Any reproduction or distribution of this document, in whole or in part, is prohibited unless permission is granted by the authors

EE-559
Deep Learning

What's on today?

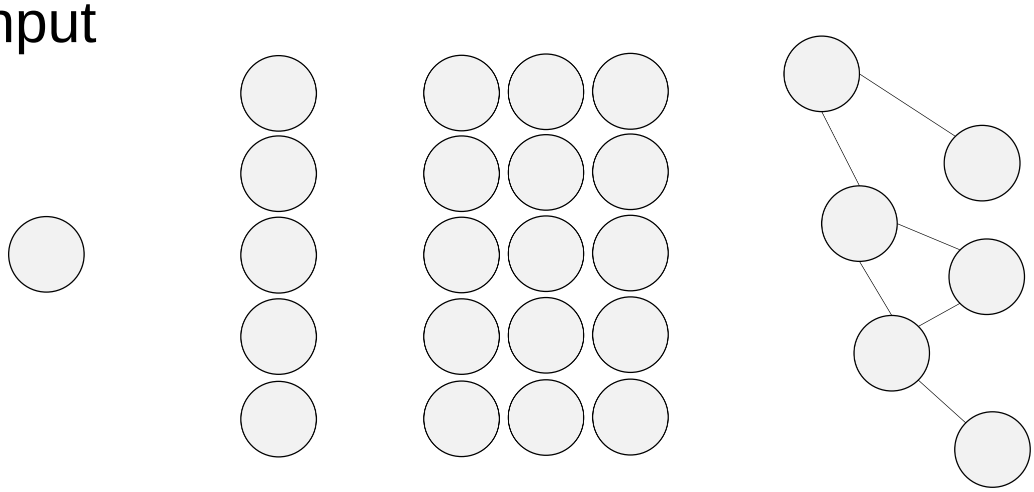
- Graphs: on nodes, edges and structure
- Simple graph: on aggregation and parameter sharing
- Tasks on graphs: how to perform regression and classification
- Graph convolutional networks: on deep learning with graphs
- Graph attention: on weighted, learned, neighbor feature aggregation
- Training: how to deal with the structure
- Line graphs: on the complementary graph
- Graph types: on the diversity of graph representations
- Exercises: message passing and graph classification

Graphs

$$y = f(x)$$

$$y = f(x; \Theta)$$

Input



$x \in \mathbb{R}$ $\quad$ $\boldsymbol{x} \in \mathbb{R}^{W}$ $\quad$ $\boldsymbol{X} \in \mathbb{R}^{W \times N}$ $\quad$ $\boldsymbol{X} \in \mathbb{R}^{W \times N}$, $\mathbf{A} \in \mathbb{R}^{N \times N}$

slido

What type of real-world problems can be modeled effectively using a graph representation?

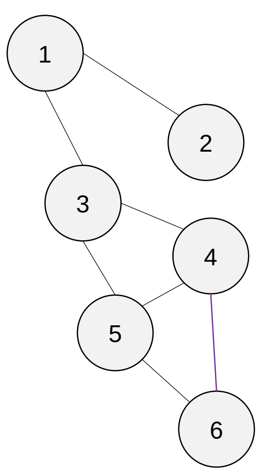
ⓘ Start presenting to display the poll results on this slide.

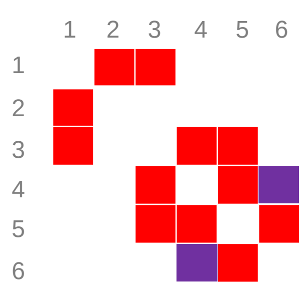
$$y = f(\boldsymbol{X}, \boldsymbol{A}; \boldsymbol{\Theta})$$

$$y = f(\boldsymbol{X}, \boldsymbol{A}; \boldsymbol{\Theta})$$

adjacency matrix

Adjacency matrix



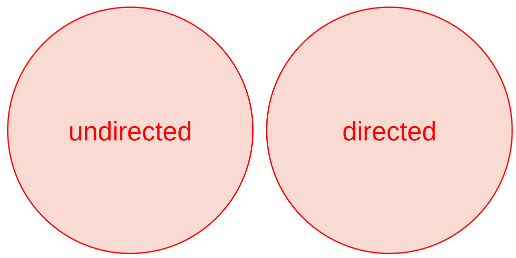


$\boldsymbol{A} \in \mathbb{R}^{N \times N}$

$N = 6$

Concepts:
Node indexing, walks of length one

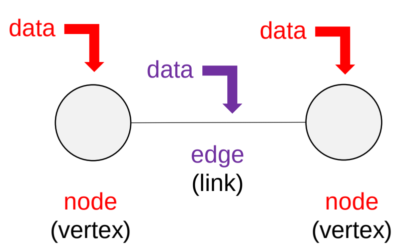
Edges



$\boldsymbol{A}$ is symmetric

$\boldsymbol{A}$ can be asymmetric

Node and edge embeddings

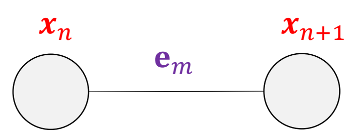


Examples: Social networks, citation networks, train map (stations, lines), protein interactions in a cell, molecule (component, bound)

Structure and embeddings

structure of the graph	node embeddings	edge embeddings
$\boldsymbol{A} \in \mathbb{R}^{N \times N}$	$\boldsymbol{X} \in \mathbb{R}^{W_x \times N}$	$\mathbf{E} \in \mathbb{R}^{W_e \times E}$
$a_{nm} \in \{0,1\}$	$\boldsymbol{x}_n \in \mathbb{R}^{W_x}$	$\mathbf{e}_m \in \mathbb{R}^{W_e}$

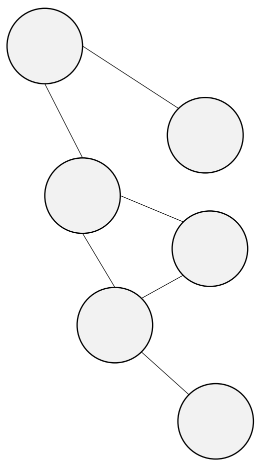
- N number of nodes
- E number of edges
- W_x size of the node embeddings
- W_e size of the edge embeddings



Concept:
The adjacency matrix, $\boldsymbol{A}$, is symmetric for undirected graphs

Simple graph

Simple graph



$\boldsymbol{A} \in \mathbb{R}^{N \times N}$ defines neighboring edges (**structure of the graph**)

$\mathcal{N}_n$ set of all neighbors of node n

$\boldsymbol{X} \in \mathbb{R}^{W_x \times N}$ node embeddings $\boldsymbol{x}_n \in \mathbb{R}^{W_x}$

$\boldsymbol{E} \in \mathbb{R}^{W_e \times E}$ edge embeddings $\mathbf{e}_m \in \mathbb{R}^{W_e}$

Concepts:
At most one edge between any two nodes, undirected edges, no self-edges

Walks from a node

$\boldsymbol{A} \in \mathbb{R}^{N \times N}$

$a_{nm} \in \{0,1\}$

$\boldsymbol{x}_n \in \{0,1\}^N$

encoding each node as a one-hot vector

$\boldsymbol{A}\boldsymbol{x}_n$	number of walks of length 1 from node n to each other node
$\boldsymbol{A}^2\boldsymbol{x}_n$	number of walks of length 2 from node n to each other node
⋮	
$\boldsymbol{A}^L\boldsymbol{x}_n$	number of walks of length $\boldsymbol{L}$ from node n to each other node

Concept:
Neighbourhood of each node

Permutation of node indices

$\boldsymbol{P}$ permutation matrix

one entry in each row and column is 1, the others are 0

$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$

permutes the columns

$\boldsymbol{A}' = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{P}$

permutes the rows

Concept:
Node indexing in a graph is *arbitrary* (any processing applied to the graph should be indifferent to permutations)

Importance of parameter sharing

invariance

dependence only on the structure, $\boldsymbol{A}$, *not on the labelling of the nodes*

equivariance

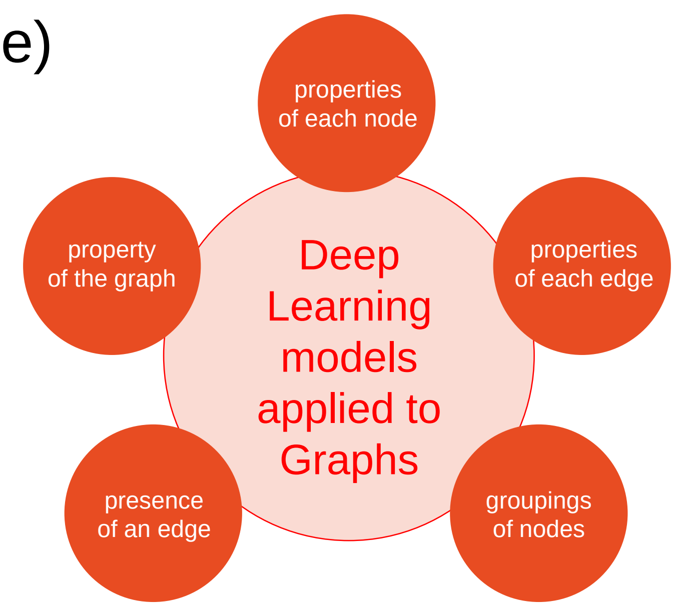
output permuted consistently with permutation of $\boldsymbol{A}$

scaling

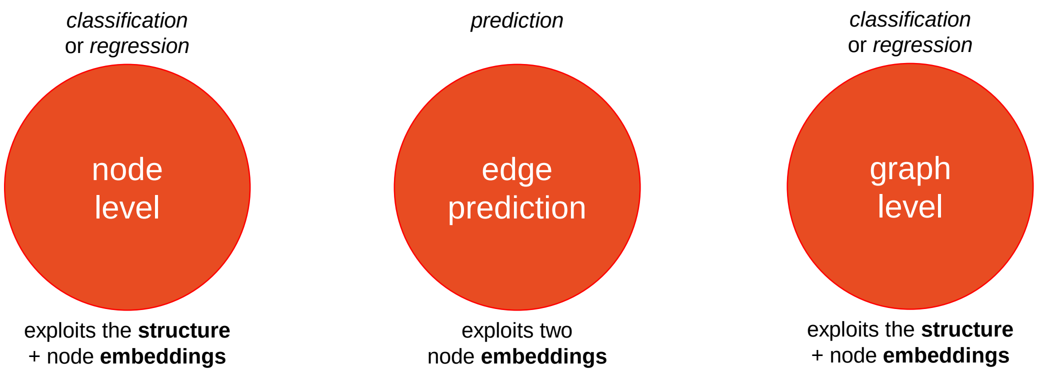
handling the growing of the graph size

Tasks on graphs

Tasks (inference)



Tasks on a graph



Tasks on a graph

$\boldsymbol{X}$	(input) node embeddings (data)
$\boldsymbol{A}$	adjacency matrix (structure)
$\boldsymbol{H}^k$	hidden representation (layer k)
$\boldsymbol{h}_n^k$	(modified) node n embedding

$$P(y_{nm} = 1 | \boldsymbol{X}, \boldsymbol{A}) = \mathrm{sigmoid}\big((\boldsymbol{h}_m^{\mathrm{K}})^{\mathrm{T}} \boldsymbol{h}_n^{\mathrm{K}}\big)$$ edge prediction

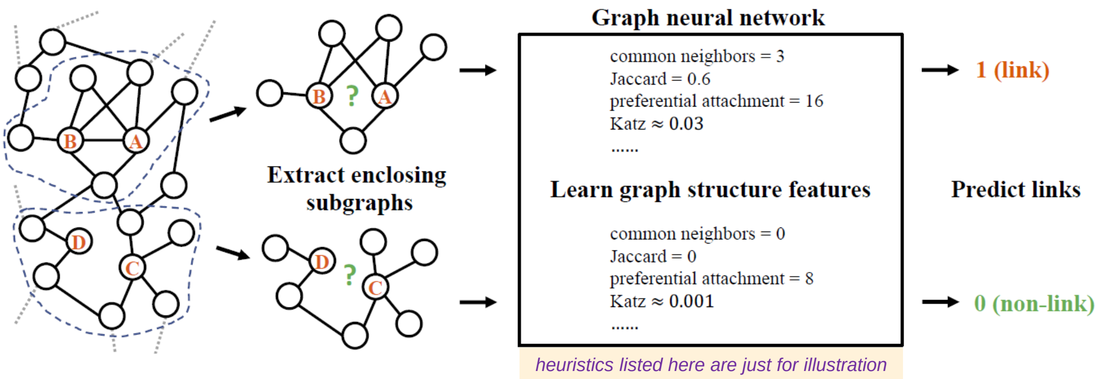
$$P(y_n = 1 | \boldsymbol{X}, \boldsymbol{A}) = \mathrm{sigmoid}(\Theta_1^{\mathrm{K}} + \boldsymbol{\Theta}_1^{\mathrm{K}} \boldsymbol{h}_n^{\mathrm{K}})$$ node level

$$P(y = 1 | \boldsymbol{X}, \boldsymbol{A}) = \mathrm{sigmoid}(\Theta_1^{\mathrm{K}} + \boldsymbol{\Theta}_1^{\mathrm{K}} \boldsymbol{H}^{\mathrm{K}} \boldsymbol{1}/N)$$ graph level

mean pooling

Concepts: *Edge prediction*: binary classification task, *node level*: independently for each node, *graph level*: combining output node embeddings

Edge prediction example



arXiv:1802.09691

Concepts: *Graph structure features* inside the observed node and edge structures, relative degree of influence of a node (*Katz centrality*), intersection over the union of the sets of neighbors of a node (*Jaccard index*)

slido

Is there anything about the mini-project that you would like clarification on?

ⓘ Start presenting to display the poll results on this slide.

Graph convolutional networks

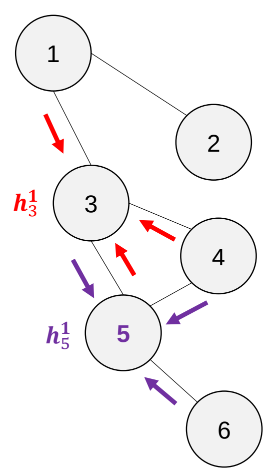
Graph convolutional networks

$$\boldsymbol{H}^1 = f(\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{\Theta}^0)$$

$$\boldsymbol{H}^2 = f(\boldsymbol{H}^1, \boldsymbol{A}, \boldsymbol{\Theta}^1)$$

$$\vdots$$

$$\boldsymbol{H}^K = f(\boldsymbol{H}^{K-1}, \boldsymbol{A}, \boldsymbol{\Theta}^{K-1})$$



$\boldsymbol{X}$	input node embeddings
$\boldsymbol{A}$	adjacency matrix (structure)
$\boldsymbol{H}^k$	modified node embeddings (layer k)
$\boldsymbol{\Theta}^k$	parameters that map from layer k to $k+1$

Concepts: Relational inductive bias, *message passing*, spatial-based convolutional graph neural network (GCN), updating the *local representation* at each node by gathering information from its neighbors by passing messages

GCN layer

$$\mathcal{G}_n^k = \sum_{m \in \mathcal{N}_n} \boldsymbol{h}_m^k$$

set of indices of the neighborhood of node n

$$\boldsymbol{h}_n^{k+1} = a[\Theta_0^k + \boldsymbol{\Theta}_1^k \boldsymbol{h}_n^k + \boldsymbol{\Theta}_1^k \mathcal{G}_n^k]$$

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k \boldsymbol{A}]$$

$$= a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k (\boldsymbol{A} + \boldsymbol{I})]$$

aggregated neighborhood

combine “messages” from adjacent nodes
(sum them with the transformed current node)

Concept: Aggregating information from neighboring nodes (*sum of node embeddings*), *local* function of the embedding of the previous layer, combine *messages* from adjacent nodes

Oversmoothing problem

repeated graph convolutions make
node embeddings indistinguishable
(oversmoothing)

↓

$k \uparrow \Rightarrow$ performance gradually decreases

↓

normalization layer

k : number of layers (network depth)

arXiv:1909.12223

Graph prediction example

Is a molecule toxic?

$$\boldsymbol{H}^k = a[\Theta_0^{k-1}\mathbf{1}_N^T + \boldsymbol{\Theta}_1^{k-1}\boldsymbol{H}^{k-1}(\boldsymbol{A}+\boldsymbol{I})]$$

$$f(\boldsymbol{X},\boldsymbol{A},\boldsymbol{\Theta}) = \text{sigmoid}(\Theta_1^K + \boldsymbol{\Theta}_1^K\boldsymbol{H}^K\mathbf{1}/N)$$

mean pooling

$\mathbf{A} \in \mathbb{R}^{N \times N}$
molecular structure

$\boldsymbol{X} \in \mathbb{R}^{118 \times N}$
matrix of one-hot vectors indicating which element of the periodic table is present

Variants for aggregation

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k\mathbf{1}_N^T + \boldsymbol{\Theta}_1^k\boldsymbol{H}^k(\boldsymbol{A}+\boldsymbol{I})]$$

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k\mathbf{1}_N^T + \boldsymbol{\Theta}_1^k\boldsymbol{H}^k(\boldsymbol{A}+(1+\epsilon_k)\boldsymbol{I})]$$

diagonal enhancement

learned scalar

$$\boldsymbol{H}^{k+1} = \begin{bmatrix} a[\Theta_0^k\mathbf{1}_N^T + \boldsymbol{\Theta}_1^k\boldsymbol{H}^k\boldsymbol{A}] \\ \boldsymbol{H}^k \end{bmatrix}$$

residual connection
concatenation with the node

$$\mathcal{G}_n^k = \max_{m \in \mathcal{N}_n} \boldsymbol{h}_m^k$$

max pooling aggregation
element-wise maximum

Mean aggregation

$$\mathcal{G}_n^k = \sum_{m \in \mathcal{N}_n} \boldsymbol{h}_m^k \Rightarrow \mathcal{G}_n^k = \frac{1}{|\mathcal{N}_n|} \sum_{m \in \mathcal{N}_n} \boldsymbol{h}_m^k$$

mean aggregation
when *embedding information* is more important than *structural information*

$\boldsymbol{D} \in \mathbb{R}^{N \times N}$ **diagonal matrix**
each non-zero element is the number of neighbors of the corresponding node

$\Rightarrow$ $(\boldsymbol{D})^{-1}$ **inverse matrix**
each non-zero element is the denominator to compute the average

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k (\boldsymbol{A}(\boldsymbol{D})^{-1} + \boldsymbol{I})]$$

Spatial and spectral methods

spatial methods
use the graph structure

spectral methods
use the Fourier domain

Graphs and transformers

- handle variable-length input
- transformer w/o positional embedding ~ GNN on a fully connected graph
- use of attention

w/o: without

~: equivalent to

GNN: Graph Neural Network

Graph attention

Graph attention layers

$$\widehat{\boldsymbol{H}^k} = \Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k$$ linear transformation on node embedding

$$s_{mn} = a\left[(\boldsymbol{\phi}^k)^{\mathrm{T}}\begin{bmatrix}\widehat{\boldsymbol{h}_m^k}\\ \widehat{\boldsymbol{h}_n^k}\end{bmatrix}\right]$$

$\boldsymbol{S} \in \mathbb{R}^{N \times N}$ similarity of every node to every other node

↑ vector of learned parameters

$$\boldsymbol{H}^{k+1} = a\left[\widehat{\boldsymbol{H}^k}\ \mathrm{softmask}[\boldsymbol{S}, \boldsymbol{A} + \boldsymbol{I}]\right]$$ attention weights applied to transformed embedding

softmax for each column of $\boldsymbol{S}$
(set to zero non-neighboring nodes)

Concepts: Aggregation by attention, concatenation of node embeddings, weights depend on the *data at the nodes* (previous cases depended instead on neighbors equally or on graph topology)

Aggregation: summary

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k(\boldsymbol{A} + \boldsymbol{I})]$$

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k(\boldsymbol{A} + (1 + \epsilon_k)\boldsymbol{I})]$$

diagonal enhancement
learned scalar ϵ_k

$$\boldsymbol{H}^{k+1} = \begin{bmatrix} a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k \boldsymbol{A}] \\ \boldsymbol{H}^k \end{bmatrix}$$

residual connection
concatenation with the node

$$\mathcal{G}_n^k = \max_{m \in \mathcal{N}_n} \boldsymbol{h}_m^k$$

max pooling aggregation
element-wise maximum of the $\boldsymbol{h}_m^k$

$$\boldsymbol{H}^{k+1} = a[\Theta_0^k \mathbf{1}_N^T + \boldsymbol{\Theta}_1^k \boldsymbol{H}^k(\boldsymbol{A}(\boldsymbol{D})^{-1} + \boldsymbol{I})]$$

mean aggregation
less importance to structural information

$$\boldsymbol{H}^{k+1} = a\left[\widehat{\boldsymbol{H}^k}\ \mathrm{softmask}[\boldsymbol{S}, \boldsymbol{A} + \boldsymbol{I}]\right]$$

aggregation by attention
attention weights on transformed embedding

Training

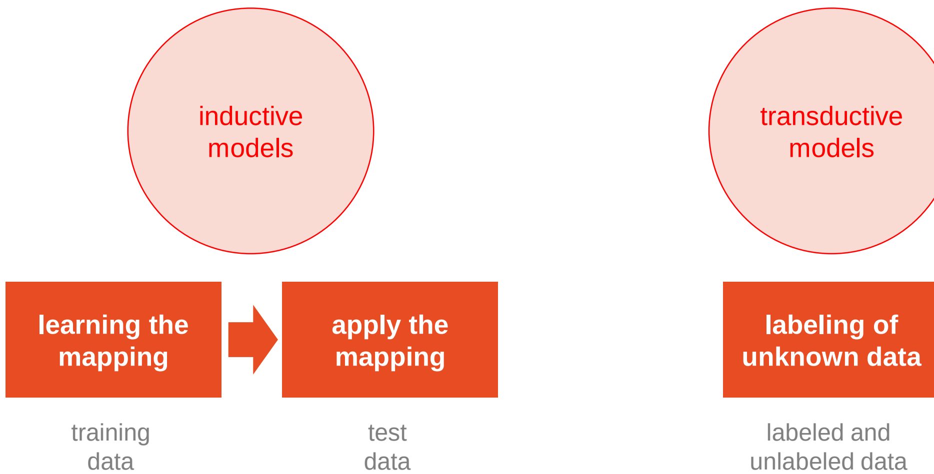
$$\{x_i, y_i\}_{i=1}^{M}$$

$$\{\boldsymbol{X}_i, \boldsymbol{A}_i, y_i\}_{i=1}^{M}$$

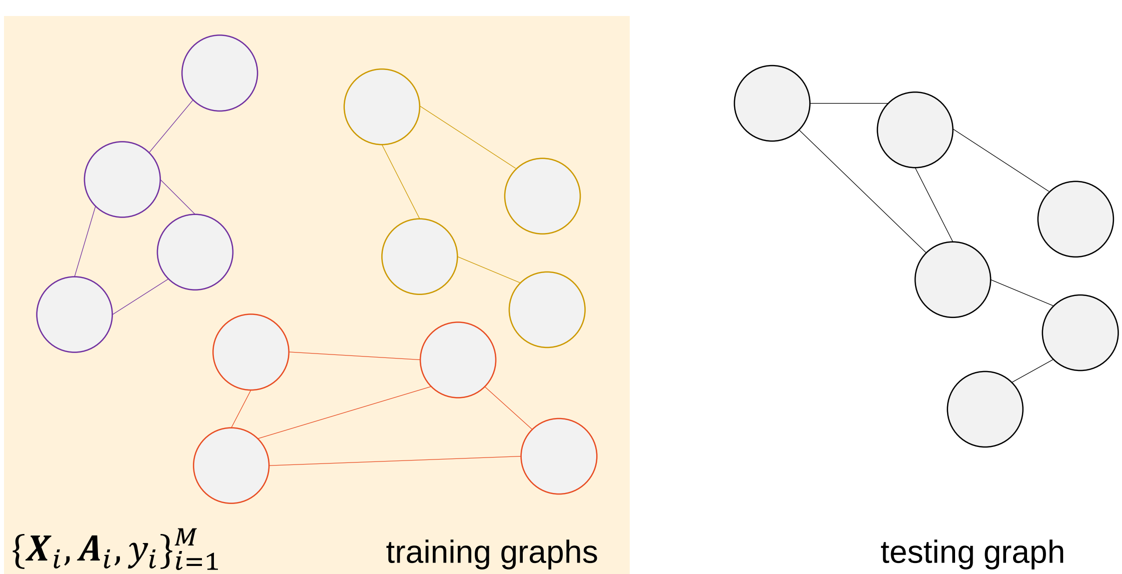
$$y = f(\boldsymbol{X}, \boldsymbol{A}; \boldsymbol{\Theta})$$

learned with SGD
& binary cross-entropy loss

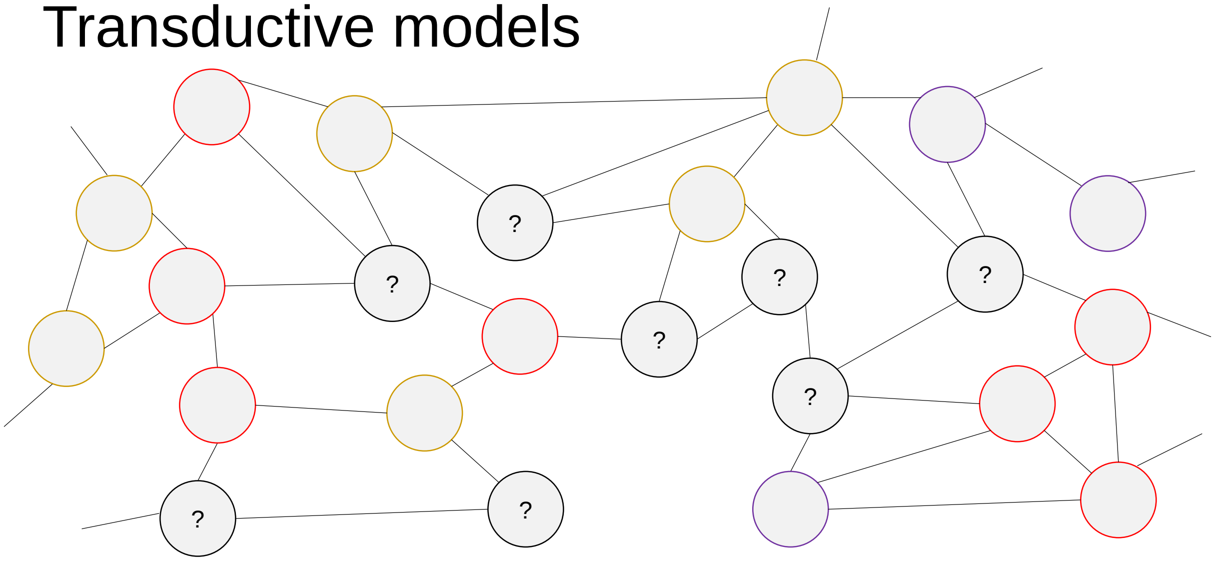
Inductive and transductive models



Inductive models



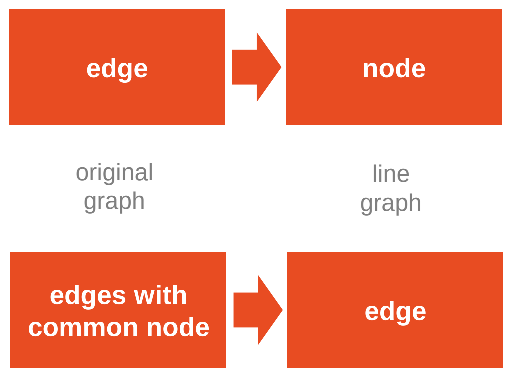
Transductive models



Concept:
Train to predict the known labels, then examine the predictions at the unknown nodes

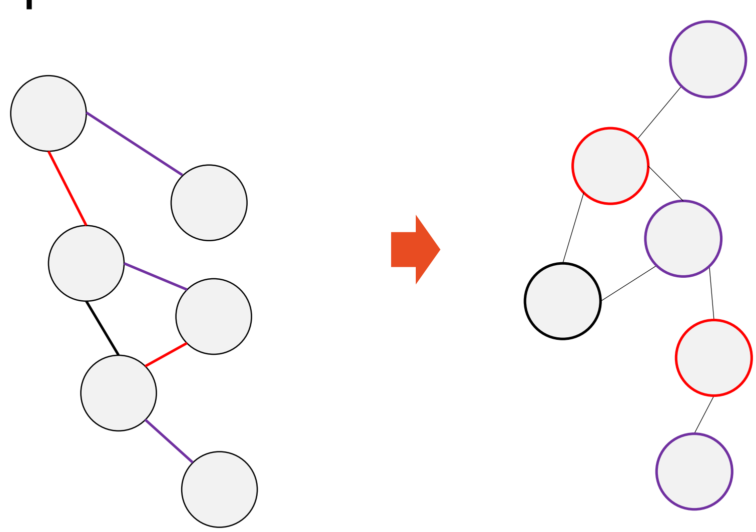
Line graphs

Line or edge graph



Concept:
Edge graph, complementary graph to process *edge embeddings*

Example



Graph types

Graph types

heterogeneous

nodes / edges
of **different types**

directed
multigraph

multiple edges
between the same
pair of nodes

geometric

defined by
geometric means
(**nodes in a space
with a metric**)

hierarchical

defined also by
an acyclic
parent function
$p: \mathcal{N} \rightarrow \mathcal{N}$

Concepts:
Simple graphs, spatial elements associated to geometric objects, parent function defines the hierarchy

Exercises

Today’s exercises

Practice. You will become familiar with:

- **PyTorch geometric**: load datasets, view information about the graphs
- **Graph Neural Networks**: message passing and graph classification

What did we learn today?

- Graphs
- Simple graph
- Tasks on graphs
- Graph convolutional networks
- Graph attention
- Training
- Line graphs
- Graph types
- Exercises

EE-559

Deep Learning

andrea.cavallaro@epfl.ch