Graphs and Machine Learning

Jan-Hendrik de Wiljes

July 9, 2024

Introduction, Motivation & Basics

(Potential) Topics

- Graphs
  - What are they (mathematically)?
  - Where do they appear naturally?
  - Where do they not appear naturally?
  - Typical notations, notions, or concepts?
  - How to store them (as data structures)?
  - ...
- Graphs & Machine Learning
  - Appearance of graphs
  - How to make graphs appear, even when they are not there
  - Application problems (and corresponding tools)
    - Graph Partitioning
    - Community Detection
    - (Link/Property/...) Prediction
    - ...

Motivation

The following aspects[1] have become more important or complex in the last decades:

- social ties among friends
  - more possibilities: distant travel, global communication, digital interaction
  - not as local as before
- the information we consume
  - few with high-quality information
  - varying perspectives, reliabilities, and motivating intentions
- technological and economic systems
  - difficult to reason about
  - riskier to tinker with
  - localized breakdowns can turn into cascading failures or financial crises
- ...

[1] all belonging to the "complex 'connectedness' of modern society"

What are graphs/networks?

"In the most basic sense, a network is any collection of objects in which some pairs of these objects are connected by links." (Easley, Kleinberg)

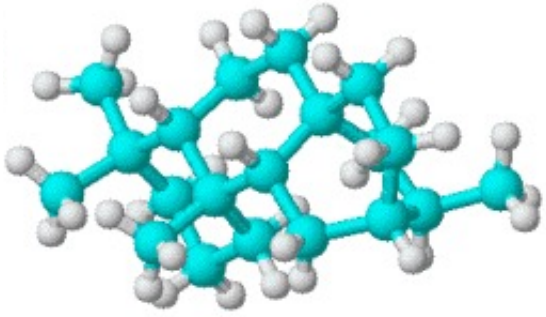
This "definition" is very flexible:

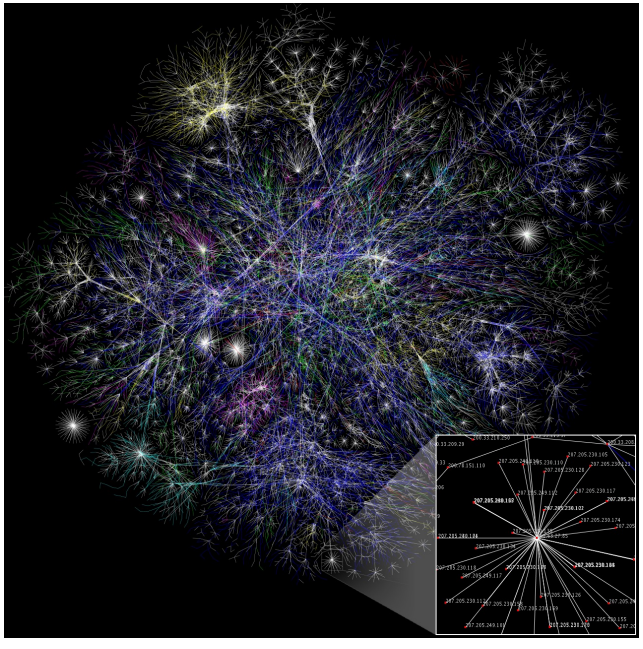
- many different forms of relationships/connections
- can be applied to various domains
- also shows how important and difficult it is to choose an appropriate model

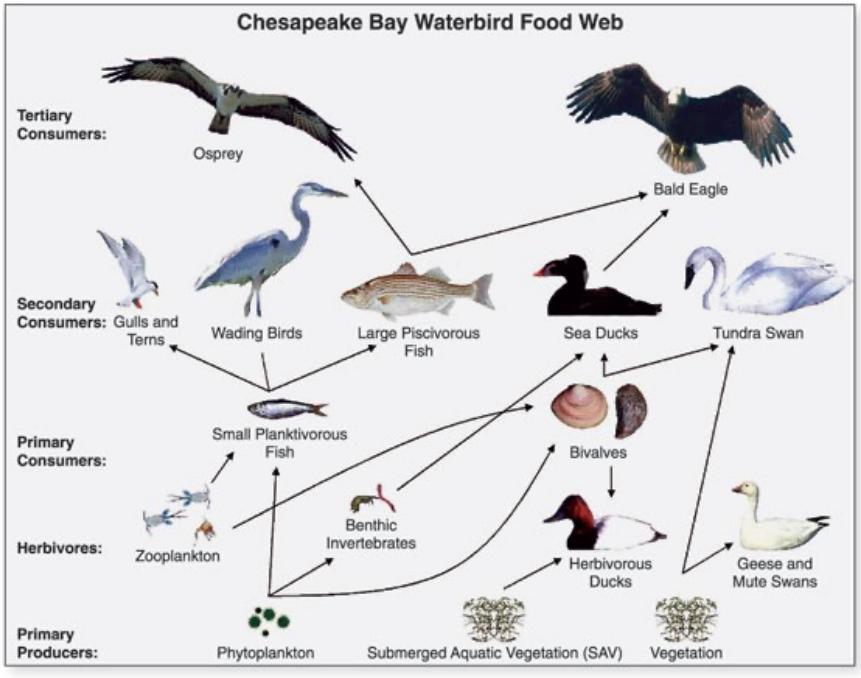
Graphs/Networks are difficult to study:

- often highly complex (number of possible networks grows exponentially)
- can be difficult to summarize the whole network

Some examples







Matthew C. Perry, Public domain, via Wikimedia Commons

Studying graphs/networks

The following aspects are of particular interest:

- structure of a graph/network
  - tightly-linked regions
  - connectedness (in the sense of structure)
  - influential objects (in general or inside tightly-linked regions)
  - reachability
- graph/network dynamics
  - connectedness (in the sense of behaviour, 'actions have consequences')
  - predicting new connections
  - failure of links or objects

Interesting properties

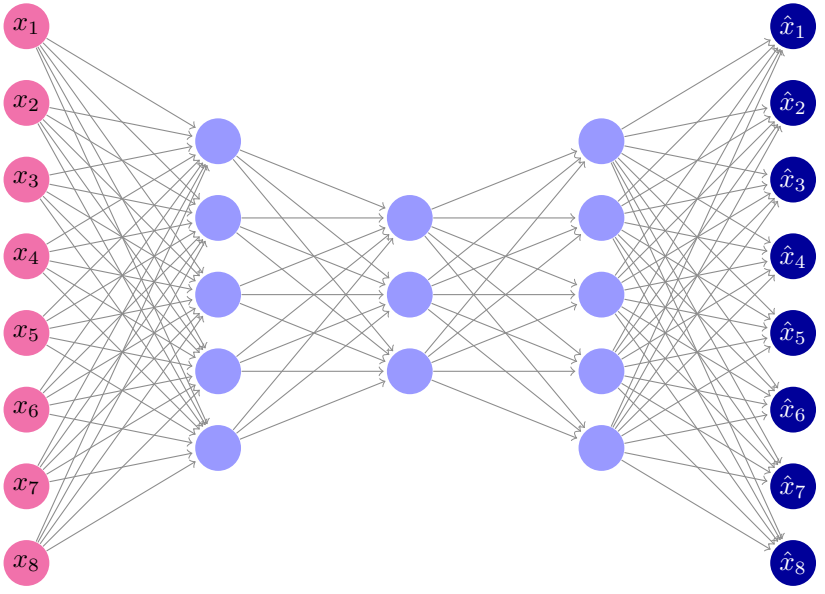
The following concepts are observed in many graphs/networks.

- small number of objects with very many links (these object are called hubs)
- small-world effect/phenomenon (any two objects are reachable by a small number of links)
- tightly-knit substructures
- ...

Data Science (in general)

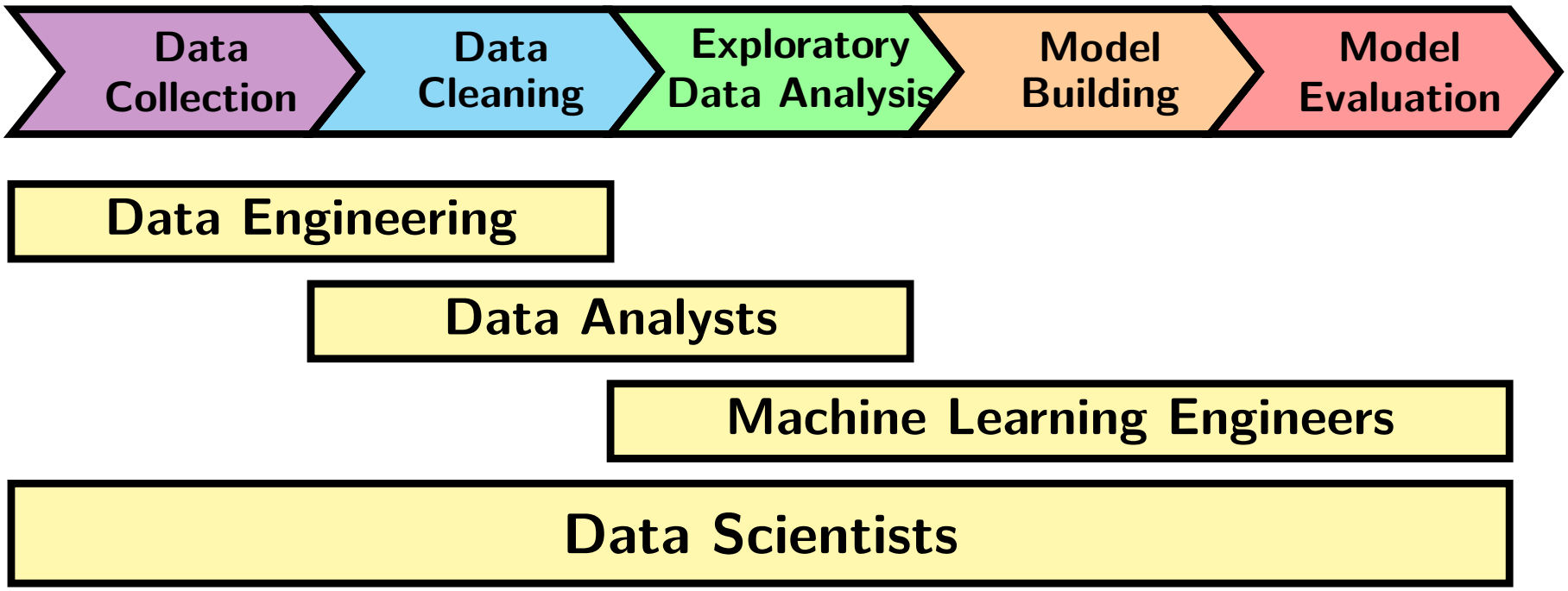
Data

Potential Deployable Model



How can we efficiently process the data to learn functions with a high prediction ability?

Overview Workflow

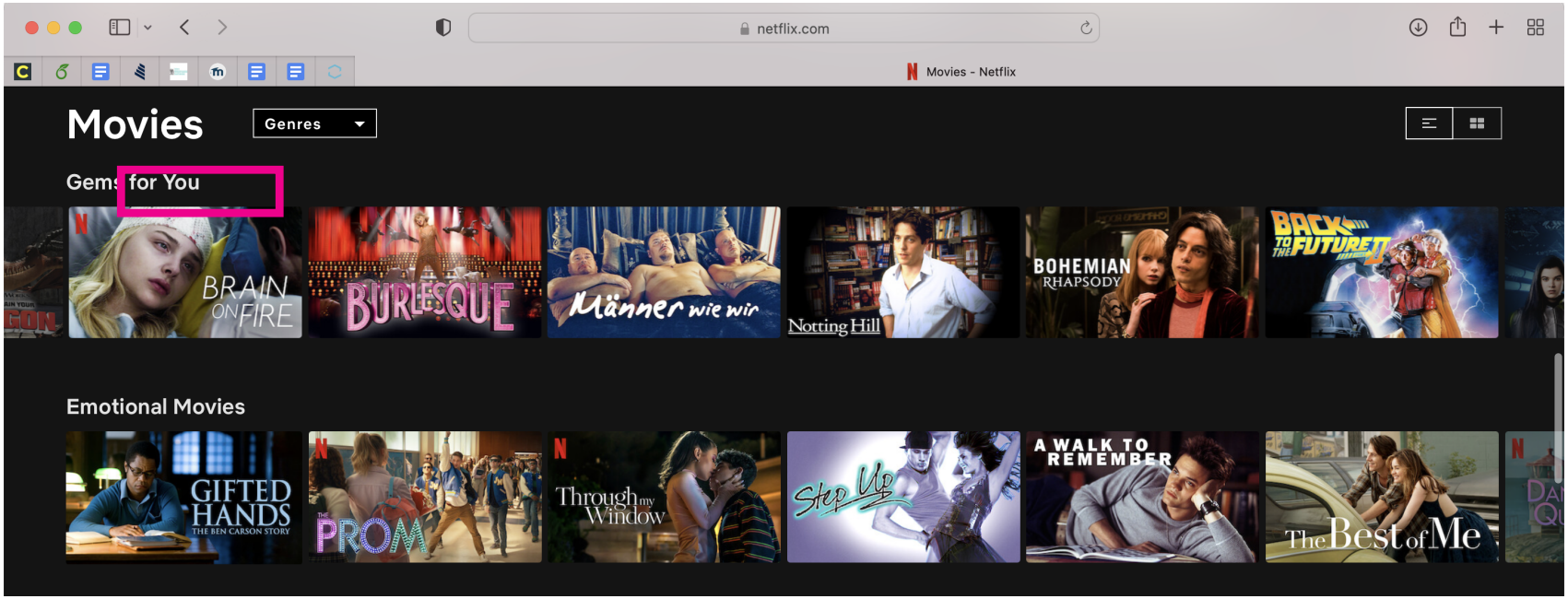


Data Collection

Key objectives:

- identify relevant data and associate data sources for considered problem /statement
- data acquisition might require long-term planing, i.e., space mission
- identify best quality data
- choose scraping software

Recommender System example: observe quantites such as rating, number of clicks or of times watched



Model Building

Key objective: Approximate function f, that describes the link between two random variables X and Y which have the joint distribution $\pi(z) = \pi(x, y)$
Choice of parametrization:

- choose model class $\mathcal{H}$
- and appropriate loss functional $l(y, h(x))$

Definition

For $h \in \mathcal{H}$ we define the **expected risk** as follows

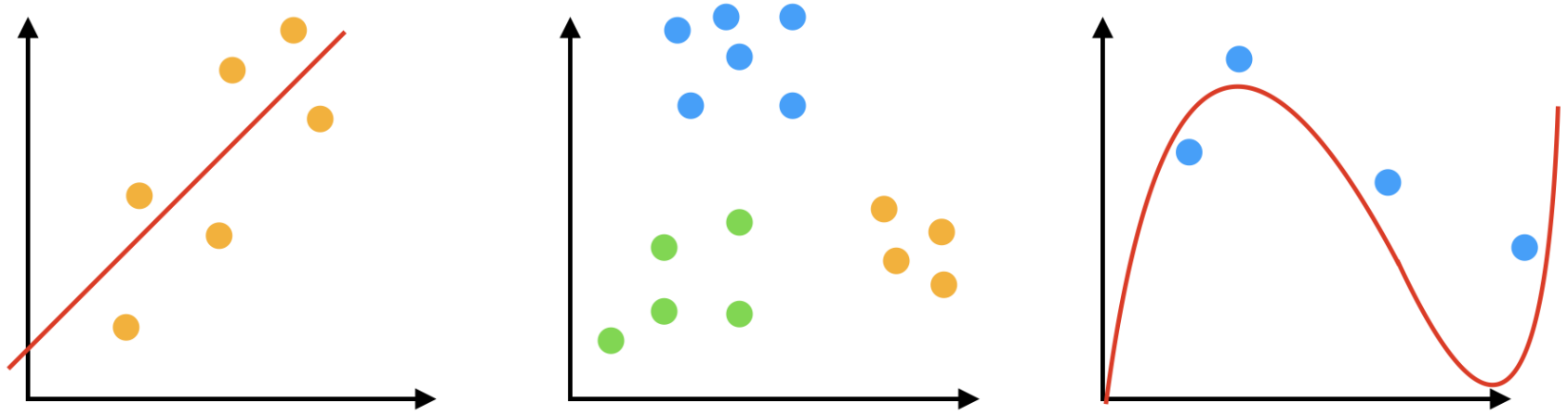
$$R(h) = \int_{\mathbf{Z}} l(y, h(x))\pi(z)dz$$

Approach: Want to find $h^* \in \mathcal{H}$ with

$$h^* = \arg\min_{h \in \mathcal{H}} R(h)$$

Empirical Risk

Given in practice: independent and identical distributed samples $S = \{(x_i, y_i)\}_{i=1}^{N}$ with $(x_i, y_i) \sim \pi(x, y)$ for $i \in \{1, \ldots, N\}$



Definition

For a given sample set S we define the corresponding **empirical risk** as follows

$$R_S(h) = \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i))$$

Empirical Risk-Minimizer

Definition

A learning algorithm $\hat{h}_N$ with $S = \{(x_i, y_i)\}_{i=1}^N$ where $(x_i, y_i) \sim \pi(x, y)$ of the form

$$\hat{h}_N \in \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(y_i, h(x_i))$$

is called **Empirical Risk-Minimizer**.

Supervised learning

Linear regression with regularization: data $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^h$ and loss function:

$$l(x_i, h_w(x_i)) = \frac{1}{2}\|y_i - wx_i\|_2^2 + \|w\|_2^2$$

Unsupervised learning

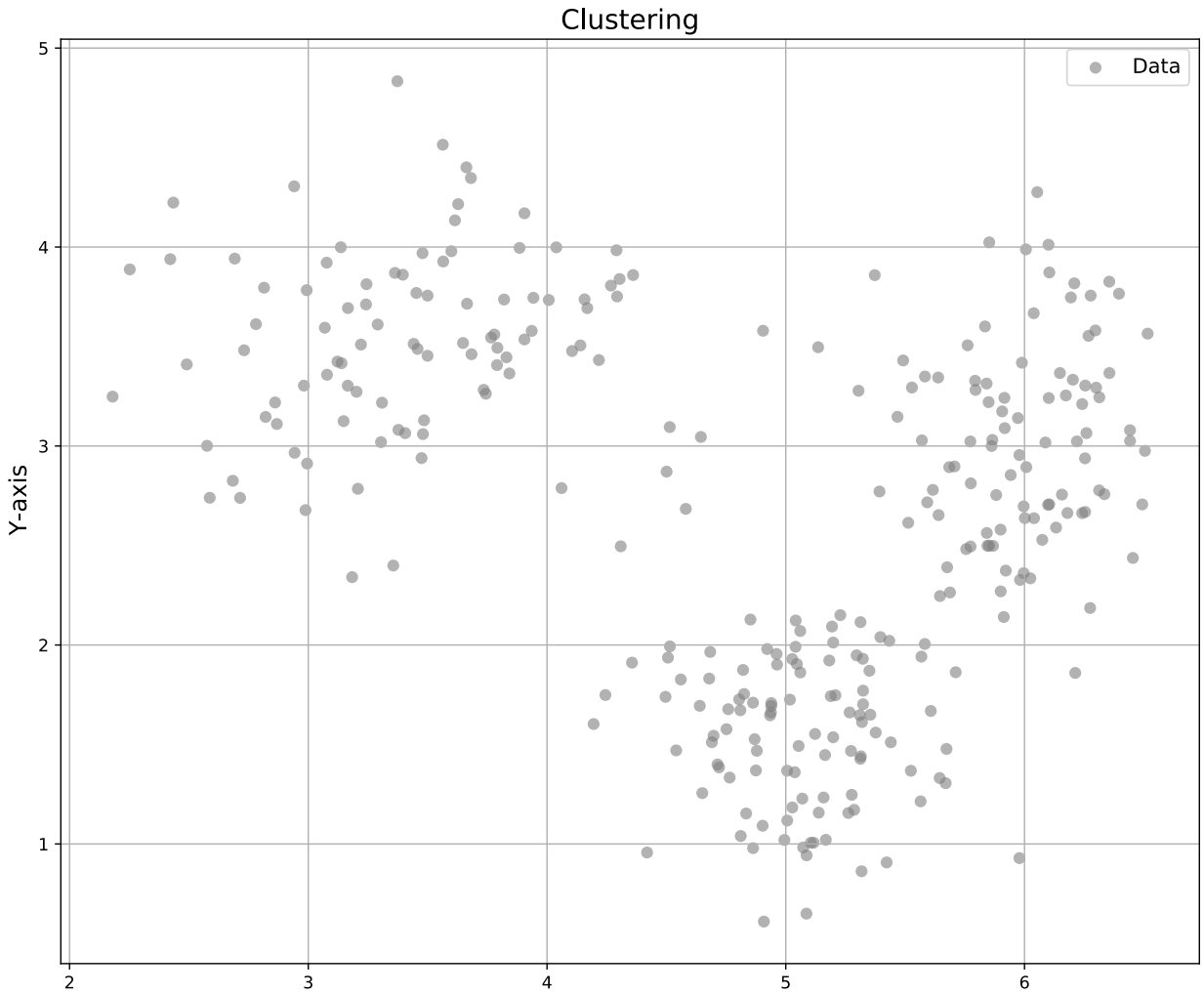
k-**Means:** data $x_i \in \mathbb{R}^d$, no labels y_i and loss function:

$$l(x_i, h_w(x_i)) = \min_{w_k} \frac{1}{2}\|x_i - w_k\|_2^2,$$

where $w_1, \ldots, w_k$ are cluster centers .

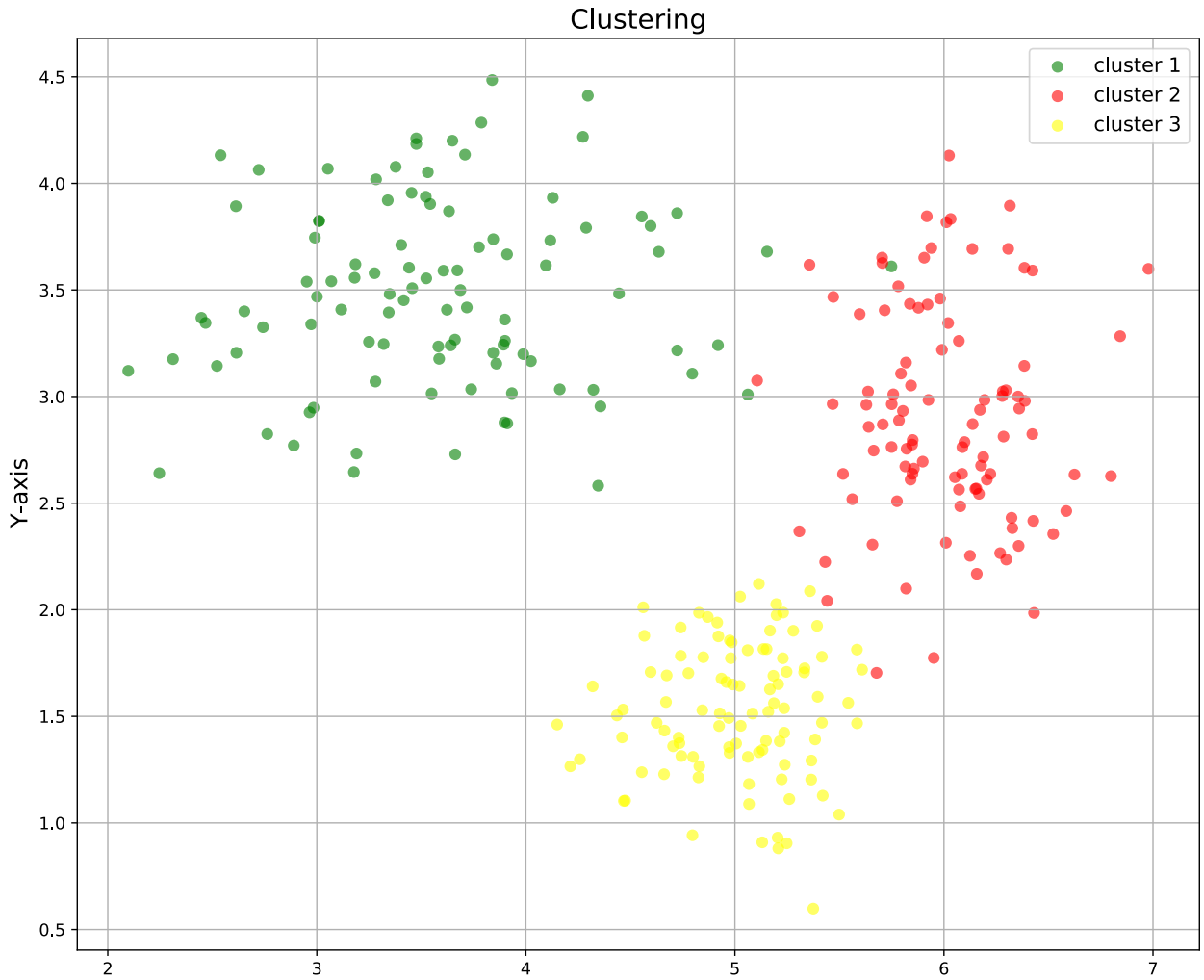
Clustering

Given: unlabed data
Goal: find pattern (described by parameters $C = \{c_1, \ldots, c_k\}$) and reduce loss $l(x_i, h_C(x_i))$



Clustering

Given: unlabed data
Goal: find pattern (described by parameters $C = \{c_1, \ldots, c_k\}$) and reduce loss $l(x_i, h_C(x_i))$



K-means

Algorithm 1 K-Means Clustering Algorithm

Input: Data points $X = \{x_1, x_2, \ldots, x_n\}$, number of clusters k
Output: Cluster centroids $C = \{c_1, c_2, \ldots, c_k\}$ and cluster assignments for each data point
Initialize cluster centroids $C = \{c_1, c_2, \ldots, c_k\}$ randomly from the data points
repeat
    for each data point $x_i \in X$ **do**
        Compute the distance $d(x_i, c_j)$ between x_i and each centroid c_j
        Assign x_i to the cluster with the nearest centroid:

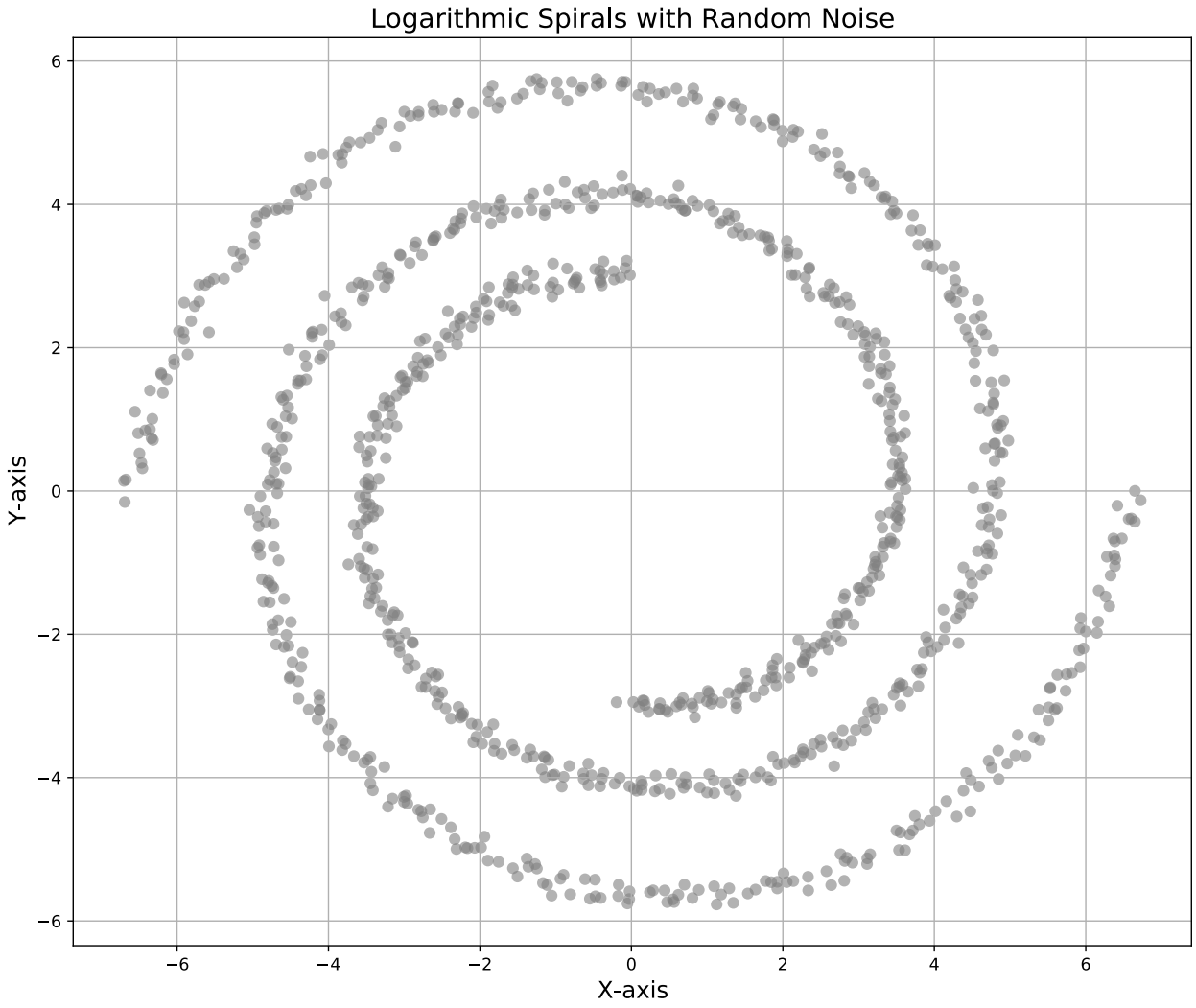
$$\text{cluster}(x_i) = \arg\min_j d(x_i, c_j)$$

    end for
    for each cluster j **do**
        Update the centroid c_j to be the mean of all data points assigned to cluster j:

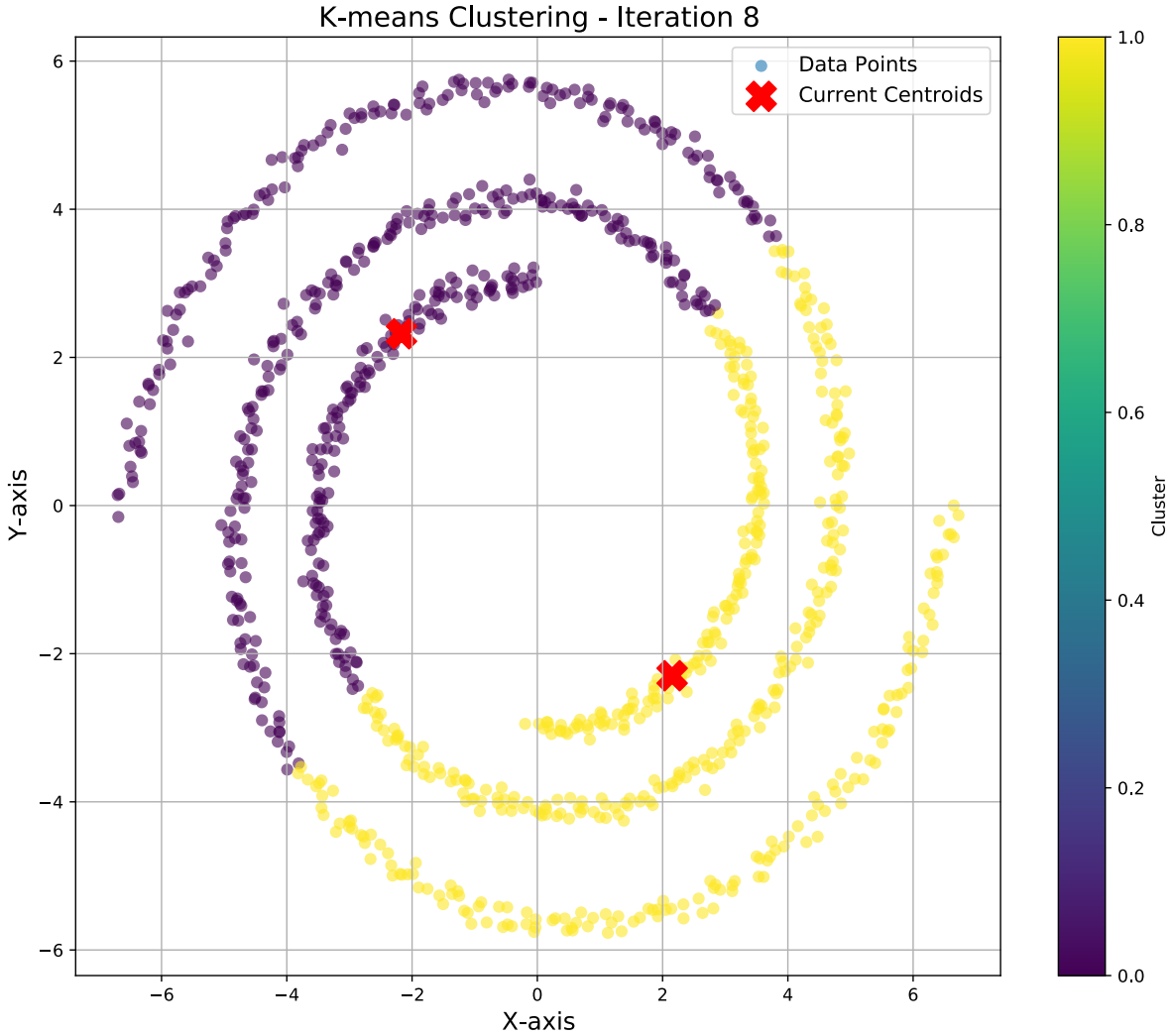
$$c_j = \frac{1}{|\{x_i \mid \text{cluster}(x_i) = j\}|} \sum_{\{x_i | \text{cluster}(x_i) = j\}} x_i$$

    end for
until convergence
return C, cluster assignments

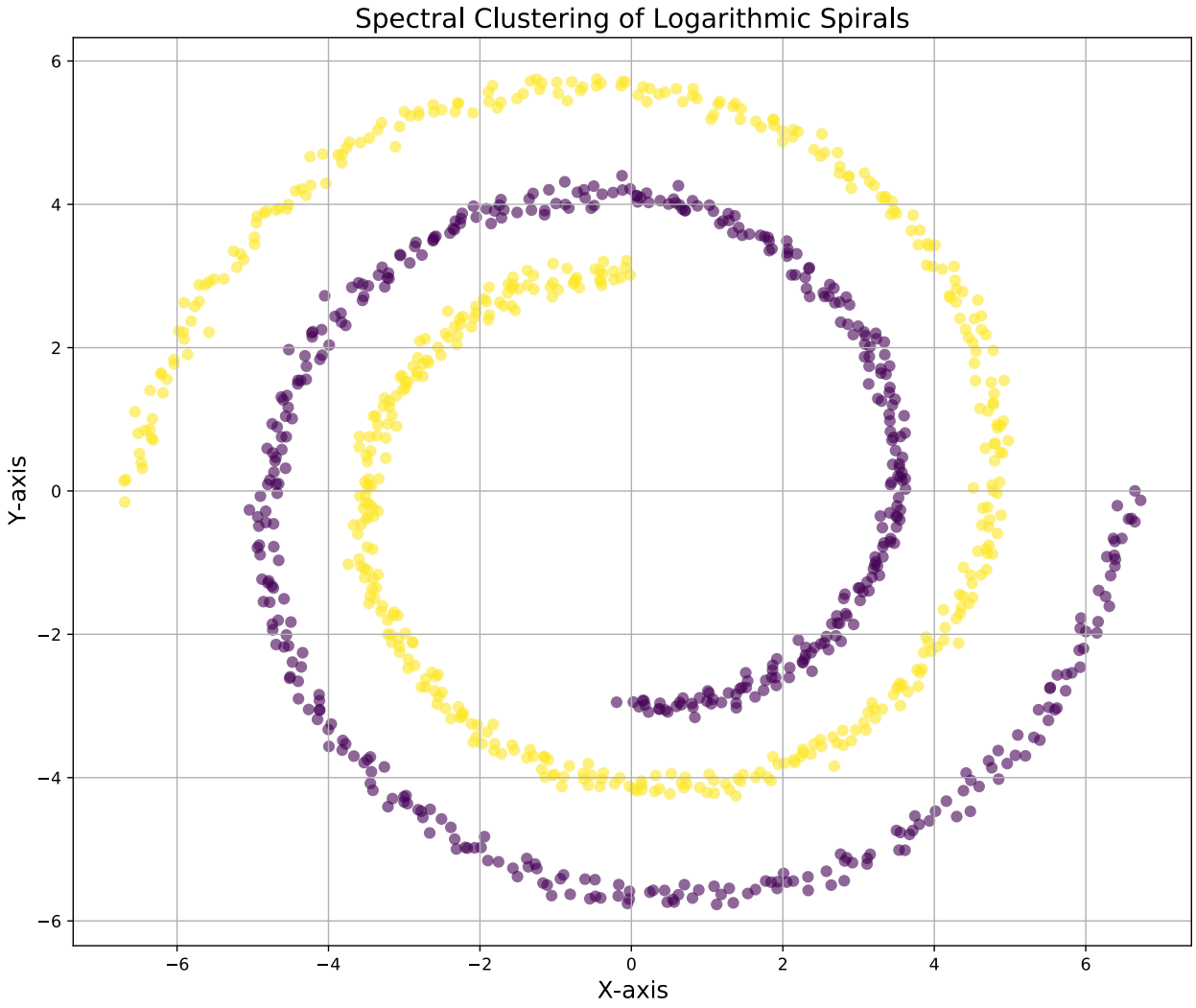
Spiral data


Logarithmic Spirals with Random Noise
Y-axis
X-axis
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

K-means on spiral data



Spectral clustering



Classification of and problems in graphs/networks

Network classes

Networks can roughly be classified in the following way:

- Technological networks
- Social networks
- Networks of information
- Biological networks
- ...

Technological networks

"The physical infrastructure networks that have grown up over the last century or so and form the backbone of modern technological societies."

(Newman)

Examples are:

- the Internet
- telephone networks
- power grids
- transportation networks
- delivery and distribution networks

Social networks

"Networks in which the vertices are people, or sometimes groups of people, and the edges represent some form of social interaction between them, such as friendship." (Newman, 2010)

Examples are:

- online social networks (Facebook, MySpace, . . .)
- human interaction networks (in general)
- archival data networks
- affiliation networks (comembership, Kevin Bacon, Paul Erdős)
- ego-centered networks (special case)

Networks of information

"Networks consisting of items of data linked together in some way." (Newman, 2010)

Examples are:

- World Wide Web
- citation networks
- recommender networks
- keyword index

Biological networks

"Networks are widely used in many branches of biology as a convenient representation of patterns of interaction between appropriate biological elements." (Newman)

Examples are:

- biochemical networks
  - metabolic networks
  - protein-protein interaction networks
  - genetic regulatory networks
- neural networks
- ecological networks (such as food webs)

Typical (research) problems/questions

- How to obtain a "map" of the Internet? What does its topology look like?
- How do failures in power grids affect the whole network (or other parts)? This is very complicated.
- Find subcommunities in a social network.
- Which new friendships/relationships will be formed in the (near) future?
- Who are the most influential people? (How do we measure influence?)
- How to obtain a model of the social network (easy for online social networks, usually difficult)?
- How does the World Wide Web look like (topology)? What about connectivity?
- Which authors work in the same field?
- Understand structure of recommender networks to build better recommender systems (same for keyword indexes).

Graph Analysis Tasks

Graph Level

- graph generation (e.g. drug discovery)
- graph evolution
- graph level prediction (classification or regression tasks from graphs)

Node Level

- node property prediction (e.g. position of atom in molecule folding)

Edge Level

- edge property prediction (e.g. side effects of drugs via adverse side effects)
- missing edge/link prediction (Netflix, Parship,...)

Sub-Graph Level

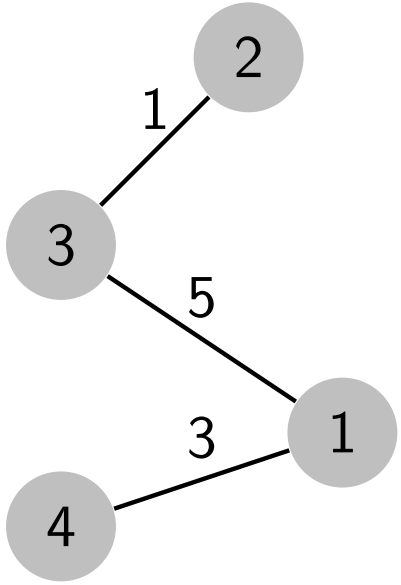
- community detection
- subgraph property prediction in itinerary systems

Modelling graphical/network data

What is a graph (formally)?

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

- $G = (V, E, \omega)$, where $V \neq \emptyset$ 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**
- usually we choose (or rename) $V = \{1, 2, \ldots, n\}$ and use the notations $ij = \{i, j\}$ for $\{i, j\} \in E$ and $\omega_{ij} = \omega(ij)$
- 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)
- $d_i := d(i) := |N(i)|$ is the **degree** of i



$w(23) = 1$,
$N(4) = \{1\}$,
$d(1) = |\{3, 4\}| = 2$

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

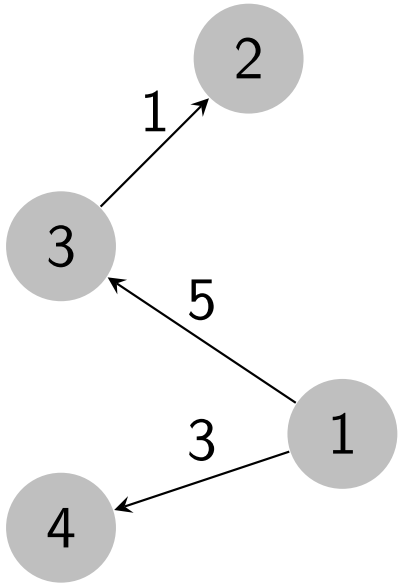
A set of vertices in a graph which are all adjacent to each other (they **induce** a complete (sub)graph), is called **clique**.

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

What is a digraph (formally)?

Edges can have a direction.

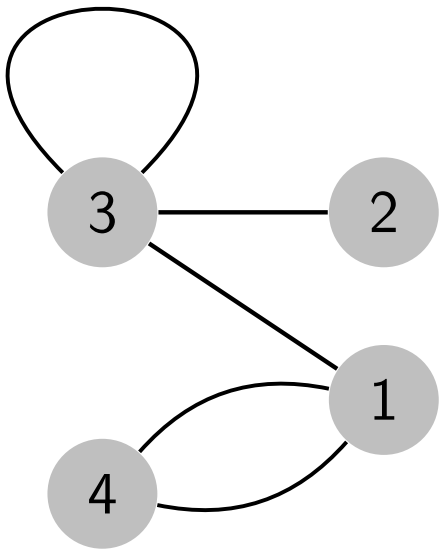
- $G = (V, E, \omega)$, where $V \neq \emptyset$ is a set, $E \subset V \times V$ (this is sometimes also called the **set of arcs**) and $\omega : E \to \mathbb{R}^+$, is called a **(weighted) digraph**
- for $(i, j) \in E$ the vertex i is called **predecessor** of j and j is called **successor** of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

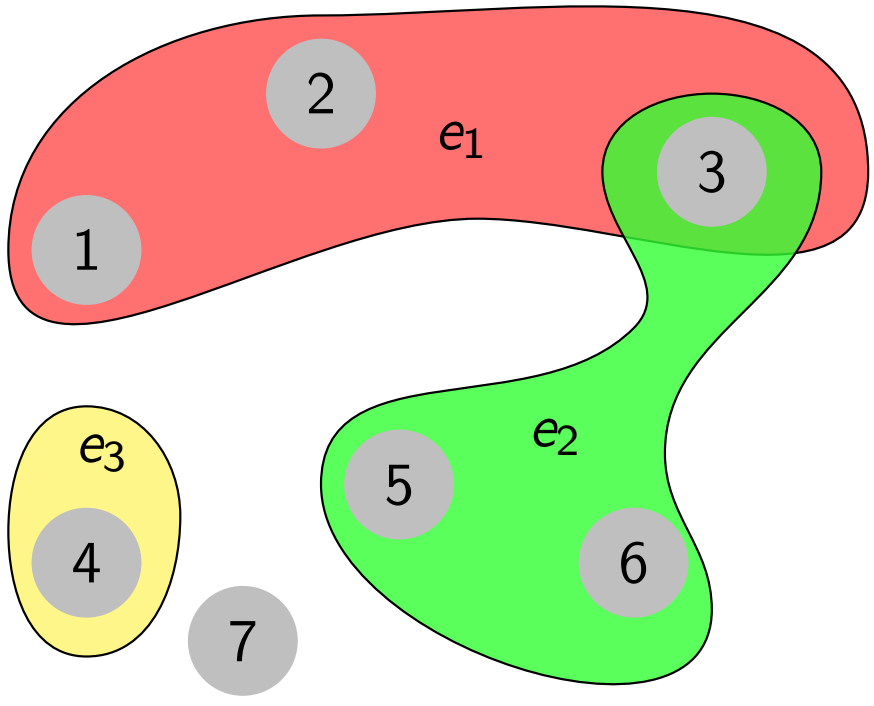
It is sometimes necessary to allow multiple edges between two vertices or a **loop** (a self-edge). In that case we use the term **multigraph**.



What is a hypergraph (formally)?

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

- natural generalisation is a **hypergraph** $H = (V, E)$, where
  - $V \neq \emptyset$ 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**
- similar notation simplifications as before



Storing graphs

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

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

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

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

All the above constructions are valid for directed graphs.

How to transform a digraph into a graph?

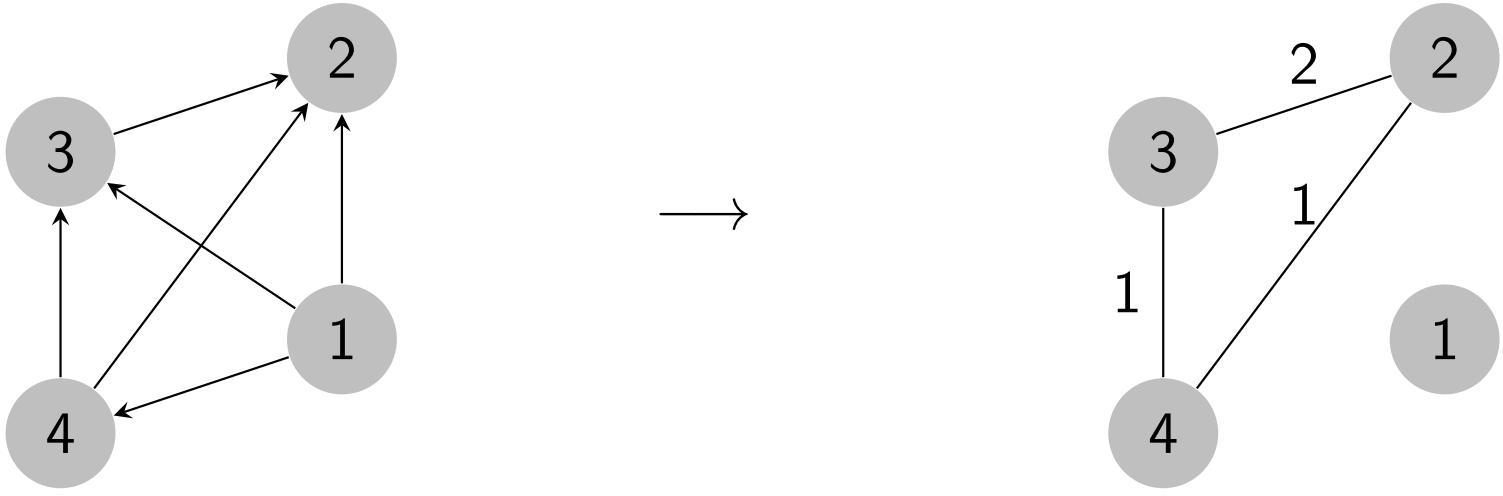
Consider the following three approaches.

- 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:

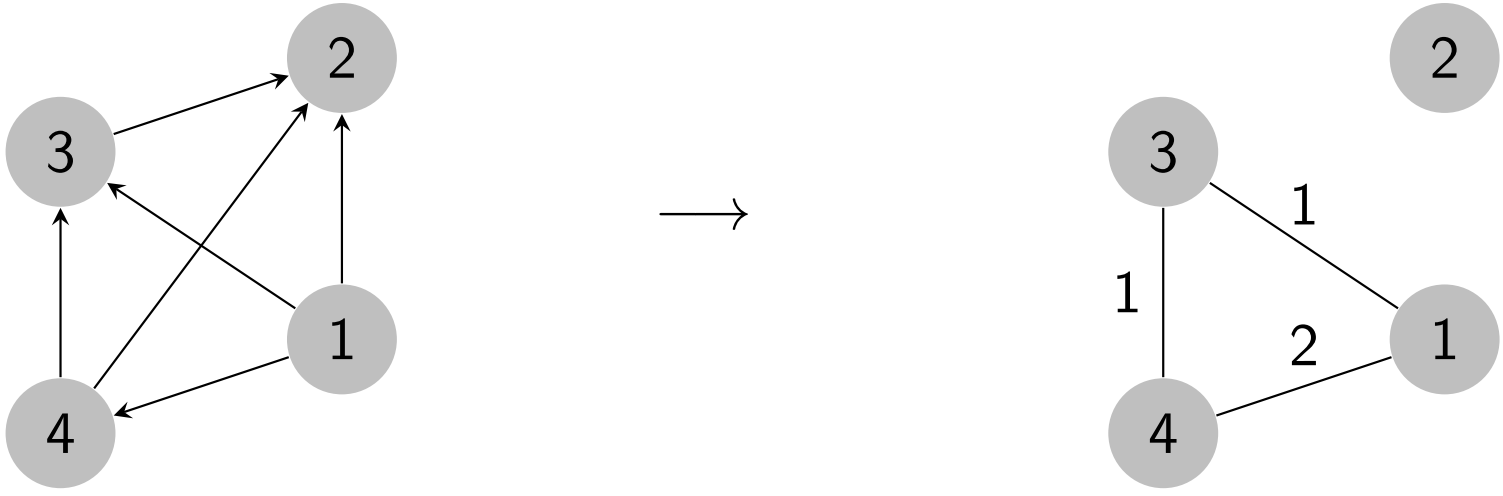
- **cocitation** c_{ij} of $i, j \in V$ is the number of common predecessors of i and j
- the **cocitation network** has vertex set V and an edge between i and j iff $c_{ij} > 0$
- it is also possible to obtain a weighted graph with weights c_{ij}
- note that $c_{ij} = \sum_{k=1}^{n} a_{ki} a_{kj}$,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



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 $B = AA^T$



How to transform a hypergraph into a graph?

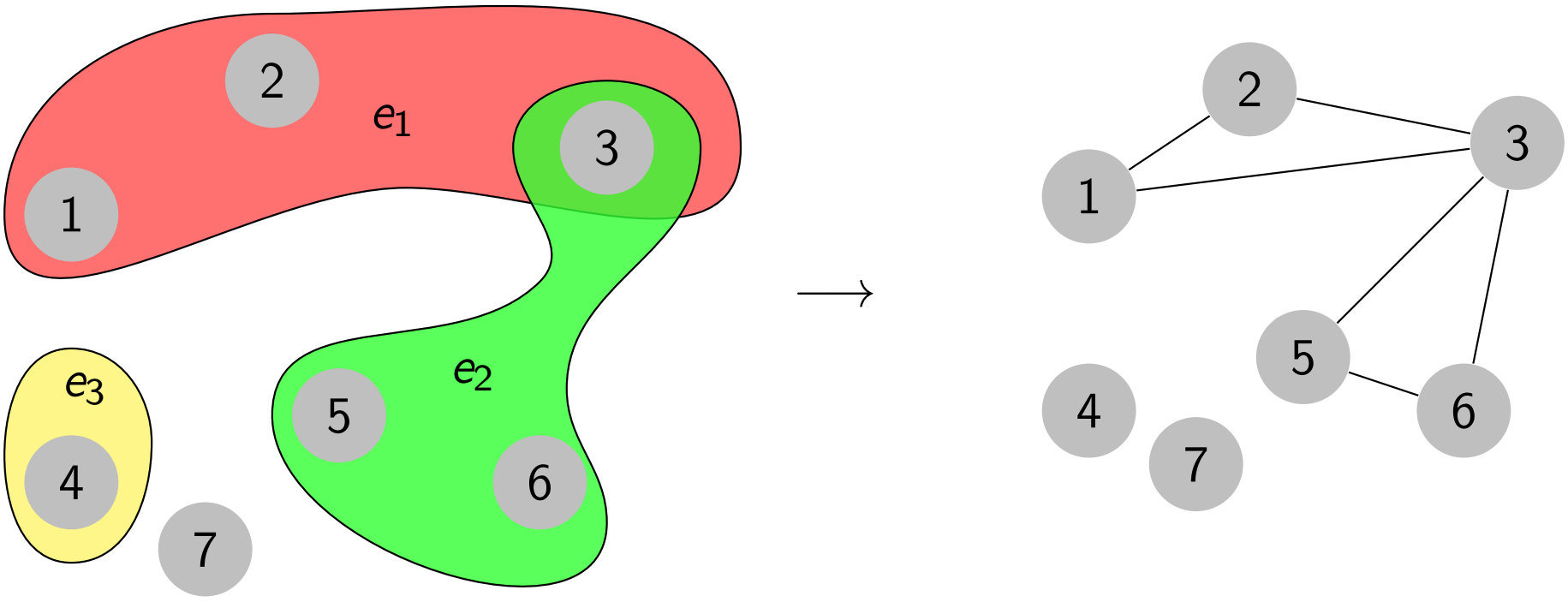
The following constructions are standard.

- 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 $V \cup E$
  - 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:

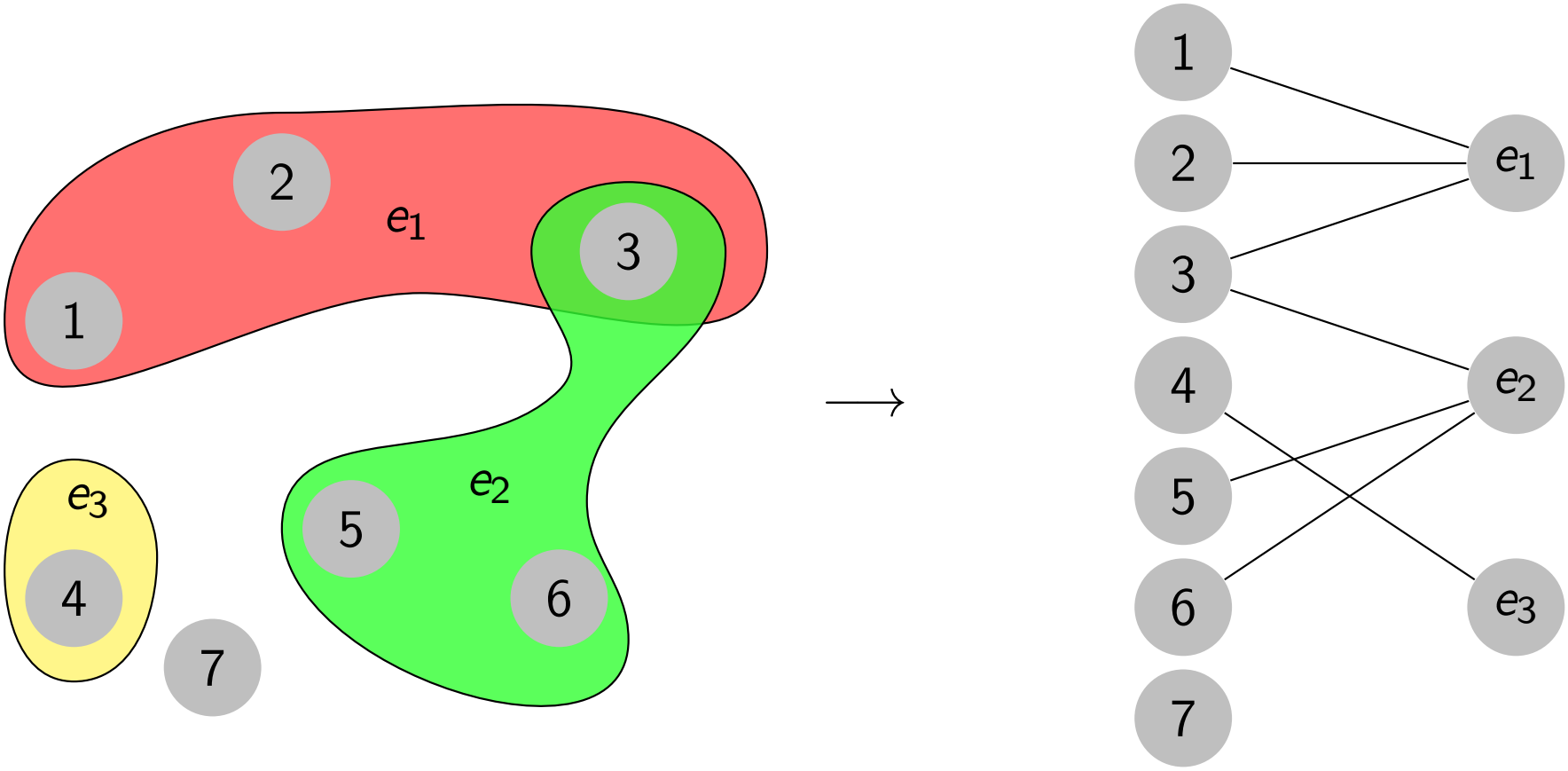
- $V^x = V$
- $E^x = \{\{i, j\} : \exists e \in E \text{ with } i, j \in e\}$



Star expansion

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

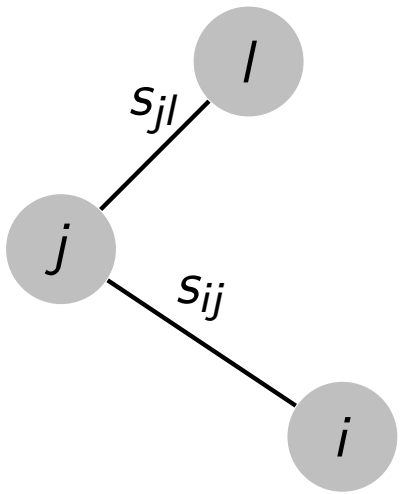
- $V^* = V \cup E$
- $E^* = \{\{i, e\} : i \in e, e \in E\}$



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[2] $s_{ij} \geq 0$ 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 $s_{il} = 0$

[2] We will discuss this topic in more detail later.

The ε-neighbourhood graph

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

- vertices are data points
- fix some $\varepsilon > 0$
- 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
- obtain an undirected graph by deleting all non-symmetric edges

In both cases weights are just the similarities.