

# Machine Learning for Complex Networks

**Chester Tan**

Machine Learning for Complex Networks  
Center for Artificial Intelligence and Data Science (CAIDAS)  
Julius-Maximilians-Universität Würzburg  
Würzburg, Germany

[chester.tan@uni-wuerzburg.de](mailto:chester.tan@uni-wuerzburg.de)

**Lecture 07**  
**Node Similarities and Unsupervised Link Prediction**

June 12, 2024

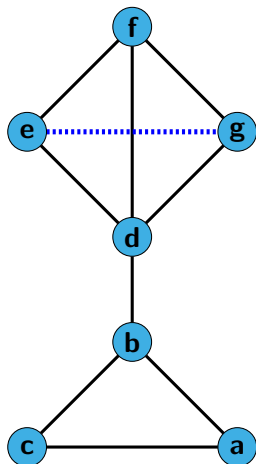


## Notes:

- **Lecture L07:** Node Similarities and Unsupervised Link Prediction 11.06.2024
- **Educational objective:** We introduce approaches to assess the similarity of nodes and show how we can use them to address link prediction in an unsupervised fashion.
  - Unsupervised Link Prediction
  - Neighborhood-based Node Similarities
  - Walk-based Node Similarities
  - Evaluating Link Prediction
- **Exercise sheet 06:** Node similarities and link prediction due 20.06.2024

# Motivation

- ▶ we have explored **statistical methods to detect communities**
- ▶ important **unsupervised learning problem** in complex networks
- ▶ so far, we assumed **complete knowledge** about the network
- ▶ what if we **partially observe links**
- ▶ can we predict which (unobserved) links are likely to exist?



## Notes:

- In the past five lectures, we have introduced different approaches to address community detection in complex networks. Generalizing the clustering problem to graphs, community detection is one of the most fundamental challenges in unsupervised learning in complex networks.
- Today, we will introduce another unsupervised learning task that has similar importance in practice. Its importance is due to an unrealistic assumption that we have always made so far, which is that we have complete knowledge about the network that we seek to study. However, we often only partially observe links.
- Consider a social network that you measure based on a survey with human respondents, who are asked to name their friends. What if some respondents missed to give some names? We can turn this into a learning problem, where – given a subset of observed links in an unknown ground truth network – we are asked to predict the missing links.
- In a different context, we can also use this for recommender systems. In an evolving social network, we can use the current friend relationships to predict which users are likely to form a link in the near future. Such recommendation schemes building on link prediction are commonly applied in social media platforms like, e.g. Twitter, LinkedIn or Instagram.

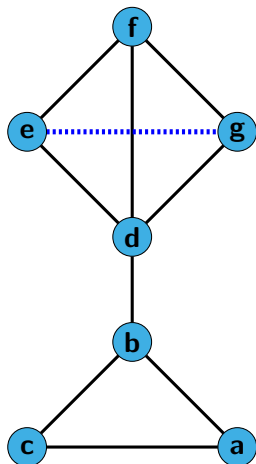
# Link Prediction in Networks

## link prediction problem

For graph  $G = (V, E)$  with subset  $E' \subset E$  of **observed links**, link prediction refers to the problem of identifying node pairs  $v, w$  such that  $(v, w) \in E - E'$ , i.e. we want to **predict unobserved links**

## exemplary applications

- ▶ product recommendations in online shops
  - ▶ friend recommendations in social networks
  - ▶ predicting drug-target associations
- 
- ▶ we can cast this as **binary classification problem**, i.e. assign class  $\{0, 1\}$  to node pairs  $(v, w) \in V \times V$
  - ▶ we can use **supervised and unsupervised** learning techniques



example network with nine observed links, **one unobserved link** and **11 not-connected node pairs**

## Notes:

- Let us first formalize the link prediction problem. We consider a (ground truth) graph  $G = (V, E)$  with a subset  $V' \subset V$  of observed links. We are interested in a binary classifier, which –for each pair of nodes  $v, w \in V \times V$  predicts whether  $(v, w) \in E$ . We are mostly interested in predictions for those pairs of nodes that are not already in the set of observed links. However, we could also consider networks with multi-edges, where edges between the same nodes  $v, w$  could occur both in the subset  $V'$  and in  $V - V'$ .
- In principle, we can consider unsupervised and supervised techniques for link prediction. With unsupervised link prediction we refer to methods that do not need a test set to train a classifier, i.e. methods that can directly predict a link based on the set of nodes and edges in the training set.

# Node similarities

- ▶ hypothesis: nodes more likely to be connected if they are “**more similar**”
- ▶ we can use **node attributes** to assess node similarities

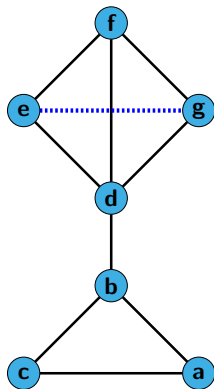
## examples

- ▶ term frequencies in citation networks
  - ▶ age/gender/beliefs/interests in social networks
  - ▶ positions of nodes in (latent) space
  - ▶ community memberships
- 
- ▶ in absence of attributes, we can consider **topological similarity** between nodes, i.e.

$$\text{sim} : V \times V \rightarrow \mathbb{R}$$

- ▶ yields **similarity matrix**  $\mathbf{S} \in \mathbb{R}^{n \times n}$  with

$$S_{ij} = \text{sim}(i, j) \text{ for } i, j \in V$$



$$\text{sim}(e, g) = ?$$

$$\text{sim}(c, e) = ?$$

## Notes:

- Most unsupervised link prediction techniques operate based on a simple hypothesis, which is that those nodes that are connected by a link are – in some way – more similar to each other than those nodes not connected by a node. Our task is to find a way to assess this similarity based on the training data set.
- This task is often easy if we have additional information on node attributes. Consider a citation network of articles, where we have access to the full text of the articles. Here we could use the text of articles to create term frequency vectors that characterize the nodes. We could then predict that two articles are likely to cite each other if their term frequency vectors are similar. In social networks, we could use demographic attributes like age, gender or we could use personal characteristics like beliefs or interests. For networks that are embedded into a space (e.g. infrastructure networks) we could predict links based on the distance in the underlying space. We could finally use community memberships, which is interesting as we could also learn those directly from the graph topology.
- In absence of such attributes, we can consider similarities between nodes that are based on the topology. In our example, we would intuitively say that nodes  $e$ ,  $g$  are more similar to each other than nodes  $c$ ,  $e$ . But why would we say this and how can we operationalize this notion of similarity?



# Inverse path length

- ▶ **topological distance** is intuitive measure of “dissimilarity”
- ▶ we can use **inverse path length** to define similarity as

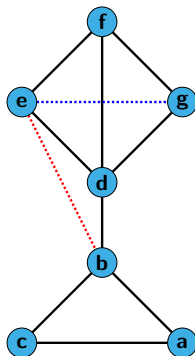
$$d^{-1}(v, w) := \frac{1}{\text{dist}(v, w)} \in (0, \infty]$$

- ▶  $d^{-1}(v, w) = 1$  iff  $(v, w) \in E$

## example

$$d^{-1}(e, g) = \frac{1}{2}$$
$$d^{-1}(e, b) = \frac{1}{2}$$

- ▶ does not consider whether nodes belong to **same community**



	a	b	c	d	e	f	g
a	$\infty$	1.00	1.00	0.50	0.33	0.33	0.33
b	1.00	$\infty$	1.00	1.00	0.50	0.50	0.50
c	1.00	1.00	$\infty$	0.50	0.33	0.33	0.33
d	0.50	1.00	0.50	$\infty$	1.00	1.00	1.00
e	0.33	0.50	0.33	1.00	$\infty$	1.00	0.50
f	0.33	0.50	0.33	1.00	1.00	$\infty$	1.00
g	0.33	0.50	0.33	1.00	0.50	1.00	$\infty$

## Notes:

- A first idea is to relate the notion of (topological) similarity with the concept of distance in a graph, where smaller distance between nodes indicates a larger similarity. Our first attempt to measure similarity is thus the inverse path length between nodes  $v, w$ .
- For the similarity of a node to itself, we have path length zero and we define the inverse path length as infinity, i.e. a node is infinitely similar to itself. We further note that, by definition, nodes that are connected by a link have similarity of one, since the shortest path distance between such nodes can not be larger than one. We thus obtain a score that assumes the special value  $\infty$  only on the diagonal, while all other values are between zero and one. Values close to zero indicate small similarity.
- Despite its appealing simplicity, this simple approach does not work well in our example. Since both node pairs  $b, e$  and  $e, g$  are connected by a shortest path of length two, both pairs have the same similarity score of 0.5.
- But why would we want nodes  $e, g$  to have a larger similarity than nodes  $b, e$ ?  $e$  and  $g$  are member of the same group of densely connected nodes, i.e. they are in the same community. This is not considered by the inverse path length, which only considers shortest paths.

# Common neighbours

- ▶ what distinguishes  $(e, g)$  from  $(e, b)$ ?
- ▶ for  $v, w \in V$  in undirected network  $G = (V, E)$  we can count **number of common neighbours**, i.e.

$$C(v, w) := |N_v \cap N_w|$$

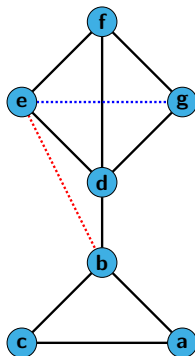
with

$$N_v := \{x \in V : (v, x) \in E\}$$

**example**

$$C(e, g) = 2$$

$$C(e, b) = 1$$



	a	b	c	d	e	f	g
a	2	1	1	1	0	0	0
b	1	3	1	0	1	1	1
c	1	1	2	1	0	0	0
d	1	0	1	4	1	2	1
e	0	1	0	1	2	1	2
f	0	1	0	2	1	3	1
g	0	1	0	1	2	1	2

## Notes:

- The simplest method to assess whether two nodes are member of the same group of densely connected nodes is to count the number of neighbours they share. This is the simplest type of neighbourhood-based similarity scores, which –despite their simplicity and scalability –often perform remarkably well in real networks.
- We only need to calculate the intersection of the neighbour sets of two nodes  $v$  and  $w$ . This yields an integer number with a minimal score of zero if there is no overlap between the neighbours.
- For our example, we find that nodes  $e$ ,  $b$  have a single neighbour ( $d$ ) in common, while  $e$  and  $g$  have two neighbours in common, because they are both member of the same community.

# Jaccard similarity

- number of common neighbors depends on **node degrees**
- makes it difficult to compare node pairs with different degrees
- normalization by total number of neighbors yields **Jaccard index**

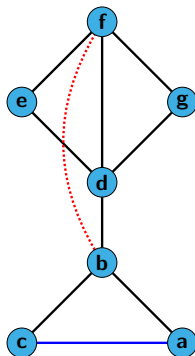
$$J(v, w) := \frac{|N_v \cap N_w|}{|N_v \cup N_w|} = \frac{|N_v \cap N_w|}{d_v + d_w - |N_v \cap N_w|}$$

- $J(v, w) = 1$  iff  $N_v = N_w$

## example

$$C(a, c) = C(b, f) = 1$$

$$J(a, c) = 0.33 > 0.2 = J(b, f)$$



	a	b	c	d	e	f	g
a	1.00	0.25	0.33	0.20	0.00	0.00	0.00
b	0.25	1.00	0.25	0.00	0.25	0.20	0.25
c	0.33	0.25	1.00	0.20	0.00	0.00	0.00
d	0.20	0.00	0.20	1.00	0.20	0.40	0.20
e	0.00	0.25	0.00	0.20	1.00	0.25	1.00
f	0.00	0.20	0.00	0.40	0.25	1.00	0.25
g	0.00	0.25	0.00	0.20	1.00	0.25	1.00

## Notes:

- Common neighbours similarity considers the absolute number of neighbours that two nodes have in common. Let us consider the two node pairs  $a, c$  and  $b, f$ , both have exactly one neighbour in common. Should those pairs have the same similarity?
- Intuitively, from the perspective of neighbour sets, nodes  $a$  and  $c$  should be more similar to each other than nodes  $b$  and  $f$ . The problem is that nodes  $b$  and  $f$  have degree three, while nodes  $a$  and  $c$  have degree two. Considering that  $a$  and  $c$  also have a link to each other, they cannot possibly have more than one common neighbour, i.e. the number of common neighbours is maximal. Due to their larger degree, for  $b$  and  $f$  this is not the case.
- We can address this with a normalization that considers the degrees of both nodes. We can use the so-called Jaccard similarity, which was originally proposed by the Swiss botanist Paul Jaccard as a *coefficient of community* between two sets of plant species → P Jaccard, 1912 . It normalizes the size of the intersection between two sets by the size of the union of the two sets.
- The size of the union of the neighbours of two nodes  $v$  and  $w$  is given by the sum of the node degrees  $d_v$  and  $d_w$  minus the number of common neighbours. We note that the Jaccard index is one iff the neighbour sets of two nodes are exactly identical. By definition, this holds for the Jaccard index of a node with itself. In our example, the Jaccard index of nodes  $e$  and  $g$  is one, as those two nodes have exactly the same neighbours. Note that the Jaccard index of nodes  $a$  and  $c$  is smaller than one, which is due to the fact that those nodes are connected to each other (which introduces one neighbour for each node that they do not have in common unless there is a self-loop).

# Szymkiewicz–Simpson coefficient

- consider  $v, w \in V$  with  $N_v \subset N_w$ , i.e. neighbors of one node subset of other node's neighbors
- overlap coefficient** normalizes common neighbors by smaller degree, i.e.

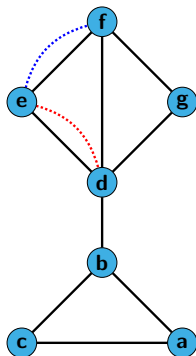
$$O(v, w) := \frac{|N(v) \cap N(w)|}{\min(d_v, d_w)}$$

- $O(v, w) = 1$  iff  $N_v \subseteq N_w$  or  $N_w \subseteq N_v$

## example

$$J(e, d) = 0.2 < 0.5 = O(e, d)$$

$$J(e, f) = 0.25 < 0.5 = O(e, f)$$



	a	b	c	d	e	f	g
a	1.00	0.50	0.50	0.50	0.00	0.00	0.00
b	0.50	1.00	0.50	0.00	0.50	0.33	0.50
c	0.50	0.50	1.00	0.50	0.00	0.00	0.00
d	0.50	0.00	0.50	1.00	0.50	0.67	0.50
e	0.00	0.50	0.00	0.50	1.00	0.50	1.00
f	0.00	0.33	0.00	0.67	0.50	1.00	0.50
g	0.00	0.50	0.00	0.50	1.00	0.50	1.00

## Notes:

- While the Jaccard index provides us with a normalized similarity score, we can still have situations with different levels of neighbourhood overlap that are not properly distinguished by the measure. Consider two nodes with different degrees, where the neighbours of the node with the smaller degree are contained in the set of neighbours of the other node. In this situation, the Jaccard index can yield a small score even though the overlap between neighbours is as large as it can possibly be.
- To see this issue, consider the node pairs  $e, f$  and  $e, d$  in the example above (which also happen to be connected by a link). While  $e$  has a degree of two, the two nodes  $f$  and  $d$  have degrees of three and four respectively. We further see that the neighbours of  $e$  (except for the one link to the respective neighbours, which is not considered as overlap since those nodes do not have self-loops) fully overlap with those of  $d$  and  $f$ . However, due to their larger degree, nodes  $d$  and  $f$  have additional neighbours not shared with  $e$ . As a consequence the Jaccard similarities of  $e, d$  and  $e, f$  are relative small with 0.2 and 0.25. Also, we note that those node pairs have different similarities, even though for both all neighbours of  $e$  (except the one that is the node itself) are contained in the neighbour sets of  $d$  and  $f$ .
- We can address this with the Szymkiewicz-Simpson coefficient, which is also called overlap coefficient. Rather than normalizing by the size of the union, here we normalize by the smallest degree of the two nodes. This again yields a normalized similarity score in the range  $[0, 1]$ , but we now assign larger values to node pairs with different degrees, where the neighbours of one node are contained in the neighbour set of the other node.

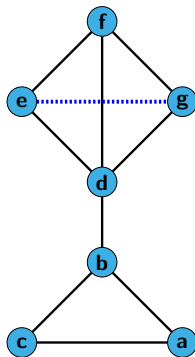


# Adamic-Adar index

- ▶ consider two nodes  $v, w$  and shared neighbour  $u$  with  $(v, u)$  and  $(v, w) \in E$
- ▶ we should consider nodes more similar if they have **common neighbours with small degree**
- ▶ **Adamic-Adar index** considers **number of shared links** between nodes  $v, w$  as

$$A(v, w) := \sum_{u \in N_v \cap N_w} \frac{1}{\log d_u}$$

- ▶ division by  $\log d_u$  reduces impact of common neighbors with large degree



	a	b	c	d	e	f	g
a	2.35	1.44	0.91	0.91	0.00	0.00	0.00
b	1.44	3.61	1.44	0.00	0.72	0.72	0.72
c	0.91	1.44	2.35	0.91	0.00	0.00	0.00
d	0.91	0.00	0.91	4.71	0.91	2.89	0.91
e	0.00	0.72	0.00	0.91	1.63	0.72	1.63
f	0.00	0.72	0.00	2.89	0.72	3.61	0.72
g	0.00	0.72	0.00	0.91	1.63	0.72	1.63

## Notes:

- One issue with the similarity measures before is, that few nodes with very large degrees can influence the similarity score of a large number of other nodes to which they are connected.
- To address this, we can define an unnormalized measure that is given above. This so-called Adamic-Adar index was introduced in → L Adamic, E Adar, 2003 . We note that, different from the measures before, it also accounts for the degrees of nodes  $u$  that two nodes  $v$  and  $w$  have in common. By this, we reduce the similarity of nodes that mainly share common neighbours with high degrees.
- The rational behind this is that the similarity of two nodes  $v, w$  that share one neighbour  $u$  that is exclusively connected to  $v$  and  $w$  (i.e. has degree two) should be larger than the similarity of two nodes that share a neighbour  $u$  that is also connected to many other nodes.

# Cosine similarity

- ▶ let  $\mathbf{A}_v \in \mathbb{R}^n$  denote column vector of adjacency matrix  $\mathbf{A}$  that refers to node  $v$

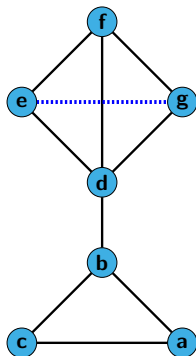
- ▶ **cosine similarity** of  $v$  and  $w$  is defined as

$$\text{cosim}(v, w) := \cos(\alpha) = \frac{\mathbf{A}_v \cdot \mathbf{A}_w}{\|\mathbf{A}_v\| \|\mathbf{A}_w\|}$$

- ▶ corresponds to **common neighbors** normalized by **geometric mean of degrees**, i.e.

$$\text{cosim}(v, w) = \frac{C(v, w)}{\sqrt{d_v \cdot d_w}}$$

→ exercise sheet



	a	b	c	d	e	f	g
a	1.00	0.41	0.50	0.35	0.00	0.00	0.00
b	0.41	1.00	0.41	0.00	0.41	0.33	0.41
c	0.50	0.41	1.00	0.35	0.00	0.00	0.00
d	0.35	0.00	0.35	1.00	0.35	0.58	0.35
e	0.00	0.41	0.00	0.35	1.00	0.41	1.00
f	0.00	0.33	0.00	0.58	0.41	1.00	0.41
g	0.00	0.41	0.00	0.35	1.00	0.41	1.00

## Notes:

- As final example for measures that are based on neighbours, consider the adjacency matrix  $\mathbf{A}$  of a network. For a given node  $v$ , we further consider the column vector  $\mathbf{A}_v$ , i.e. a binary vector of length  $n$  that gives a one-hot encoding of the neighbours of  $v$ . In directed networks, we can use the row or column vector depending on whether we consider nodes pointing to node  $v$  or the nodes to which node  $v$  points.
- A simple yet powerful approach to learning in graphs is to interpret the vector  $\mathbf{A}_v$  as a feature of  $v$  in an  $n$  dimensional feature space, which captures the neighbourhood of node  $v$ . If two vectors  $\mathbf{A}_v$  and  $\mathbf{A}_w$  are identical,  $v$  and  $w$  have exactly the same neighbours. If they are orthogonal, the two nodes have no neighbours in common. This motivates the idea that we can use the angle between two vectors  $\mathbf{A}_v$  and  $\mathbf{A}_w$  to measure the similarity of two nodes in terms of their neighbourhoods. We can use cosine similarity to capture the cosine of the angle in the range from  $[-1, 1]$ , where 0 means that two vectors are orthogonal, while 1 means that they are co-linear, i.e. they point in the same direction. A value of minus one would indicate that two vectors point in opposite directions, however in an adjacency matrix with positive entries we cannot have negative values for cosine similarity. The situation is different in signed networks, where we can have “anti-links”, i.e. negative relationships that are often captured with  $-1$  entries in the adjacency matrix.
- For our example, we find that  $e$  and  $g$  have the maximum cosine similarity. Different from the measures introduced before, cosine similarity does not account for degrees of the involved nodes or for subset relationships. This approach to measure the similarity of items in relational data was described in → G Salton, MJ McGill, 1983

# Practice session

- ▶ we implement **topology-based similarity scores**
- ▶ we **rank node pairs** based on their similarity

## 07-01 - Topology-based Similarity Scores

June 8 2022

An important approach to predict links in networks is to calculate scores that capture pair-wise similarities between nodes. In the first practice session, we implement different approaches to calculate such node similarities and test them in a simple toy example.

```
import networkx as nx
import numpy as np

from matplotlib import pyplot as plt
from collections import defaultdict

import pandas as pd
```

Python

We first create our well-known toy example.

```
plot_style = {
    'edge_color': 'white'
}
```

Python

```
n = nx.Network(directed=False)
n.add_edge('a', 'b')
n.add_edge('b', 'c')
n.add_edge('c', 'a')
n.add_edge('a', 'd')
n.add_edge('d', 'f')
n.add_edge('f', 'g')
n.add_edge('g', 'd')
n.add_edge('d', 'e')
n.add_edge('b', 'e')
n.plot(**plot_style)
```

Python

### practice session

see notebooks 07-01 in gitlab repository at

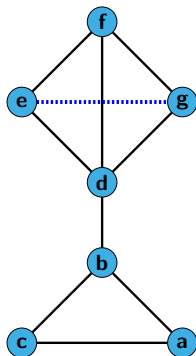
→ [https://gitlab.informatik.uni-wuerzburg.de/ml4nets\\_notebooks/2024\\_sose\\_ml4nets\\_notebooks](https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks)

## Notes:

- In the first practice session, we implement topology-based similarity scores and test them in small example networks. We further use them to rank node pairs based on their similarities.

# Local Similarity Scores?

- ▶ similarity scores based on neighborhoods are **local measures**
- ▶ neighbour-based scores neglect **global topology of the network**
- ▶ **inverse path length** → slide 4  
considers global topology, but neglects community structure
- ▶ can we define **node similarities that consider non-local characteristics?**



$$\mathbf{A} = \begin{matrix} & \begin{matrix} a & b & c & d & e & f & g \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 & 1 & 0 & 2 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

## Notes:

- In the practice session, we noted that the neighbourhood-based similarity scores introduced so far neglect features that are due to the global topology. As an example, we have considered a network with a community pattern that is not expressed in an overlap of the immediate neighbors of nodes. For this example, our measures failed to produce reasonable similarity scores.
- We note that on slide 4, we started our exploration with the inverse path length, a simple similarity score that does consider the global topology but that neglects community patterns (since nodes in different communities can still be connected by a short path).
- This raises the question whether we can define similarity measures that consider both the local and global topology of a network. So far, we accounted for community structures by considering the overlap of immediate neighbours of a node. However, there are other, not necessarily local ways to define community structures that enable us to study community structures at multiple “scales”.



# Walk-based similarity measures

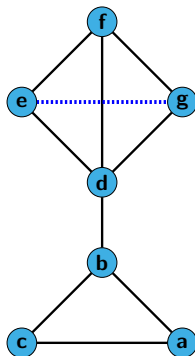
- nodes in **same community** are connected by many paths and walks
- $A^k$  counts **number of walks of length  $k$**  between all pairs of nodes
- we define **walk-based similarity** as

$$W^l(v, w) = \sum_{k=1}^l A_{v,w}^k$$

where  $l > 1$  is maximum walk length that we consider

## example

matrix of walk-based similarities  $W^2(v, w)$  between all pairs of nodes

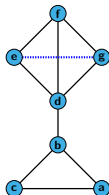


	a	b	c	d	e	f	g
a	2.00	1.00	1.00	1.00	0.00	0.00	0.00
b	1.00	3.00	1.00	0.00	1.00	1.00	1.00
c	1.00	1.00	2.00	1.00	0.00	0.00	0.00
d	1.00	0.00	1.00	4.00	1.00	2.00	1.00
e	0.00	1.00	0.00	1.00	2.00	1.00	2.00
f	0.00	1.00	0.00	2.00	1.00	3.00	1.00
g	0.00	1.00	0.00	1.00	2.00	1.00	2.00

## Notes:

- We remember that in lecture 02, we argued that nodes in the same community not only (or necessarily) share many common neighbours, they are also connected by a large number of paths and walks. We have further seen that we can use the power of adjacency matrices to count the number of walks of exactly length  $k$  between all pairs of nodes. This observation can be used for a very simple (and unfortunately not very useful) similarity measure: We can simply count all walks up to a given length  $l$ , i.e. we define a measure of similarity that sums the entries of all matrix powers  $\mathbf{A}^i$  from  $i = 1$  to a maximum length  $l$ . In lecture 02, we have seen that pairs of nodes in the same community will typically have larger values, indicating that there are many different walks that connect them.
- The example above shows the entries of the walk-based similarity for a maximum walk length  $l = 2$ . As explained in Lecture 02, for an undirected network without self-loops, we will have the node degrees on the diagonal, because each link to a neighbour can be used to independently leave and return to a given node, i.e. a walk of length two. The result above looks promising, as the largest value is obtained for the node pair  $e, g$ , which is also the most natural candidate for a predicted link.

# Walk-based similarity?



	a	b	c	d	e	f	g
a	2.00	1.00	1.00	1.00	0.00	0.00	0.00
b	1.00	3.00	1.00	0.00	1.00	1.00	1.00
c	1.00	1.00	2.00	1.00	0.00	0.00	0.00
d	1.00	0.00	1.00	4.00	1.00	2.00	1.00
e	0.00	1.00	0.00	1.00	2.00	1.00	2.00
f	0.00	1.00	0.00	2.00	1.00	3.00	1.00
g	0.00	1.00	0.00	1.00	2.00	1.00	2.00

$$W^2(v, w)$$

	a	b	c	d	e	f	g
a	2.00	4.00	3.00	1.00	1.00	1.00	1.00
b	4.00	2.00	4.00	6.00	1.00	2.00	1.00
c	3.00	4.00	2.00	1.00	1.00	1.00	1.00
d	1.00	6.00	1.00	4.00	6.00	6.00	6.00
e	1.00	1.00	1.00	6.00	2.00	5.00	2.00
f	1.00	2.00	1.00	6.00	5.00	4.00	5.00
g	1.00	1.00	1.00	6.00	2.00	5.00	2.00

$$W^3(v, w)$$

## issues

- ▶ choice of maximum walk-length  $l$ ?
- ▶ equal influence for walks of all lengths?

## Notes:

- Unfortunately, this simple approach to measure similarity has multiple issues that we need to address. A first obvious issue is that it introduces a parameter  $l$  that we need to choose, i.e. the question is which value we should use to obtain a reasonable similarity score for a given network. For the example network above, we obtain different similarity scores for  $l = 2$  and  $l = 3$ . In our example for  $l = 2$  we have the highest score for the node pair  $e, g$  while for  $l = 3$  the same score is assigned for the node pair  $b, f$ .
- Another issue is due to the definition that simply sums the counts of all walks of any length  $k$  up to  $l$ . This means that all lengths  $k$  have the same influence on the resulting similarities, however –due to the combinatorial explosion– there are typically many more walks for larger  $k$  compared to smaller  $k$ . This undermines the usefulness of our measure for larger values of  $l$ , where node similarities are dominated by the number of longest walks.

# Katz index

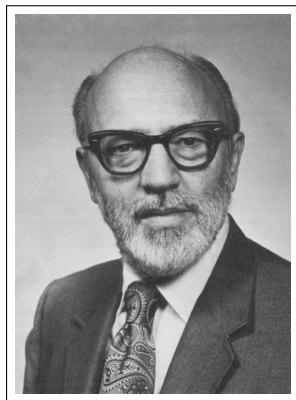
- ▶ we can **drop parameter**  $l$  if we count walks of **any length**, i.e.  $l \rightarrow \infty$
- ▶ for  $\beta \in (0, 1)$  we can exponentially **decrease influence of longer walks** by  $\beta^l$
- ▶ **Katz index** of a pair of nodes  $v, w$  is defined as **infinite series**  $\rightarrow$  L Katz, 1953

$$K(v, w) = \sum_{k=1}^{\infty} \beta^k A_{v,w}^k = \beta A_{v,w} + \beta^2 A_{v,w}^2 + \dots$$

- ▶ in matrix form, **series converges** to

$$K := (\mathbf{I} - \beta \mathbf{A})^{-1} - \mathbf{I}$$

where entries  $K_{v,w}$  give **Katz similarity** of nodes  $v$  and  $w$



Leo Katz

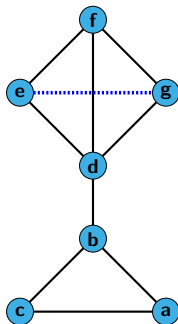
1914 - 1976

image credit: Applied Probability Trust, 1977

## Notes:

- The Katz index addressed both issues at once. A simple idea to get rid of the need to choose the maximum walk length  $l$  is to consider walks of any length, i.e. we let  $l$  go to infinity and take the limit.
- For this limit to exist (and to reduce the influence of long walks) we introduce a scaling parameter  $\beta$  that is smaller than one. For each  $k$  we then multiply the entries of the  $k$ -th power with  $\beta^k$ , i.e. we exponentially reduce the influence of long walks, which counters the exponential increase in the possible number of walks as  $k$  increases.
- For each pair of nodes, this definition yields an infinite series. In matrix form, for parameters  $\beta \in (0, 1)$  this series converges to the inverse of the identity matrix minus the adjacency matrix scaled by the parameter  $\beta$ , again subtracting the identity matrix, i.e. the Katz similarity index can be easily calculated based on matrix inversion.
- Note that the Katz index was originally proposed as a status measure (i.e. centrality measure) for actors in social networks in → L Katz, 1953 . It has inspired the PageRank centrality, which can be seen as a random walk variant of the Katz index that also works for weakly connected directed networks → Statistical Network Analysis .

# Katz index: example



	a	b	c	d	e	f	g
a	0.00	0.02	0.02	0.00	0.00	0.00	0.00
b	0.02	0.00	0.02	0.02	0.00	0.00	0.00
c	0.02	0.02	0.00	0.00	0.00	0.00	0.00
d	0.00	0.02	0.00	0.00	0.02	0.02	0.02
e	0.00	0.00	0.00	0.02	0.00	0.02	0.00
f	0.00	0.00	0.00	0.02	0.02	0.00	0.02
g	0.00	0.00	0.00	0.02	0.00	0.02	0.00

$$\beta = 0.02$$

	a	b	c	d	e	f	g
a	0.12	0.29	0.28	0.07	0.02	0.02	0.02
b	0.29	0.17	0.29	0.28	0.07	0.09	0.07
c	0.28	0.29	0.12	0.07	0.02	0.02	0.02
d	0.07	0.28	0.07	0.27	0.33	0.39	0.33
e	0.02	0.07	0.02	0.33	0.13	0.32	0.13
f	0.02	0.09	0.02	0.39	0.32	0.20	0.32
g	0.02	0.07	0.02	0.33	0.13	0.32	0.13

$$\beta = 0.2$$

## Notes:

- Considering that the limit exists for any parameter  $\beta \in (0, 1)$  it is interesting to explore how the choice of this parameter influences the resulting similarity measure. Intuitively, for smaller values of  $\beta$  that are closer to zero, the scaling factor  $\beta^k$  will vanish more quickly for larger values of  $k$ , which has a stronger impact on longer walks compared to shorter walks. We thus obtain similarities that are more strongly influenced by short rather than long walks.
- The opposite is true for larger values of  $\beta$  with values closer to one. Here, we obtain similarities that are more strongly influenced by longer walks compared to shorter walks.
- In a nutshell, we can use the parameter  $\beta$  to adjust the scale at which we consider global topological characteristics of the network, where larger values of  $\beta$  allows us to incorporate topological features farther away from the two nodes.
- Considering our discussion of neighbour-based similarity scores, we can immediately see another issue that the Katz index does not account for. It does not consider the degrees of nodes, which influence how many different walks we expect to exist between a pair of nodes. Hence, the resulting similarities are likely to be correlated with the node degrees, just as we have seen that nodes with higher degree have a larger influence on neighbourhood-based similarity scores.



# Leicht-Holme-Newman index

- ▶ consider similarities

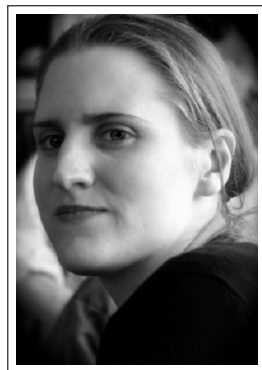
$$L(v, w) = C_{v,w;1}^{-1} A_{v,w} + C_{v,w;2}^{-1} A_{v,w}^2 + \dots$$

where  $C_{v,w;k}$  is **expected number of walks of length  $k$**  from  $v$  to  $w$

- ▶ in matrix form this converges to

$$\mathbf{L} = \mathbf{D}^{-1} \left( \mathbf{I} - \frac{\alpha \mathbf{A}}{\lambda_1} \right)^{-1} \mathbf{D}^{-1}$$

with  $\alpha \in (0, 1)$ ,  $\lambda_1$  largest eigenvec. of  $\mathbf{A}$   
and  $\mathbf{D}$  degree matrix → E Leicht, P Holme, M Newman, 2005



Elizabeth A Leicht

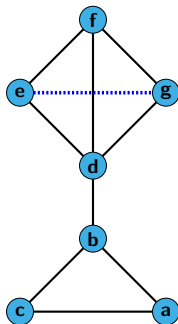
image credit: [www.cabdyn.ox.ac.uk](http://www.cabdyn.ox.ac.uk)

- ▶ entries  $L_{v,w}$  are **Leicht-Holme-Newman similarity score** of node pair  $(v, w)$
- ▶ small  $\alpha$  assign more weight to short walks

## Notes:

- This can be fixed by defining the same series, where we also consider the **expected number of walks of length  $k$  between nodes**. We first consider a definition where we divide each  $k$ -th power of the adjacency matrix with the expected number of walks of length  $k$ , i.e. we not only correct for the fact that there are more walks for larger length  $k$  but also for the expected number of such walks given the length and the node degrees.
- At first glance, this appears to be a rather complicated definition, as we have a different coefficient for each walk length  $k$ . But interestingly, it has been shown that this series converges to a rather simple expression that includes the largest eigenvalue of the adjacency matrix, the degree diagonal matrix **D** and a single parameter  $\alpha$  which needs to be chosen in  $(0, 1)$ . The resulting similarity measure was proposed by Elisabeth Leicht and collaborators in → EA Leicht, P Holme, M Newman, 2005 .

# LNH index: example



	a	b	c	d	e	f	g
a	8.64	0.06	0.09	0.00	0.00	0.00	0.00
b	0.06	3.84	0.06	0.03	0.00	0.00	0.00
c	0.09	0.06	8.64	0.00	0.00	0.00	0.00
d	0.00	0.03	0.00	2.16	0.05	0.03	0.05
e	0.00	0.00	0.00	0.05	8.64	0.06	0.00
f	0.00	0.00	0.00	0.03	0.06	3.84	0.06
g	0.00	0.00	0.00	0.05	0.00	0.06	8.64

$\alpha = 0.02$

	a	b	c	d	e	f	g
a	8.85	0.69	1.03	0.06	0.01	0.01	0.01
b	0.69	3.98	0.69	0.32	0.08	0.06	0.08
c	1.03	0.69	8.85	0.06	0.01	0.01	0.01
d	0.06	0.32	0.06	2.27	0.53	0.39	0.53
e	0.01	0.08	0.01	0.53	8.86	0.71	0.22
f	0.01	0.06	0.01	0.39	0.71	3.99	0.71
g	0.01	0.08	0.01	0.53	0.22	0.71	8.86

$\alpha = 0.2$

## Notes:

- The influence of the parameter  $\alpha$  mimics that of the parameter  $\beta$  in the Katz index, i.e. we can use to either the influence of either shorter or longer walks. For small values of  $\alpha$  similarities are more strongly influenced by shorter walks, i.e. we obtain a similarity score which emphasizes local topological features. For larger values of  $\alpha$  the influence of longer walks is stronger.
- This can be seen in the example above, where the similarity of  $e, g$  is close to zero for  $\alpha = 0.02$ , while those nodes are considered to be the most similar (unconnected) nodes for  $\alpha = 0.2$ .

# Practice session

- ▶ we implement and compare **walk-based similarity scores** for nodes in a network

## 07-01 - Topology-based Similarity Scores

June 8 2022

An important approach to predict links in networks is to calculate scores that capture pair-wise similarities between nodes. In the first practice session, we implement different approaches to calculate such node similarities and test them in a simple toy example.

```
import networkx as nx
import numpy as np

from networkx import nx_olap as nx_olap
from collections import defaultdict

import pandas as pd
```

Python

We first create our well-known toy example.

```
plot_style = {
    'edge_color': 'white'
}
```

Python

```
n = nx.Network(directed=False)
n.add_edge('a', 'b')
n.add_edge('b', 'c')
n.add_edge('c', 'a')
n.add_edge('a', 'd')
n.add_edge('d', 'e')
n.add_edge('e', 'g')
n.add_edge('g', 'd')
n.add_edge('d', 'f')
n.add_edge('b', 'e')
n.plot(plot_style)
```

Python

### practice session

see notebooks 07-02 in gitlab repository at

→ [https://gitlab.informatik.uni-wuerzburg.de/ml4nets\\_notebooks/2024\\_sose\\_ml4nets\\_notebooks](https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks)

## Notes:

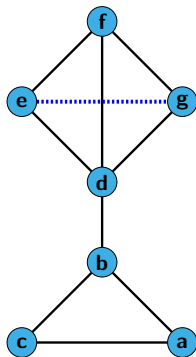
- In the second practice session, we show how we can implement walk-based similarity measures. We further test them in small example networks and use them to rank node pairs based on their similarity.

# From Node Similarities to Link Prediction

- ▶ consider graph  $G = (V, E)$  where  $E_{\text{train}} \subset E$  is set of **observed edges**
- ▶ use  $G = (V, E_{\text{train}})$  to calculate **similarity score**  $\text{sim} : V \times V \rightarrow \mathbb{R}$
- ▶ for  $k \in \mathbb{N}$  consider set  $\text{Top}^k \subset V \times V$  of  $k$  tuples  $(v, w)$  with largest  $\text{sim}(v, w)$
- ▶ we obtain **unsupervised binary classifier for link prediction**

$$\text{predict}(v, w) = \begin{cases} 1 & \text{if } (v, w) \in \text{Top}^k \\ 0 & \text{else} \end{cases}$$

- ▶ how can we **evaluate performance of link prediction**?



	a	b	c	d	e	f	g
a	8.85	0.69	1.03	0.06	0.01	0.01	0.01
b	0.69	3.98	0.69	0.32	0.08	0.06	0.08
c	1.03	0.69	8.85	0.06	0.01	0.01	0.01
d	0.06	0.32	0.06	2.27	0.53	0.39	0.53
e	0.01	0.08	0.01	0.53	8.86	0.71	0.22
f	0.01	0.06	0.01	0.39	0.71	3.99	0.71
g	0.01	0.08	0.01	0.53	0.22	0.71	8.86

## Notes:

- So far, we only considered measures that allow us to assess the similarity of two nodes. We now consider how we can use this to address the link prediction problem, which is a key problem in graph learning with important practical applications in information systems, information retrieval, social networks, and bioinformatics.
- We first assume that we have a graph  $G = (V, E)$  where we only observe a subset of the edges are observed. We denote the subset of observed edges as  $E_{train}$  as we will use it to calculate the similarity scores of all node pairs.
- For a given  $k \in \mathbb{N}$  we can now consider the  $k$  tuples  $(v, w)$  that have the largest similarity scores. For each  $k$  this yields a binary classifier, where we predict a link  $(v, w)$  iff  $(v, w) \in \text{Top}^k$ . The parameter  $k$  serves as a discrimination threshold, i.e. for larger  $k$  we predict more links. For larger  $k$ , we expect this classifier to be more **sensitive** and less **specific**, which are too technical terms in the evaluation of binary classifiers that we formally define in the following.



# Confusion matrix

- ▶ we can use similarity scores in  $G = (V, E_{\text{train}})$  to **predict links** for tuples  $(v, w) \in (V \times V) - E_{\text{train}}$
- ▶ we can evaluate prediction against **ground truth** in test  $E_{\text{test}} := E - E_{\text{train}}$
- ▶ **performance of binary classifier** captured in **confusion matrix**  $\mathbf{C} \in \mathbb{N}^{2 \times 2}$

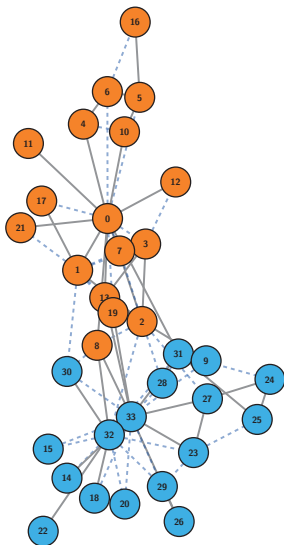
		ground truth	
		negative (no link)	positive (link)
prediction	negative (no link)	true negative $TN$	false negative type II error $FN$
	positive (link)	false positive type I error $FP$	true positive $TP$
		$N := TN + FP$	$P := TP + FN$

- ▶ entries of confusion matrix are basis to define **evaluation metrics**

## Notes:

- To evaluate the accuracy of a binary classifier we first assume that we have a prediction of binary classes (given by our classifier from before) as well as a ground truth, i.e. a test set with known class labels against which we can evaluate this prediction. In our link prediction setting, the test set of actual links that we seek to predict with high accuracy is given by  $E_{\text{test}} = E - E_{\text{train}}$ . We further only predict links for those node pairs not already connected by a link in the training set, i.e. the tuples  $V \times V - E_{\text{train}}$ .
- The evaluation of binary classifiers is inherently related to the evaluation of a hypothesis test, where we have a binary ground truth (whether the null hypothesis should be rejected or not) as well as a binary prediction (whether the test rejects the null hypothesis or not). We can put the four possible outcomes in a  $2 \times 2$  confusion matrix, where diagonal entries capture cases where our class predictions are correct. The off-diagonal elements capture type I and type II errors, i.e. a false positive or a false negative (where we map dichotomous classes to positive/1 and negative/0).
- We can take the column sums of the matrix above to get the total number of positive and negative instances in our data. The relation between those two numbers gives an indication whether we have a **balanced** or **imbalanced** classification problem, i.e. whether one class is much more frequent than another class (more on this in a moment).

# Link Prediction in Karate club network



Karate club network with training (gray) and test (blue dashed) edges

## empirical example

- ▶ social network of Karate club → W Zachary, 1977
- ▶  $n = 34$  nodes,  $m = 77$  links
- ▶  $|E_{\text{train}}| = 39$ ,  $|E_{\text{test}}| = 38$

**confusion matrix** of binary classifier predicting links for  $k = 20$  node pairs with largest **Common Neighbour** similarity

		true class	
		negative	positive
predicted class	negative	$TN = 512$	$FN = 28$
	positive	$FP = 10$	$TP = 10$
		$N = 522$	$P = 38$

## Notes:

- We are now ready to address link prediction in a first empirical network, the Karate club network that we already used in Lectures 04 and 05.

# Accuracy and specificity

- **accuracy** is probability that a prediction (pos. or neg.) is correct

$$ACC = \frac{TP + TN}{P + N}$$

		true class	
		negative	positive
predicted class	negative	$TN = 512$	$FN = 28$
	positive	$FP = 10$	$TP = 10$
		$N = 522$	$P = 38$

$$ACC = \frac{10 + 512}{560} = 0.932$$

- **specificity** is probability that negative class is correctly predicted

$$SPC = TNR = \frac{TN}{N}$$

$$SPC = \frac{512}{522} \approx 0.981$$

## Notes:

- Using the entries in the confusion matrix, we can define metrics that capture different aspects of a classifier's performance.
- The accuracy is the probability that a prediction/classification is correct, irrespective of the predicted class. It is simply the sum of the diagonal entries in the confusion matrix, divided by the total number of instances that are classified. A problem of this metric is that it does not distinguish between positive and negative predictions, which can be problematic if (i) we care more about one class than the other one, or (ii) if classes are imbalanced.
- The specificity is defined as the probability that an instance that should be classified as negative is correctly classified. This metric is useful in situations where type I errors (i.e. negative class wrongly predicted as positive) are particularly harmful. A high specificity implies that we have few type I errors. A classifier that simply always assigns a negative class will have a specificity of one, but it can still have a high type II error.
- We should favour high specificity in scenarios where a false positive has catastrophic consequences (e.g. in criminal investigation or court decisions).

# Precision, recall and $f$ -score

- **recall/sensitivity** is probability that positive class is correctly predicted

$$REC = TPR = \frac{TP}{P}$$

		true class	
		negative	positive
predicted class	negative	$TN = 512$	$FN = 28$
	positive	$FP = 10$	$TP = 10$
		$N = 522$	$P = 38$

- **precision** is probability that positive class prediction is correct

$$PRC = \frac{TP}{TP + FP}$$

$$REC = \frac{10}{38} = 0.263$$

$$PRC = \frac{10}{10 + 10} \approx 0.5$$

- **$F_1$ -score** is harmonic mean of precision and recall

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

$$F1 = \frac{20}{20 + 10 + 28} \approx 0.345$$

## Notes:

- The recall or sensitivity is the equivalent to specificity, focusing on type II errors. It is the fraction of instances in the positive class that is correctly classified. Again, we can easily create a classifier with perfect recall sensitivity simply by always predicting the positive class, but such a classifier is likely to have a low specificity.
- We should favour high recall in scenarios where a false negative has severe consequences (e.g. in medical diagnostics or quality assurance).
- The precision captures the probability that an instance that is classified as positive actually belongs into the positive class. It is easy to see that this corresponds to type I errors, i.e. we can see it as a complement to the recall/sensitivity metric. Intuitively, it is easier to reach high precision if we predict less positive instances, i.e. classifying only those as positive where we have high levels of confidence. However, such a classifier would probably have a low recall/sensitivity.
- The overall accuracy of a classifier both in terms of type I and type II errors can be measured in terms of the  $F_1$  score, which is the harmonic mean of precision and recall.



# Link prediction and class imbalance

- ▶ in sparse networks, link prediction exhibits **extreme class imbalance**, i.e.  $|E| \ll |V|^2$  and thus  $P \ll N$
- ▶ class imbalance challenges **interpretation of evaluation metrics**
- ▶ **large computational effort** to compute  $\text{sim}(v, w)$  for all node pairs
- ▶ for each link in test set  $E_{\text{test}}$ , we can generate **negative sample**, i.e.  $(v, w) \notin E_{\text{train}} \cup E_{\text{test}}$
- ▶ yields **balanced problem** with equal number of positive/negative samples
- ▶ limits number of tuples for similarity computation to  $2 \cdot |E_{\text{test}}|$

		true class	
		negative	positive
predicted class	negative	$TN = 35$	$FN = 21$
	positive	$FP = 21$	$TP = 17$
		$N = 38$	$P = 38$

evaluation of binary classifier predicting links for  $k = 20$  node pairs with largest **Common Neighbour** similarity (and negative sampling of 38 node pairs)

$$ACC = 0.684$$

$$REC = 0.4473$$

$$PRC = 0.85$$

$$F1 = 0.586$$

## Notes:

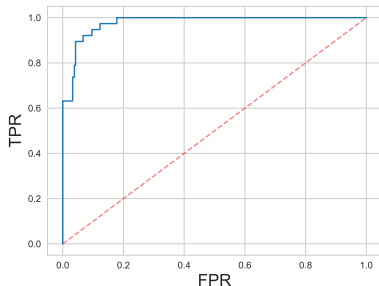
- A common issue in the evaluation of link prediction algorithms is due to the fact that real networks are typically sparsely connected, i.e. only a tiny fraction of all possible links exists. This introduces two separate challenges:
- A first challenge is due to the fact that class imbalance complicates the interpretation of the evaluation metrics introduced before. In particular, depending on the size of the network, the sparsity of links, and the metric that we use we may be tricked into thinking that a link prediction algorithm is either better or worse than it actually is. In our example, the accuracy of 0.932 our Common Neighbours link prediction may seem high, however it is simply high because it only predicts few links and the majority of node pairs are not connected. Even a classifier that randomly predicts a small number of links in a sparse network will be correct for most of the node pairs. Conversely, the precision of our classifier may seem low as it is only 0.5. However, since it is much harder to find a pair of nodes that is connected than one that is not connected, a random classifier would have a much smaller precision so 0.5 is actually good.
- A second challenge is due to the computational effort to evaluate link prediction in larger networks, where we would have to calculate node similarities for an unfeasible number of nodes pairs.
- We can mitigate both issues by means of negative sampling, We can, for instance, randomly sample the same number of node pairs that are not connected by a link as the number of edges in our test set. We now have a balanced binary classification problem. This makes it easy to interpret evaluation metrics and it we only need to calculate similarities for the same number of pairs as we have edges in the test set.

# Receiver-Operating Characteristic

- ▶ more **sensitive** classifiers tend to be less **specific**, i.e. use of larger  $k$  for  $\text{Top}^k$  increases TP and FN
- ▶ consider **tunable discrimination threshold**  $\epsilon$  for similarity score  $\text{sim}(v, w)$ , i.e.

$$\text{predict}(v, w) = \begin{cases} 1 & \text{if } \text{sim}(v, w) > \epsilon \\ 0 & \text{else} \end{cases}$$

- ▶ plot of true vs. false positive rate for  $\epsilon \in [0, 1]$  is called **Receiver Operating Characteristic (ROC)**
- ▶ **area under ROC curve (AUC)** captures discriminative power
  - ▶  $\gg 0.5$ : better than random
  - ▶  $\ll 0.5$ : worse than random



link prediction using  
**Newman-Holme-Leicht index**  
( $\alpha = 0.2$ )

$$AUC \approx 0.98$$

## Notes:

- We can obtain a classifier with perfect recall/sensitivity one if we classify all instances as positive. The number of false positives corresponds to the number of negative instances. Similarly, we trivially find a classifier with specificity one if we classify all results as negative. The recall/sensitivity of this classifier will be zero.
- For similarity-based link prediction, we can thus always obtain a classifier with perfect sensitivity if we set  $k = \binom{n}{2} - |E_{\text{train}}|$ , i.e. if  $\text{Top}^k$  trivially contains all node pairs. Similarly for  $k = 0$  we obtain a classifier that has perfect specificity.
- A plot that shows the true and the false positive rates of a binary classifier as we increase a discrimination threshold  $\epsilon$  from its minimal value to its maximum value is called Receiver-Operating characteristic or ROC curve. For link prediction, rather than using the top  $k$  most similar pairs, we can set a **discrimination threshold**, where we predict a link if the similarity between nodes is larger than  $\epsilon$ . The true and false positive rate will increase equally with  $\epsilon$  if all of the node pairs that are additionally predicted as links are true and false positives with equal probability. For a classifier that has discriminative power, the true positive rate should initially increase faster than the false positive rate. If the opposite is true, we can switch the predicted class to obtain a classifier that has discriminative power.
- We can evaluate the performance of a classifier across different discrimination thresholds with the area under the (ROC) curve (AUC). A perfect classifier has  $AUC = 1$ , in which case we get zero false positives and a true positive rate one for infinitesimal discrimination thresholds. The ROC curve helps us to choose the optimal discrimination threshold for a binary classifier. It enables us to evaluate predictive power across all discrimination thresholds.

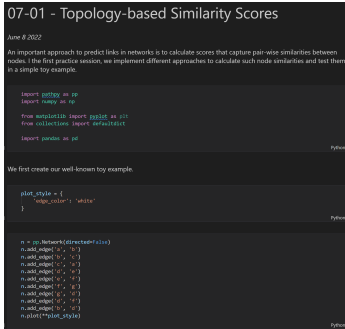
# Practice session

- ▶ we implement **similarity-based classifiers for link prediction**
- ▶ we **evaluate predictive power** of similarity-based link prediction in an empirical network
- ▶ we compare the performance of similarity scores in **graph reconstruction**

## practice session

see notebooks 07-03 and 07-04 in [gitlab repository](https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks) at

→ [https://gitlab.informatik.uni-wuerzburg.de/ml4nets\\_notebooks/2024\\_sose\\_ml4nets\\_notebooks](https://gitlab.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_sose_ml4nets_notebooks)



## Notes:

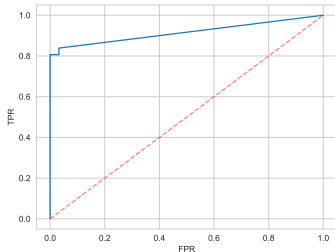
- In the final practice session, we use neighbour- and walk-based similarity scores to implement unsupervised link prediction. We further implement negative sampling and evaluate the discriminative power of the resulting classifiers in empirical social networks.

# Predicting student cooperations

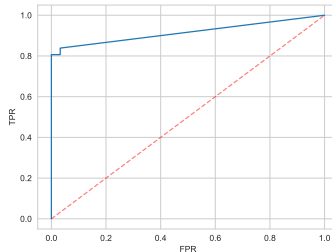
1/2

## example network

- ▶ network of homework cooperations between students at Ben-Gurion University
- ▶  $n = 185$  nodes,  $m = 311$  links
- ▶  $E_{\text{train}} = 90\%$  of links



Adamic-Adar index  
AUC = 0.91



Cosine Similarity  
AUC = 0.91

## Notes:

- We show two of the results from the practice session, where we have used 90 % of the edges in a network of student cooperations to calculate node similarities. The remaining 10 % of edges constitute our test set and we employ negative sampling to generate the same number of node pairs not connected by an edge either in the test or training set.
- For this network, we find that the Adamic-Adar index and the Cosine similarity perform exactly identical, both obtaining an AUC of 0.91.
- Note that the result above is for a single random split in a test/training network. In a real evaluation we would average across multiple random splits and report the average and standard deviation of our results.

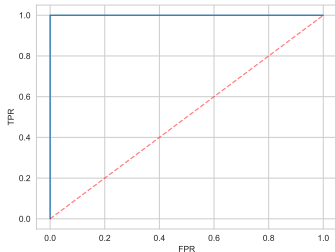


# Predicting student cooperations

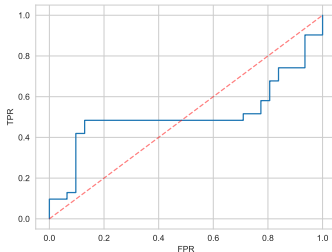
2/2

## example network

- ▶ network of homework cooperations between students at Ben-Gurion University
- ▶  $n = 185$  nodes,  $m = 311$  links
- ▶  $E_{\text{train}} = 90\%$  of links



Katz index ( $\beta = 0.2$ )  
AUC = 1



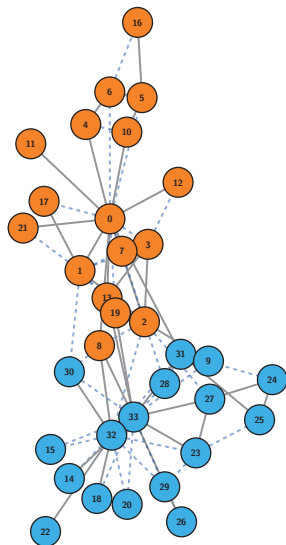
Katz index ( $\beta = 0.3$ )  
AUC = 0.51

## Notes:

- We further test the performance of the Katz index, and show that it crucially depends on the parameter  $\beta$  that balances the influence of short and long walks. For  $\beta = 0.2$  we obtain a perfect prediction in our test set. For a slightly smaller value of  $\beta = 0.3$ , we obtain an AUC that is not better than random! We thus need to treat  $\beta$  as a hyperparameter that we need to tune to the specific data set in which we want to predict links. This can make it complicated to apply the Katz index in some practical settings (e.g. where we do not have sufficient data for hyperparameter tuning).

# In summary

- ▶ we have considered **unsupervised link prediction** in complex networks
- ▶ link prediction algorithms commonly based on notions of **similarity between nodes**
- ▶ we can assess similarity based on **node attributes** that provide **additional information on node features**
- ▶ we can (additionally) assess **topological node similarities**
- ▶ we can use **local and non-local similarity scores** to define **binary classifier** for links




## Notes:

- We summarize today's lecture. We have introduced the link prediction problem in complex networks, which is a key problem in graph learning with many important applications in practice. We can address it in a supervised and unsupervised fashion.
- For unsupervised link prediction, we often build classifiers that are based on a notion of node similarities. Such similarities can often be calculated in a rather easy fashion if we have additional node attributes. In cases where we do not have such information, or where we want to add an additional layer of information, we can use the topology of the network to assess the similarity of two nodes.
- In today's lecture, we have introduced both local and non-local similarity scores. We have discussed how those can be turned into binary classifiers that we can use to address link prediction, and how we can evaluate such a binary classifier.
- In the next lectures, we will introduce advanced techniques that can, among other tasks, be used both for supervised and unsupervised link prediction.

# Exercise sheet 6

- ▶ **sixth exercise sheet** will be released today
  - ▶ explore topology-based similarity scores
  - ▶ implement and evaluate link prediction in empirical networks
  - ▶ consider link prediction based on stochastic block model
- ▶ solutions are due **June 20th** (via WueCampus)
- ▶ present your solution to earn bonus points



Machine Learning for Complex Networks  
SoSe 2022

Prof. Dr. Ingo Scholtes  
Chair of Informatics XV  
University of Würzburg

**Exercise Sheet 06**

Published: June 15, 2022  
Due: June 22, 2022  
Total points: 10

Please upload your solutions to WueCampus as a scanned document (image format or pdf), a typesetted PDF document, and/or as a jupyter notebook.

1. Topology-based Similarity Scores

- (a) Create a synthetic network with heterogeneous degrees in which a link prediction based on the Adamic-Adar index has larger predictive power (in terms of AUC) compared to the Szymkiewicz-Simpson coefficient. 2P
- (b) Consider the cosine similarity score between two nodes  $u$  and  $v$  in an undirected network. Prove that it corresponds to the number of common neighbours normalized by the geometric mean of the degrees  $d_u$  and  $d_v$ . 2P
- (c) Consider a random network with heterogeneous node degrees. Compute the Katz index and the Leicht-Helme-Newman index for all node pairs and plot the correlation between node similarities and the product of the node degrees for all node pairs. What do you observe? 8P

2. SimRank and Generative Models

- (a) The so-called SimRank scores provides a general approach to assess the similarity of objects in relational data. It can also be used to assess the similarity of nodes in a directed graph. Intuitively, SimRank assigns large similarity to a pair of nodes if they are pointed to by the same set of other nodes. 3P
- (b) Consider the Stochastic Block Model and explain how we can use it for link prediction in networks that exhibit a community structure. Implement and evaluate your idea in synthetic networks with community structure that are generated using the stochastic block model. Compare the performance of your approach to that of neighbourhood and walk-based similarity scores. 2P

## Notes:

# Self-study questions

1. Find a network where the Leicht-Holme-Newman index performs better than the Katz index. Which characteristics should such networks have?
2. Construct an example for a network, where walk-based similarities perform better than neighbour-based similarity scores.
3. Prove that the cosine similarity corresponds to the number of common neighbours divided by the geometric mean of the degrees of the nodes.
4. Give an example for an evaluation metric where class imbalance can make a classifier appear to perform better or worse than it actually does.
5. In link prediction, what is the sensitivity/specificity of a  $\text{TOP}^k$  similarity-based classifier for  $k = |V|^2$ ?
6. Investigate how the Katz index can be used to assess the centrality of a node in a network.
7. Create a network in which the Adamic-Adar index performs than the Szymkiewicz-Simpson coefficient.
8. Explain how the Area Under Curve (AUC) score is defined for link prediction. Give examples for similarity scores for which we can expect AUC values close to zero, 0.5 and one.

## Notes:



# References

## reading list

- ▶ P Jaccard: **The Distribution of the Flora in the Alpine Zone 1**, New Phytologist, 1912
- ▶ L Katz: **A New Status Index Derived from Sociometric Analysis**, Psychometrika, 1953
- ▶ L Adamic, E Adar: **Friends and neighbors on the web**, Social Networks, 2003
- ▶ EA Leicht, P Holme, MJ Newman: **Vertex similarity in networks**, Phys. Rev. E 73, 2006
- ▶ F Fouss, A Pirotte, JM Renders, Marco Saerens: **Random-Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation**, IEEE Transactions on Knowledge and Data Engineering, 2007
- ▶ RN Lichtenwalter, JT Lussier, NV Chawla: **New Perspectives and Methods in Link Prediction**, Proc. of SIGKDD, 2010
- ▶ D Liben-Nowell, J Kleinberg: **The Link-Prediction Problem for Social Networks**, Journal of the American Society for Information Science and Technology, 2007
- ▶ G Salton, MJ McGill: **Introduction to modern information retrieval**, 1983
- ▶ L Lü, T Zhou: **Link prediction in Complex Networks: A Survey**, Physica A 390, 2011

## Friends and Neighbors on the Web

Lada A. Adamic  
Xerox PARC  
3333 Coyote Hill Rd  
Palo Alto, CA 94304  
(650) 612-4770

ladamic@parc.xerox.com

Eytan Adar  
HP SRI and HP Laboratories  
1501 Page Mill Road, Rm. 1U-19  
Palo Alto, CA 94304  
(650) 857-2396  
eytan@hpl.hp.com

### ABSTRACT

The Internet has become a rich and large repository of information about its individuals. Averting from the tasks and work on a user's homepage to the making links the user subscribes to are reflections of social structures in user links in the web world. In this paper we devise techniques to mine this information in order to perfect relationships between individuals. Further we show that some pieces of information are better indicators of social connections than others, and that these indicators vary between user populations and provide a glimpse into the social lives of individuals in different communities. Our techniques provide potential applications in automatically inferring web-world connections and discovering, finding, and characterizing communities.

### Keywords

Homepage analysis, social network, social worlds, web communities

### 1. INTRODUCTION

One of the first large scale web applications was the serving of individual homepages. These generally autobiographical pages reflect a user's interests and experiences. They include everything from photographs of the user's pet to the user's essay or resume. Homepages are not dissimilar to the web, but point to and are presented by their users, are "breaks and emphases" on the web. These links can represent anything from friendship to collaboration, to general interest in the material on the other user's homepage. In this way individual homepages become part of a large community structure.

Recent work [5, 7, 11] has attempted to use analysis of link topology to find "web communities." These web communities are web page collections with a shared topic. For example, any page dealing with data mining and linking to other pages on the same topic would be part of the data mining page collection. Such a page is not necessarily a homepage or even associated with a particular individual. In contrast, our work focuses on individuals' homepages and the connections between them, essentially allowing us to tap into both virtual and real world communities of people.

Although homepage identification has been researched as a separate problem [2, 12], we use knowledge that in the first link analysis on a network of homepages. Rather than discarding the previous concept that pages which share a topic are likely to link to one another, we use more use it to characterize relationships between people. For example, not people who mention "baseball" likely to link to each other? Consequently, we use

links on homepages to predict where connections between individuals will exist? And furthermore, which items are best at predicting connections, or "baseball" a better predictor than "baseballing"? Here we describe and evaluate techniques to answer the above questions. While the name of homepages is to provide a view of the individual user and their social relationships to others, as a side effect they provide an interesting view of whole communities!

### 1.1 Information Side Effects

Information side effects are by-products of data intended for one use which can be mined in order to understand user targeted, and possibly larger scale, phenomena. A more example of information side effects is the RadioCensus census [4]. RadioCensus mines information from cell phone base stations that show the land use or even users in order to determine traffic conditions. Congested roadways yield show a increased load on base stations due to more web use.

Just as it is possible to extract global traffic patterns from a device intended to provide communications between two individuals, we can likewise extract large social networks from individualized homepages. Users linking to one another form a peer social network which is easy to harvest and provides a lot of information about the community. Gathering information on relationships between people and the content of those relationships, which can range from collaborations (or friendships) to shared interests (or hobbies), is an inherent task for social network researchers. Data is acquired through user-consenting phone or live interviews. We are able to harvest this information easily and automatically because it is already available as a side effect of people living a digital life. This presents an unprecedented opportunity to discover new and interesting social and cultural phenomena.

The data we study in this paper is shown in Figure 1, chosen from the following three different sources:

<sup>1</sup>All the information used in this analysis, with the exception of the MIT mailing lists, was publicly available. While we do not consider ourselves to be in violation of the spirit in which this information was made available, the potential for release of methods such as ours leads to an interesting set of ethical questions.

<sup>2</sup>Work done while author was at Xerox PARC

## Notes: