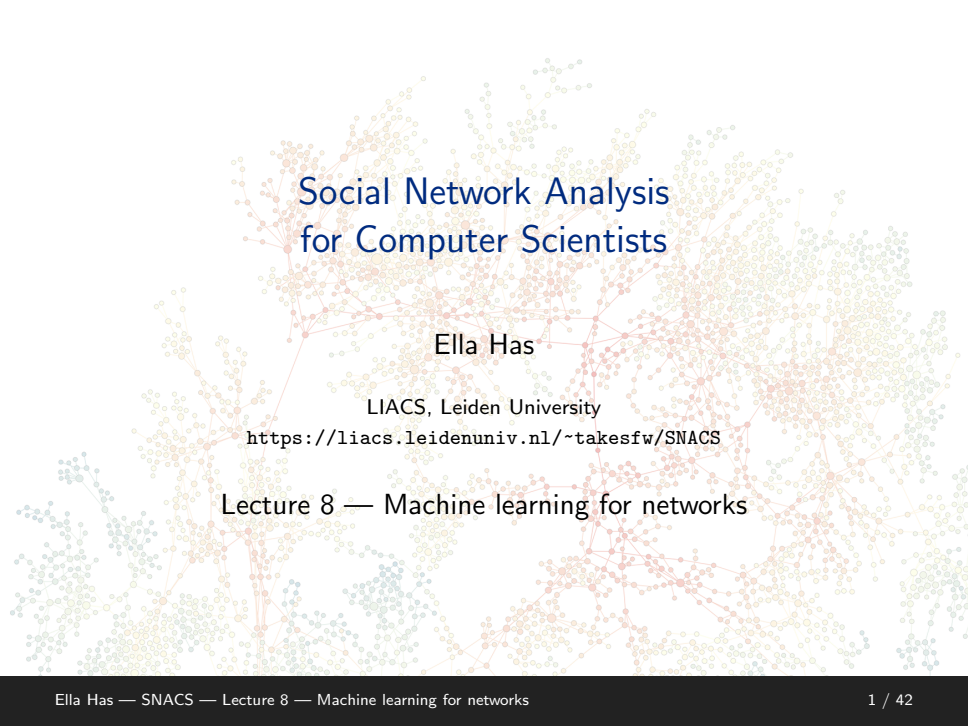




Social Network Analysis for Computer Scientists

Ella Has

LIACS, Leiden University

<https://liacs.leidenuniv.nl/~takesfw/SNACS>

Lecture 8 — Machine learning for networks

Today

- Recap
- Introduction
- Random-Walk Embeddings
- Graph Neural Networks
- Applications

Recap

Notation

Concept

- Network (graph)
- Nodes (objects, vertices, ...)
- Links (ties, relationships, ...)
 - Directed — $E \subseteq V \times V$ — “links”
 - Undirected — “edges”
- Number of nodes — $|V|$
- Number of edges — $|E|$
- Degree of node u
- Distance from node u to v

Symbol

$$G = (V, E)$$

 V E n m $\deg(u)$ $d(u, v)$

Real-world networks

- 1 Sparse networks density
- 2 Fat-tailed power-law degree distribution degree
- 3 Giant component components
- 4 Low pairwise node-to-node distances distance
- 5 Many triangles clustering coefficient
- Many examples: social networks, communication networks, citation networks, collaboration networks, protein interaction networks, information networks, webgraphs, financial networks, ...

Centrality measures

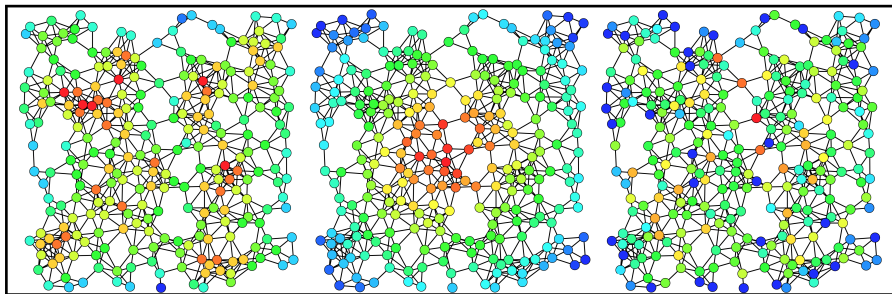


Figure: Degree, closeness and betweenness centrality.

Source: "Centrality" by Claudio Rocchini, Wikipedia File:Centrality.svg

Attribute assortativity

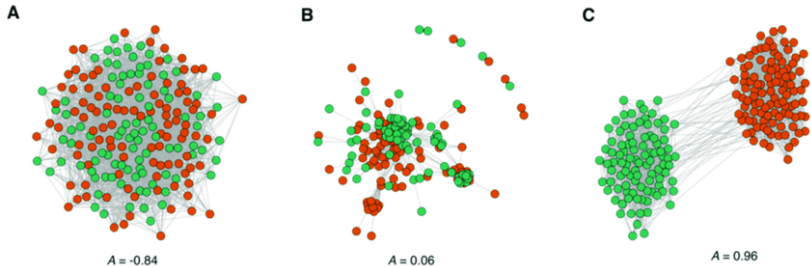


Figure: Attribute assortativity.

Image: Moya-García, A. et al. Identification of New Toxicity Mechanisms ... *Genes*, 13(7), 1292, 2022.

Projection

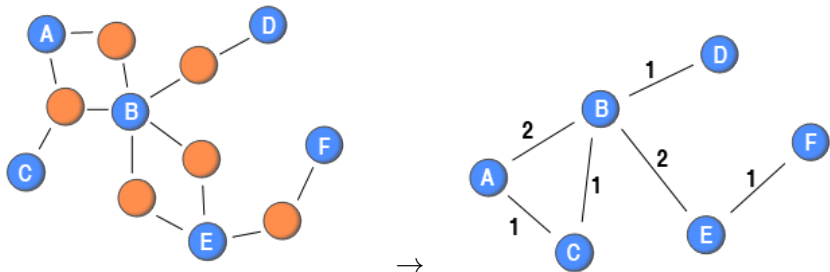


Image: <http://toreopsahl.com>

Community detection

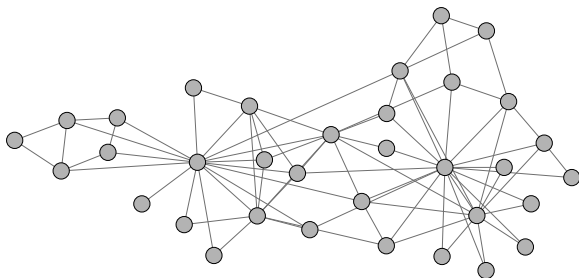


Figure: Communities: node subsets connected more strongly with each other.

- Partitioning a given graph into densely connected groups of nodes

Community detection

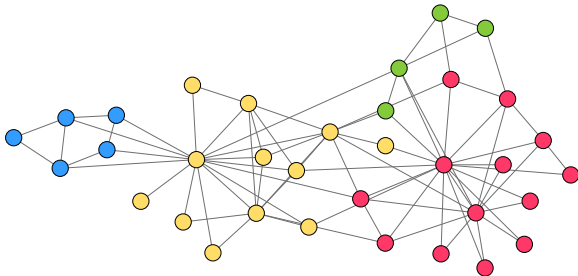


Figure: Communities: node subsets connected more strongly with each other.

- Divisive algorithms (Girvan-Newman), local search algorithms (Louvain) optimizing a quality function (e.g., modularity), label propagation and random walk based approaches.

Core/periphery structure

- Many large networks have a densely connected **core** and a **periphery** of nodes attaching to only a few neighbors
- **Core/periphery structure**: there is **one** dense area
- **Community structure**: there are **several**, with little periphery

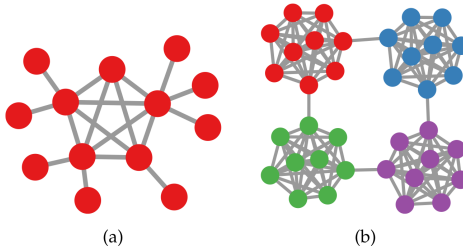
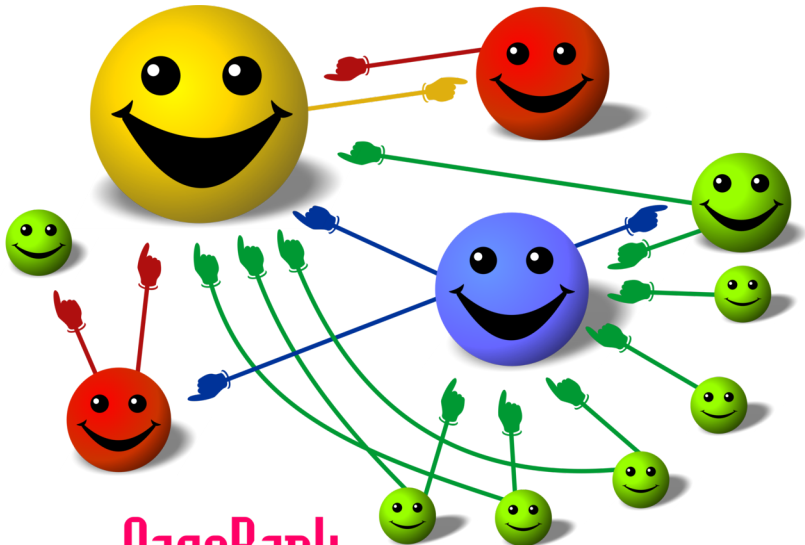
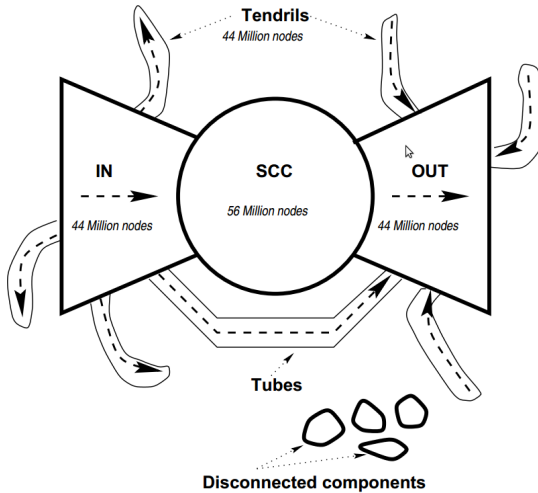


Figure: (a) Core/periphery structure and (b) Community structure.



PageRank

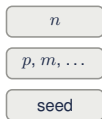
Webgraph



Broder et al., Graph structure in the web, Computer Networks 33(1): 309–320, 2000.

Network models

Input parameters



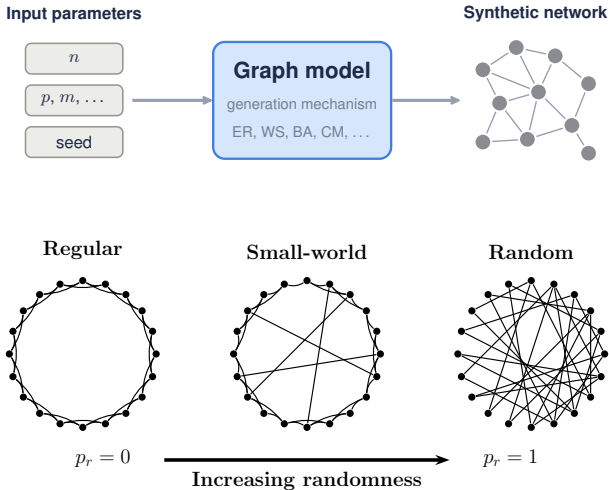
Graph model

generation mechanism
ER, WS, BA, CM, ...

Synthetic network



Network models



Based on: Watts, D., Strogatz, S. Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998).

Lancichinetti–Fortunato–Radicchi (LFR) graphs

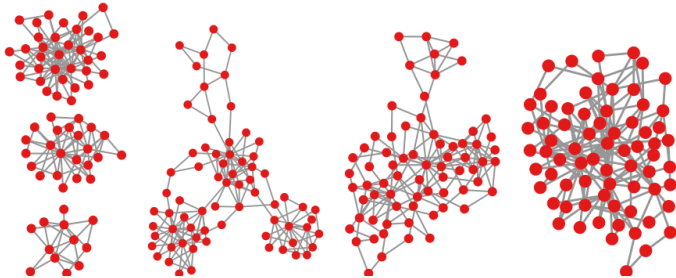


Figure: LFR models with community mixing parameter $\mu = 0, 0.05, 0.2, 1$.

Centrality-based centralization

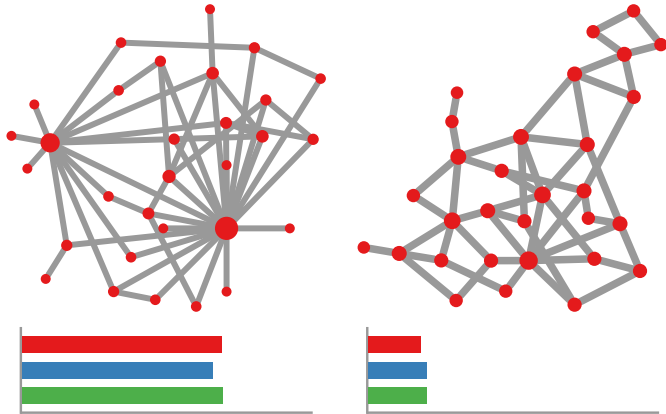


Figure: Centralization of two networks according to degree (red), betweenness (blue) and closeness (green) centrality; bar length is the centralization value.

Graph evolution

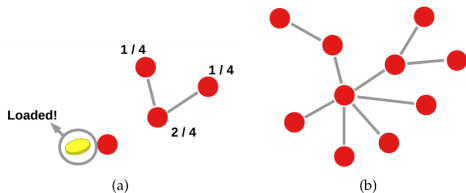


Figure: PA: target node based on degree (a), graph formed through PA (b).

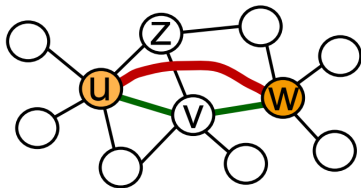


Figure: Triadic closure

Network sampling

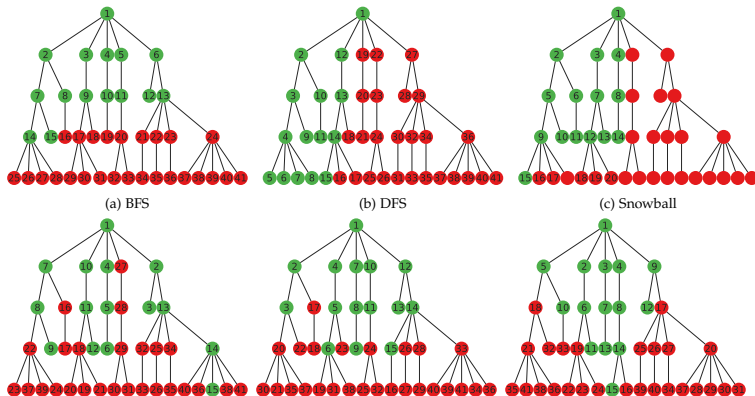


Figure: Six sampling strategies on the same network with a budget of 15 nodes. Labels give the exploration order; nodes are sampled (green) or not (red).

Machine Learning and Networks

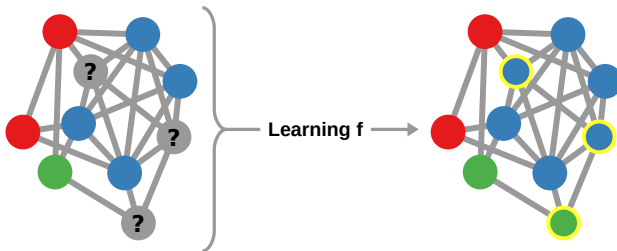
Network tasks and machine learning

Many network tasks are prediction tasks:

- Node classification
- Link prediction
- Anomaly detection

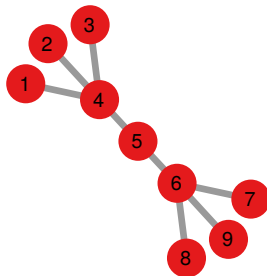
Machine Learning models are useful for prediction, but

- nodes and edges cannot be parsed
- adjacency matrices are too large and sparse

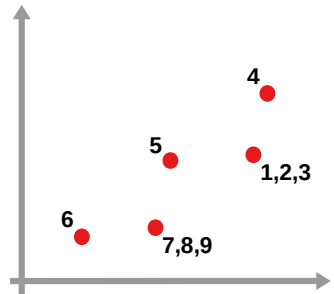


Network embedding

Solution: dense vector representations to represent the network
Similar nodes $\rightarrow$ similar vectors

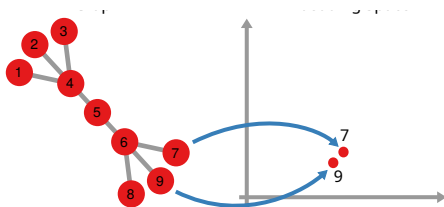


Node	Embedding
1	$\{0.88, 0.69\}$
2	$\{0.88, 0.69\}$
3	$\{0.88, 0.69\}$
4	$\{1, 1\}$
5	$\{0.48, 0.66\}$
6	$\{0, 0\}$
7	$\{0.42, 0.1\}$
8	$\{0.42, 0.1\}$
9	$\{0.42, 0.1\}$



Properties of good embeddings

- Structural properties of the network should be captured
- Similar nodes should be close together in vector space
 - Neighbors
 - Structurally equivalent nodes
- Allow for good performance downstream machine learning tasks



Loss functions

Different approaches to optimizing for good embeddings

- Node similarity (unsupervised/self-supervised)
- Graph reconstruction (self-supervised)
- Task performance (supervised)

Embedding methods

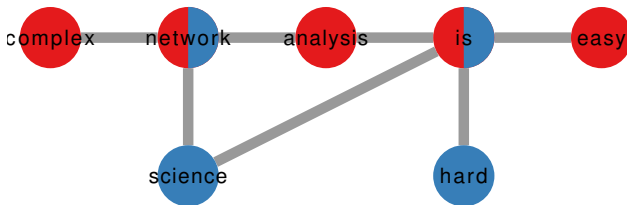
- Random-walk embeddings
- Graph neural networks (GNNs)
- Matrix factorization/spectral embeddings (not covered here)

Random-walk Embedding

Analogy to language

word2vec learns word representations from their context

- Input: corpus of text
- Output: vector representation for each word in text
- Words appearing in similar context close together

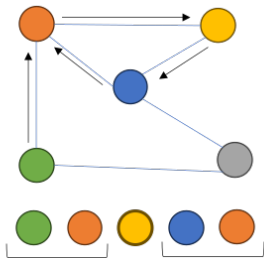


Complex network analysis is easy.
Network science is hard.

Nodes instead of words

- Input: (un)directed, (un)weighted unattributed graph
- Sample random walks
- Sequence of nodes visited can act like a “sentence”
- Nodes with similar contexts get similar representations
- **Context:** the nodes appearing in random walks before and after a node, at a certain distance (window size)

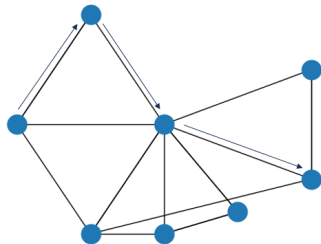
The powerful **king** ruled over the kingdom.



DeepWalk

DeepWalk samples uniform random walks

- 1 Start a walk from node v_0
- 2 Uniformly sample one of v_i 's neighbors v_{i+1} , and repeat
- 3 Train embeddings with SkipGram
 - Use negative sampling to estimate co-occurrence probabilities



SkipGram

- Learn representation Φ that maximizes probability of the context given representation
 - 1 Cycle over the target nodes v_i in the random walks
 - 2 Per node u_j in context window of size c around v_i :
 - Loss: $J(\Phi) = -\log \Pr(u_j | \Phi(v_i))$
 - Gradient descent: $\Phi = \Phi - \alpha \cdot \frac{\partial J}{\partial \Phi}$
- Probability is expensive to compute for each target-context pair: $O(c |V|^2)$, so it is approximated with **negative sampling**

Negative sampling

Sample k nodes from **noise distribution** $Pr_n(v)$

- Objective: distinguish sampled nodes from nodes in context based on representation

$$J(\Phi) = -\log \sigma(\Phi(v_i)^\top \Phi'(u_j)) - k \mathbb{E}_{w \sim Pr_n(v)} [\log \sigma(-\Phi(v_i)^\top \Phi'(w))]$$

- Noise distribution choice
 - Uniform
 - Proportional to total frequency in corpus

Biased walks

Uniform walks are inefficient exploration

Node2vec: parameterize probabilities

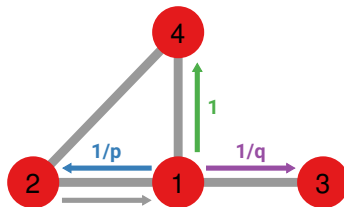
- Parameters:

- p : regulates backtracking (breadth-first search)
- q : regulates depth-first search

- Unnormalized probabilities:

- Returning to previous node: $\frac{1}{p}$
- Moving to common neighbor: 1
- Into depth: $\frac{1}{q}$

Parameters can be chosen or learned from data



Embedding Zachary's Karate Club network

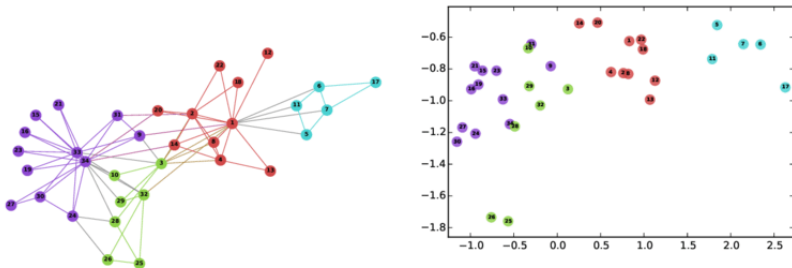
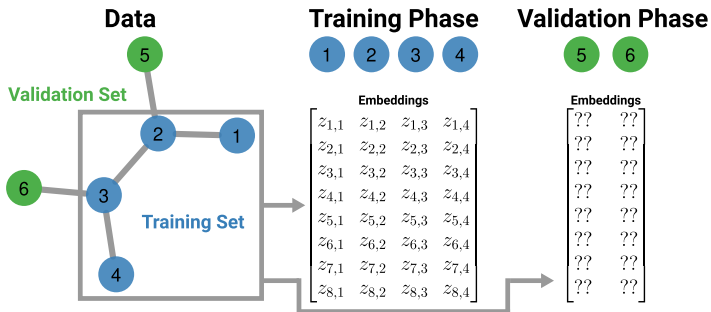


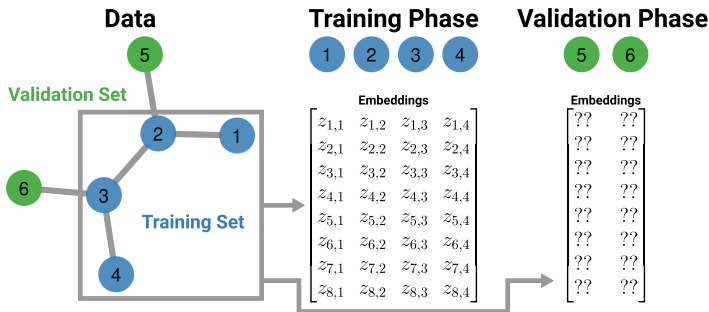
Figure: DeepWalk embedding of Zachary's Karate Club. Source: Rizi (2021)

- Embedding clusters nodes by community (roughly)
- Structurally similar nodes near each other (purple low-degree nodes)

Limitations



Limitations



- **Transductive:** cannot embed unseen nodes without complete retraining
 - Train/test split not possible
 - New nodes over time
- Does not incorporate node features

Graph Neural Networks

Graph Neural Networks

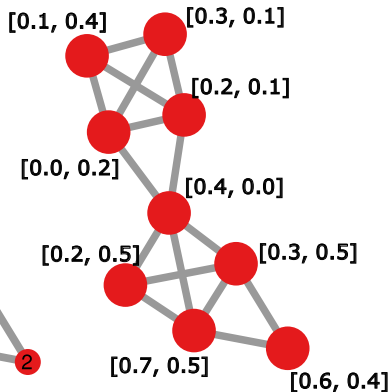
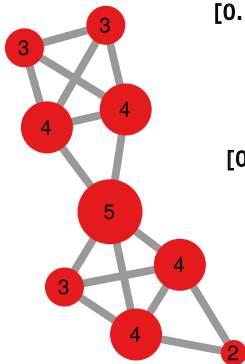
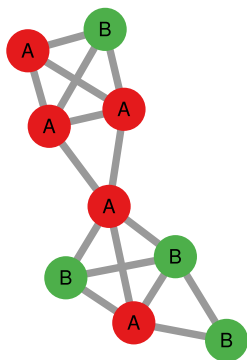
Graph neural networks offer more flexibility

- Use additional node features (tabular data about each node)
- Less retraining for new nodes
- Leveraging innovations in neural network architectures for matrix data

Graph Neural Networks

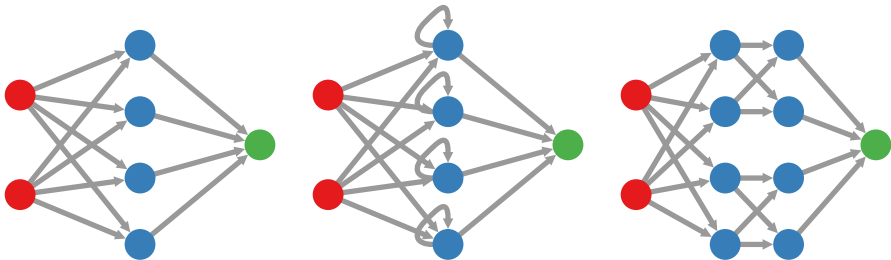
- Node features
- From image convolution to graph convolution
- Message passing
- Training
- Challenges
- Architectures

Attributed networks



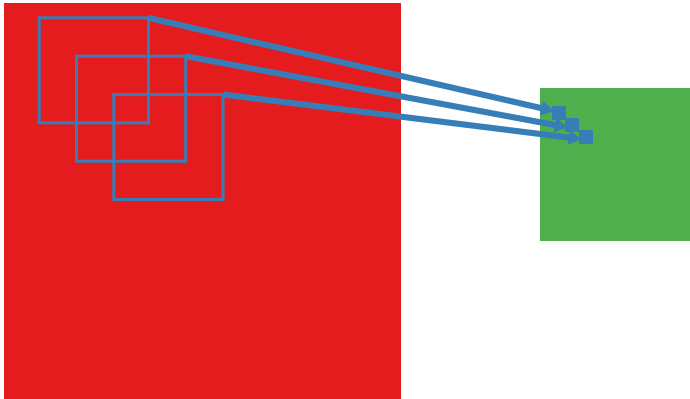
Neural networks

- Layers of neurons
- Connections with weights
- Non-linear activation functions

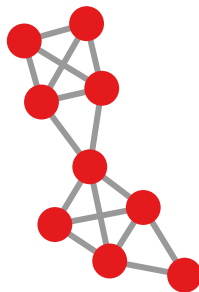
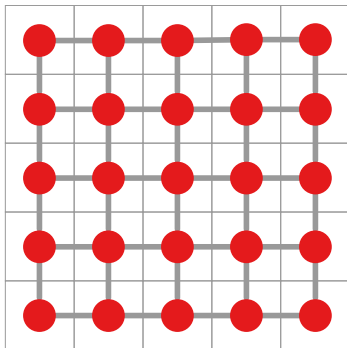


Convolutional Neural Networks

- Kernel: shared weights across pixels
- Weighted sum of each pixel's neighboring pixels

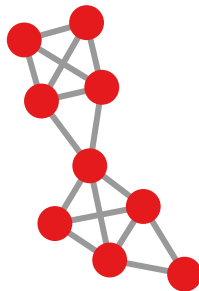
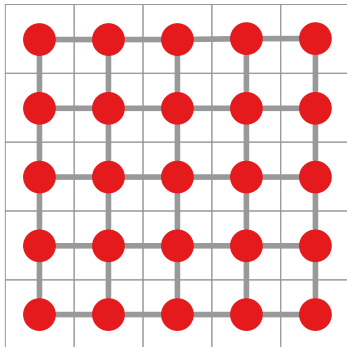


Graph Convolutional Networks (GCN)



Differences image and graph data

Graph Convolutional Networks (GCN)

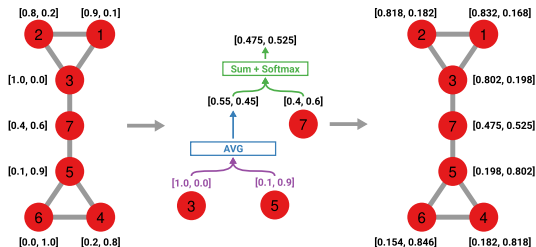


Differences image and graph data

- Number of neighbors (constant or variable)
- Relevance of order of neighbors (clear ordering or arbitrary)

Basic GCN embedding

- 1 Input: *attributed*, (un)directed, (un)weighted graph
- 2 Initialize feature vector for each node
- 3 For K layers: (usually $K = 2$)
 - For each node:
 - Sample neighbors
 - Message passing: aggregate neighbors' features
 - Fully connected neural network layer (layer-specific)
 - Normalize feature vectors
- 4 Embeddings: output feature vectors last layer



Training

- Learn parameters fully connected layers (and parameters aggregation function if applicable)
- Loss function
 - Self-supervised: e.g., similar nodes near each other, or graph reconstruction performance
 - Supervised: task-specific performance
- Stochastic gradient descent

Message passing

Let's simulate the message-passing mechanism

- Take some (3-5) pieces of paper and write down how many months you have been at Leiden University (your node feature value) on each piece

Message passing

Let's simulate the message-passing mechanism

- Take some (3-5) pieces of paper and write down how many months you have been at Leiden University (your node feature value) on each piece
- Pass the pieces of paper to your neighbors next to you, in front, and behind

Message passing

Let's simulate the message-passing mechanism

- Take some (3-5) pieces of paper and write down how many months you have been at Leiden University (your node feature value) on each piece
- Pass the pieces of paper to your neighbors next to you, in front, and behind
- Take the average of the numbers you received (intermediate result)

Message passing

Let's simulate the message-passing mechanism

- Take some (3-5) pieces of paper and write down how many months you have been at Leiden University (your node feature value) on each piece
- Pass the pieces of paper to your neighbors next to you, in front, and behind
- Take the average of the numbers you received (intermediate result)
- Incorporate your own initial feature value by taking the average of your feature and the intermediate result

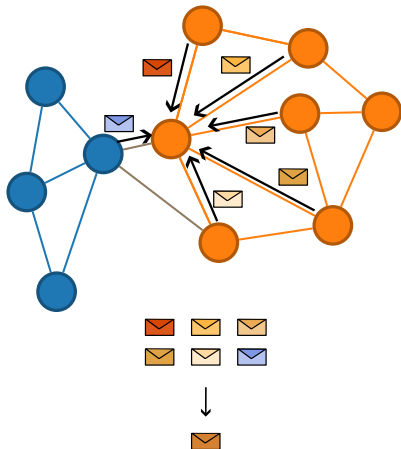
Message passing

Let's simulate the message-passing mechanism

- Take some (3-5) pieces of paper and write down how many months you have been at Leiden University (your node feature value) on each piece
- Pass the pieces of paper to your neighbors next to you, in front, and behind
- Take the average of the numbers you received (intermediate result)
- Incorporate your own initial feature value by taking the average of your feature and the intermediate result
- See how your and your neighbors' embeddings have changed from your original node features
- What does this new value represent?

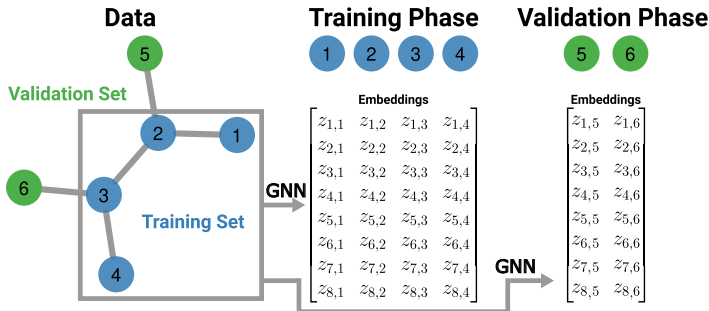
Message passing

- After one round, embeddings contain information from direct neighbors
- Another round adds information from neighbors of neighbors
- After K steps, each node has information from K -hop neighbors



Inductive embedding

- **Inductive embedding:** new (unseen) nodes can be embedded using the trained network
- Only K -hop neighbors are affected by addition of a node and need to update embeddings



Challenges

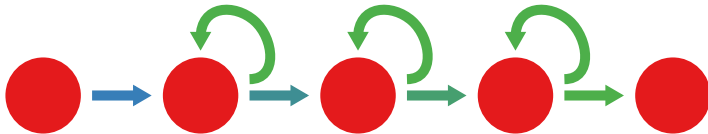
- Over-smoothing: repeating message passing too many times $\rightarrow$ whole graph has the same features

Challenges

- Over-smoothing: repeating message passing too many times $\rightarrow$ whole graph has the same features
 - Skip-connection: aggregate passed message with original features (like we did before)
 - Jumping knowledge: concatenate embeddings from all layers as final embedding

Challenges

- Over-smoothing: repeating message passing too many times $\rightarrow$ whole graph has the same features
 - Skip-connection: aggregate passed message with original features (like we did before)
 - Jumping knowledge: concatenate embeddings from all layers as final embedding
- Over-squashing: signal from far-away nodes gets diluted



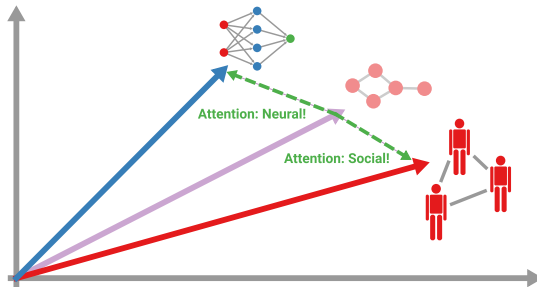
GNN architectures

Multiple GNN architectures derived from "regular" neural network architectures, including:

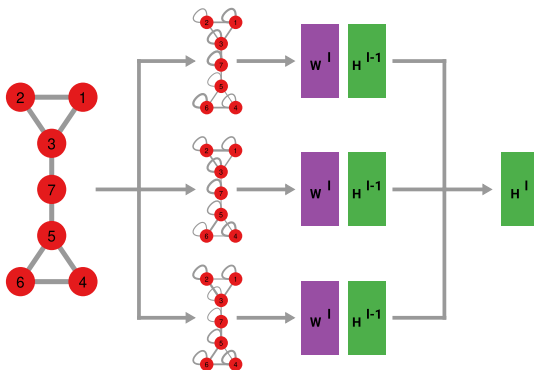
- Graph attention
- Graph transformers
- Autoencoders

Graph Attention Network (GAT)

- Attention: which neighbors to weigh heavier in message passing
- Extra degree of freedom: learn which neighbors are more important based on features of previous layer and trainable weights

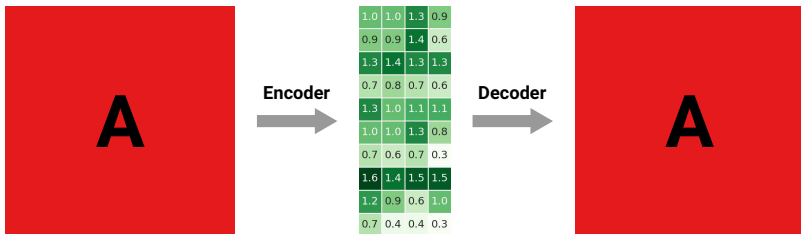


Graph Transformer



- Multiple attention heads $\rightarrow$ transformer
- Allows learning of different divisions of attention in parallel

Autoencoder



- Train encoder to produce embeddings that capture adjacency matrix
- Train decoder to reconstruct adjacency matrix from embeddings
- Hidden embedding layer captures relevant features of the network
- Variational autoencoder (VAE): learns distribution over hidden embeddings → allows generation of new similar networks
- Encoder and decoder layers can be GCN, GAT, etc.

Applications

Recommendations

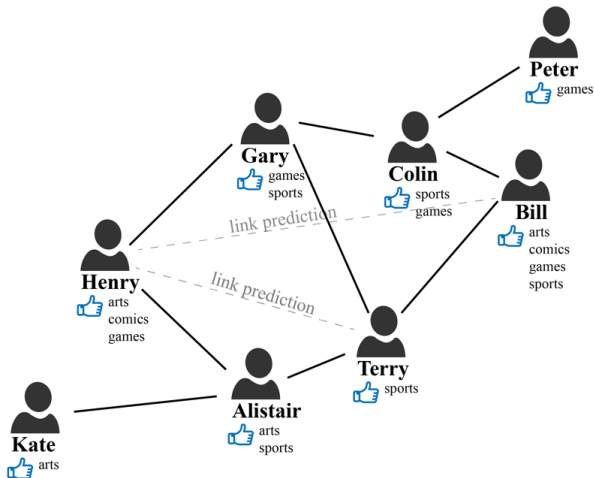
- Social media: people to follow, posts you may like
- Web shops: "others also bought..."
- Streaming services: what to watch next

People you may know at LinkedIn

 B Software Engineer, Machine Learning ... 6 mutual connections Connect	 C Staff Software Engineer at LinkedIn 32 mutual connections Connect	 D Software Engineer 11 mutual connections Connect	 A Software Engineer at LinkedIn 8 mutual connections Connect
--	---	--	---

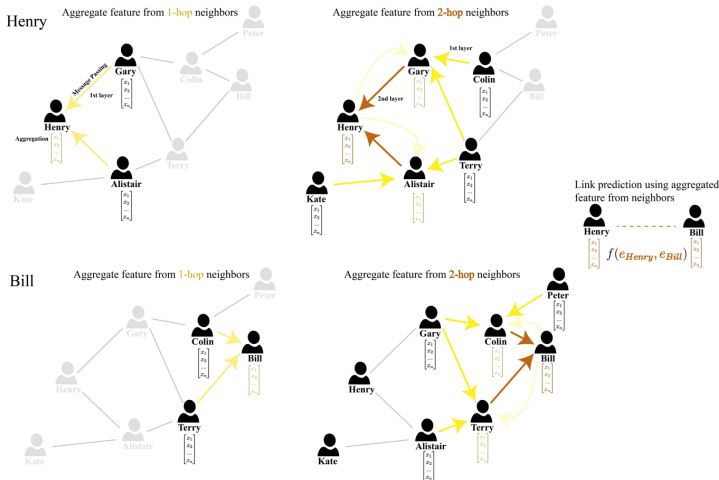
<https://www.linkedin.com/blog/engineering/member-customer-experience/optimizing-pymk-for-equity-in-network-creation>

Recommendation systems



<https://aws.amazon.com/blogs/machine-learning/graph-based-recommendation-system-with-neptune-ml-an-illustration-on-social-network-link-prediction-challenge>

Embeddings for Link Prediction



<https://aws.amazon.com/blogs/machine-learning/graph-based-recommendation-system-with-neptune-ml-an-illustration-on-social-network-link-prediction-challenge>

Anomaly detection

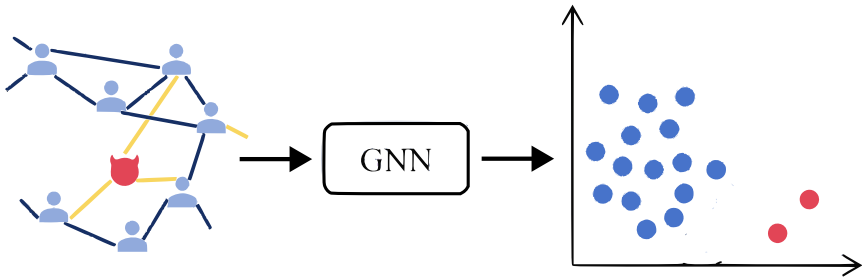


Figure: Adapted from Huang et al. (2026), Fig. 1.

Summary

- Network embeddings are lower-dimensional representations of networks
- Network embeddings allow the application of machine learning models
- Random-walk embedding methods sample walks on (unattributed) graphs and learn embeddings based on context/neighborhood similarity
- GNNs generate network embeddings informed by structure and node features to optimize a loss function

Upcoming lab session and “Homework”

- Continue on Assignment 2

Credits

- Several images are gratefully reused from Michele Coscia, The Atlas for the Aspiring Network Scientist, v3 of <https://arxiv.org/abs/2101.00863>