Statistical Data Analysis

Dr. Jana de Wiljes

11. Januar 2022

Universität Potsdam

What is a graph (formally)?

The objects on the following slides will play a major role in this course.

- $G = (V, E, \omega)$, where $\boxed{V \neq \emptyset}$ is a set (called the **vertex set**), $E \subset \binom{V}{2} = \{\{u,v\} : u, v \in V\}$ (called the **edge set**) and $\omega : E \to \mathbb{R}^+$, is called a **(weighted) graph**

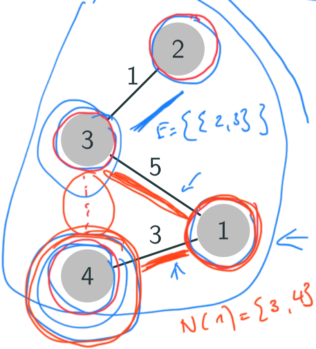
$\{\{,\}\{,\}\ldots\}$

- usually we choose (or rename) $V = \{1, 2, \ldots, n\}$ and use the notations $ij = \boxed{\{i,j\}}$ for $\{i,j\} \in E$ and $\omega_{ij} = \omega(ij)$

$\omega(\{i,j\})$

- for every $i \in V$ define $N(i) := \{j \in V : ij \in E\}$, called the **neighbourhood** of i (in G); elements of $N(i)$ are called **neighbours** of i (those elements are **adjacent** to i)

$(\{a,b,c,d\})$



$w(23) = 1$,

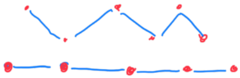
$\boxed{N(4) = \{1\}}$, $N(4) = \{1,3\}$

$\boxed{d(1)} = |\underbrace{\{3,4\}}| = 2$

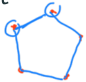
degree

Graph classes

Well known graph classes are:

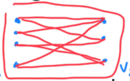


- the **path graph** P_n has vertex set $\{1, 2, \ldots, n\}$ and edge set $\{\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}\}$
- the **cycle graph** C_n has vertex set $\{1, 2, \ldots, n\}$ and edge set $\{\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}, \{n, 1\}\}$



- the **complete graph** K_n consists of n vertices which are all adjacent to each other

- the **complete bipartite graph** $K_{m,n}$ has two sets V_1 and V_2 of vertices of sizes m and n, such that the edge set consists of all possible edges between V_1 and V_2



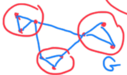
$V_1 \cup V_2 = V$

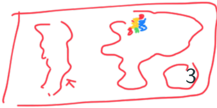
$V_1 \cup V_2 \cup V_3 = V$

k-partit

A set of vertices in a graph which are all adjacent to each other (they **induce** a complete (sub)graph), is called **clique**.

The graph $K_{1,n}$ is called a **star**.



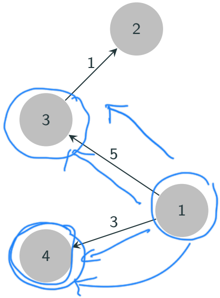


What is a digraph (formally)?

$\{i,j\} = \{j,i\}$

Edges can have a direction.

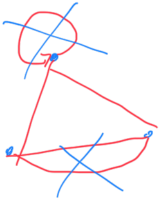
- $G = (V, E, \omega)$, where $V \neq \emptyset$ is a set, $E \subset V \times V$ (this is sometimes also called the **set of arcs**) and $\omega : E \to \mathbb{R}^+$, is called a **(weighted) digraph**
- for $(i, j) \in E$ the vertex i is called **predecessor** of j and j is called **successor** of i
- similar notation simplifications as before
- $N^+(i) := \{j \in V : (i, j) \in E\}$ is the **out-neighbourhood** of i, $N^-(i) := \{j \in V : (j, i) \in E\}$ is the **in-neighbourhood** of i
- $d^+(i) := |N^+(i)|$ is the **out-degree** of i and $d^-(i) := |N^-(i)|$ is the **in-degree** of i

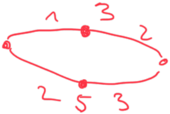


$N^-(3) = \{1\}$,
$N^+(4) = \emptyset$,
$d^+(1) = 2$,
$d^-(2) = 1$

Example of a multigraph

It is sometimes necessary to allow multiple edges between two vertices or a **loop** (a self-edge). In that case we use the term **multigraph**.





What is a hypergraph (formally)?

Sometimes more than two vertices need to form an edge (certain real life situations' have this property).

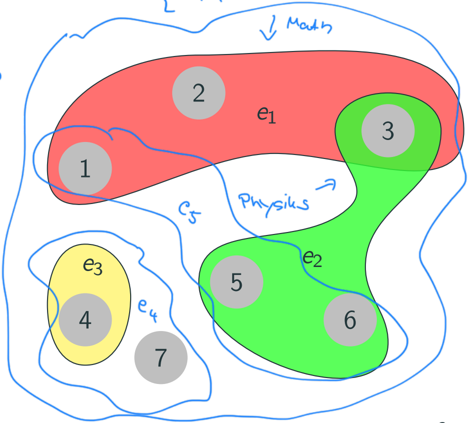
$E \subset \mathcal{P}(V)$

$\mathcal{P}(V) = \{\emptyset, \{1\}, \{2\}, \{\ \}, \{1\ 2\}\ \{\ldots\}$
$\{1, 3\ 2\} \ldots \}$

- natural generalisation is a **hypergraph** $H = (V, E)$, where

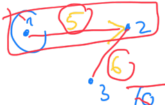
  $\{1, \ldots, 7\}$

  - $V \neq \emptyset$ is (also) a set, but
  - E can be an arbitrary subset (the elements are called **hyperedges**) of the power set $\mathcal{P}(V)$
- if all hyperedges are of the same size r, then H is called r-**uniform**



Storing graphs

Certain matrices and lists can be associated with a graph (we will see more examples later).



- **affinity matrix** $W(G)$:

$$w_{ij} = \begin{cases} \omega_{ij} & \text{if } \{i,j\} \in E, \\ 0 & \text{else.} \end{cases}$$

$$\begin{pmatrix} 0 & 5 & 0 \\ 5 & 0 & 6 \\ 0 & 6 & 0 \end{pmatrix}$$

- **adjacency matrix** $A(G)$: special case of $W(G)$, where $w_{ij} = 1$ for all $ij \in E$.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$N(v)$

- **adjacency list**:
  - associate list to every vertex containing its neighbours
  - call list of these lists adjacency list of the graph (treated differently in the literature)
  - not very useful for mathematical arguments
  - especially useful (for storing) when $A(G)$ is sparse

$|V| = n$

$|E| = m$

$\boxed{m << n}$

$n = 1000$

All the above constructions are valid for directed graphs.

$E = \{\{1, 999\}, \{7, 3\}, \{99, 2\}\}$

How to transform a digraph into a graph?

Consider the following three approaches.

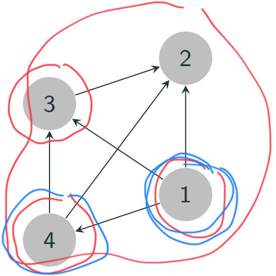
- ignore the directions
- carry out **cocitation coupling**
    - existence of common predecessors induce edges
    - weights are naturally given by number of common predecessors
- carry out **bibliographic coupling**
    - existence of common successors induce edges
    - weights are naturally given by number of common successors

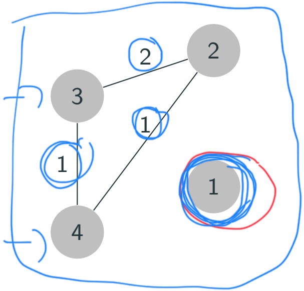
Cocitation coupling

A (undirected) graph is constructed via:

- **cocitation** c_{ij} of $i, j \in V$ is the number of common predecessors of i and j
- the **cocitation network** has vertex set V and an edge between i and j iff $c_{ij} > 0$
- it is also possible to obtain a weighted graph with weights c_{ij}
- note that $c_{ij} = \sum_{k=1}^{n} a_{ki} a_{kj}$, therefore $C = A^T A$

$\longrightarrow$

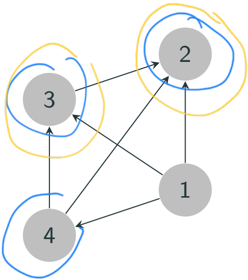




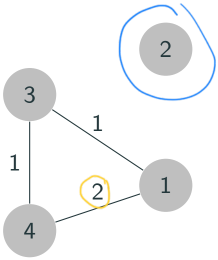
Bibliographic coupling

A (undirected) graph is constructed via:

- **bibliographic coupling** b_{ij} of $i, j \in V$ is the number of common successors of i and j
- the **bibliographic coupling network** has vertex set V and an edge between i and j iff $b_{ij} > 0$
- it is also possible to obtain a weighted graph with weights b_{ij}
- note that $b_{ij} = \sum_{k=1}^{n} a_{ik} a_{jk}$, therefore $B = AA^T$

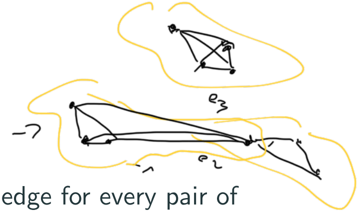


$\longrightarrow$

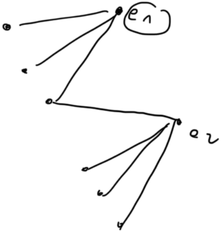


How to transform a hypergraph into a graph?

The following constructions are standard.



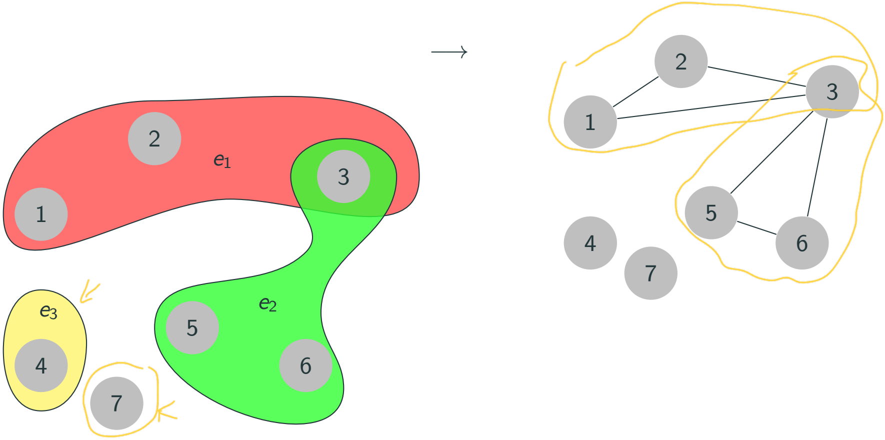
- clique expansion
  - the vertex set is V
  - each hyperedge e is replaced by an edge for every pair of vertices in e
  - this construction yields cliques for every hyperedge
- star expansion
  - vertex set is $\boxed{V \cup E}$
  - edge between u and e iff $u \in e$
  - every hyperedge corresponds to a star
- there are more...



Clique expansion

The clique expansion $G^x = (V^x, E^x)$ is constructed from $H = (V, E)$ via:

- $V^x = V$
- $E^x = \{\{i, j\} : \exists e \in E \text{ with } i, j \in e\}$



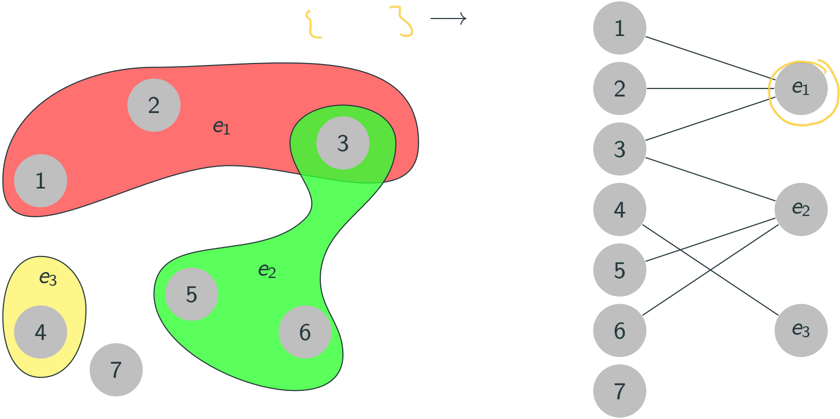
Star expansion

The star expansion $G^* = (V^*, E^*)$ is constructed from $H = (V, E)$ via:

$V = \{1, \ldots, 7, e_1, e_2, e_3\}$

- $V^* = \boxed{V \cup E}$
- $E^* = \{\{i, e\} : i \in e, e \in E\}$

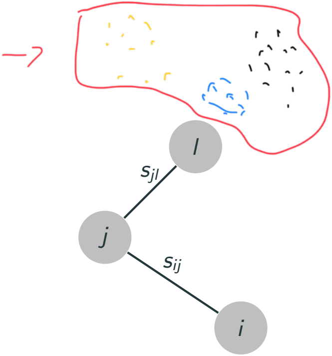
$\{\ldots\}$



What if data without network structure is given?

Solution: Build your own graph!

- given a set of data points $x_1, x_2 \ldots, x_n$ and some notion of similarity[1] $s_{ij} \geq 0$ between all pairs of data points x_i and x_j
- build graph $G = (V, E)$, where the vertex i represents the data point x_i, so $V = \{1, 2, \ldots, n\}$
- $\{i, j\} \in E$ if $s_{ij} > 0$
- edge weight $\omega_{ij} = s_{ij}$ (edge weights represent similarities)
- G is called **similarity graph** (although with this particular choice of edges it is often referred to as the **fully connected graph**)



graph for $\{x_i, x_j, x_l\}$ with $s_{ij}, s_{jl} > 0$ and $s_{il} = 0$

The ε-neighbourhood graph

The ε-**neighbourhood graph** is constructed as follows:

- vertices are data points
- fix some $\boxed{\varepsilon > 0}$
- connect all vertices whose similarities are smaller than ε
- since ε is usually small, values of existing edges are roughly of the same scale
- hence usually unweighted

The (mutual) k-nearest neighbour graph

The k-**nearest neighbour graph** is constructed as follows:

- vertices are data points
- fix some $k > 0$
- connect i to the k nearest (w.r.t. s_{ij}) k vertices via an edge starting at i
- obtain an undirected graph by ignoring the directions

The **mutual** k-**nearest neighbour graph** is constructed as follows:

- vertices are data points
- fix some k
- connect i to the k nearest (w.r.t. s_{ij}) k vertices via an edge starting at i

only if they are mutal

- obtain an undirected graph by deleting all non symmetric edges

Graph Partitioning and Community Detection

Difference

Graph Partitioning (GP)

- partition vertices into given number of groups
- sizes of groups are (roughly) fixed
- many edges inside groups, few edges between groups
- goal: dividing network into smaller more manageable pieces
- example:
  - numerical solution of network processes on a parallel computer

Community Detection (CD)

- partition vertices into groups
- sizes of groups are not fixed
- many edges inside groups, few edges between groups
- goal: understanding structure of a network
- examples:
  - collaboration
  - related web pages

Why is partitioning hard?

Problem

Partition vertex set into two parts (*graph bisection*).

n vertices into parts of sizes n_1 and n_2 ($n_1 + n_2 = n$):

$V = V_1 \cup V_2$

- $\frac{n!}{n_1!n_2!}$ possibilities (half of it if order is ignored and $n_1 = n_2$)
- using Stirling's formula $\boxed{n! \approx \sqrt{2\pi n}(n/e)^n}$ we get

$\binom{8}{4}$ 4   4

$$\frac{n!}{n_1!n_2!} \approx \frac{n^{n+1/2}}{n_1^{n_1+1/2} n_2^{n_2+1/2}}$$

- for a balanced partition ($n_1 \approx n_2$):

$$\boxed{\text{roughly } \frac{2^{n+1}}{\sqrt{n}} \text{ possibilities}}$$

V_1 V_2

Therefore, exhaustive search is usually unfeasible.