Tutorial: Scalable Graph Neural Networks with Deep Graph Library

George Karypis, Zheng Zhang, Minjie Wang, Quan Gan, and Da Zheng

Agenda

- 9:00-10:00 Overview of graph neural networks
- 10:00-10:30 Overview of deep graph library (DGL)
- 10:30-11:00 Virtual coffee break
- 11:00-12:00 (Hands-on) GNN models for basic graph tasks
- 12:00-13:00 Virtual lunch break
- 13:00-14:30 (Hands-on) GNN training on large graphs
- 14:30-16:00 (Hands-on) Distributed GNN training

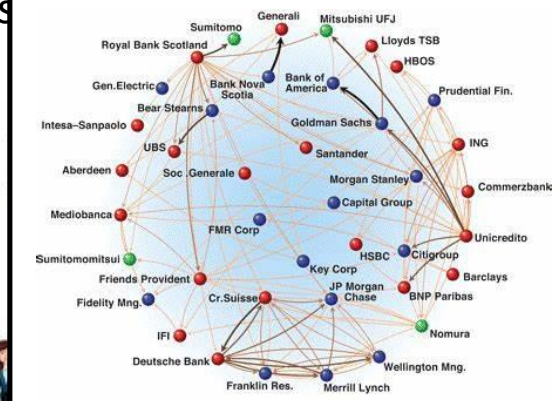
Overview of Graph Neural Networks

George Karypis

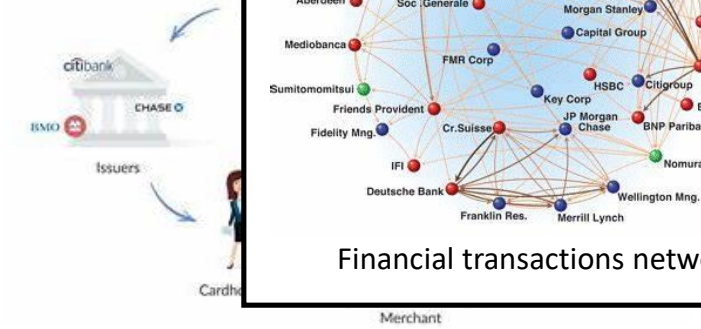
Senior Principal Scientist, AWS Deep Learning Science,
Palo Alto, CA, USA, gkarypis@amazon.com
Professor, Department of Computer Science & Engineering, University of Minnesota,
Minneapolis, MN, USA, karypis@umn.edu, (on leave)

Fraud & Abuse

Detect malicious accounts, fraudulent financial transactions, fake reviews, fraudulent ins

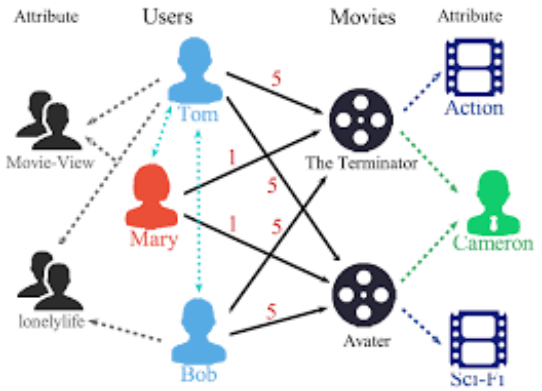


Financial transactions network

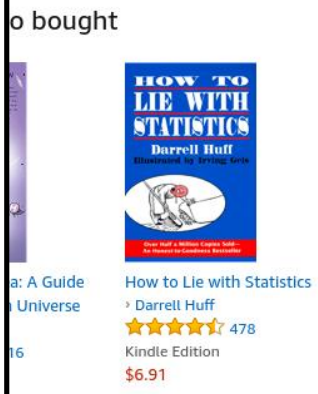


Recommendations

Products, media, articles, experiences, jobs, courses, spouses, …



Collaborative knowledge graph



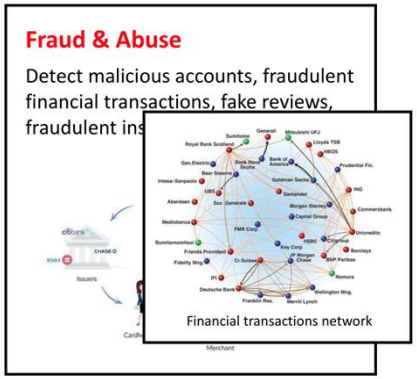
Marketing

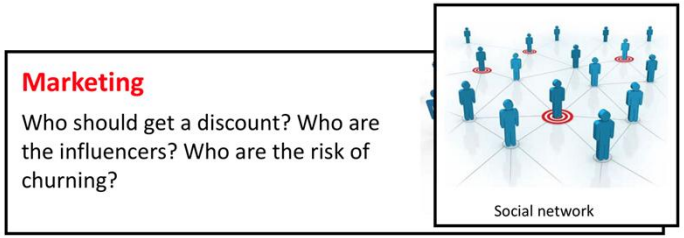
Who should get a discount? Who are the influencers? Who are the risk of churning?

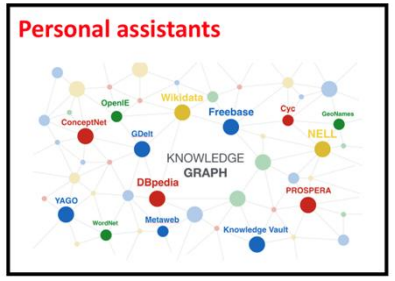
Social network

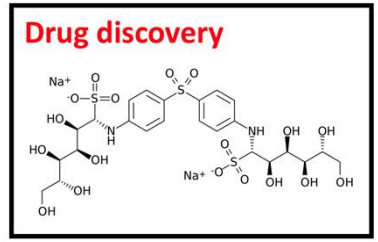
Tasks in graph learning

- Node classification
  - Detect malicious accounts
  - Target right customers
- Link prediction
  - Recommendations
  - Predict missing relations in a knowledge graph
- Graph classification
  - Predict the property of a chemical compound



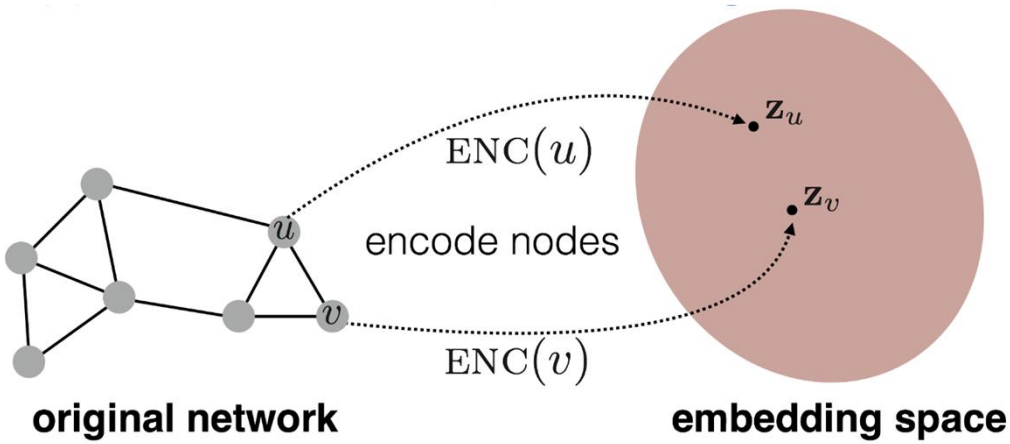






Graph learning and node embeddings

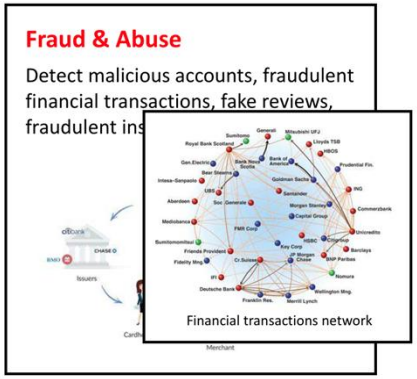
- Embed nodes to a low-dimension space so that these embeddings capture the essential task-specific information and use them to train off-the-self classifiers.
  - For example, node similarities in the embedding space approximates the similarities in the original graph.

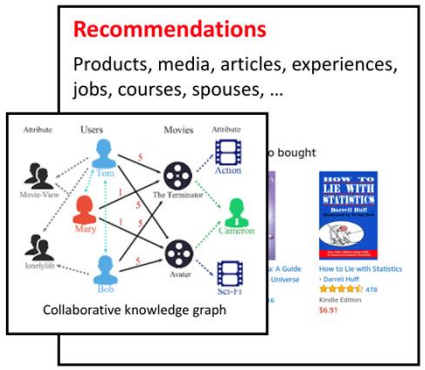


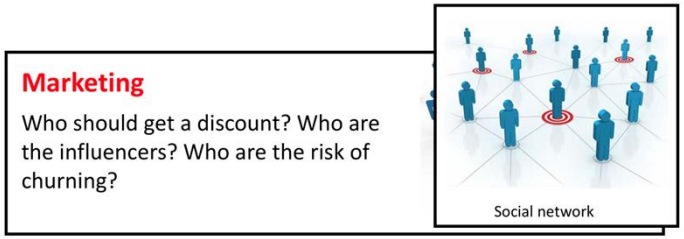
Representation Learning on Networks, snap.stanford.edu/proj/embeddings-www, WWW 2018

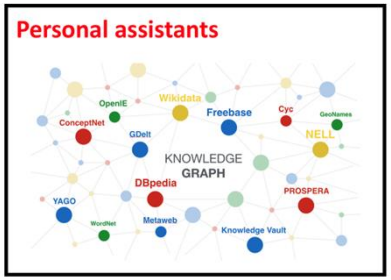
Traditional node embedding approaches

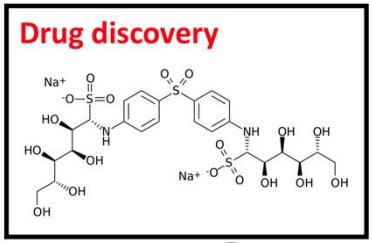
- Generate embeddings by manual feature engineering
  - Requires domain expertise, involves considerable manual fine-tuning, time consuming, does not scale, …
- Automatically generate embeddings using unsupervised dimensionality reduction approaches
  - Singular value decomposition, tensor decomposition, co-factorization, deep walks, etc.
  - Cannot effectively combine rich attributes with network structure.
  - Employ mostly (multi-)liner models.
  - Do not allow for end-to-end learning.







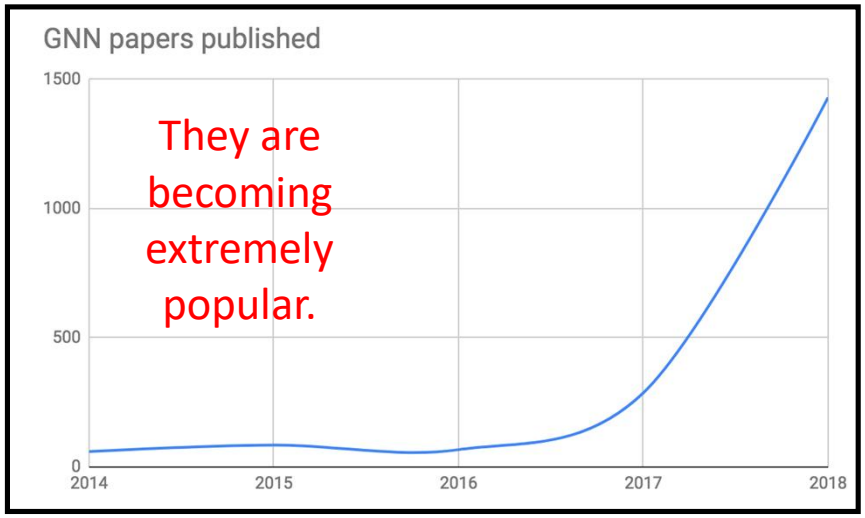




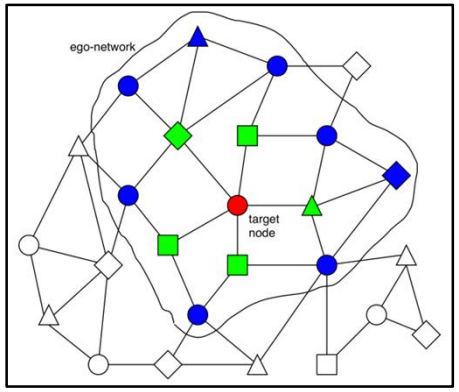
Can we do better?

Graph Neural Network (GNN)

A family of (deep) neural networks that learn node, edge, and graph embeddings.



How do GNNs work?



An ego-network around each node is used to learn an embedding that captures task-specific information.

The embeddings use both the structure of the graph and the features of the nodes and edges.

The embeddings are learned in an end-to-end fashion; thus, the predictions are a function of the target node's ego-network.

A general graph neural network formalism

Graph neural networks are based on ***message-passing***

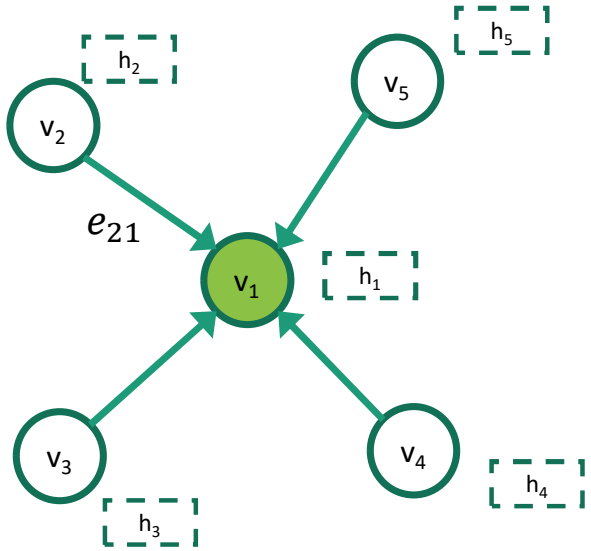
Reduce/Aggregate

Message

$$m_v^{(l)} = \sum_{w \in N(v)} M^{(l)}(h_v^{(l-1)}, h_w^{(l-1)}, e_{vw})$$

$$h_v^{(l)} = U^{(l)}(h_v^{(l-1)}, m_v^{(l)})$$

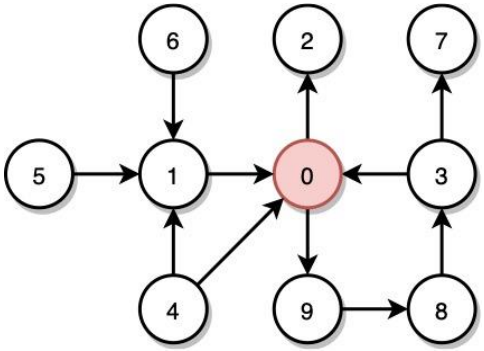
Update

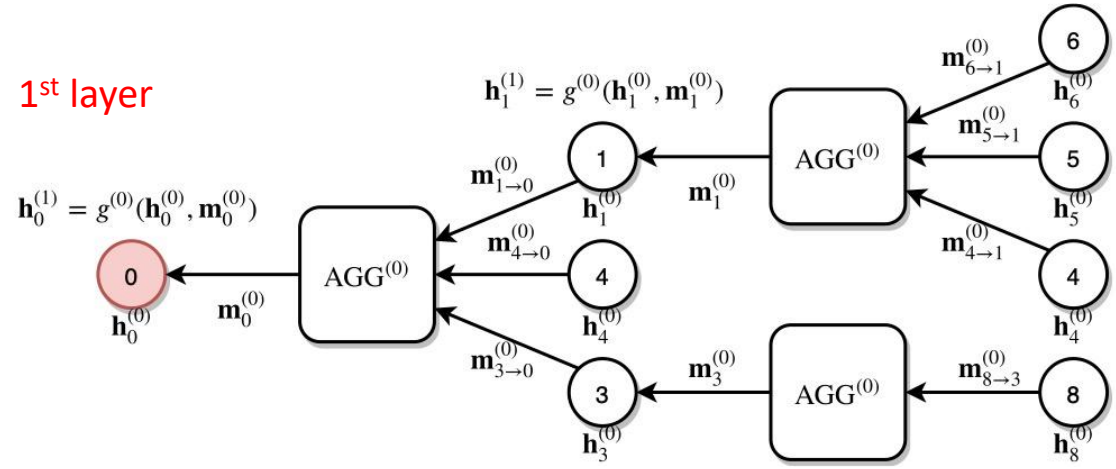


Neural Message Passing for Quantum Chemistry

A multi-layer GNN

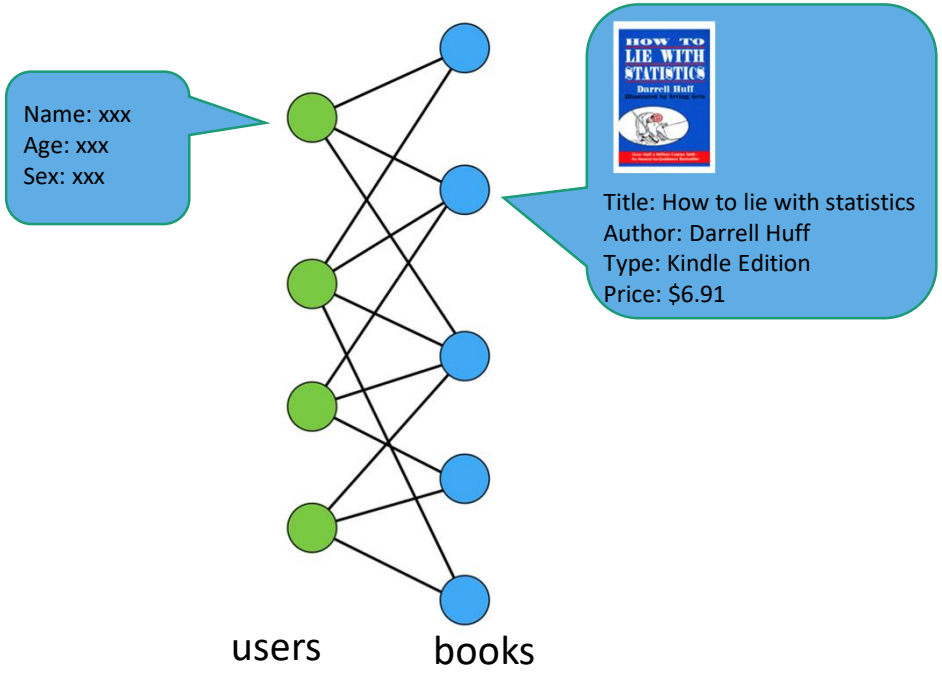
Multiple GNN layers can be stacked together.





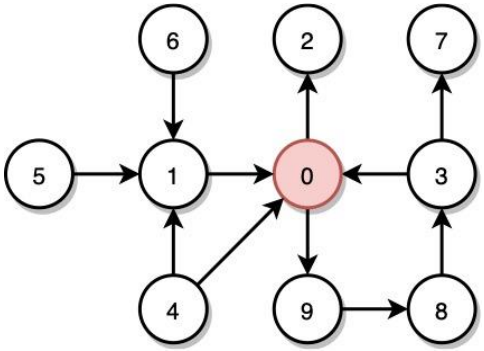
Why are graph neural networks better?

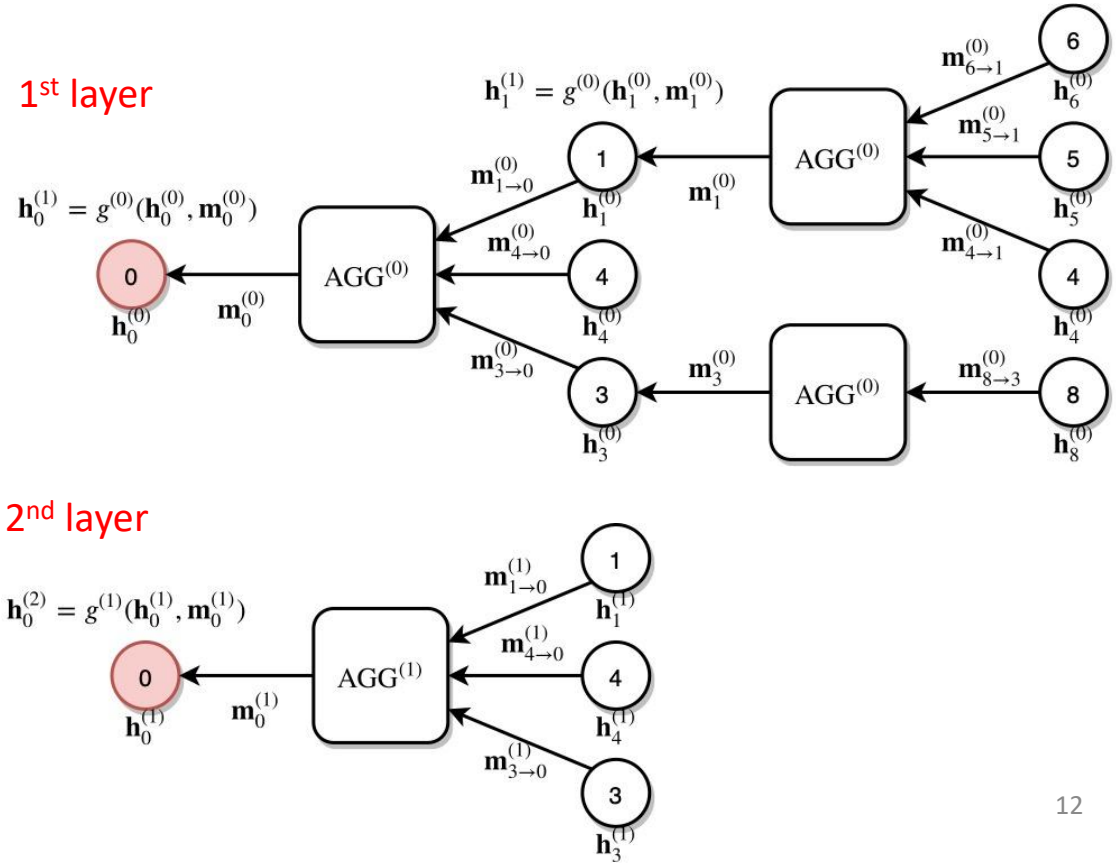
GNNs compute node embeddings using both the structure of the graph and the features of the nodes and edges.



Why are graph neural networks better?

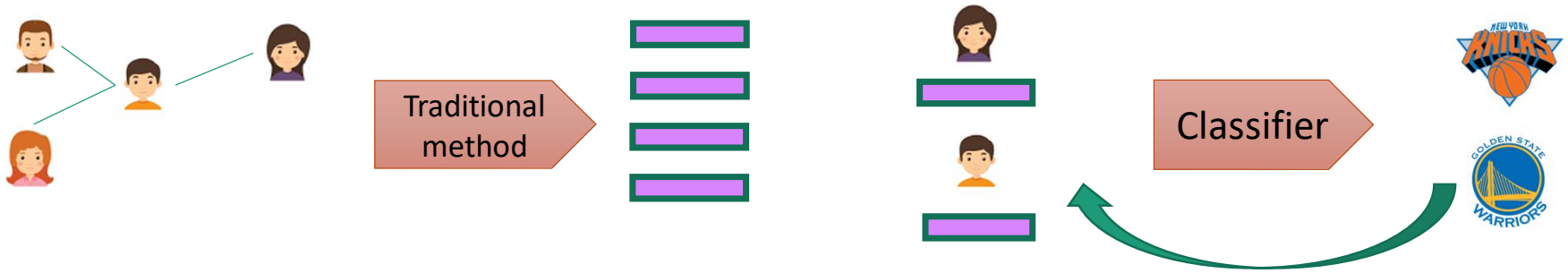
GNNs can *integrate* topologically distant information in a non-linear fashion.

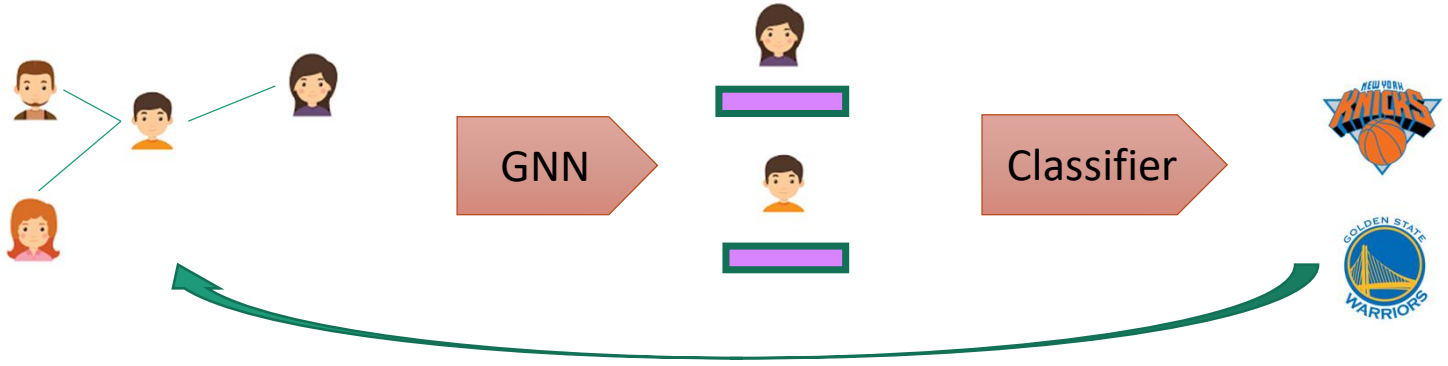




Why are graph neural networks better?

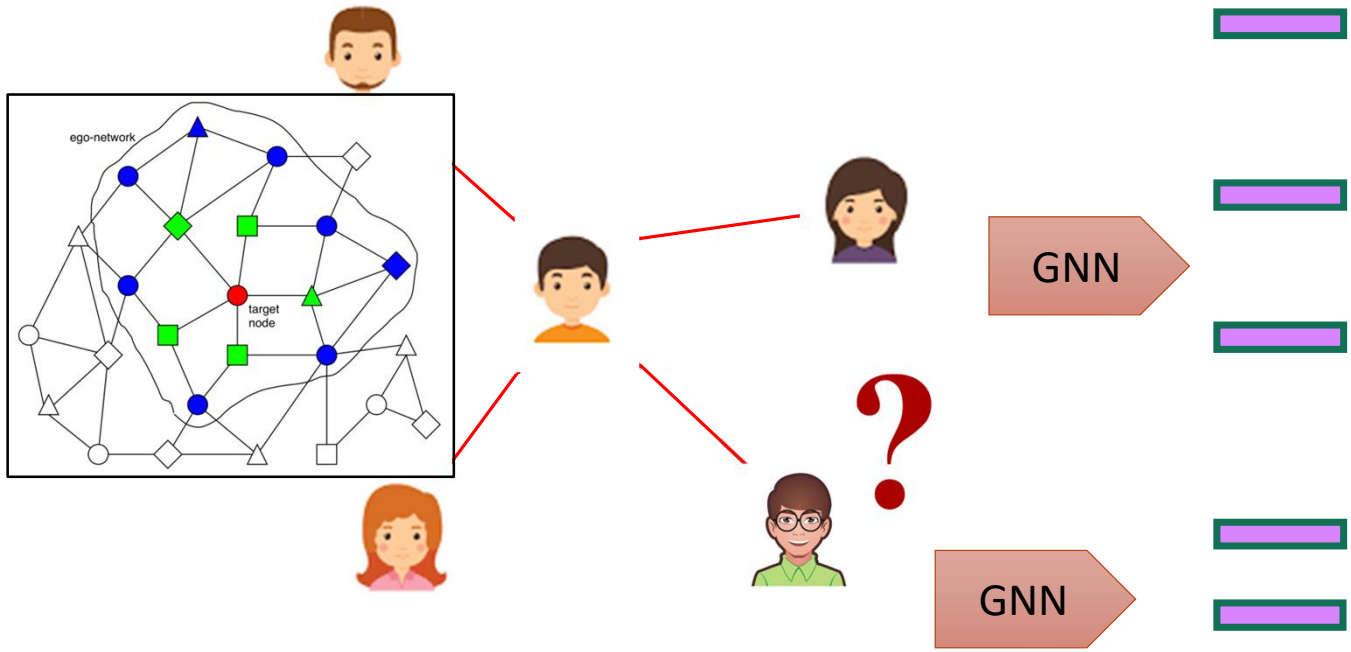
GNNs and the downstream classification/regression models can be trained in an end-to-end fashion.





Why are graph neural networks better?

GNNs are naturally inductive because they learn the same neural networks on all the nodes and edges.

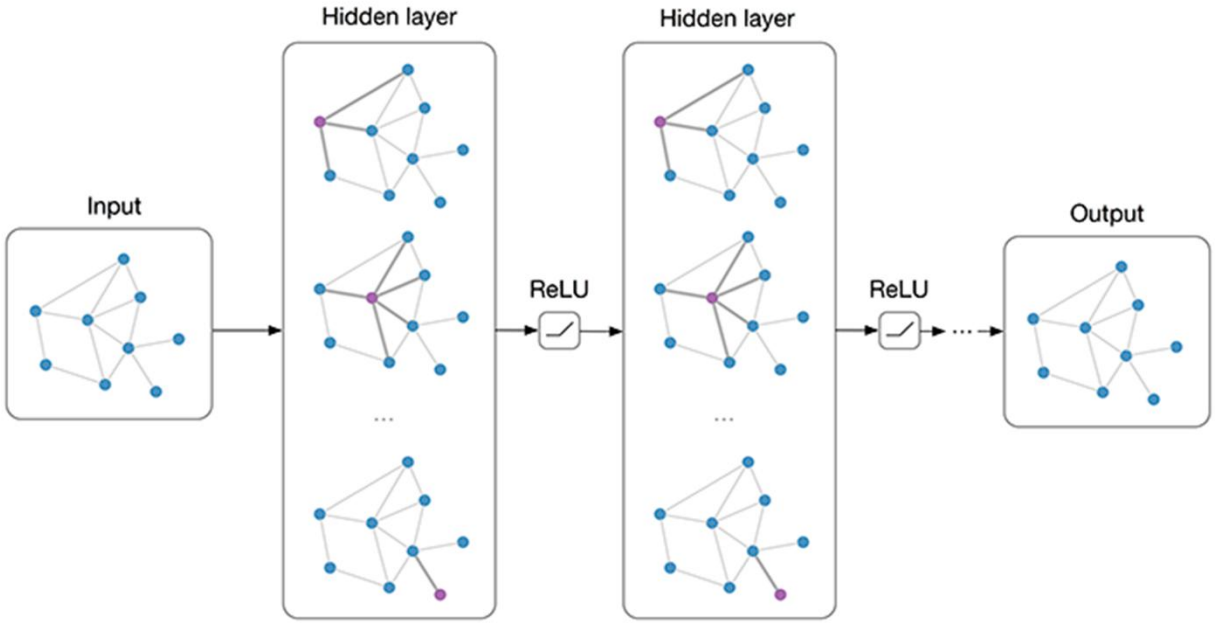


Example 1: Graph convolution network (GCN)

$$M_{vw}^{(l)} = \frac{h_w^{(l-1)}}{d_v+1}$$

$$m_v^{(l)} = \sum_{w \in N(v) \cup \{v\}} M_{vw}^{(l)}$$

$$h_v^{(l)} = \phi(m_v^{(l)} W^{(l)})$$



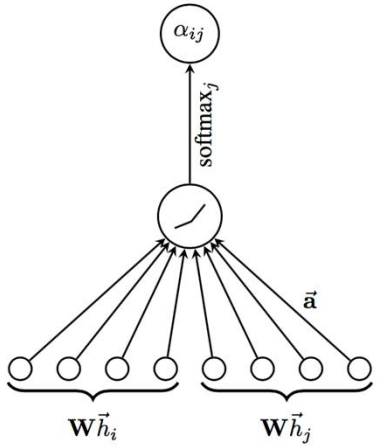
Example 2: Graph attention networks (GAT)

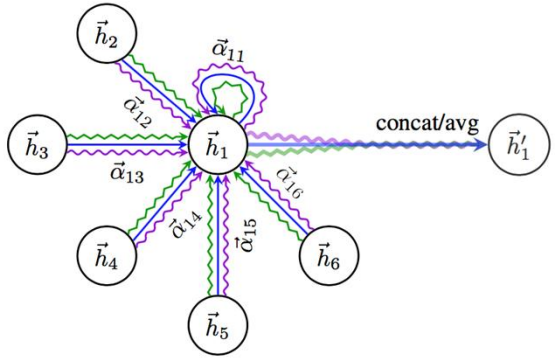
GAT provides weighted sum over the neighborhood—Enables to selectively integrate information.

$$M_{vw}^{(l)} = \alpha_{vw} h_w^{(l-1)}$$

$$m_v^{(l)} = \sum_{w \in N(v) \cup \{v\}} M_{vw}^{(l)}$$

$$h_v^{(l)} = \phi(m_v^{(l)} W^{(l)})$$

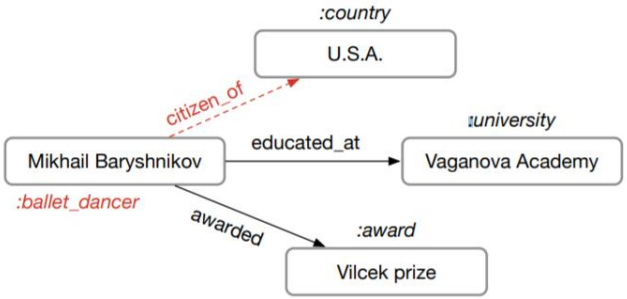




$$\alpha_{vw} = \frac{\exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_v || W\vec{h}_w]))}{\sum_{k \in N_v} \exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_v || W\vec{h}_k]))}$$

Example 3: Relational graph convolution networks (RGCN)

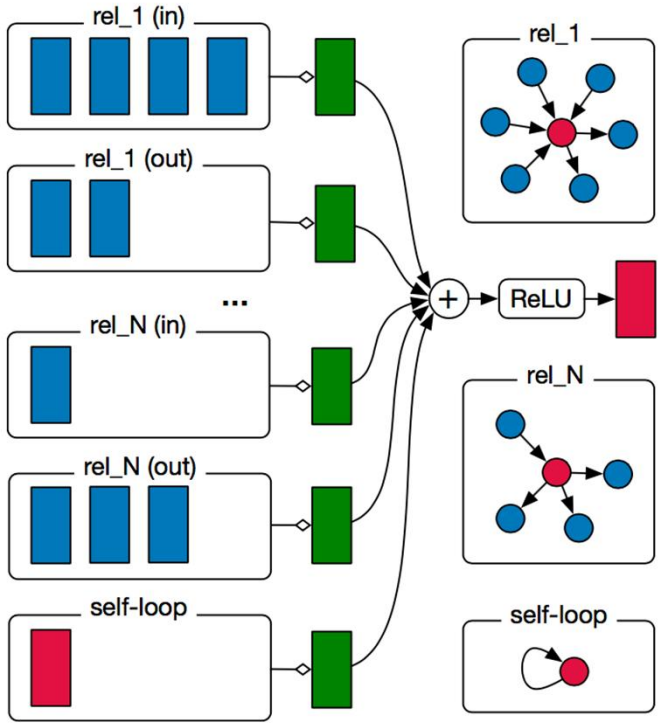
Handles graphs whose nodes are connected with different relations.



$M_{vw}^{(l)} = \frac{1}{c_{v,r}} W_r^{(l)} h_w^{(l-1)}$, r is the relation of e_{vw}.

$$m_v^{(l)} = \sum_{w \in N(v) \cup \{v\}} M_{vw}^{(l)}$$

$$h_v^{(l)} = \sigma(m_v^{(l)} W^{(l)})$$

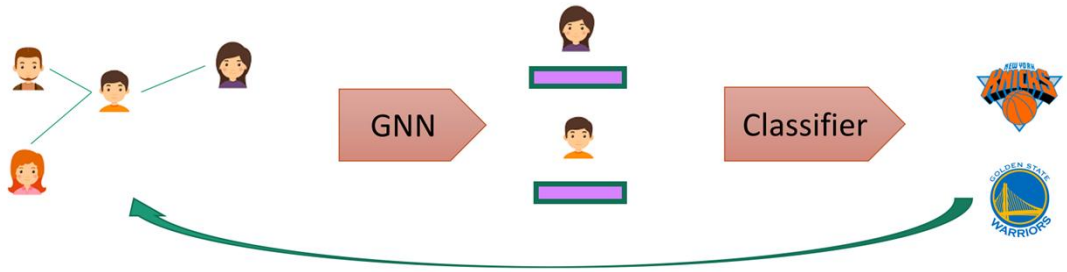


How to train an end-to-end GNN?

- An L-layer GNN computes node embeddings

$$m_v^{(l)} = \sum_{w \in N(v)} M^{(l)}(h_v^{(l-1)}, h_w^{(l-1)}, e_{vw})$$

$$h_v^{(l)} = U^{(l)}(h_v^{(l)}, m_v^{(l)})$$

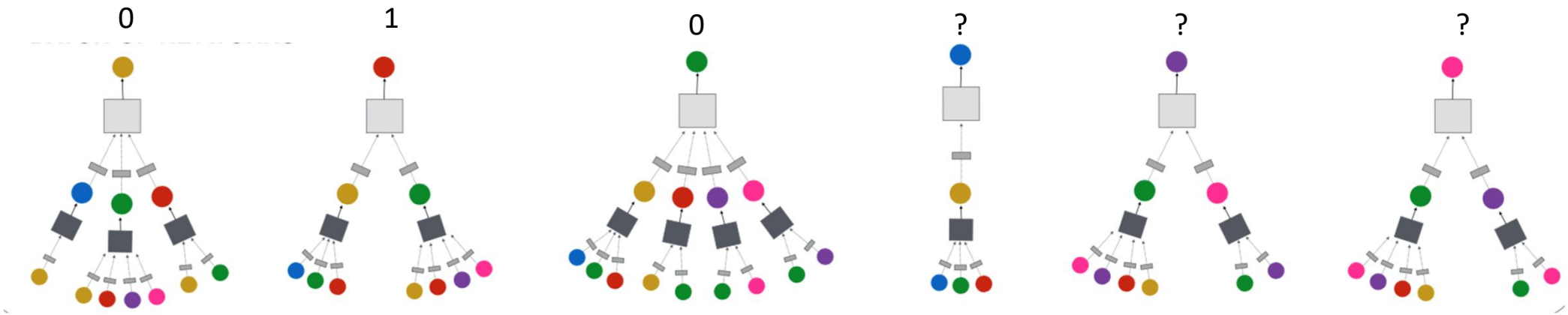


- We use $h_v^{(l)}$ in a downstream model, often an MLP, trained with any loss function.

Node classification with GNN

- Node classification is trained in the semi-supervised setting.

$$\text{loss} = \text{CrossEntryLoss}(h_v^{(l)} W, label_v)$$



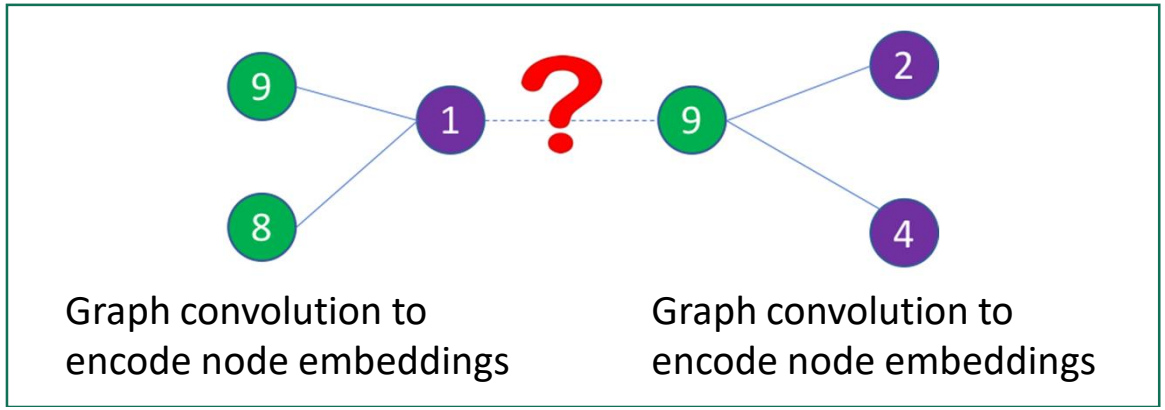
Link prediction with GNN

- We train a link prediction model with connectivity of nodes as the training signal.
  - Positive edges are trained against a few negative edges

$$\text{loss} = -\log\left(\sigma(h_v^T h_w)\right) - Q \cdot E_{u_n \sim P_n(v)} \log\left(\sigma(-h_v^T h_{u_n})\right)$$

Positive edges

Negative edges

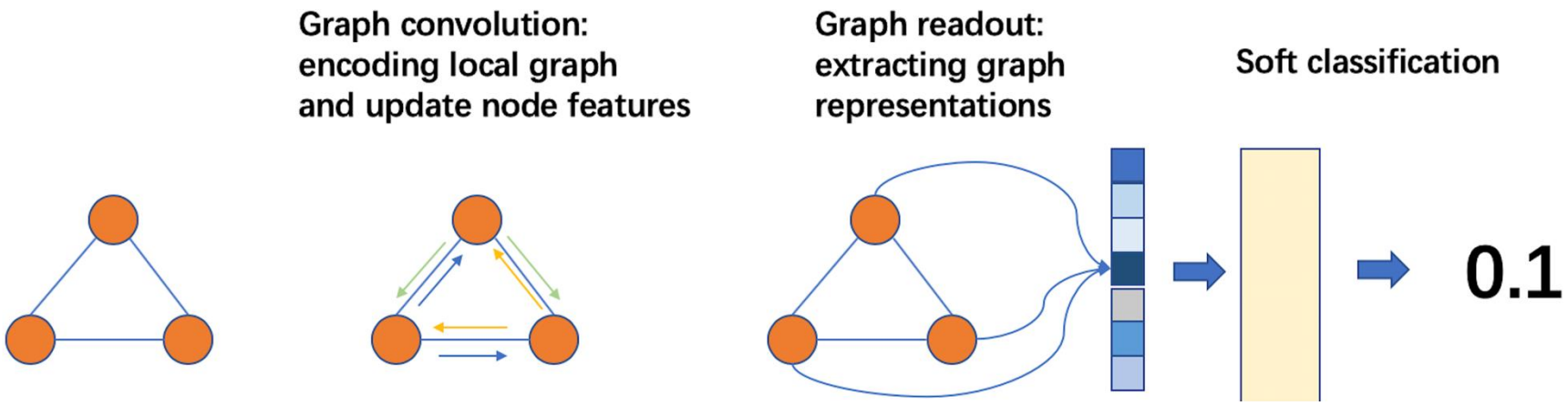


Graph classification

- Graph readout to compute graph embeddings.
- Train a graph classifier on the graph embedding.

$$g = readout\left(h_1^{(l)}, h_2^{(l)}, \ldots, h_n^{(l)}\right)$$

$$\text{loss} = \text{CrossEntryLoss}(g\,W, label)$$

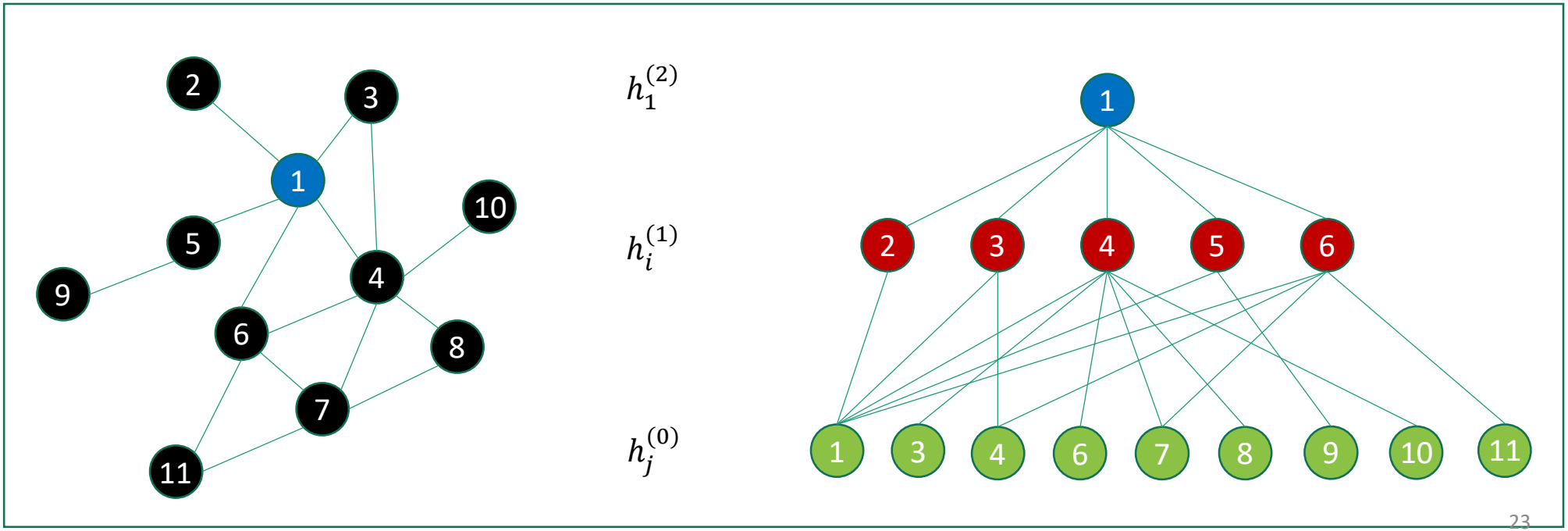


How to train GNN on large graphs

- Many applications have extremely large graphs—millions of nodes and billions of edges:
  - Social networks
  - Recommendation
  - Knowledge graph
  - …
- A typical training method: mini-batch training

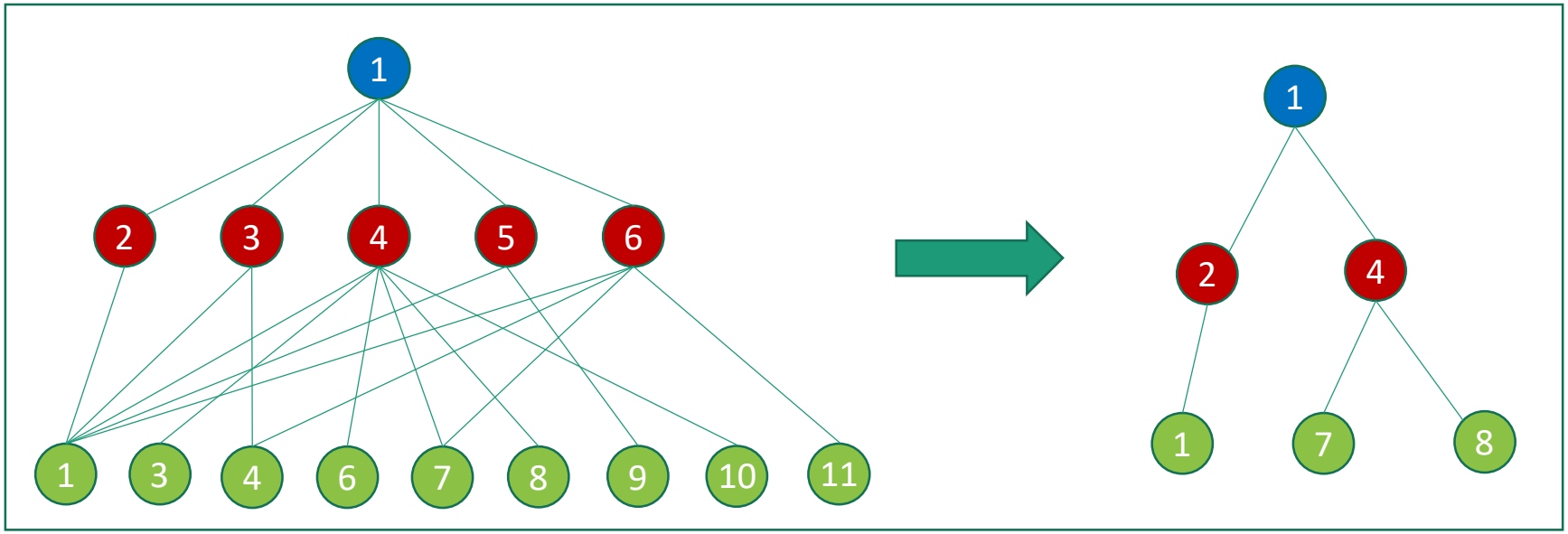
A glance of mini-batch training on graphs

- A mini batch represents the computation graph for target nodes.
- Small-world graphs lead to a huge computation graph.



Mini-batch with neighbor sampling

- Prune the computation graph:
  - Sample neighbors from a neighbor list of a vertex.



Summary

- The solution to many applications can be formulated as graph learning problems.
- Graph neural networks are a new technique for graph learning.
  - They have multiple advantages over traditional methods.
- It is an exciting area of research and they are increasingly find their way into production applications.