Complex-Network Modelling and Inference

Lecture 18: Application: Graph Matching

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
https://roughan.info/notes/Network_Modelling/

School of Mathematical Sciences,
University of Adelaide

March 7, 2024

Section 1

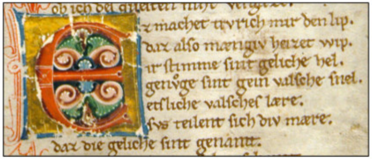
Graph Comparison: Edit Distance

Graph Edit Distance (GED)

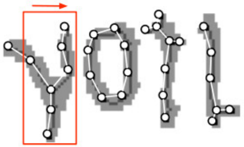
- Graph edit distance $d(H, G)$ between graphs H and G
  - defined as the minimum number of operations you need to do to convert from G to H
  - typical allowed operations
    - node insertions and deletions
    - edge insertions and deletions
    - rewiring?
    - label changes?
    - edge splitting (subdivision) or contraction
- Allows comparison between graphs
  - where the graphs represent data, we can do pattern matching

GED application

- fingerprint recognition [NB05]
  - graphs express relationships between features: whorls, loops, ...
  - graph distance used because exact print taken from fingers differ in part of finger, and “squashing”
- handwriting recognition [FSF+13]
  - handwriting graph roughly looks like the actual writing


ob ich der [illegible]
z machet tuvrich mir den lip.
dar alſo manigiv herzet wip.
ir ſtimme ſint geliche hel.
genvge ſint gein valſche ſnel.
etſliche valſches lære.
ſus teilent ſich div mære.
dar die geliche ſint genant.

(a) original image



(b) skeleton graph, sliding window

Edit distance problems

- Calculating GED is NP-complete
  - approximations exist
- Assumes labelled nodes
  - with unlabelled graphs, even testing isomorphism is difficult.

Section 2

(Unlabelled) Graph Matching [GM18]

Graph Isomorphism

Isomorphism expresses the fact that often the labels on a graph are arbitrary.

Definition

A *isomorphism* between two graphs G and H is a bijection[1] between the nodes $N(G)$ and $N(H)$ of the two graphs

$$f : N(G) \to N(H),$$

such that any two vertices $u, v \in N(G)$ are adjacent if and only if $f(u), f(v) \in N(H)$ are adjacent. We say two graphs are *isomorphic* is such a function exists.

So an isomorphism is a change of labels, preserving the edges.

[1] Loosely, a bijection is a function between two sets where each element of one set is paired with exactly one element of the other set and visa versa.

Graph Isomorphism

Questions:

- Are two (unlabelled) graphs isomorphic?
  - this is hard in general, but we can often show they aren't using simpler properties, such as the node degree distribution.
- Is one (of two graphs) isomorphic to a subgraphs of the other?
  - even harder (this is NP-complete)
  - or even hard: find the maximum common subgraph
- How far are two graphs from isomorphic (given edit distance)?
  - even harder, because I have to find the closest match of all possible matches

If these are all so hard, then what can we hope to do for real data (where the data can be large)?

Applications

Two or more networks, correlated in some way.

- Biochemistry: protein-protein and protein-gene interaction networks are of great interest. *Orthologs* are genes that have evolved (changed) between two species, but still hold the same function (and in our application, the same functional relationships). Similarly proteins may be chemically different, but have the same function.
- Online social networks: multiple networks exist, and information passes between networks, not just inside networks. Any application looking at such networks would be enhanced by matching accounts in multiple networks.
- Fraud detection: fraudsters try to hide by changing their names, but their relationships (*e.g.*, as seen in telephone calls, emails or online social networks) are the same.

We want an approximate match between the two.

Names

The idea has been reinvented several times, so has many names

- random graph isomorphism
- network (graph) alignment
- network (graph) reconciliation
- network (graph) matching
- network (graph) de-anoymisation

The Problem

Two (or more) networks, correlated in some way. For the sake of argument, let us propose the model

- There is an underlying graph $G = (N, E)$
- Each observed network $G_i = (N_i, E_i)$ has
  - the same set of nodes $N_i = N$
    The labels are unknown to us, but for the sake of the model, assume they are the same.
  - The edges are formed as follows
    If $e \in E$ then
    $$P\{e \in E_i \mid e \in E\} = 1 - s_i$$
    If $e \notin E$ then
    $$P\{e \in E_i \mid e \notin E\} = r_i$$
- Intuition
  - Think of r and s as probabilities of Type I and Type II errors.
  - If $r = s = 0$ then G and G_i are isomorphic.
  - We don't know r or s, but we presume they are relatively small, *i.e.*, $r \ll P\{e \in E\}$, so there aren't a vast number of false edges.

What can we work out?

Assume G is a Gilbert-Erdős-Rényi random graph $G(n,p)$ then

- G_i is a Gilbert-Erdős-Rényi random graph $G(n,p_i)$ where

$$\begin{aligned} p_i &= P\{e \in E_i \mid e \in E\}P\{e \in E\} + P\{e \in E_i \mid e \notin E\}P\{e \notin E\} \\ &= (1-s_i)p + r_i(1-p) \end{aligned}$$

- We can think of G_1 and G_2 as two correlated random graphs (for simplicity take $r_i = r$ and $s_i = s$, and take r, s and p small)

$$\begin{aligned} P\{e \in G_1 \;\&\; e \in G_2\} &= (1-s)^2 p + r^2(1-p) \\ &\simeq (1-s)^2 p \\ P\{e \in G_1 \;\&\; e \notin G_2\} &= (1-s)sp + r(1-r)(1-p) \\ &\simeq r(1-r)(1-p) \\ P\{e \notin G_1 \;\&\; e \in G_2\} &\simeq r(1-r)(1-p) \\ P\{e \notin G_1 \;\&\; e \notin G_2\} &= s^2 p + (1-r)^2(1-p) \\ &\simeq (1-r)^2(1-p) \end{aligned}$$

What can we work out?

Assume G is a Gilbert-Erdős-Rényi random graph $G(n, p)$ then

- Covariance of G_1 and G_2 (treating edges as 1, and non-edges as zero, then $\mu_i = p_i$), and taking $p_i = p$

$$\begin{aligned} cov(G_1, G_2) &\simeq (1-s)^2 p(1-p)^2 + 2r(1-r)(1-p)(-p)(1-p) \\ &\quad +(1-r)^2(1-p)(-p)^2 \\ &= (1-p)p\Big[(1-s)^2(1-p) - 2r(1-r)(1-p) + (1-r)^2 p\Big] \end{aligned}$$

- Makes sense:
  - If $s = r = 0$ then $cov(G_1, G_2) = var(G) = p(1-p)$.
  - If $r = 0$ and $s > 0$, then $cov(G_1, G_2) = p(1-p)[(1-s)^2(1-p) + p]$ so as s increases, the covariance decreases, *i.e.*, they are less correlated.

Discussion

- We don't know (and never see directly) the underlying graph G
- In real applications, often $N_i \neq N$
- In real applications, often there are some labels that
  - already match, *e.g.*, some social network names match, call these *beacons* or *landmarks*
  - have some "likelihood" of a match, *e.g.*, names might match, minus some spelling errors, or genes sequences may resemble each other (be "homologous") because of a (recent) common ancestor
- In real applications, the underlying network is unlikely to be Gilbert-Erdős-Rényi

These are research topics.

Two Main Approaches

There are MANY algorithms [GM18], with two main approaches:

- *Local:* find some quantity that is (locally) conserved by isomorphism, and try to match regions that having matching quantities
- *Global:* try to match the whole graph, *e.g.*, through optimising some match criteria, or minimising a cost function

Global is much harder (computationally and conceptually) so we focus on the former.

Local alignment features

- Homologous (nodes with similar descriptors) in the two networks
- Node degree is preserved by isomorphism
  - of course, many nodes may have same (or similar) degree
  - focus on high-degree nodes, *e.g.*, [BES80]
- Look for more extensive motifs: *e.g.*, certain small subgraphs (*e.g.*, almost cliques)
  - mine-and-merge approaches
    1. mine graphs to identify subgraphs of interest
    2. merge the components

    Even though merge is NP-hard, it is on a smaller graph of subgraphs
- Greedy approaches find some key "beacons" or "seeds", which can be identified (*e.g.*, highest degree nodes), and grow from these
- Divide-and-conquer: work out ways to break graphs into pieces which can be matched more quickly
- Lots of hybrid variants

NetAligner [PCA12]

1. Construct a (weighted) *alignment* graph (a common strategy [PCA12, CMG$^+$12])
   - its nodes are all pairs of *possible* matches
     - ★ nodes can be given a weight (a probability of being a match) based on scoring of local metrics, *e.g.*, node degree, or number of length 2 paths through the node [CMG$^+$12], or other data
   - its edges are based on edges that are likely given the pairwise match (also weighted)

   Nodes and edges below a threshold are pruned
2. Clustering:
   - *e.g.*, extract from the alignment graph connected components of k-nodes
   - score these (sum of scores of nodes and edges)
3. Select "seeds" and expand
   - greedy: select highest score first, ...

Quality of results

No panacea

1. some algorithms can't deal with noise (larger s, r)
2. some approaches can't deal with large networks
3. depends what results you need: do you need an exact match, or does some approximation work?

Reverse Application

If I was to try to anonymise a network dataset, how many edges and nodes should I add so that the anonymisation can't be reversed?

Further reading I

Lászó Babai, Paul Erdös, and Stanley M. Selkow, *Random graph isomorphism*, SIAM J. Comput. **9** (1980), no. 3, 628–635.

G. Ciriello, M. Mina, P.H. Guzzi, M. Cannataro, and C. Guerra, *AlignNemo: A local network alignment method to integrate homology and topology*, PLoS ONE **7** (2012), no. 6, `https://doi.org/10.1371/journal.pone.0038107`.

Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke, *A fast matching algorithm for graph-based handwriting recognition*, pp. 194–203, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Pietro Hiram Guzzi and Tijana Milenković, *Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin*, Briefings in Bioinformatics **19** (2018), no. 3, 472–481.

Further reading II

Michel Neuhaus and Horst Bunke, *A graph matching based approach to fingerprint classification using directional variance*, pp. 191–200, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

R.A. Pache, A. Céol, and P. Aloy, *Netaligner – a network alignment server to compare complexes, pathways and whole interactomes.*, Nucleic Acids Res. **40** (2012), PMCID: PMC3394252.