Ch. CS224W: Graph ML

Date / /

§01. Introduction

① Traditional Method:
Graphlets, Graph Kernels

② Node Embedding:
DeepWalk, Node2Vec

③ Graph Neural Networks:
GCN, GraphSAGE, GAT, GNNs.

④ graphs & reasoning:
TransE, BetaE

⑤ Deep Generative models:
GraphRNN

- 图机器学习，图数据挖掘，图神经网络（GNN），图卷积神经网络（GCN），知识图谱，知识计算，对话问答，推荐系统，社会计算，媒体计算，网络智能，搜索引擎。

<Why Graphs ?>

- Graphs are a general language for describing and analyzing entities with interactions.（图：描述关联数据的通用语言。）

独立同分布 (i.i.d) — 彼此无关　　Graph = (V, E)（注意与由 pixel 构成的 image 区别）

- Event graphs / computer networks / disease pathways / particle networks / knowledge graphs / regulatory networks / scene graphs / 3D shapes …

Königsberger Brückenproblem 哥尼斯堡七桥问题（欧拉一笔画）

<如何对图数据进行挖掘？>

↳ relational graph.

- modern deep learning toolbox is designed for simple sequences & grids.

complex {
- Arbitrary size and complex topological structure.（任意尺寸输入 — no spatial locality）
- No fixed node ordering or reference point（没有固定的节点顺序和参考锚点）
- Often dynamic and have multimodal features.（动态变化 + 多模态特征）

Input: Network (Graphs) → Graph convolution → Activation function → Regularization (Dropout) → predictions. Node Lables, New links, Generated Graph…

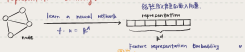
Raw Data → Graph Data → Learning Algorithm → Model

No Feature Engineering（无需人工特征工程）

Representation Learning: Automatically learn the feature 自动学习特征（类似 CNN）

⟵ Downstream prediction task ⟶

<Representation Learning>



- map nodes to d-dimensional embeddings（图嵌入） such that similar nodes in the network are embedded together

- PyG (PyTorch Geometric): the ultimate library for Graph Neural Networks
  Graph Gym: Platform for designing Graph Neural Networks.
  ↳ modularized GNN implementation, simple hyperparameter tuning …

<Different types of Tasks>

Graph-level prediction
Graph generation（生成新的分子结构）

- Node level（信用卡欺诈）
- Community (subgraph) level（用户聚类）
- Edge level（连接 — 推荐"可能认识的人"）

① Node Classification, 节点分类: predict a property of a node.

② Link prediction: 连接预测: predict whether there are missing links between 2 nodes.

③ Graph classification: 图分类: Categorize different graphs

Ch.
Date /

④ Clustering 聚类: Detect if nodes form a community.

⑤ { Graph generation: Drug discovery
  Graph evolution: Physical simulation

§02. Graph Representation (图的表示)

{ Objects: nodes, vertices (N)
  Interactions: links, edges (E)
  System: network, graph G=(N, E)

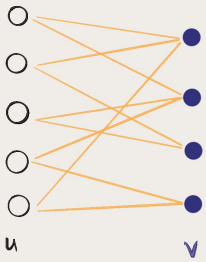
How to build a graph: { What are edges? ⇒ 如何设计图的本体 ontology.
  What are nodes? ⇒ 取决于将来要解决什么问题.

<图的种类>.

{ undirected: 无向图 — symmetrical, reciprocal (互相的)
  directed: 有向图 — arcs

• Heterogeneous Graph: 异质图. → node & edge 可以有不同的类型.

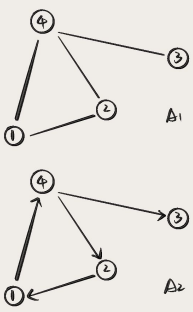
G=(V, E, R, T) { Nodes with node types: $v_i \in V$
  Edges with relation types: $(v_i, r, v_j) \in E$
  Node type: $T(v_i)$
  Relation type: $r \in R$.



• Bipartite Graph: 二分图.
  └ a graph whose nodes can be divided into two disjoint sets U & V, such that every link connects a node in U to one in V
  U and V are Independent sets.

<Node Degrees>. 结点的连接数

• Node degree k_i: the number of edges adjacent to node i.
• Avg. degree: $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$.
• In / out - degree: $\bar{k}^{in} = \bar{k}^{out}$.



<Adjacency Matrix> 邻接矩阵.

{ $A_{ij}=1$ if there is a link from node i → j.
  $A_{ij}=0$ otherwise

$A_1 = \begin{bmatrix} 0&1&0&1 \\ 1&0&0&1 \\ 0&0&0&1 \\ 1&1&1&0 \end{bmatrix}$ node 2 → 4

$A_2 = \begin{bmatrix} 0&0&0&1 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&1&1&0 \end{bmatrix}$ 1 → 4 → not symmetric

<无向> node 4 → node 2

<有向> 4→2 4→3.

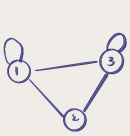
Ch.

Date /

- Networks are Sparse Graph (稀疏图)
  - $E \ll E_{max}$, $k \ll N-1$.
  - Edge list: 只记录存在连接的 node pair.
  - adjacency list: 记录与每个 node 相关的信息.

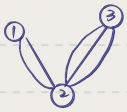
- unweighted: 无权重
  - $E = \frac{1}{2}\sum A_{ij}$
  - $\bar{k} = \frac{2E}{N}$

- weighted: 有权.
  - $E = \frac{1}{2}\sum \text{non-zero}(A_{ij})$
  - $\bar{k} = \frac{2E}{N}$.



- self-edges (self-loop)
  - $E = \frac{1}{2}\sum_{i \neq j} A_{ij} + \sum A_{ii}$

- multigraph
  - $E = \frac{1}{2}\sum \text{non-zero}(A_{ij})$
  - $\bar{k} = \frac{2E}{N}$



< Connectivity > 连通性. 任意两个节点之间都有路线.

- connected graph: Any two vertices can be joined by a path.

  A disconnected graph is made up by two or more connected components. 连通块.

* The adjacency matrix of a network with several components can be written in a block-diagonal form, so that non-zero elements are confined to squares, with all other elements being zero. 分块对角

有向图中，任意两个点都可以相互触达.

- Strongly: has a path from each node to every other node and vice versa (A→B, B→A)
- weakly: is connected if we disregard (忽略) the edge directions.

- Strongly connected components (SCCs): 强连通块.
  - ↳ can be identified, but not every node is part of a non-trivial strongly connected component.

§03. Traditional Methods for ML in Graphs

- Design features for nodes/links/graphs 抽取 D 个特征. 输出为 D 维向量. features: 节点属性 / 结构特征 — 连接信息.
- Obtain features for all training data

- Traditional ML pipeline uses hand-designed features (人工构造的特征 (特征工程)) on undirected graphs. 无向图.

<Goal>: Make predictions for a set of objects

- Features: D-dimensional vectors
- Objects: Nodes, edges, sets of nodes, entire graphs.
- Objective function: What task are we aiming to solve?

< Node-level > 节点层面. $f: V \longrightarrow \mathbb{R}$

- Characterize the structure and position of a node in the network.

without capturing importance ← ① Node degree: 度数 → treats all neighboring nodes equally.

take node importance ← ② Degree Centrality: 重要度. [Adjacency Matrix] × [Column Vector] = [Degree Centrality].

(Eigenvector / Betweenness / Closeness ...)

Ch.

Date /

< Node Centrality > 重要度 / 质量. → node importance in a graph

- Eigenvector centrality: 特征向量.

A node v is important if surrounded by important neighbouring nodes $u \in N(u)$.

↓ sum of the centrality of neighbouring nodes:

recursive 递归 ⇐ $C_v = \frac{1}{\lambda} \boxed{\sum_{u \in N(v)} C_u}$ 相邻节点的重要度之和.

常数 normalization constant. → largest eigenvalue of A.

$\lambda c = Ac$ { A: Adjacency; c: Centrality vector ⇒ Eigenvector of A.; λ: Eigenvalue

By Perron - Frobenius Theorem: largest eigenvalue λ_{max} (C_{max}) is always postive and unique.

- Betweenness centrality

A node is important if it lies on so many shortest paths between other nodes.

$C_v = \sum_{s \neq v \neq t} \frac{\#(\text{shortest paths btw } s \& t \text{ that contains } v)}{\#(\text{shortest paths btw } s \& t)}$ 其中有多少对节点的最短路径通过 v.

除节点 v 之外 两两节点对数

$C_A = C_B = C_E = 0$.

$C_C = 3$.

(A-C-B, A-C-D, A-C-D-E)

$C_A = \frac{1}{2+1+2+3} = \frac{1}{8}$

$C_D = \frac{1}{2+1+1+1} = \frac{1}{5}$

- Closeness Centrality (去哪都近)

A node is important if it has small shortest path lengths to all other nodes.

$C_v = \frac{1}{\sum_{u \neq v} \text{shortest path length btw } u \& v}$ (节点 v 到其他节点 u 最短路径长度之和)

< Clustering Coefficient >: 集群系数

↳ measures how connected v's neighboring nodes are:

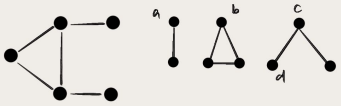
counts the triangles in the ego-network (自我中心网络)

$e_v = \frac{\#(\text{edges among neighboring nodes})}{\binom{k_v}{2}} \in [0,1]$

v 节点相邻节点也相连个数 (三角形个数)

→ v 节点相邻节点两两对数.

counts pre-specified subgraphs ← < Graphlet >



- Graphlet Degree Vector (GDV): A count vector of graphlets rooted at a given node.

↳ 描述 u 节点邻域的信息.

small subgraphs that describe the structure of node u's network neighbourhood.

provides a measure of a node's local network topology. 比较两个节点的 GDV 向量, 计算两点的相似度.

- Induced Subgraph 导出子图 ≠ 同构.

由该图顶点的一个子集, 和该图中两端点在该子集中的所有边的集合. that subset.

formed from a subset of vertices and all of the edges connecting the vertices in

Isomorphism 同构

↳ 两个图之间的完全的对应关系.

Two graphs which contain the same number of nodes connected in the same way.

▲ Importance-based:

- node degree → number of nodes
- node centrality measure → importance.

Structure-based:

- node degree
- clustering coefficient → connected
- Graphlet counter vector → occurrences

Ch.
Date /

PART 2. Link Prediction Task and Features
连接层面的特征工程

- The task is to predict **new links** based on the existing links. 通过已知连接，补全未知连接.

  node pairs (with no existing links) are ranked, and top k node pairs are predicted.

{ 直接提取 link 的特征，把 link 变成向量 ✓
{ 把 link 两端节点的 D维向量拼在一起 ✗ → 丢失 link 本身的连接结构信息.

客观静态图 (蛋白质、分子)

① Links missing at random:
 └ remove a random set of links and then aim to predict them.

随时间变换的图. (论文引用、社交网络).

② Links over time:
 Given $G[t_0, t_0']$ (时间段区间) a graph defined by edges up to time t_0'. output a ranked list L of edges (not in $G[t_0, t_0']$) that are predicted to appear in time $G[t_1, t_1']$.

- evaluation { $n = |E_{new}|$: # new edges that appear during the test period $[t_1, t_1']$
  { Take top n elements of L and count correct edges. 比较预测出的 top n 个连接和真实的 n 个连接.

- Methodology: 提取连接特征 → D维向量 → 分数 c(x,y)

① For each pair of nodes (x,y) computes score c(x,y)
② Sort pairs (x,y) by the decreasing score c(x,y) ← 降序排序.
③ Predict top n pairs as new links.
④ See which of these links actually appear in $G[t_1, t_1']$.

<Link-level Features>:
{ Distance-based feature (距离)
{ Local neighborhood overlap (局部连接)
{ Global neighbourhood overlap (全图连接)

- Shortest-path distance between two nodes. ← 只看最短路径长度，而忽略个数.
 └ this does not capture the degree of neighbourhood overlap.

- Captures # neighbouring nodes shared between 2 nodes v_1 & v_2:
 { Common neighbours: $|N(v_1) \cap N(v_2)|$ — 共同好友个数
 { Jaccard's coefficient: $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$ — 交并比.
 { Adamic-Adar index: $\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$ — 共同好友的重要性.

<Limitation>: Metric is always 0 if the two nodes do not have any neighbours in common.
 $N_A \cap N_E = \emptyset$, $|N_A \cap N_E| = 0$. → 但未来仍有可能连接.

- Katz index: count the number of walks of all lengths between a given pair of nodes.
 └ 节点 u和v之间长度为k的路径个数.

$S_{v_1 v_2} = \sum_{l=1}^{\infty} \beta^l A^l_{v_1 v_2}$

$0<\beta<1$: discount factor

$S = \sum_{i=1}^{\infty} \beta^i A^i = (I - \beta A)^{-1} - I$, where $(I-\beta A)^{-1} = \sum_{i=0}^{\infty} \beta^i A^i$

by geometric series of matrices.
类似等比数列求和.

power of Adjacency matrix: $A_{uv} = 1$ if $u \in N(v)$. 若u是v的邻居节点，则为1，否则为0.

① let $P^{(k)}_{uv}$ = # walks of length k between u and v.
② $P^{(k)} = A^k$
 路径个数 = A^k_{uv} 矩阵的第u行第v列元素.
③ $P^{(1)}_{uv}$ = # walks of length 1 (direct neighbourhood) between u and v = A_{uv}.
④ $P^{(2)}_{uv}$: sum up these # walks across u's neighbours.

$P^{(2)}_{uv} = \sum_i A_{ui} * P^{(1)}_{iv} = \sum_i A_{ui} * A_{iv} = A^2_{uv}$

与u隔1步的节点i; i 再与v隔1步.

Ch.

Date /

PART 3. Graph-Level & Graph kernels

全图层面的特征工程

图的相似性问题指判断2图是否同构. (特征 + 结构)

Characterize the structure of an entire graph. 提取出的特征能反映全图的结构特点.

<Kernel Method> 将低维空间的非线性划分问题，转化为高维空间的线性划分问题.

└ design kernels instead of feature vectors.

- Graph kernels: measure similarity between 2 graphs. { Graphlet kernel; Weisfeiler-Lehman kernel. } → Bag-of-*.

* Goal: Design graph feature vector $\phi(G)$ D维向量.

Key Idea: Bag-of-Words (BoW) → word counts (一篇文章中对应词出现的个数作为特征)

Naive extension to graph: Regard nodes as words. 将图看做文章，节点看做单词.

↳ 只看每个节点有没有，不看连接结构.

Bag of Node Degrees. ⇒ 只看 Node degree 个数，看节点，不看连接结构.

<Graphlet Feature>

Let $G_k = (g_1, g_2, \cdots g_{n_k})$ be a list of graphlets of size k.

k=3, 4 graphlets k=4, 11 graphlets.

Count the number of different graphlets in a graph.

* 计数全图 Graphlet 个数，而非特定节点邻域 Graphlet 个数.

Limitation: expensive!

子图匹配：n^k (多项式复杂)

worst-case: 子图同构测试.

subgraph isomorphism test (NP-hard)

⇐ Given 2 graphs G & G': graphlet kernel is computed as:

$K(G, G') = f_G^T f_{G'}$ (G和G' Graphlet Count Vector 数量积)

Q: If G and G' have different sizes, → skew the value.

S: normalize each feature vector: (归一化)

$h_G = \frac{f_G}{\text{Sum}(f_G)}$ $K(G, G') = h_G^T \cdot h_{G'}$

Computationally efficient. (linear in #(edges)).

<Weisfeiler-Lehman kernel> — Colour Refinement

└ use neighbourhood structure to iteratively enrich node vocabulary.

Given: A graph G with a set of nodes V.

⇐ ① Assign an initial colour $c^{(0)}(v)$ to each node v.

② Iteratively refine node colours by: → 节点颜色，同图节点颜色.

$c^{(k+1)}(v) = \text{HASH}(\{c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)}\})$.

↑ maps different inputs to different colors.

③ After k steps of colour refinement.

$c^{(k)}(v)$ summarizes the structure of k-hop neighbourhood.