CS 886: Graph Neural Networks
Lecture 1: Introduction, Problems and Applications

Kimon Fountoulakis

School of Computer Science
University of Waterloo

07/01/2025

Goals of the course

- Learn about vanilla and state-of-the-art graph neural networks
- Learn about applications of graph neural networks

Goals of the course

Topics

1. Generalization performance of graph neural networks
2. Expressive power of graph neural networks
3. Large language models and graphs
4. Neural algorithmic reasoning
5. Generative graph neural networks
6. Self-supervised learning in graphs
7. Oversmoothing
8. Scalability

Architectures:

1. Spectral and spatial convolutional graph neural networks
2. Graph attention networks
3. Invariant and equivariant graph neural networks
4. General message passing graph neural networks
5. Higher-order graph neural networks
6. Graph neural networks for heterogeneous graphs

Tentative Schedule

(Tentative) Schedule

The schedule below is subject to change:

Week	Date	Topic	Readings	Slides
1	1/7	Introduction, problems and applications (Kimon lecturing)	- Geometric Deep Learning (Chapter 1) - Geometric foundations of Deep Learning - Towards Geometric Deep Learning I: On the Shoulders of Giants - Towards Geometric Deep Learning II: The Perceptron Affair - Towards Geometric Deep Learning III: First Geometric Architectures - A Gentle Introduction to Graph Neural Networks - Intro to graph neural networks (ML Tech Talks) - Foundations of Graph Neural Networks	
1	1/9	Spatial graph convolution and its theoretical performance on simple random data, Part 1 (Kimon lecturing)	- Semi-Supervised Classification with Graph Convolutional Networks - Graph Convolution for Semi-Supervised Classification: Improved Linear Separability and Out-of-Distribution Generalization - Code for reproducing the experiments - Theory of Graph Neural Networks: Representation and Learning - PyTorch code for GCN - Example code	
2	1/14	Spatial graph convolution and its theoretical performance on simple random data, Part 2 (Kimon lecturing)	- Effects of Graph Convolutions in Multi-layer Networks - Code for reproducing the experiments - Theory of Graph Neural Networks: Representation and Learning - PyTorch code for GCN - Example code	

https://github.com/opallab/cs886-winter-2025

Workload breakdown

- Class participation: 15%
- Midterm project: 20%
- Presentations: 25%
- Final project: 40%

Tentative Schedule

- No presentations the first 5 seminars.
- I will present my perspective of graph neural networks on synthetic data.
- After the first 5 seminars we will continue with student presentations (more on that later).

Presentations (25% of the total mark)

- There will be $\approx$ 19 seminars where students present a paper.
- Each student will present $\approx$ 19/x, where “x” is the number of students.
- Each presentation will be 40-50 minutes + questions.
- At each seminar 2 students will present.
- Meeting with myself (online) before the presentation for feedback and rehearsal

Presentations (25% of the total mark)

- I expect the following questions to be answered in each presentation:
  - What is the problem?
  - Why is it important?
  - Why don’t previous methods work on that problem?
  - What is the solution to the problem the authors propose?
  - What interesting research questions does the paper raise?

Presentations (25% of the total mark)

- The presentation will be in slides.
- The material has to be yours, but you are allowed to use figures, tables etc from the paper. Make sure to cite the source.
- Have a separate slide for each of the 5 “Why/What” questions.
- Have examples in your presentations.

Midterm project (20% of the total mark)

- To be delivered before the beginning of the reading week, February 15.
- 3 pages + references
- You may use your midterm project as a foundation for the final project.

Options for the midterm project:

- Option A (Empirical evaluation): Pick a problem that interests you. Implement and experiment with several graph neural network methods to tackle this problem.
- Option B (Method design): Identify a problem for which there are no satisfying approaches. Develop a new graph neural network architecture to tackle this problem. Analyze theoretically and/or empirically the performance of your technique.
- Option C (Theoretical analysis): Identify a problem or a graph neural network architecture for which theoretical performance (e.g., complexity, performance on random data, expressivity) is not well understood. Analyze the properties of this problem or technique.

Final project (40% of the total mark)

- To be delivered before the end of the term, April 4.
- 6 pages + references
- You may use your midterm project as a foundation for the final project.

Options for the midterm project:

- Option A (Empirical evaluation): Pick a problem that interests you. Implement and experiment with several graph neural network methods to tackle this problem.
- Option B (Method design): Identify a problem for which there are no satisfying approaches. Develop a new graph neural network architecture to tackle this problem. Analyze theoretically and/or empirically the performance of your technique.
- Option C (Theoretical analysis): Identify a problem or a graph neural network architecture for which theoretical performance (e.g., complexity, performance on random data, expressivity) is not well understood. Analyze the properties of this problem or technique.

Publish your work

@NeurIPS 2024

Analysis of Corrected Graph Convolutions

Robert Wang · Aseem Baranwal · Kimon Fountoulakis

Student @CS886

East Exhibit Hall A-C #3005

[Abstract]
Fri 13 Dec 11 a.m. PST — 2 p.m. PST

Course website

- Course website: https://github.com/opallab/cs886-winter-2025 <— schedule, work breakdown, grades etc.

Required mathematical background

- Basic linear algebra, i.e., vector and matrix operations, matrix decomposition, eigen- and svd-decompositions.

- Multivariate calculus

- Basic probability, i.e., common distributions, means, and so on

- Basic machine learning concepts

Programming languages

python™

Texts

- Check the course website. I have collected a large list of texts, video lectures and other material.

- https://github.com/opallab/cs886-winter-2025

Let’s start…

A social network

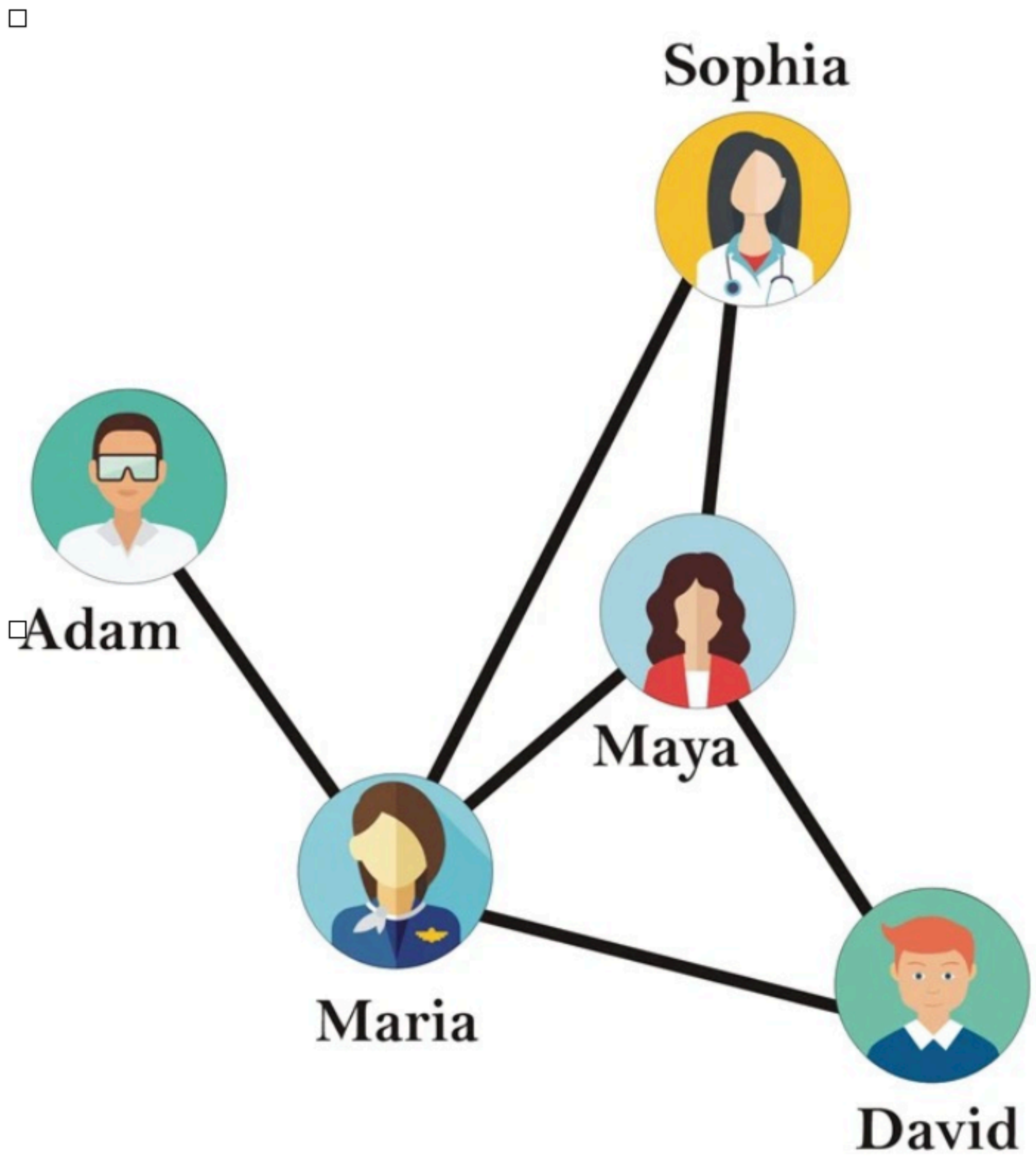
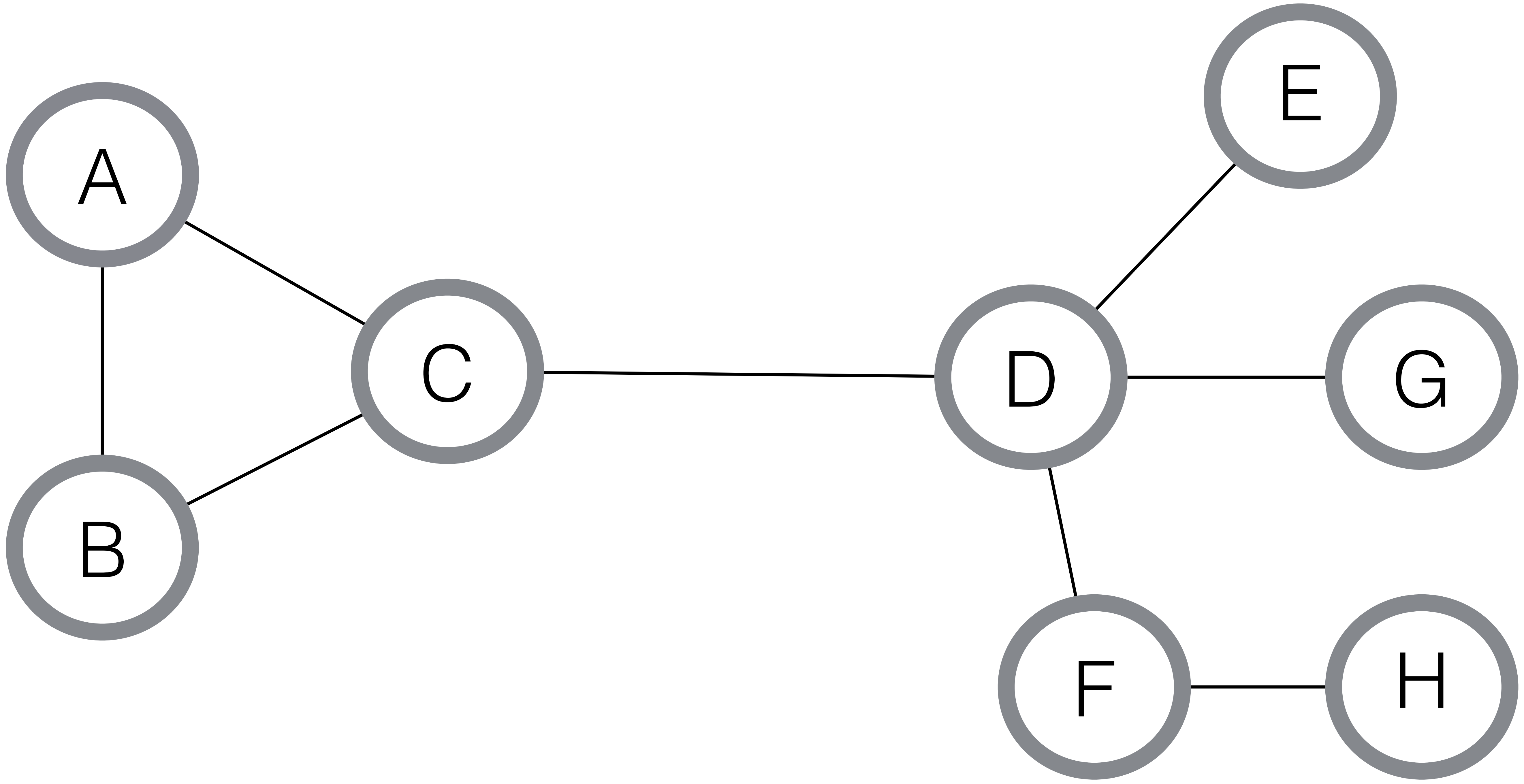
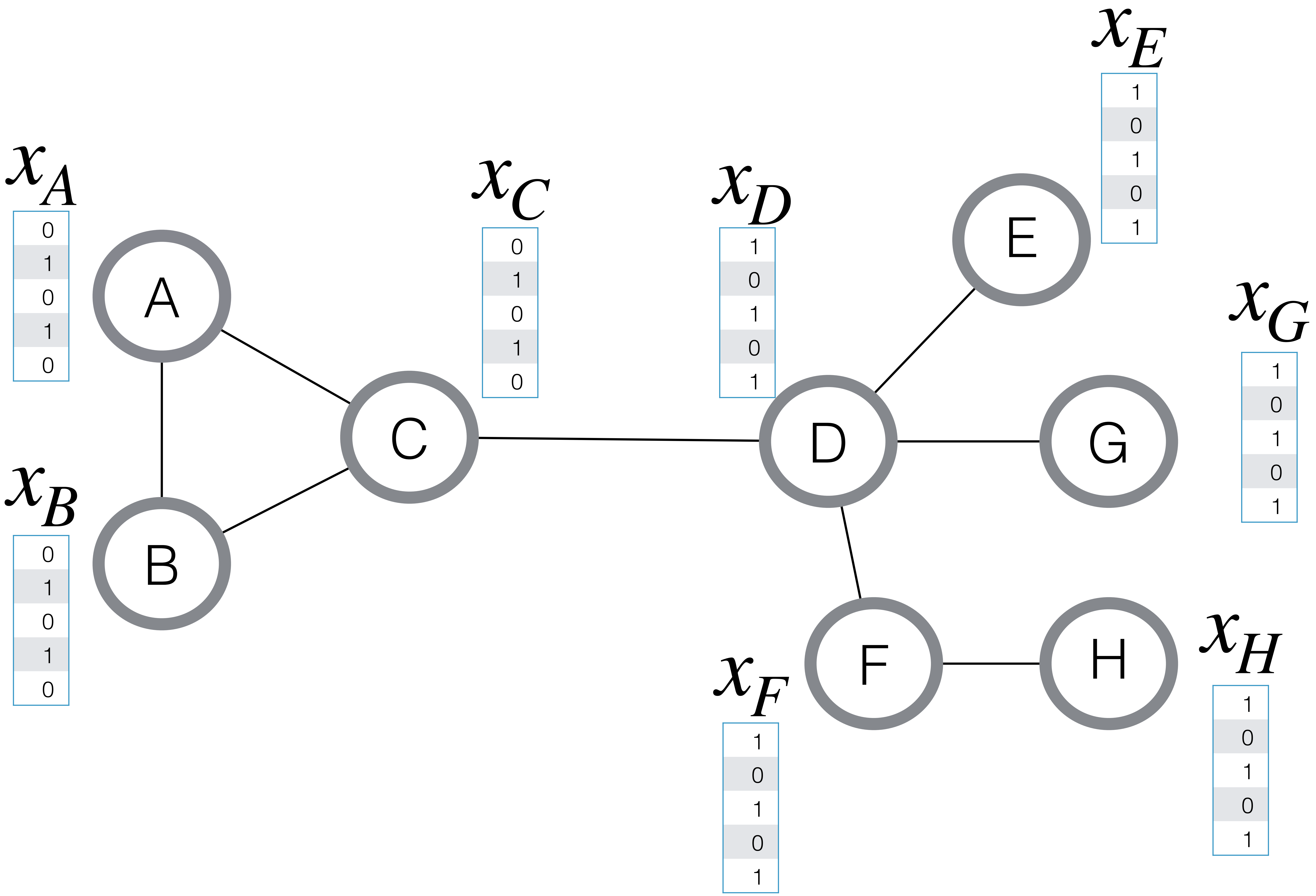


image source: https://www.nature.com/articles/s41598-019-57304-y

Graphs

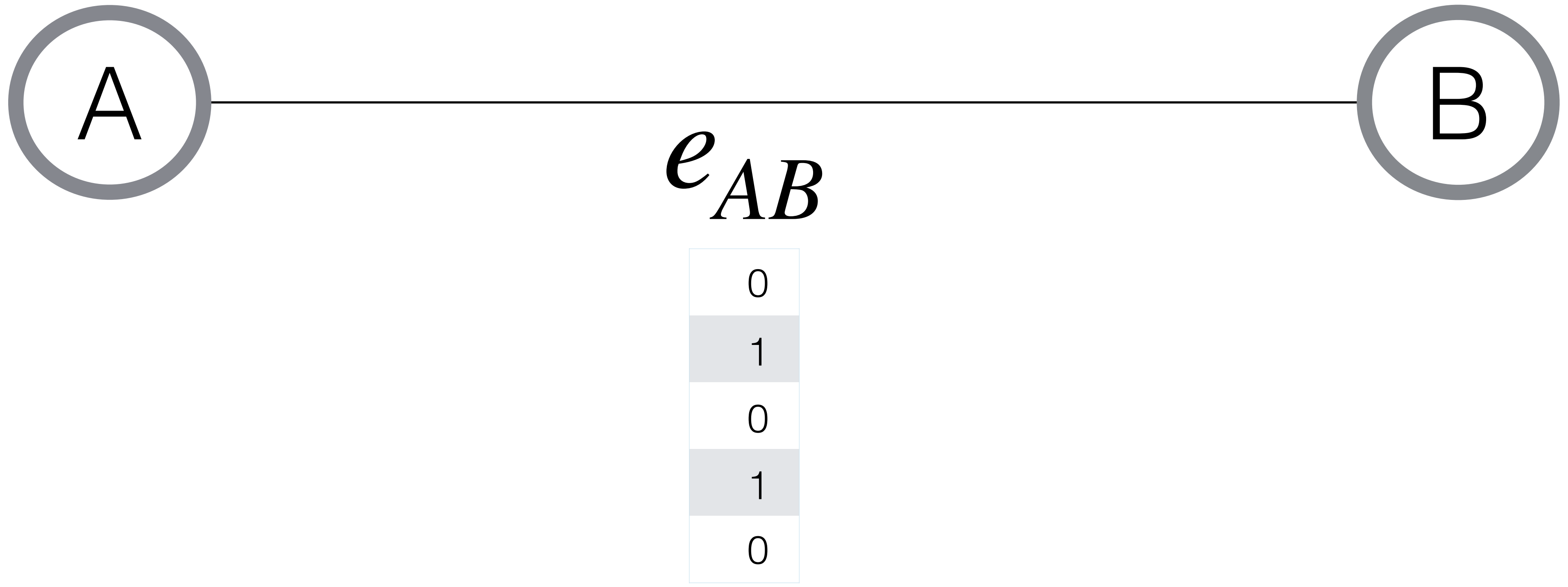


Graphs + features



- x_i is the feature vector for node i

Edge features



Dimensions

- Features can be scalars, vectors, matrices and tensors.

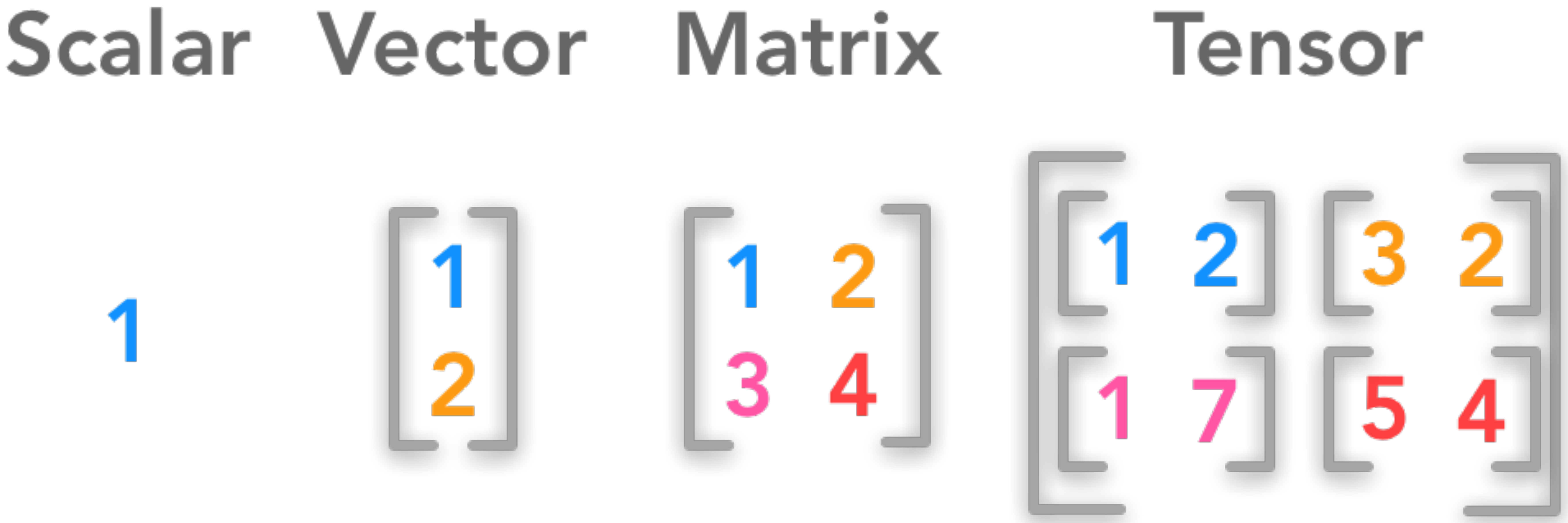
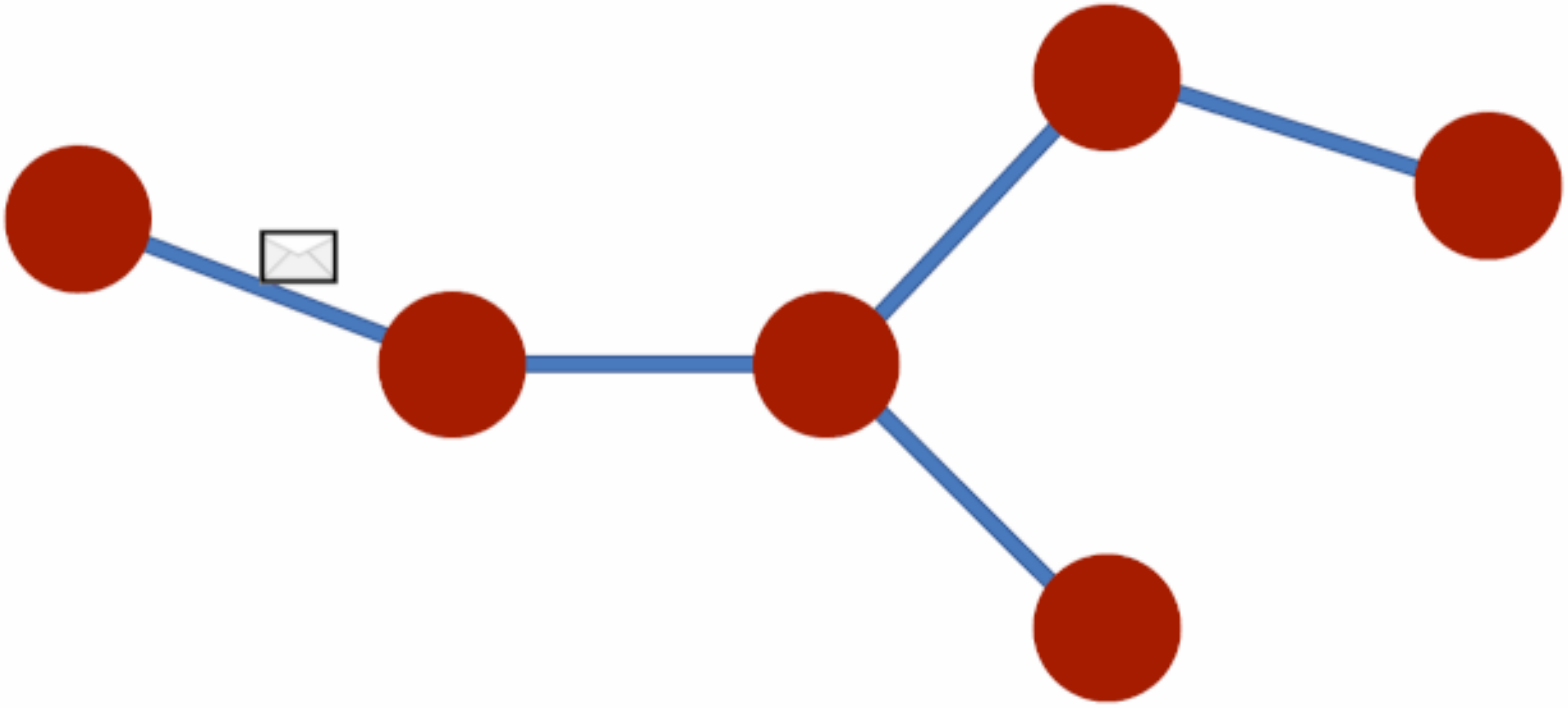


image source: “Essential Math for Data Science” book

State-of-the-art method: message passing



State-of-the-art method: message passing

📦 = activation(aggregation(✉️))

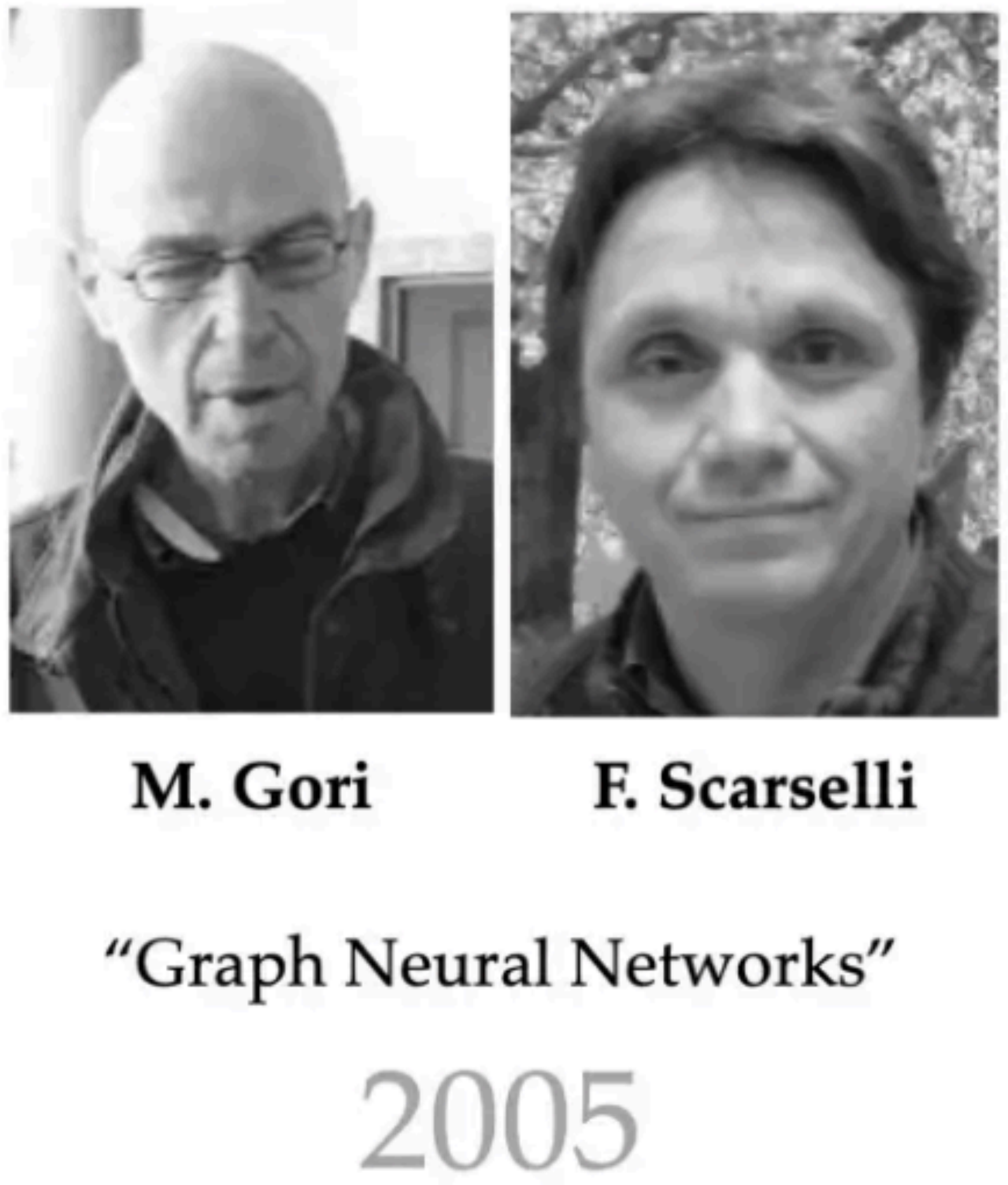
One message per node, i.e., output features

One message per node, i.e., input features

-The activation and aggregation functions depend on the architecture, details will be given in later slides.

History of graph neural networks

- The term “graph neural networks” was introduced in the works of M. Gore and F. Scarselli.



M. Gori, G. Monfardini, and F. Scarselli, A New Model for Learning in Graph Domains, Proceedings of the IEEE International Joint Conference on Neural Networks, 2005.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, The Graph Neural Network Model, IEEE Trans. Neural Networks 20(1):61–80, 2008.

source

History of graph neural networks

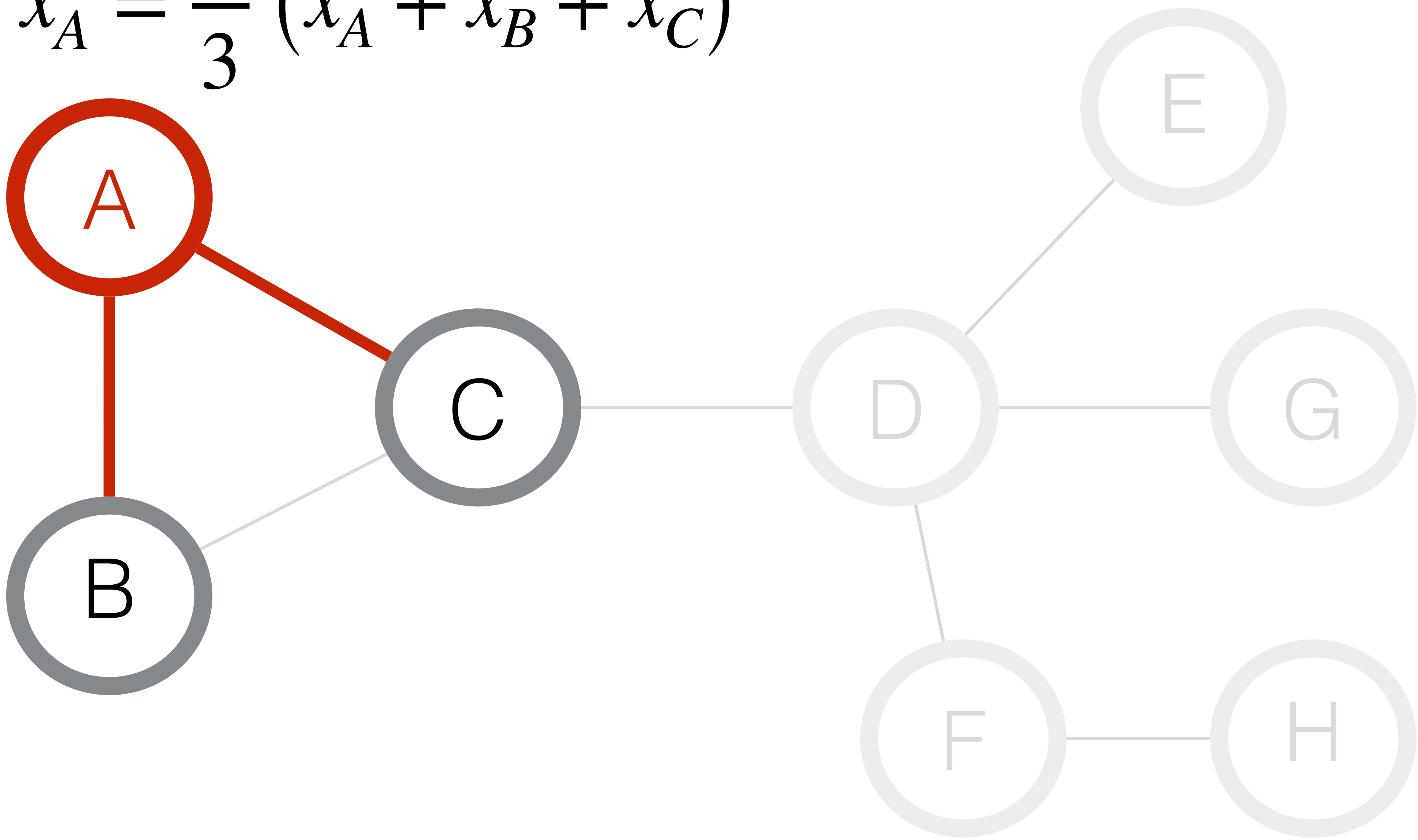
- Early works on graph neural networks appear in the 90s

[source]

Vanilla Graph Convolution Network (GCN): aggregation function

$$x'_A = \frac{1}{3}\left(x_A + x_B + x_C\right)$$

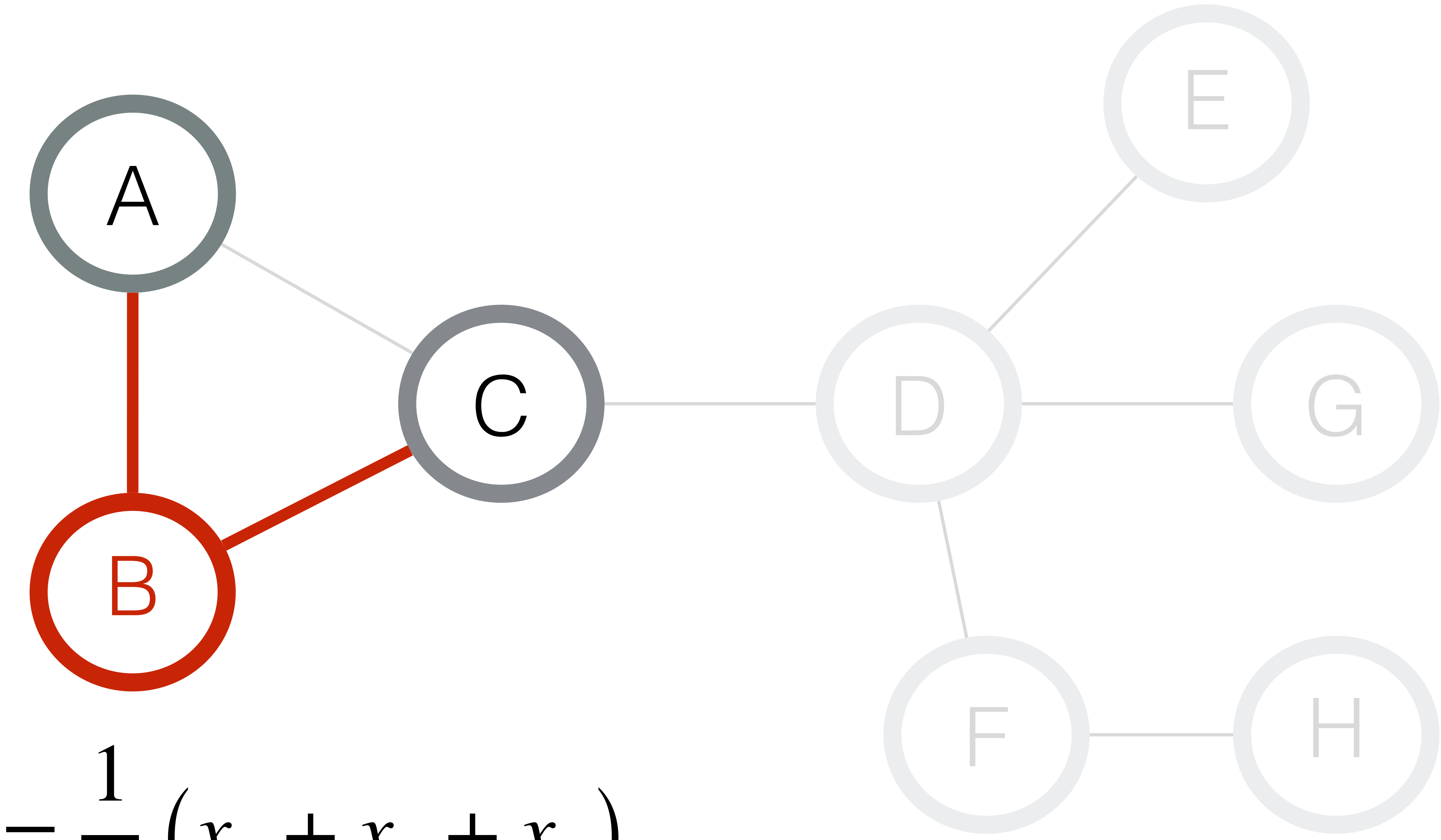
Node A gets messages x_B and x_C and combines it with its own x_A uniformly.



T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): aggregation function

Node B gets messages x_A and x_C and combines it with its own x_B uniformly



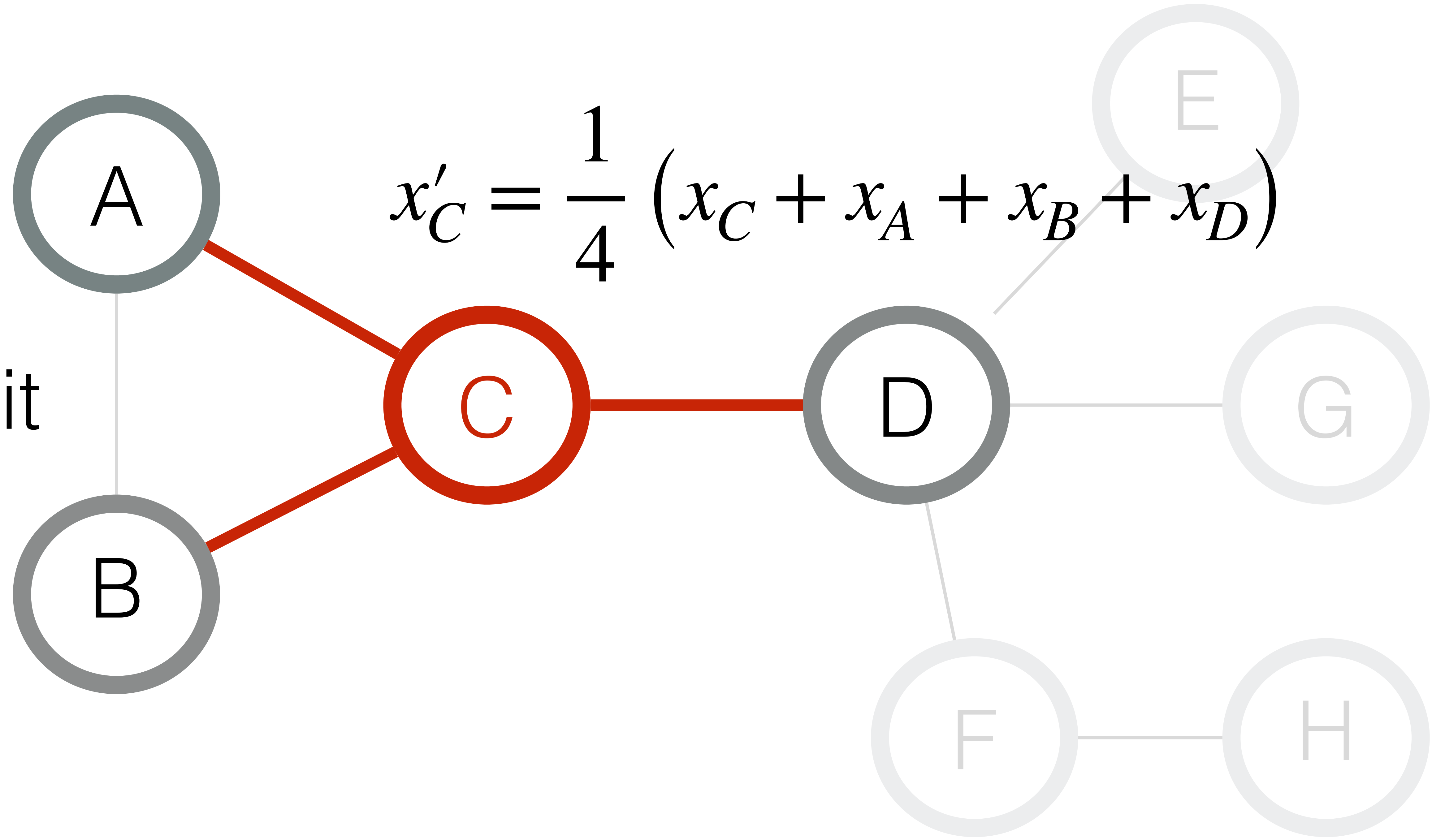
$$x'_B = \frac{1}{3}\left(x_B + x_A + x_C\right)$$

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): aggregation function

Node C gets messages x_A, x_B and x_D, and combines it with its own x_C uniformly

$$x'_C = \frac{1}{4}\left(x_C + x_A + x_B + x_D\right)$$



T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): aggregation function

$$\underbrace{X_i'}_{\text{Convolved data for node } i} := \frac{1}{\underbrace{D_{ii}}_{\text{Degree of node } i}} \sum_{j=1}^{n} \underbrace{A_{ij}}_{\text{Adjacency matrix}} \underbrace{X_j}_{\text{Data For node } j}$$

- A component of A is equal to 1 if two nodes are connected with an edge
- D is a diagonal matrix where each component shows the number of neighbors of a node

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): aggregation function in matrix form

$$X' := D^{-1} A X$$

Convolved data　　Degree matrix　　Adjacency matrix

- A component of A is equal to 1 if two nodes are connected with an edge
- D is a diagonal matrix where each component shows the number of neighbors of a node

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): learning parameters

$$X'W := D^{-1}AXW$$

-Learning matrix W. It's value are decided by minimizing a loss function.

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): activation

$$\sigma(X'W) := \sigma(D^{-1}AXW)$$

-Activation function σ. Examples include $\sigma(y) := \max(y,0)$ or $\sigma(y) :=$ sigmoid$(y) = 1/(1+e^{-y})$ which squeezes values in [0,1].

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Vanilla Graph Convolution Network (GCN): multiple layers

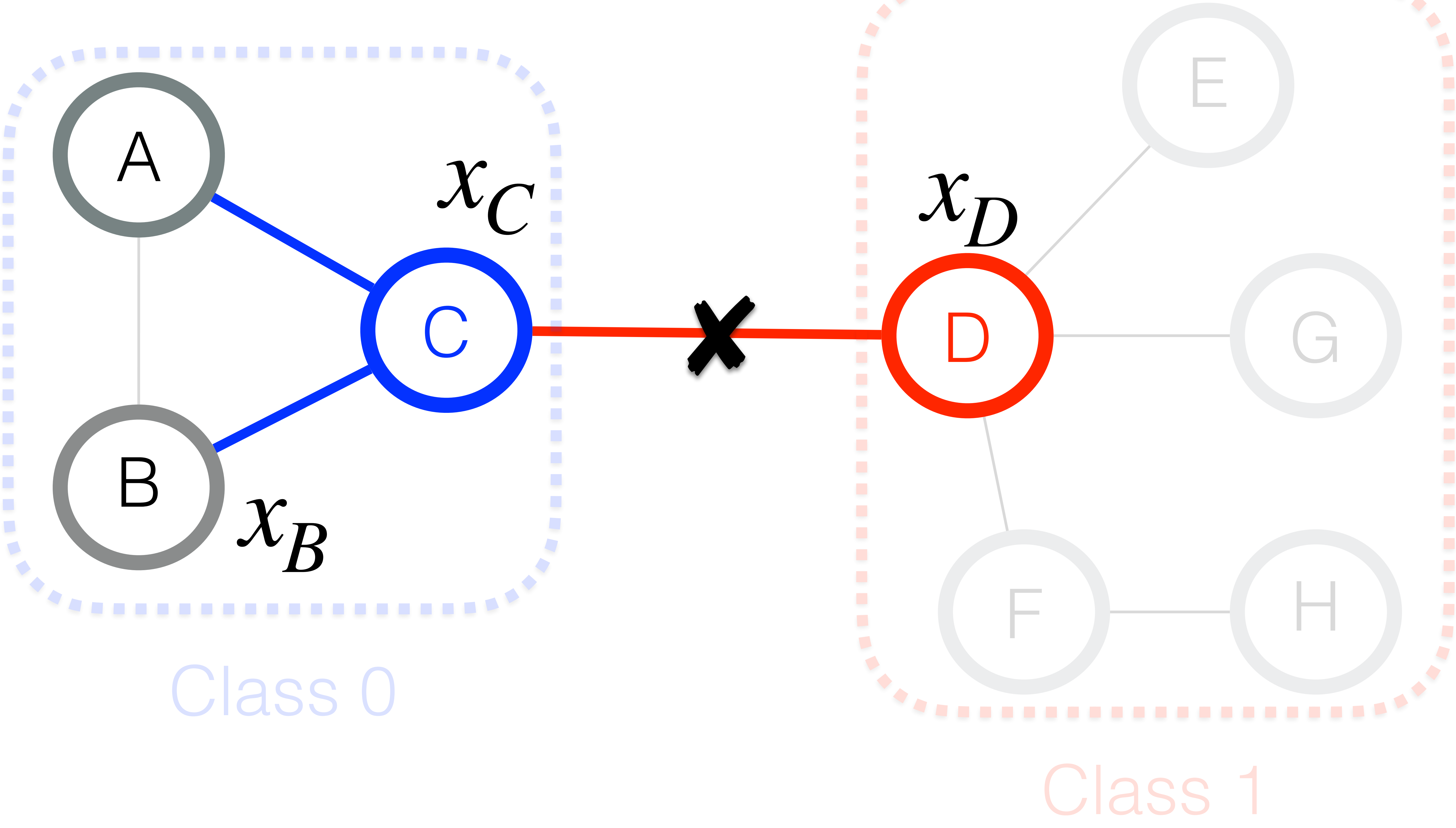
Example: 3-layer GCN

$$X' := \underbrace{\sigma_3(D^{-1}A\, \underbrace{\sigma_2(D^{-1}A\, \underbrace{\sigma_1(D^{-1}AXW_1)}_{\text{layer 1}}\, W_2)}_{\text{layer 2}}\, W_3)}_{\text{layer 3}}$$

T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

An important issue with vanilla Graph Convolution

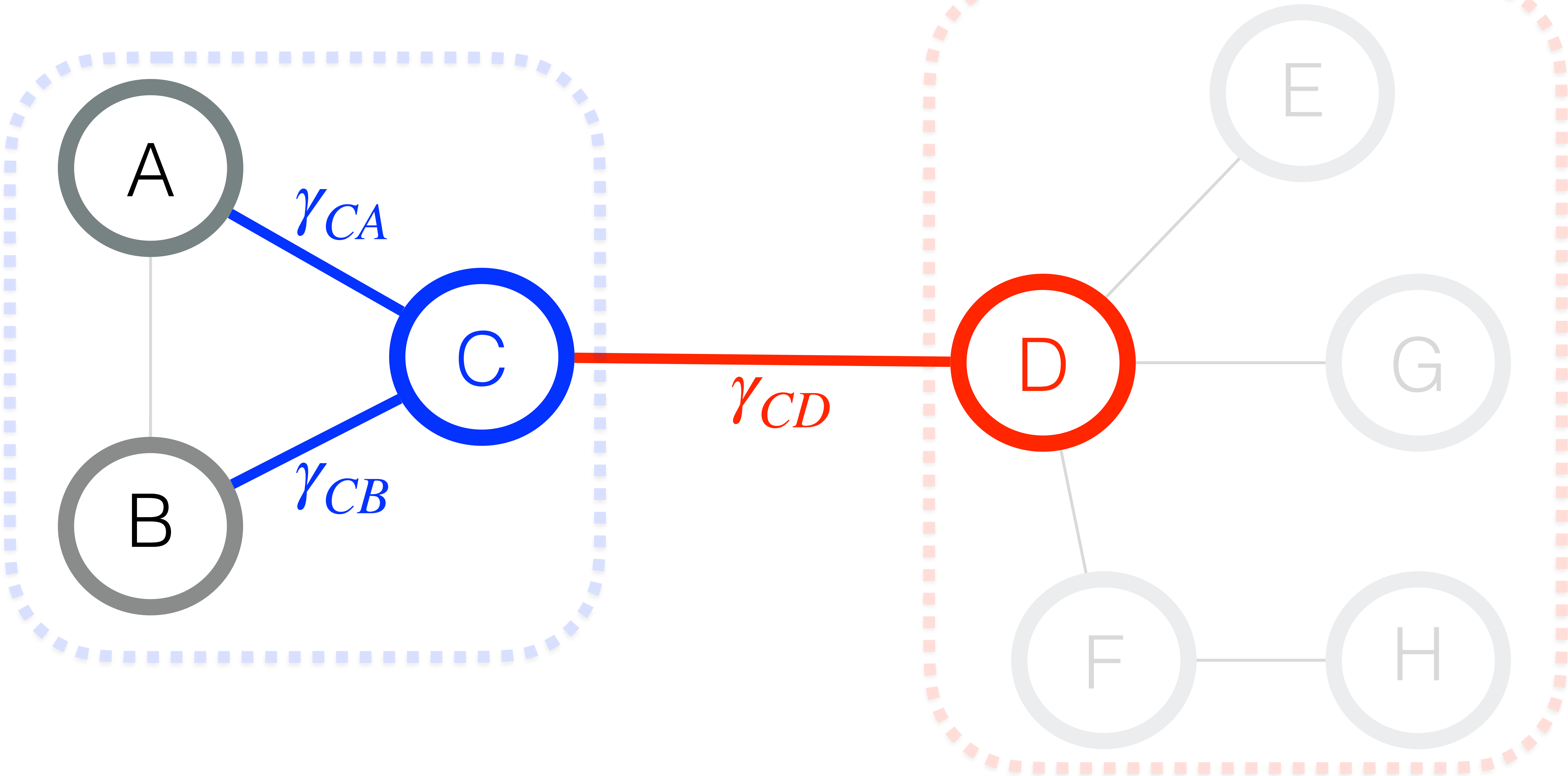
$$x'_C = \frac{1}{4}\left(x_C + x_A + x_B + \cancel{x_D}\right)$$



T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

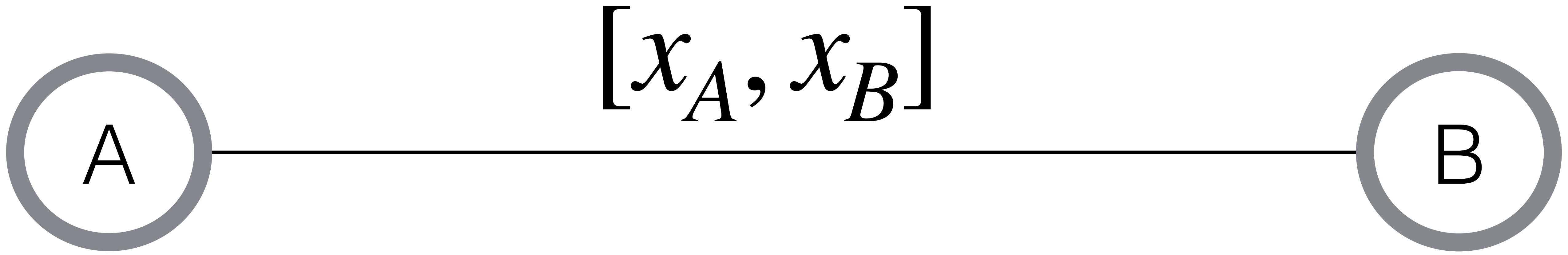
Solution: Graph Attention Network (GAT)

$$x'_C = \gamma_{CC} x_C + \gamma_{CA} x_A + \gamma_{CB} x_B + \gamma_{CD} x_D$$



P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio. Graph Attention Networks, ICLR 2018

Vanilla Attention Mechanism



P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio. Graph Attention Networks, ICLR 2018

Vanilla Attention Mechanism

$$\gamma_{AB} = \psi\left(MLP\left(\left[x_A, x_B\right]\right)\right)$$

ψ is a soft-max function

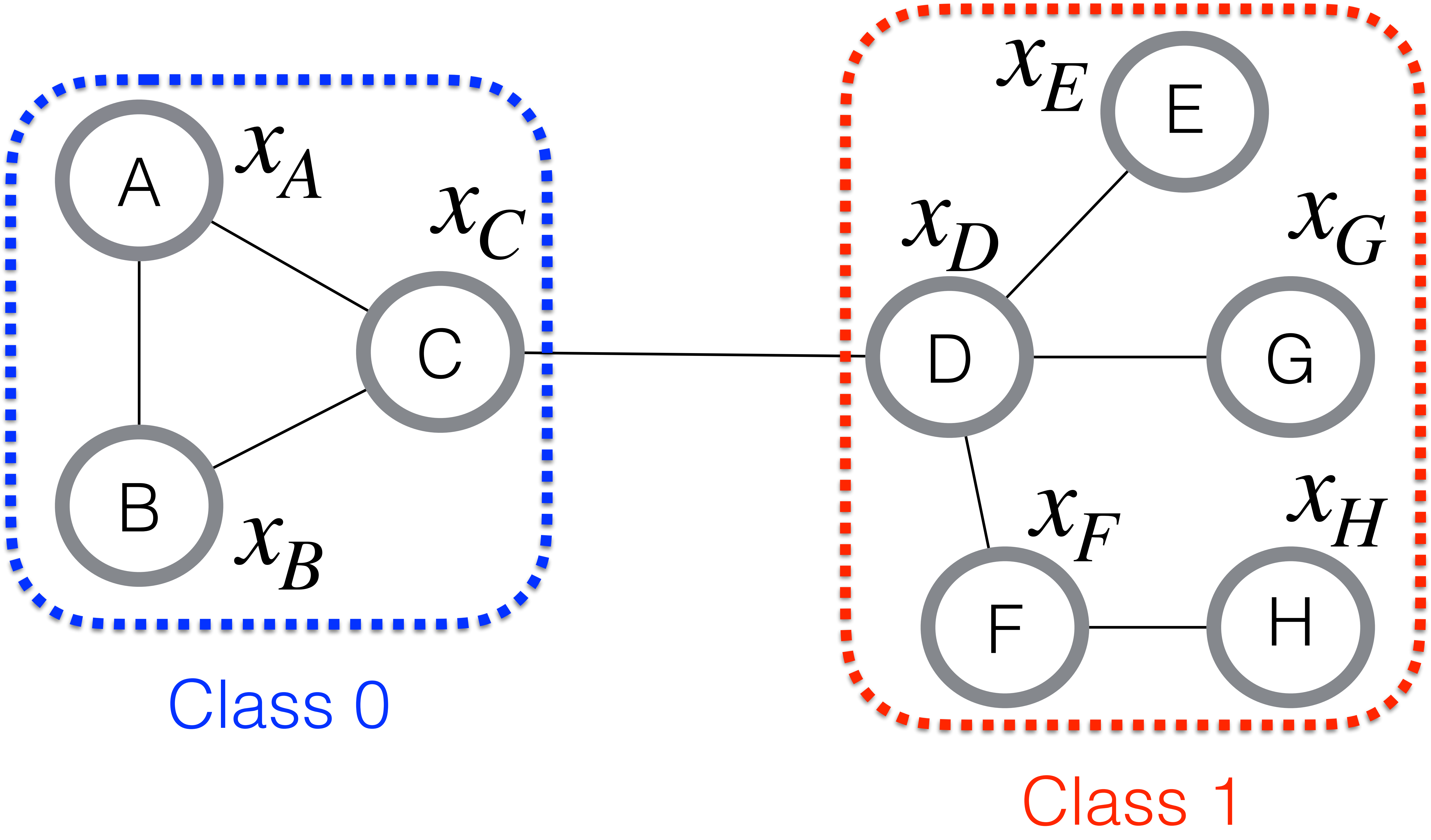
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio. Graph Attention Networks, ICLR 2018

Problems

- Node classification
- Graph classification
- Link prediction
- Regression

Leaderboard: https://ogb.stanford.edu

Node classification



- x_i is the feature vector for node i

Graph classification

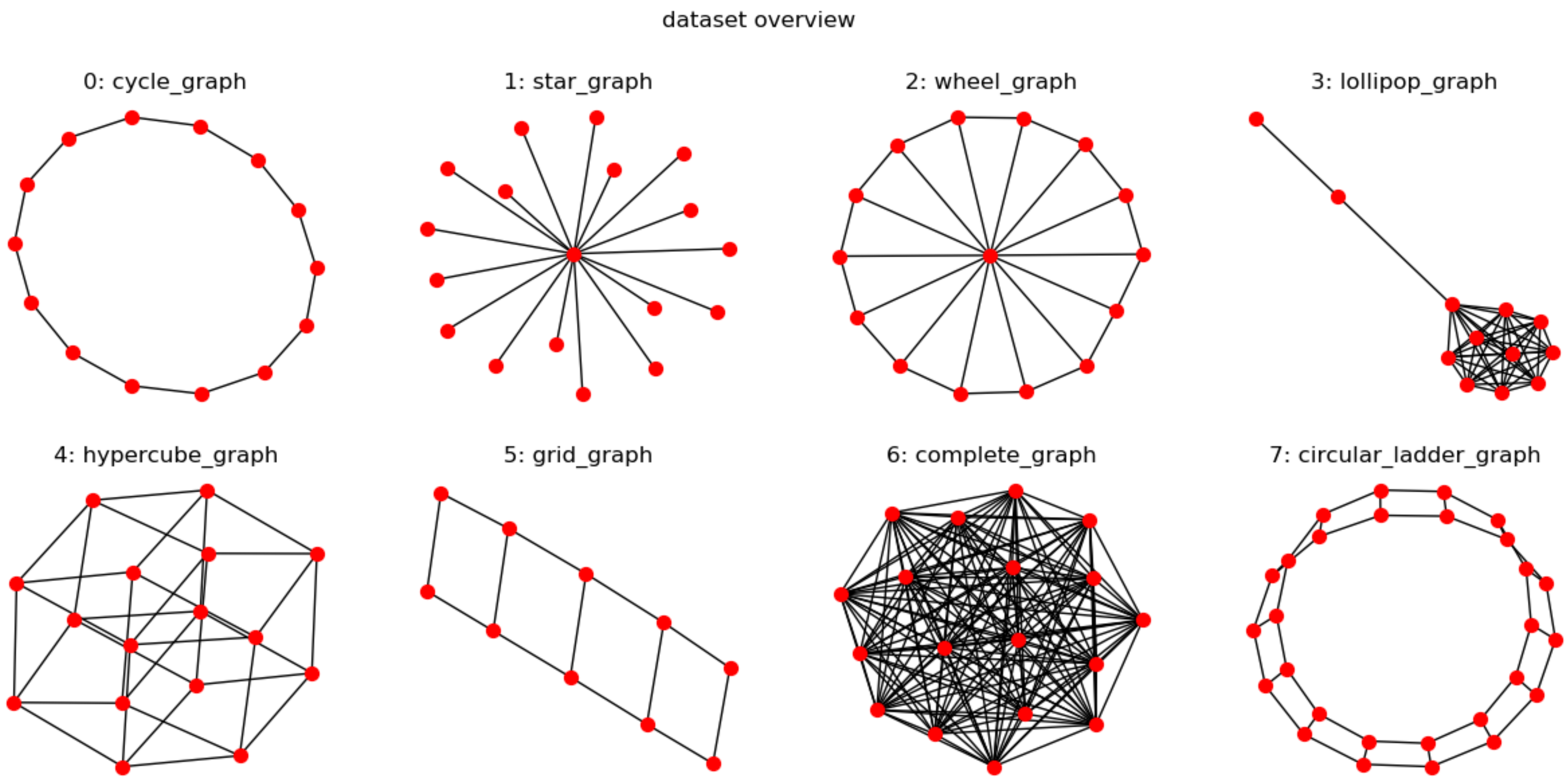


image source: https://www.dgl.ai/blog/2019/01/25/batch.html

Link prediction

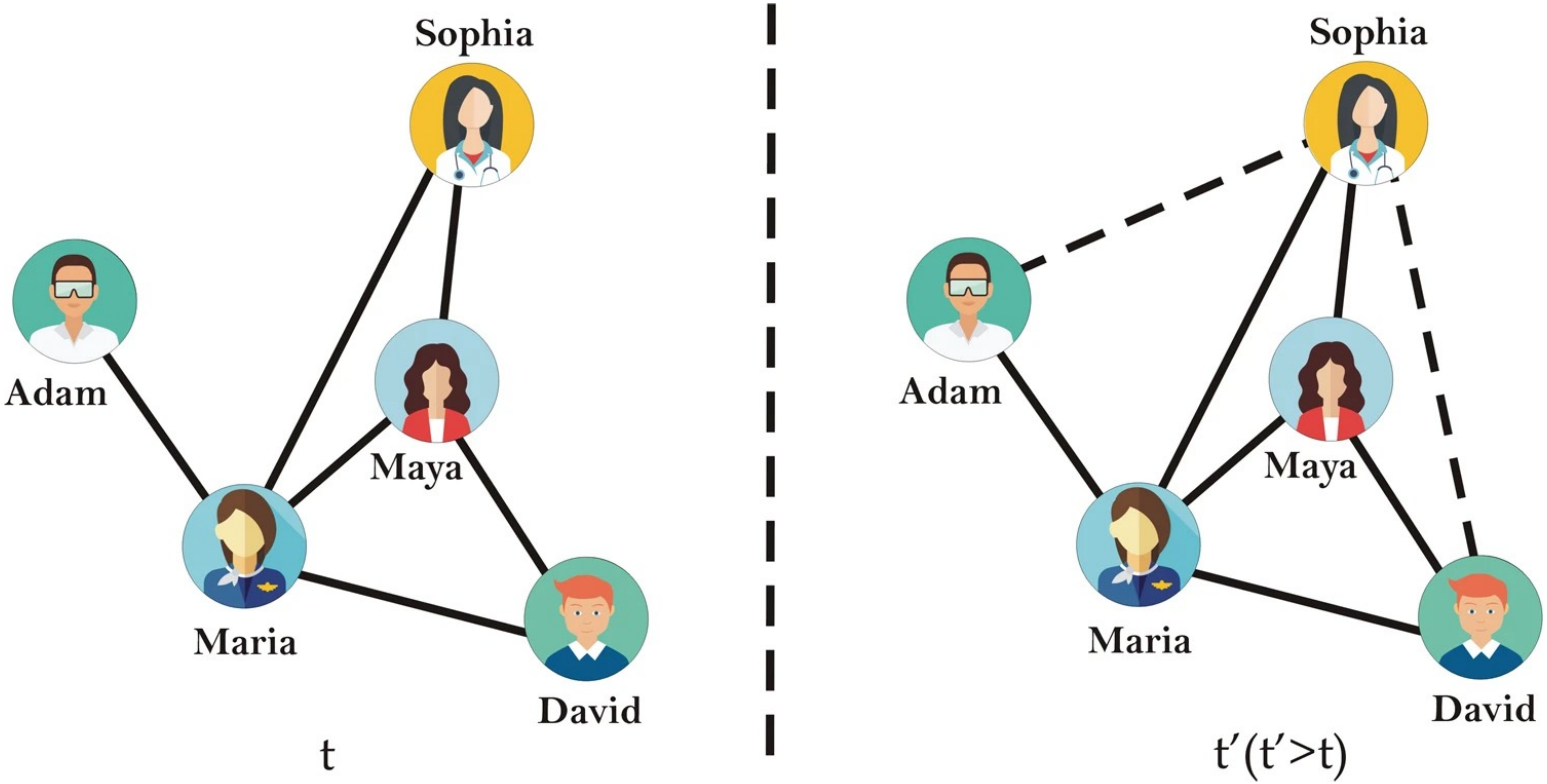
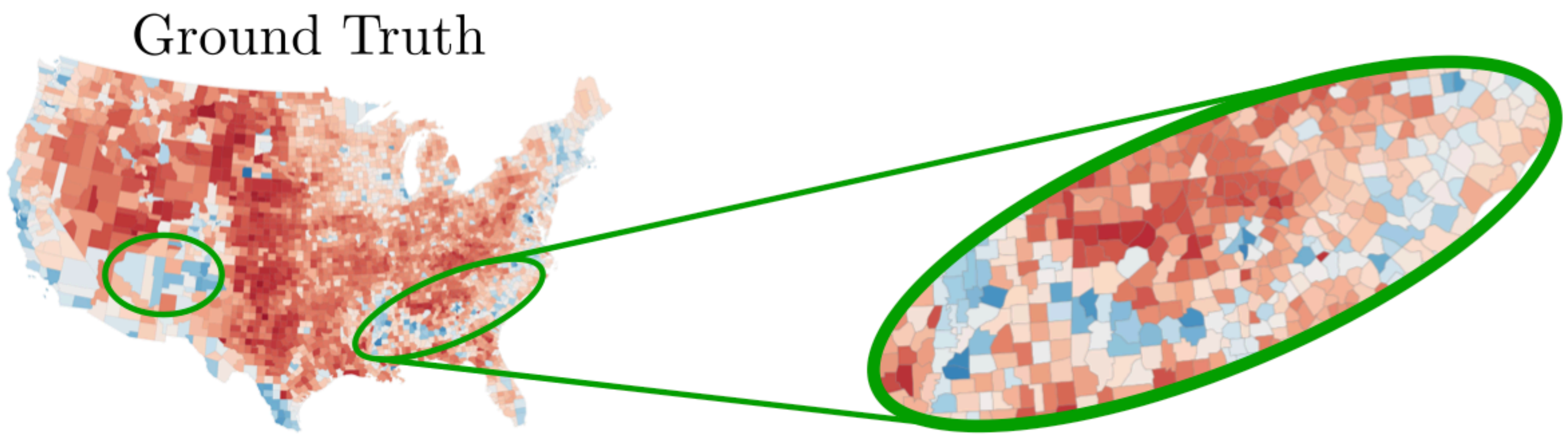


image source: https://www.nature.com/articles/s41598-019-57304-y

Regression

- The ground truth over the nodes has continuous values



- Goal: train the graph neural network to fit the continuous values over the nodes.

image source: https://dl.acm.org/doi/10.1145/3394486.3403101

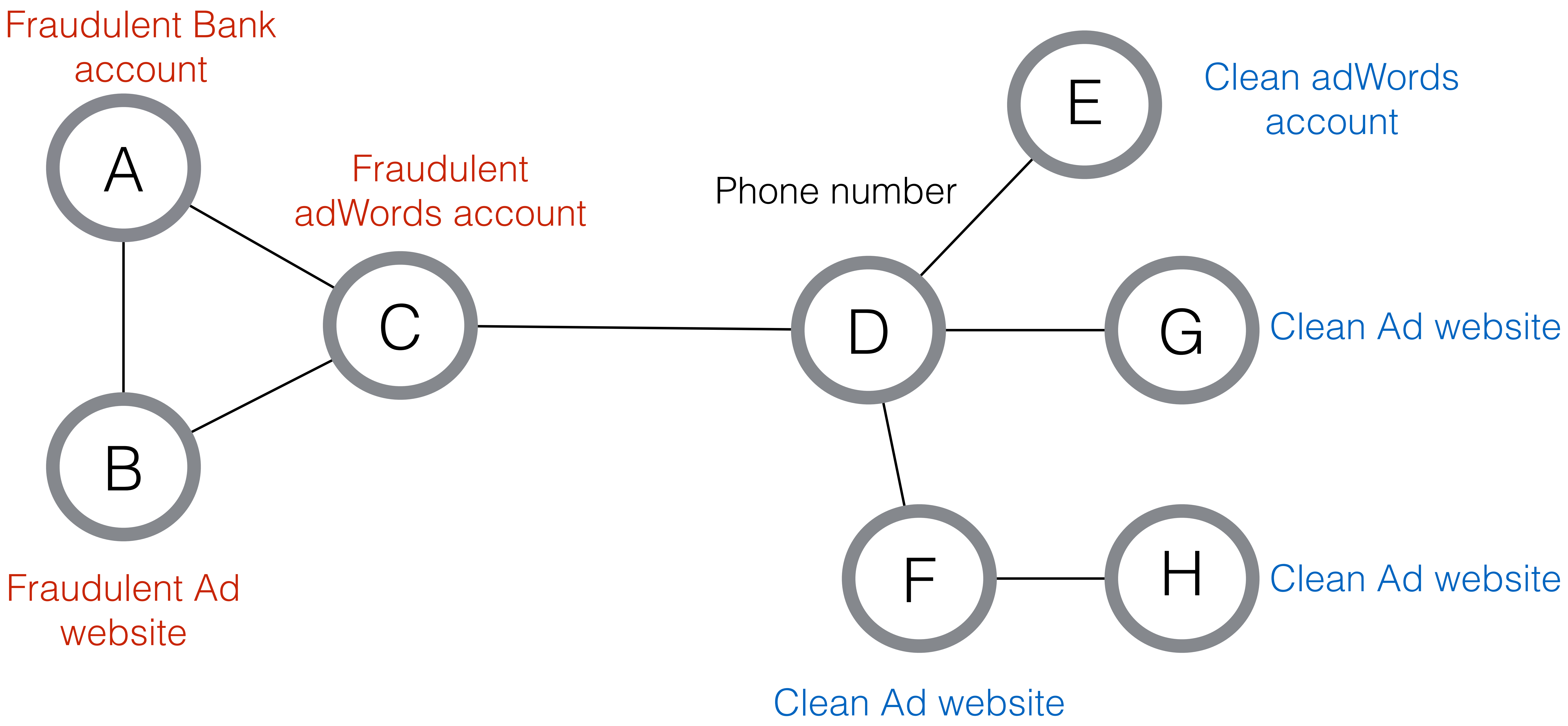
Applications

Disclaimer

-I have put serious effort for a particular application on fraud detection while I was visiting Google (see next slides),

-but, I am not an “applications” researcher as much as a “methods” researcher.

Example application: fraud detection in Google



More examples of fraud detection.

-Online payment fraud (billion dollar business). Basically someone stole your card info and uses it online.

-Cloaking. Fool web crawlers to think that an Ad is legitimate.

-Click fraud. Bots clicking pay-per-click ads to increase revenue.

Recommender systems

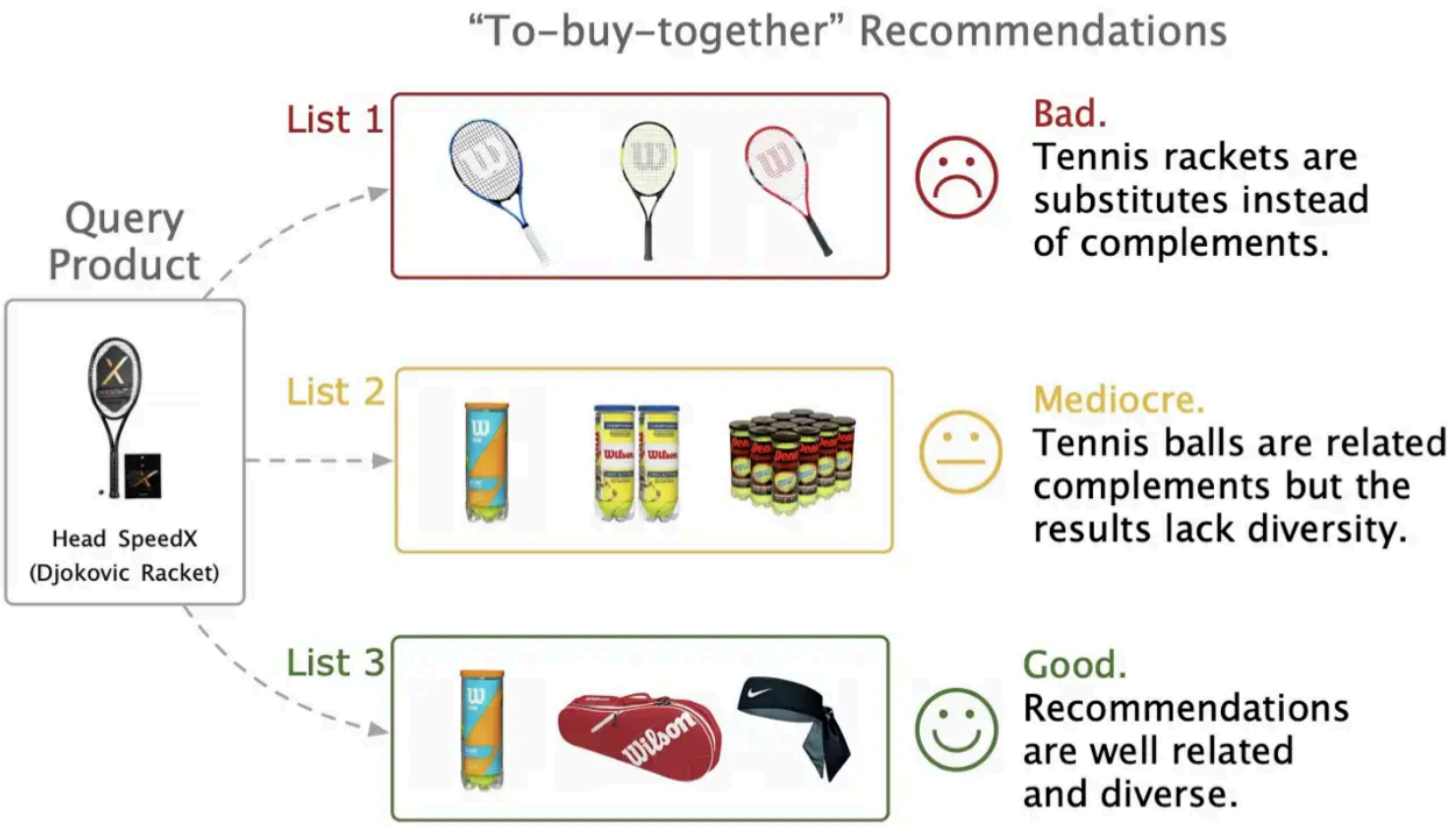


image source: Amazon

More examples of recommendation systems.

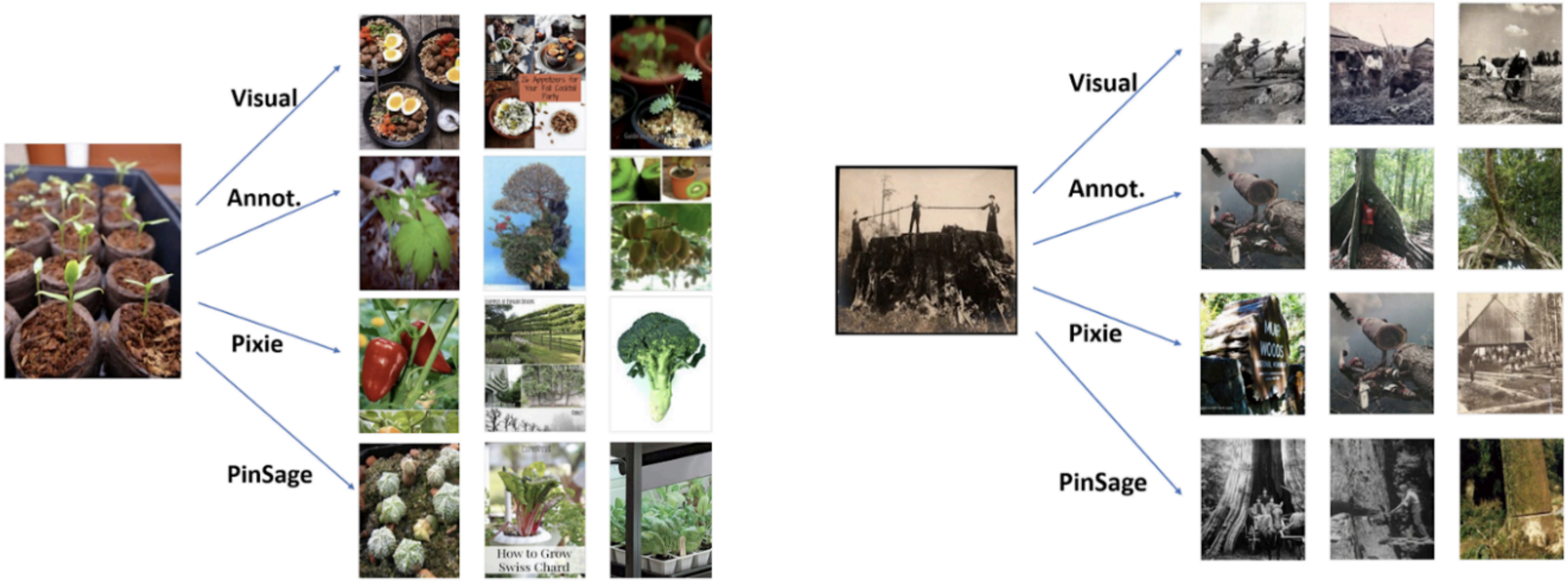


Figure 3: Examples of pins recommended by different algorithms. The image to the left is the query pin. Recommended items to the right are computed using Visual embeddings, Annotation embeddings, Pixie (purely graph-based method), and PinSage.

image source: Medium

Traffic forecasting

image source: DeepMind

Material discovery

RESEARCH

Millions of new materials discovered with deep learning

29 NOVEMBER 2023

Amil Merchant and Ekin Dogus Cubuk

Share

image source: DeepMind

Weather forecasting

GraphCast: AI model for faster and more accurate global weather forecasting

14 NOVEMBER 2023

Remi Lam on behalf of the GraphCast team

Share

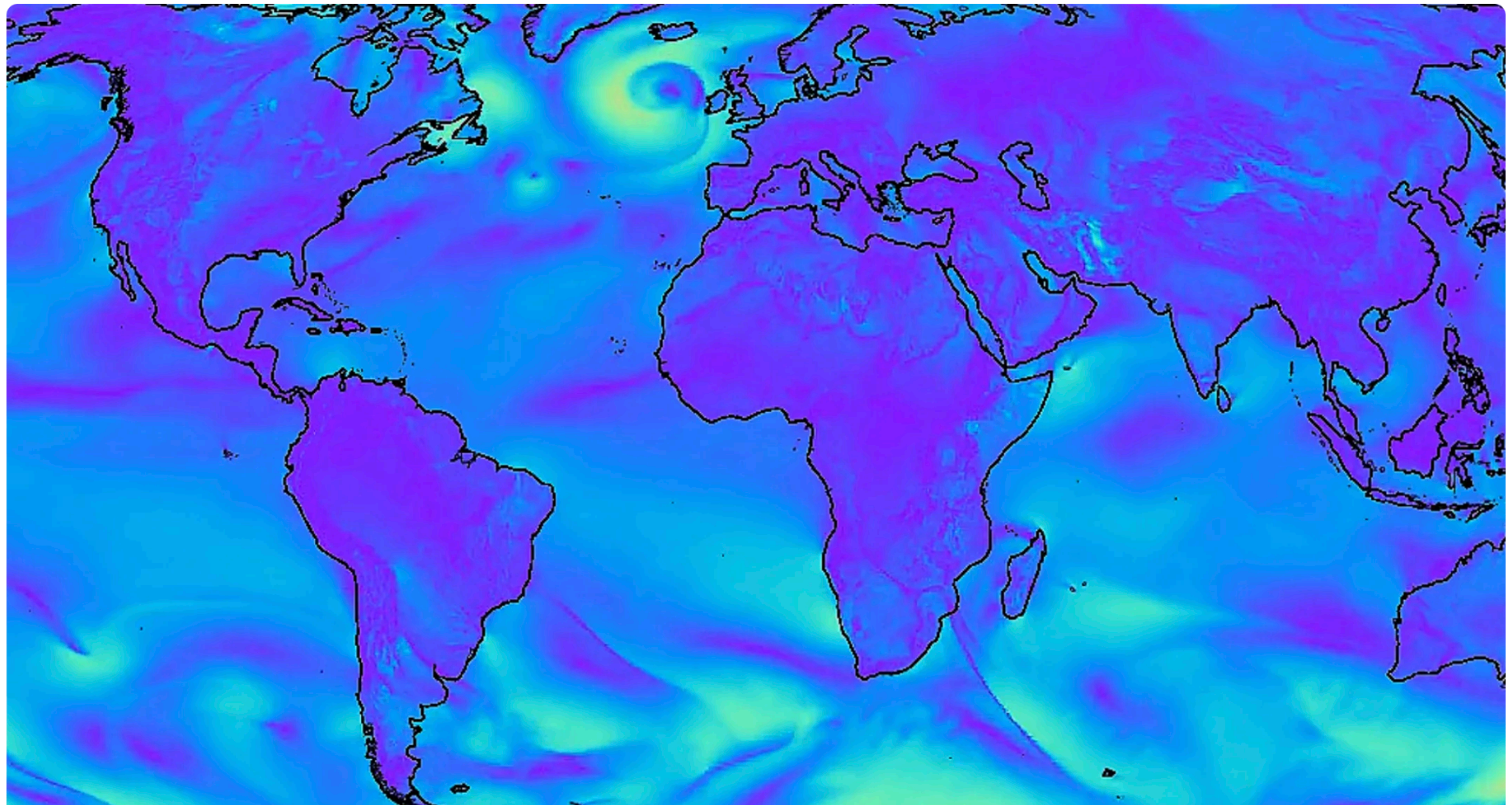
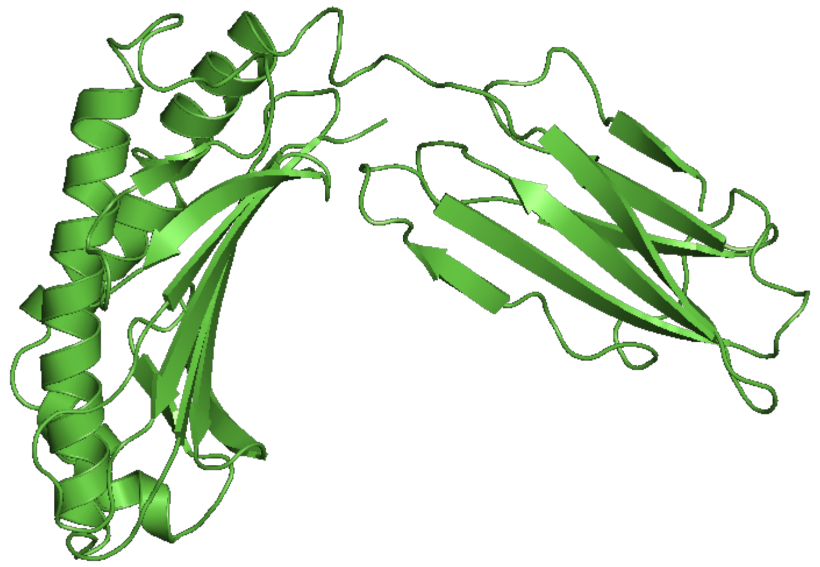


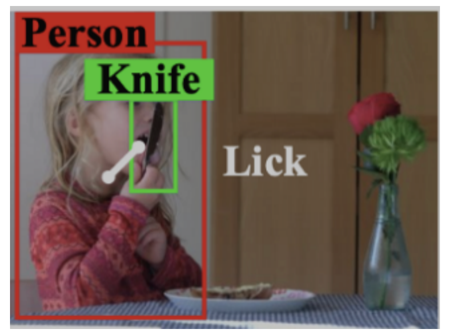
image source: DeepMind

Popular applications (that I don’t have expertise on)

- Social influence prediction
- Medical diagnosis
- Drug discovery (very popular recently)



- Modelling spread of COVID
- Traffic forecasting
- Human-object inte on



- Text classification
- Chip design

source: Medium