Graph neural networks lecture

TIF 360

Mats Granath

28/3 2023

Graph neural networks =

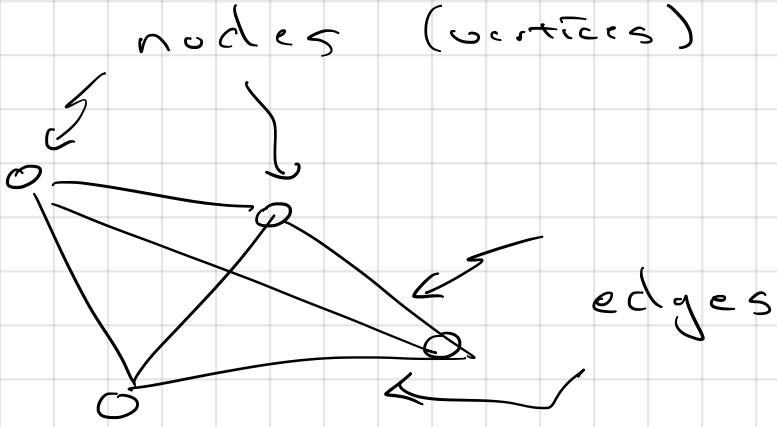
= neural networks acting on graphs

Outline

- Why interesting to apply NN to graph objects?
- How is a graph defined?
- What type of machine learning task may we want to do?
- Basic GNN layer:

  Graph convolution
- Other layers + Graph Pooling
- Homework A

Graph

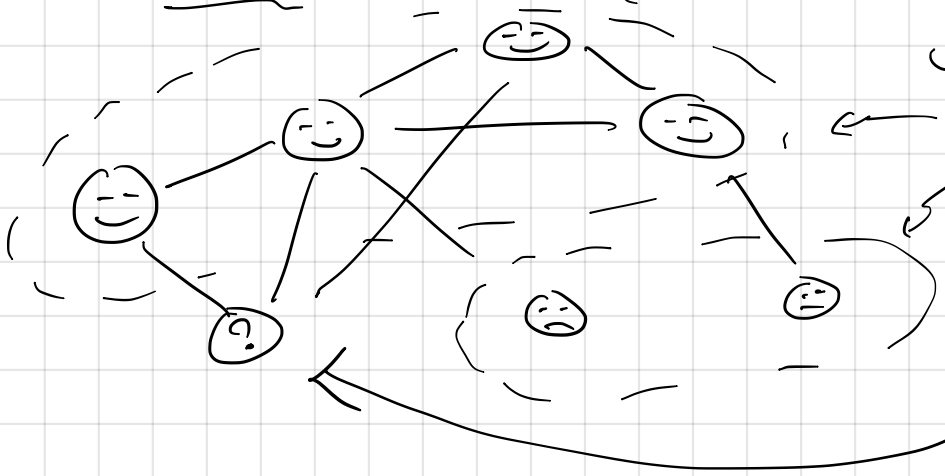


More specific shortly

Why NN on graphs / what tasks?

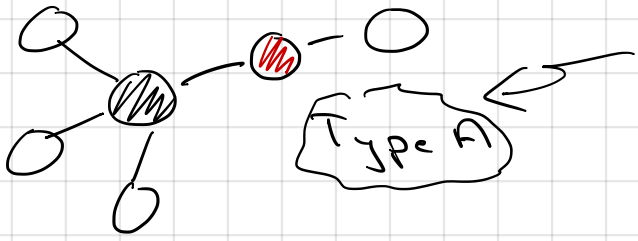
Many interesting forms of data that can be nicely represented as graphs:

Social networks:



We may want to do clustering. Or node prediction.

Molecules



We may want to do classification or regression

e.g. how chemically active is the molecule

Different methodology for different tasks

[4] Graph classification / regression

↕

Supervised learning

↕

Data: large number of labeled graphs

↑

The network needs to handle different size graphs with varying connectivity (node degree)

[4] Node-prediction

↕

Semi-supervised learning:

↕

typically single graph, node degree may vary

some labeled nodes, also edge info taken into account e.g. neighboring nodes may be more likely to be similar

[3] Clustering: can be unsupervised

The GNN may do some automatic clustering

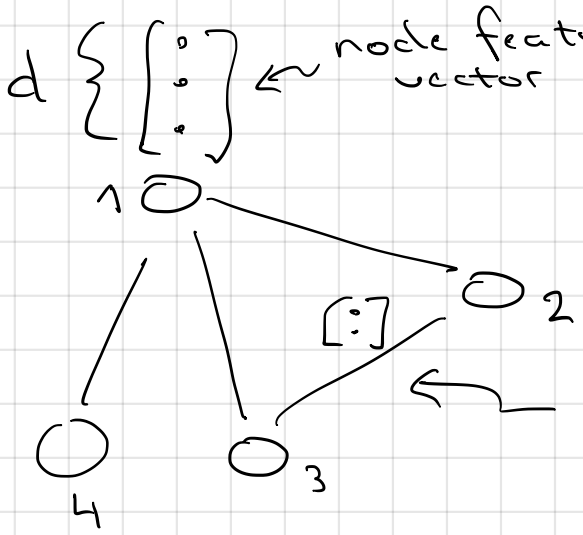
graph → new graph where info from neighboring nodes is absorbed

also, lower dim. embedding

Graphs: not only nodes and edges (in this context)

"decorated graph"

Information on nodes and edges



$\Rightarrow \vec{x}_i \quad i = 1, \dots n$

d: dimension of node feature vector

n: nr of nodes

$\vec{e}_{23}$, e_{23}: node feature vector, or scalar

$\vec{e}_{ij}, e_{ij}$

The structure is given by an

Adjacency matrix A_{ij}

$A_{ij} = 1$ if edge $i \to j$

$A_{ij} = 0$ if no edge

Undirected graph $A_{ij} = A_{ji}$

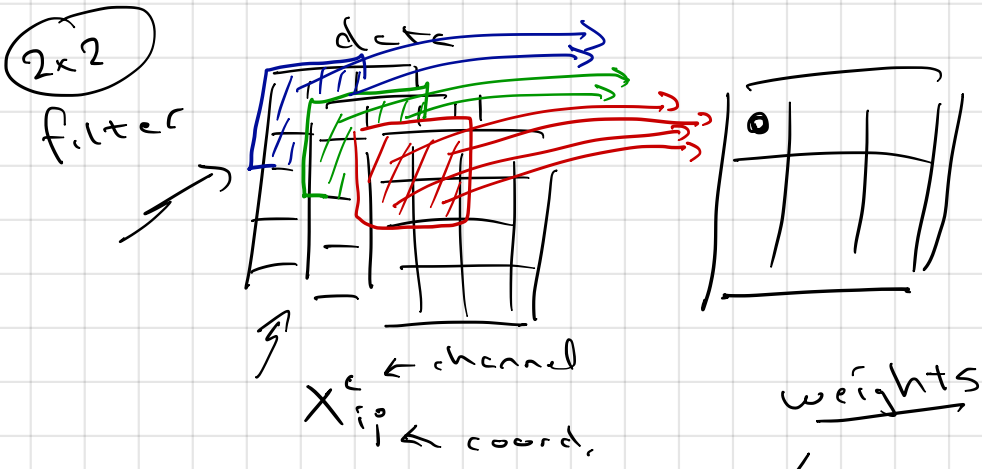
Ex above

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Convolutions on graphs

Collect information from neighboring nodes seems like a natural construction.

Ordinary convolution layer on grid:

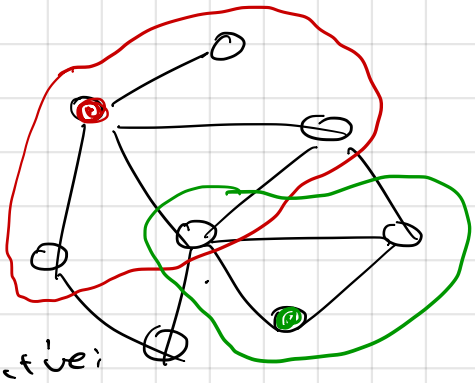


individual weight to each input.

$$X'_{kl} = \sigma\left(\sum_{\substack{m=0,1 \\ n=0,1 \\ c}} w^{c}_{mn} \cdot x^{c}_{k+m,l+n} + b\right)$$

pos — weights — bias

On a graph?



objective: new graph, new feature vectors based on neighbors

Since the number of neighbors vary, not simple to have different weights to diff nodes

Discuss ingredients

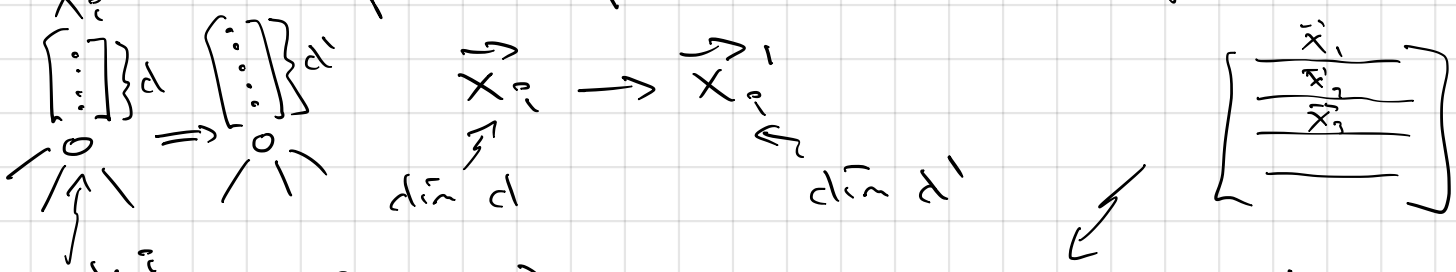
$\vec{x}_i$ feature vector on node i

A_{ij} adjacency matrix

σ: non-linear fn (i.e ReLU or other)

- Basic construction GCN Conv (Kipf & Welling)
  - Isomorphic mapping: $A'_{ij} = A_{ij}$
    same number of nodes, same connectivity
  - Include self-loops in A:
    $\tilde{A} = A + \mathbb{1}$ $\quad (\tilde{A}_{ij} = A_{ij} + \delta_{ij})$
  - Degree matrix $\tilde{D}_{ii} = \sum_{j=1}^{n} \tilde{A}_{ij} = \deg_i$
    # number of neighbors incl self
    $\tilde{D}_{ij} = 0$

$\vec{x}_i$ $\vec{x}'_i$ • Map node feature to new node feature



Write $\vec{x}_i$ as matrix $X \leftarrow n \times d$

act. fcn, incl. bias

$$X' = \sigma\left(\underbrace{\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}}_{\text{adds up all neighbor vectors normalized by degree}} X W\right)$$

W: weight matrix $d \times d'$

W transforms each node vector $d \to d'$

$$\vec{x}'_i = \sigma\left(\sum_j \frac{\tilde{A}_{ij}}{\sqrt{\deg_i}\sqrt{\deg_j}} W^T \vec{x}_j\right)$$

"just like a non-activated dense layer on each node"

Collects information "one-hop" per layer. Isotropically, i.e. all neighbors have equal influence.

- Simple variation **GraphConv**

$$\vec{x}_i' = \sigma\left(\omega_1 \vec{x}_i + \omega_2 \sum_j e_{ij} \vec{x}_j\right)$$

e_{ij}: edge weight
$e_{ij} = 0$ if $A_{ij} = 0$

takes edge weights into account

fixed

- We can also make the absorption of information from neighboring nodes adaptive (trainable).

Graph attention layer **GATConv**

$$\vec{x}_i' = \sigma\left(\sum_j \alpha_{ij} \omega \vec{x}_j\right)$$

ω: $d' \times d$ weight matrix

- $\alpha_{ij} = \text{softmax}(\varepsilon_{ij}) = \dfrac{e^{\varepsilon_{ij}}}{\sum_k e^{\varepsilon_{ik}}}$

$$\sum_j \alpha_{ij} = 1$$

$$\Sigma_{ij} = \sigma'\left(\vec{a} \cdot \underbrace{[\omega\vec{x}_i \| \omega\vec{x}_j]}_{\text{concatenation}}\right) \tilde{A}_{ij}$$

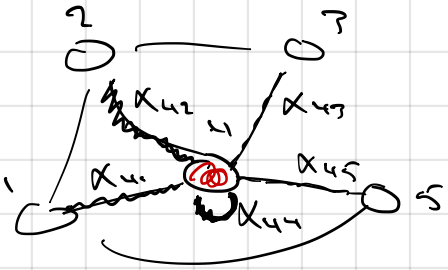
possibly other activation

$2d'$ attention vector

$$\begin{bmatrix} \omega\vec{x}_i \\ \vdots \\ \omega\vec{x}_j \end{bmatrix} \quad d' \quad d'$$

to make sure $\alpha_{ij} = 0$ if nodes are not adjacent

Attention coefficients, trainable edge weights



more input from some neighboring nodes

- One can also have several attention heads with attention coeffs α^k & weights ω^k

$$\vec{x}'_i = \Big\|_k \vec{x}'^{,k}_i = \begin{bmatrix} \vdots \\ \vec{x}'^{,1}_i \\ \vdots \\ \vec{x}'^{,2}_i \\ \vdots \\ \vec{x}'^{,3}_i \\ \vdots \end{bmatrix} \begin{matrix} \\ \text{head1} \\ \\ \text{head2} \\ \\ \text{head3} \\ \end{matrix}$$

These types of structures have had great success for Natural Language Processing

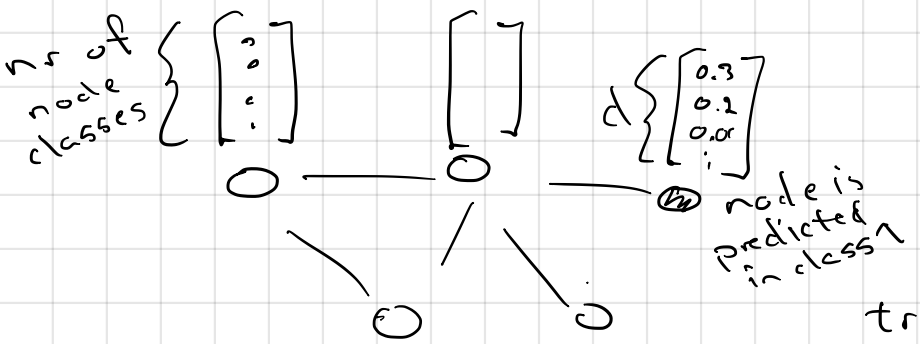
- NB! Convolutional graph layers collect information similarly to each node $\Rightarrow$ after each layer info tends to get smeared out. Too many layers can be bad. (Depending on task.)

Now we have learned how to collect information in the graph.

New graph with node features $\vec{x}'_i$ that reflect surrounding of node i.

What's next? Depends on task — Discuss

- To do node classification: end with a softmax activated convolut. layer with d = number of classes



train over loss w.r.t. labeled nodes.

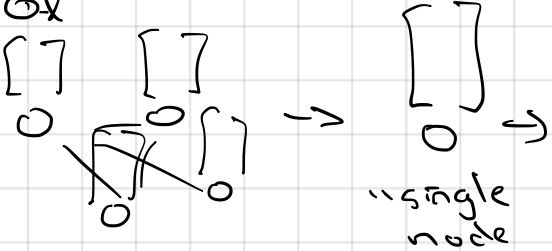
- To do graph classification we need to reduce info to one vector, a.k.a. graph embedding.

For this use Pooling layer

- Simplest Global pooling

E.g. global mean pool

$$\vec{x}'_1 = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i$$



dense → softmax → [class predic.]

- More sophisticated: max-k pooling

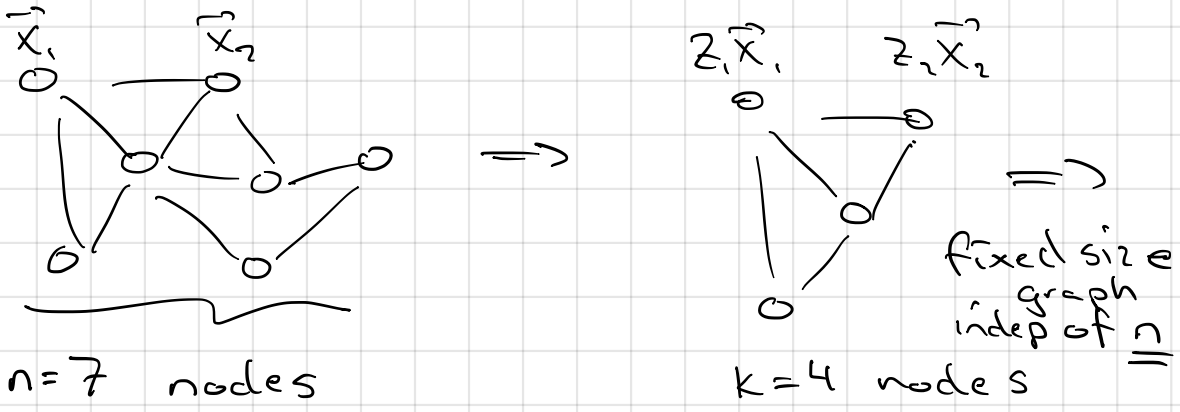
  select k most important nodes

  using "self-attention"

$$Z_i = \sigma(\vec{a} \cdot \vec{x}_i)$$

$\vec{a}$: attention vector

pick k-nodes with highest value Z



Homework A

warm up

- Prob. 1 (2p) Do semisupervised node-classification (most code available)

- Prob 2 (8p) Do graph-classification data from a problem related to quantum computing