE 的分别为属性和结构信息。属性很好处理，有现成的属性矩阵  $C$ ，但结构信息用什么呢？邻接矩阵  $A$ ？
- 现实生活中的网络高度稀疏，节点之间的连接很少。邻接矩阵只能捕获观察到的连接。
- 那么，如何获得更多的连接信息呢？



## DONE

- 参考 Page Rank 的思想，DONE 构造了一个概率矩阵  $P^t \in \mathcal{R}^{N \times N}$ ， $(P_i^t)_j$  表示从节点  $i$  出发经过  $t$  时刻后随机走到节点  $j$  的概率。
- 显然， $P_i^0$  中只有  $(P_i^0)_i = 1$ ，其余各项均为 0。

$$P_i^t = r P_i^{t-1} [D^{-1} A] + (1 - r) P_i^0$$

其中  $0 \leq r \leq 1$ ， $1 - r$  表示随机游走的重启概率。 $D$  是一个对角矩阵，有  $D_{ii} = \sum_{j=1}^N a_{ij}$ 。

- 输入网络的结构信息  $X$  为所有  $P^t$  的平均值。

$$X = \frac{1}{T} \sum_{t=1}^T P^t$$



## DONE

- 接下来，我们来看 DONE 的目标函数。
- 对于结构信息，DONE 有以下两个约束
- 接近性约束 (proximity)：以 AE 作为网络结构相对应的重构 Loss，再结合较小异常点影响的考虑。

$$\mathcal{L}_{str}^{Prox} = \frac{1}{N} \sum_{i=1}^N \log\left(\frac{1}{o_i^s}\right) \|x_i - \hat{x}_i\|_2^2$$

- 同质性约束 (homophily)：邻域内的节点结构信息也应该与当前节点相近。

$$\mathcal{L}_{str}^{Hom} = \frac{1}{N} \sum_{i=1}^N \log\left(\frac{1}{o_i^s}\right) \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|h_i^s - h_j^s\|_2^2$$



DONE

- 同样的，对于属性信息，也有接近性约束（proximity）和同质性约束（homophily）。

$$\mathcal{L}_{attr}^{Prox} = \frac{1}{N} \sum_{i=1}^N \log\left(\frac{1}{o_i^a}\right) \|c_i - \hat{c}_i\|_2^2$$

$$\mathcal{L}_{attr}^{Hom} = \frac{1}{N} \sum_{i=1}^N \log\left(\frac{1}{o_i^a}\right) \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|h_i^a - h_j^a\|_2^2$$



## DONE

- 实际上，图中节点的链路结构和节点的属性也是高度相关的。因此，最后一部分 Loss (combining structure and attributes) 写为

$$\mathcal{L}^{Com} = \frac{1}{N} \sum_{i=1}^N \log\left(\frac{1}{o_i^{com}}\right) \|h_i^s - h_i^a\|_2^2$$

- 最终的优化目标为：

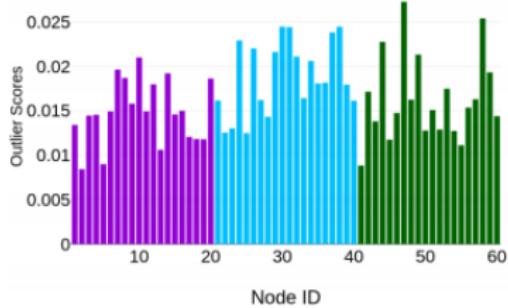
$$\min_{\Theta, O} L_{DONE}$$

$$= \alpha_1 \mathcal{L}_{str}^{Prox} + \alpha_2 \mathcal{L}_{str}^{Hom} + \alpha_3 \mathcal{L}_{attr}^{Prox} + \alpha_4 \mathcal{L}_{attr}^{Hom} + \alpha_5 \mathcal{L}^{Com}$$

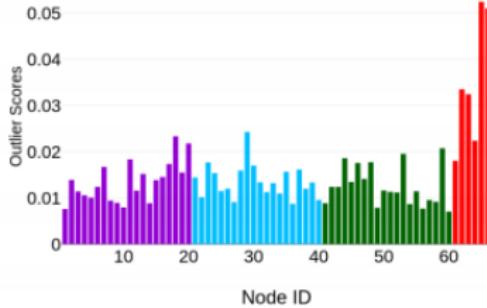


DONE

- 节点  $i$  的最终异常分数  $o_i$  由三个离群值  $o_i^s$ 、 $o_i^a$  和  $o_i^{com}$  的加权平均获得。权重由具体数据集和任务确定。
- 将异常分数由高到低排序，根据设定的异常比例划分异常节点和正常节点。



(a) Without Outliers



(b) With Outliers

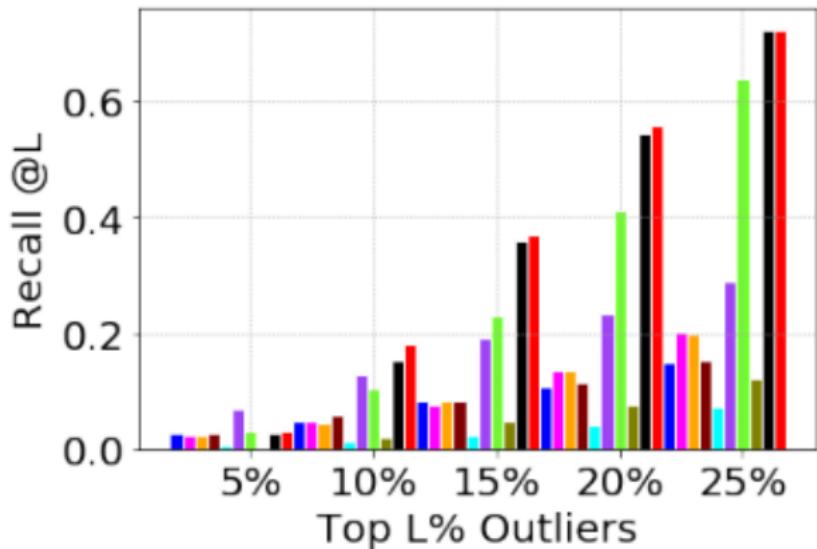
outlier score 与异常点



DONE

■ Node2Vec ■ LINE ■ SDNE ■ GraphSage ■ DGI  
 ■ SEANO ■ ONE ■ Dominant ■ DONE ■ AdONE

## Pubmed



DONE 实验结果



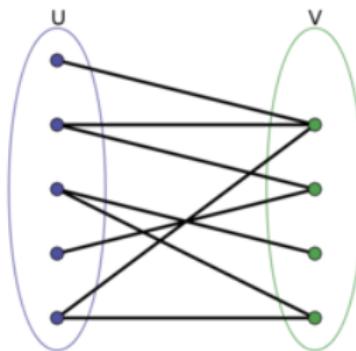
# 深度子图异常检测——DeepFD[Wang et al., 2018]

- DeepFD 是一个基于特定任务——欺诈检测的异常子图检测方法。
- 一种典型的欺诈行为模式是，欺诈者操纵大量的账户或 IP 地址来评级或点击一些特定的项目（如产品、网站）。例如，一些网购产品的虚假评论可能会误导消费者购买有缺陷的产品。
- 由此，欺诈行为将会表现出密集的交互，且欺诈用户访问的项目之间也会存在很高的相似度。



# DeepFD

- 已知图  $G = (V, E)$ , 如果节点集  $V$  可以分割为两个互不相交的子集  $(U, V)$ , 且图中的每条边  $(i, j)$  所连接的两个节点  $i$  和  $j$  都属于不同的节点集 ( $i \in U, j \in V$ ), 则称图  $G$  为一个**二分图**。

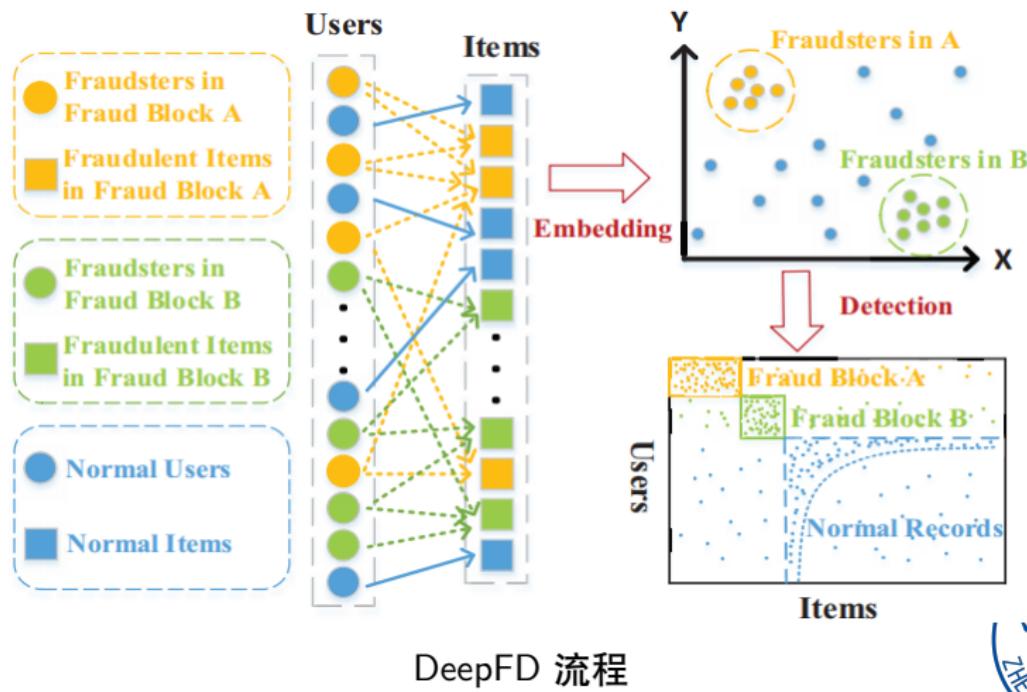


# DeepFD

- DeepFD 将用户与物品之间的交互建模为二分图：用户为源节点，物品为目标节点。由于欺诈行为有显著的密集特征，由此欺诈检测问题可以看做在二分图中发现可疑的密集块。
- 主要的思想与过程：
  - 通过网络将所有用户节点嵌入映射到低维的表征空间中，希望可疑用户在同一欺诈块中的表征尽可能接近，而正常用户的表征则是均匀分布在剩余的表征空间中。
  - 基于密度的方法检测异常块，即通过用户节点在表征空间中的位置关系区分正常用户和可疑用户。



# DeepFD



# DeepFD

我们进行如下的定义：

- $G(X, Y, E)$ : 二分图
  - $X = \{x_1, x_2, \dots, x_m\}$ :  $m$  个用户节点的集合
  - $Y = \{y_1, y_2, \dots, y_n\}$ :  $n$  个项目节点的集合
  - $E \subset \{(x_i, y_j) | x_i \in X, y_j \in Y\}$ : 从  $X$  集合到  $Y$  集合的有向边集合
  - $S = \{s_1, s_2, \dots, s_m\}$ : 用户节点的结构信息, 其中  $s_i = \{s_{ij}\}_{j=1}^n$ , 若从  $x_i$  到  $y_j$  存在边, 则  $s_{ij} = 1$ , 否则为 0。
  - $h_i$ :  $s_i$  对应的隐含层表征
  - $\hat{s}_i$ :  $s_i$  对应的重构向量



# DeepFD

## DeepFD 的表征学习过程

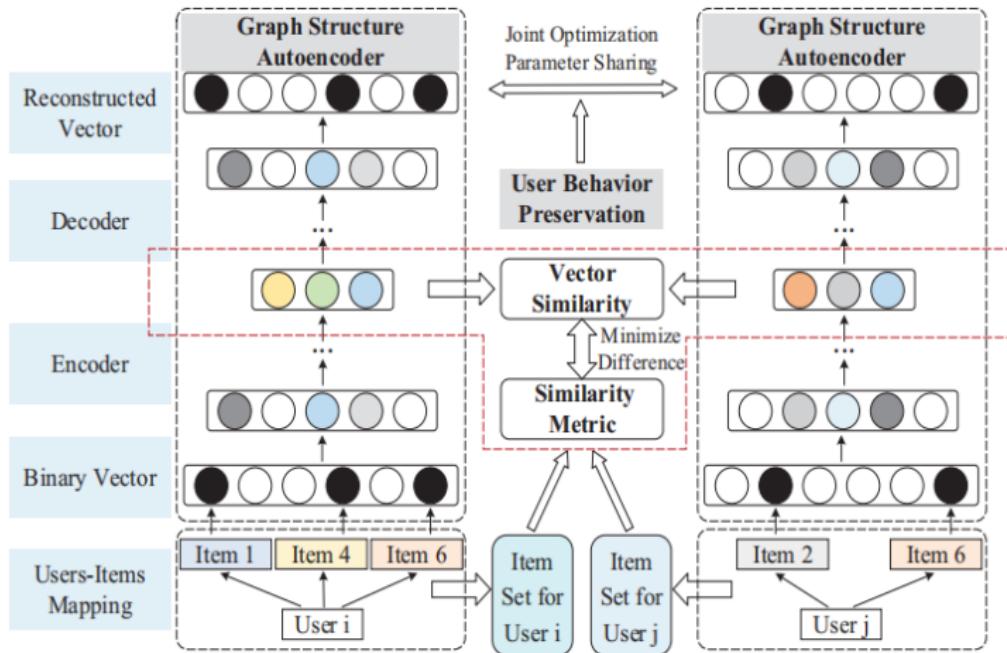
DeepFD 也选用了 AE 作为网络结构：将每个节点的结构信息输入 AE，隐含层向量即为节点的低维表征。同时，对其施加约束以达到我们想要的“密集”效果。

## DeepFD 的表征学习约束：

- 重构约束：对独立的用户节点进行约束，为 AE 必备的重构 Loss。
- 用户行为相似性约束：对用户节点 pair 进行约束，旨在使得行为相似的用户节点在低维表征空间中接近。



# DeepFD



DeepFD 的模型框架



# DeepFD

- 首先是重构约束，AE 经典的重构 Loss 为：

$$\mathcal{L}_{AE} = \sum_{i=1}^m \|\hat{s}_i - s_i\|_2^2$$

- 然而，这里也有与 DONE 中相同的问题：现实中邻接矩阵往往是非常稀疏的。如果用上面这个 Loss 进行训练，那么大部分优化精力会浪费在原来为 0 的元素上面。



# DeepFD

- 因此，DeepFD 希望通过设置权重的方法来修正 Loss 函数：

$$\begin{aligned}\mathcal{L}_{recon} &= \sum_{i=1}^m \|(\hat{s}_i - s_i) \odot g_i\|_2^2 \\ &= \|(\hat{S} - S) \odot G\|_2^2\end{aligned}$$

其中  $\odot$  表示哈达玛积，即矩阵对应位置相乘  $(A \odot B)_{ij} = A_{ij} * B_{ij}$ 。  
 $g_i$  为对应于  $s_i$  的权重向量。对于  $g_i = \{g_{ij}\}_{j=1}^n$ ，若  $s_{ij} = 0$ ，则  $g_{ij} = 1$ ，否则  $g_{ij} = \beta > 1$ 。

- 简而言之，就是对  $S$  中值为 0 的地方设置 1 的权重，值不为 0 的地方设置  $\beta > 1$  的权重。



# DeepFD 用户行为相似性约束

- 对于用户节点  $x_i$  和  $x_j$ , 它们的隐含层表征之间的距离定义为:

$$dis_{ij} = \|h_i - h_j\|_2^2$$

- 我们进一步将距离度量转换为相似度度量:

$$\widehat{sim}_{ij} = \exp(-\lambda \cdot dis_{ij})$$

其中,  $\lambda \geq 0$ 。当两个用户节点的表征距离接近于 0 时,  
 $\widehat{sim}_{ij}$  的值接近 1, 表示其表征十分相似。



# DeepFD

- 现在，我们已经有了计算出来的相似度  $\widehat{sim}_{ij}$ 。那么这两个节点实际上的行为相似性  $sim_{ij}$  是怎样的呢？
- 在实际中，为了达到欺诈的目的，可疑用户节点不可避免地与更多的相同项目节点关联，因此它们之间的相似度相对较高。而正常用户节点的行为是独立的，总体上相似度较低。
- 我们可以从最初的结构信息入手。首先定义  $N_i := \{y_j \in Y | s_{ij} = 1\}$  为与用户节点  $x_i$  相连的项目节点集合。



# DeepFD

- 相似度  $sim_{ij}$  可以表示为：

$$sim_{ij} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

- 直观的说，如果两个用户节点共享很多项目节点，那么它们的相似度会很高。
  - 不过，还存在两种极端情况：

- 集合  $N_i$  和集合  $N_j$  不共享任何共同的元素，即  $|N_i \cap N_j| = 0$
  - 集合  $N_i$  与集合  $N_j$  相同，即  $|N_i \cap N_j| = |N_i \cup N_j|$



DeepFD

- 为了使相似度度量更具有鲁棒性，作者在其中添加了一个平滑项，改进后的相似度度量如下：

$$sim_{ij} = \begin{cases} \frac{|N_i \cap N_j| + 1}{|N_i \cup N_j| + n} & |N_i \cap N_j| = \emptyset, \\ \frac{|N_i \cap N_j| + (n - 1)}{|N_i \cup N_j| + n} & |N_i \cap N_j| = |N_i \cup N_j|, \\ \frac{|N_i \cap N_j|}{|N_i \cup N_j|} & otherwise. \end{cases}$$



DeepFD

- 由此，我们将  $sim_{ij}$  作为监督信息，使  $\widehat{sim}_{ij}$  向其靠近。

$$\mathcal{L}'_{sim} = \sum_{i,j=1}^m \|\widehat{sim}_{ij} - sim_{ij}\|_2^2$$

- 此外，相似度越高的用户节点对越可能是欺诈用户。由此，应该给予它们更大的权重，修正后的目标函数如下：

$$\mathcal{L}_{sim} = \sum_{i,j=1}^m sim_{ij} \cdot \|\widehat{sim}_{ij} - sim_{ij}\|_2^2$$



# DeepFD

- 总体的目标函数如下所示：

$$\mathcal{L} = \mathcal{L}_{recon} + \alpha \mathcal{L}_{sim} + \gamma \mathcal{L}_{reg}$$

其中  $\mathcal{L}_{reg}$  是用来防止过拟合的 L2 范数正则化项。

$$\mathcal{L}_{reg} = \frac{1}{2} \sum_{l=1}^K (\|W^l\|_2^2 + \|\hat{W}^l\|_2^2 + \|b^l\|_2^2 + \|\hat{b}^l\|_2^2)$$

$W$  和  $b$  表示 encoder 中每层网络的 weight 和 bias，相对应的  $\hat{W}$  和  $\hat{b}$  表示 decoder 中每层网络的 weight 和 bias。

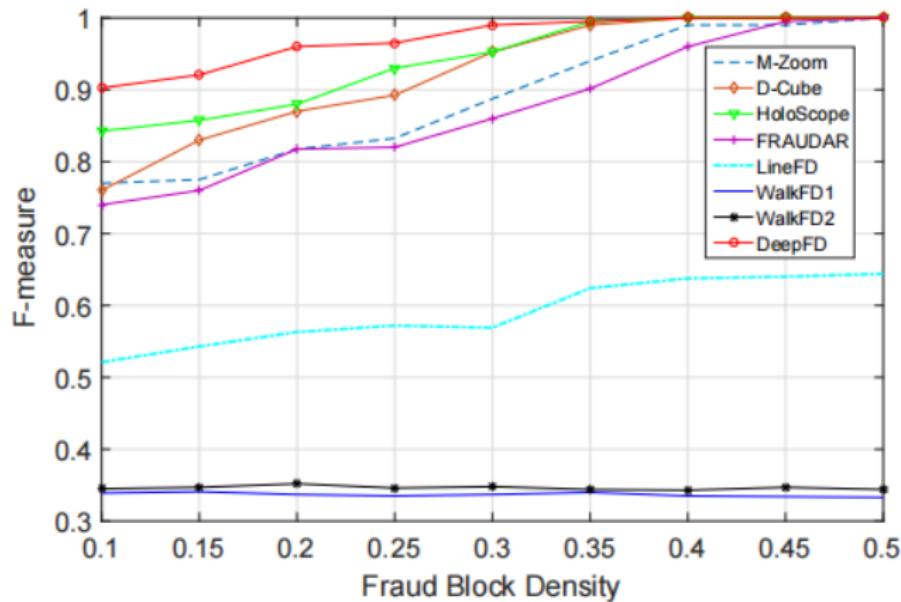


# DeepFD

- 获得了低维表征向量后，在二分图中的欺诈检测问题就可以巧妙地转化为在表征空间中寻找密集区域的问题。
- 由此，可以使用 DBSCAN 来寻找密集块：距离较近且满足一定数量的用户节点将被聚为一类，即诈骗群体。



# DeepFD

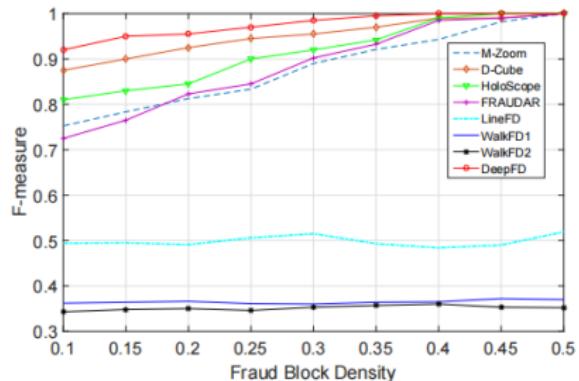


(a) Yelp

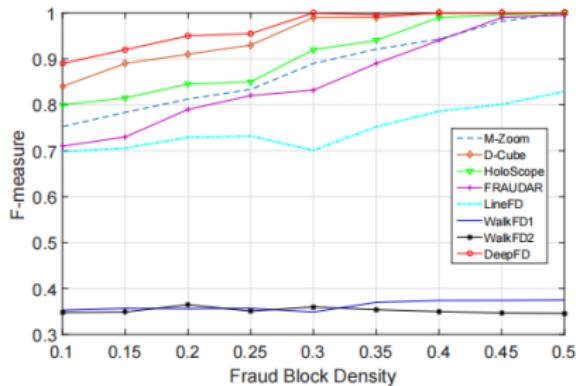
DeepFD 实验结果



# DeepFD



(b) Amazon Instrument



(c) Amazon Movie

## DeepFD 实验结果



## 1 图数据挖掘简介

## 2 图上的分类

## 3 图上的聚类

## 4 图上的异常检测



# References I

-  Akoglu, L., McGlohon, M., and Faloutsos, C. (2010).  
Oddball: Spotting anomalies in weighted graphs.  
In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 410–421. Springer.
-  Bandyopadhyay, S., Vivek, S. V., and Murty, M. (2020).  
Outlier resistant unsupervised deep architectures for attributed network embedding.  
In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 25–33.
-  Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008).  
Fast unfolding of communities in large networks.  
*Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.



## References II

-  Girvan, M. and Newman, M. E. (2002).  
Community structure in social and biological networks.  
*Proceedings of the national academy of sciences*, 99(12):7821–7826.
-  Newman, M. E. (2004).  
Fast algorithm for detecting community structure in networks.  
*Physical review E*, 69(6):066133.
-  Wang, H., Zhou, C., Wu, J., Dang, W., Zhu, X., and Wang, J. (2018).  
Deep structure learning for fraud detection.  
In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 567–576. IEEE.

