GRAPH NEURAL NETWORKS

Nathanaël Fijalkow

What ? Architectures of neural networks taking graphs as input

When ? When data is naturally presented as a graph

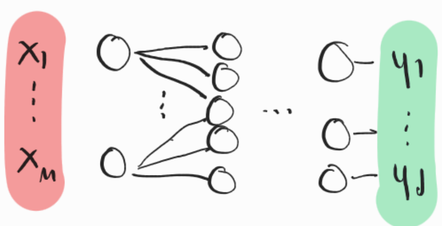
Why ? Because graphs = relations

How ? Let's see !

Background

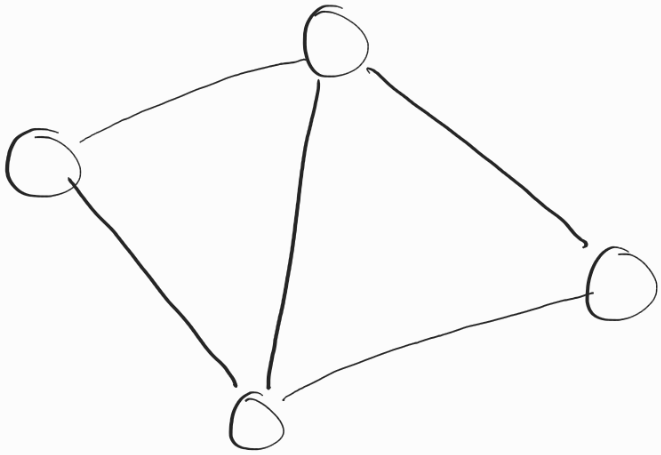
MLP multi layer perceptron

$\phi_\Theta : \mathbb{R}^n \to \mathbb{R}^d$ $\quad\Theta$ set of parameters



MLP can be trained efficiently

What is a graph?



Nodes: V

Node features: $F_v \in \mathbb{R}^n$

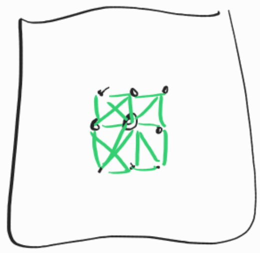
Edges: E

Edge features $F_e \in \mathbb{R}^d$

Graph features $g \in \mathbb{R}^k$

Graph structured data ?

- social networks ✓
- molecules ✓
- images



- text   this is a sentence
- transportation networks ✓

What is special about graphs?

→ represent relations between entities

DIFFICULTY nodes are typically **UNORDERED**

↳ we do not want to introduce arbitrary order!

this is a representation issue

Good representations:

- improve learning performances
- reduce bias

3 types of questions:

Graph-level

does this molecule smell good?

Node level

identify fake users in a network

↳ for each node

Edge level

identify friend relationship

↳ for each edge

Naturally: the answer should NOT depend on representation (chosen order for the nodes, ...)

A special case: Deep Sets (NeurIPS 2017)

we have a set of points $x_1, \dots, x_k \in \mathbb{R}^n$

we want to predict some function

$$F: \mathbb{R}^{n \times k} \to \mathbb{R}$$

we know that F does NOT depend on order

we already lost when we wrote

$x_1, \dots, x_k$: arbitrary order!

IDEA:

$$f(x) = \Phi_\theta\left(\sum_{i=1}^{k} \Psi_{\theta'}(x_i)\right)$$

$\Phi_\theta, \Psi_{\theta'}$ are MLPs

$\sum$ is order invariant!

(BTW, so is max and mean)

Now we have a differentiable model that only represents unordered functions!

A step back: permutation invariance

We knew that

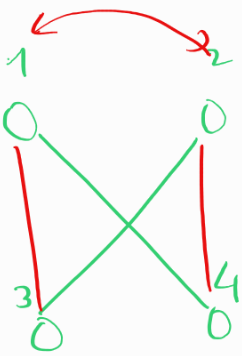
$\forall P$ permutation of $x_1, \dots, x_k$,

$$F(Px) = F(x)$$

← target function is permutation invariant

So we design a class of models which are permutation invariant:

$$f(Px) = f(x)$$



Invariance for graphs

Key difference: permuting vertices affects edges

$$F(PV, PEP^T) = F(V, E)$$ invariant

$$F(PV, PEP^T) = P \circ F(V, E)$$ equivariant

GNN layer

graph-in graph-out: maps a graph to a graph

For a node v:

$$N_v = \{ v' : (v, v') \in E \}$$

neighborhood

$$F_{N_v} = \{\{ F_{v'} : v' \in N_v \}\}$$

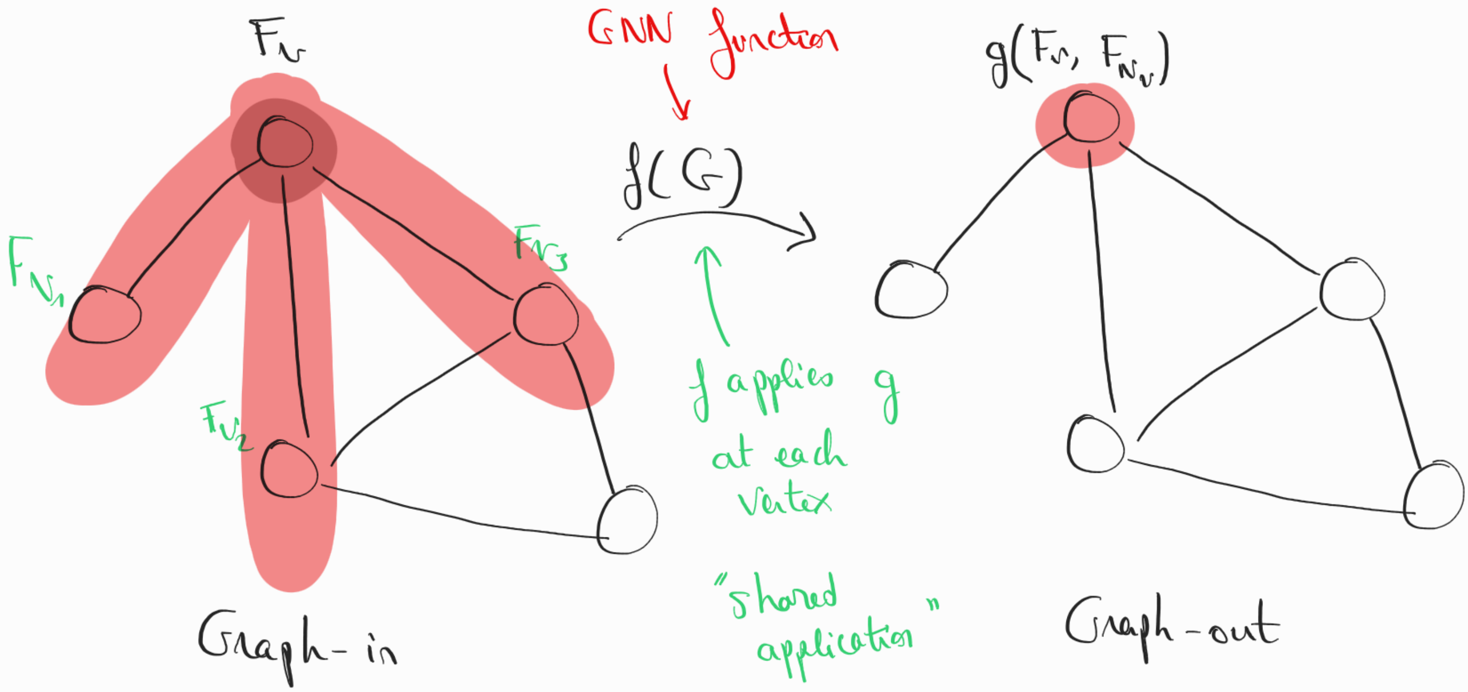
multiset

$$g(F_v, F_{E_{N_v}}, \dots)$$

g local function: $g(F_v, F_{N_v})$

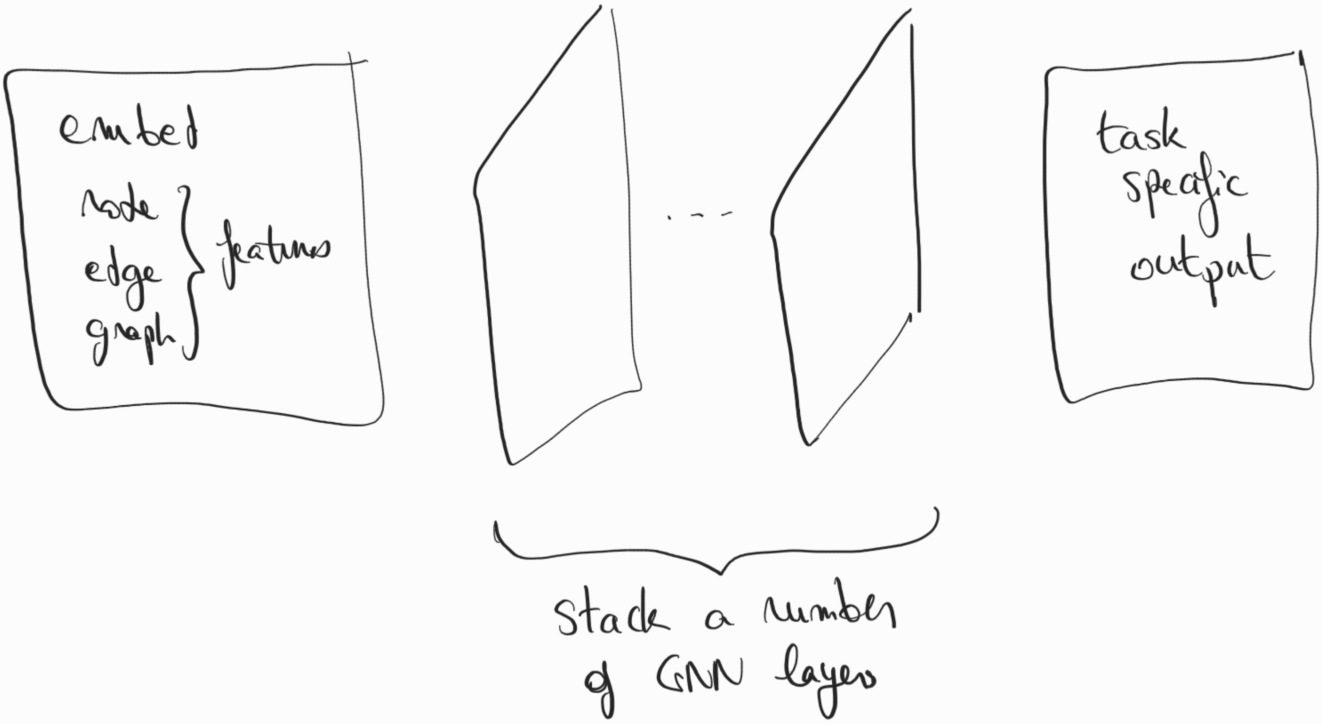
g is invariant under permutation of neighbours

(we'll talk about g soon)



Slightly more general: output node features using node + edge features, etc...

GNN: the full model



Constructing the local function g

g is called "diffusion" / "propagation" / "message passing"

General definition: **message-passing style!**

$$g(F_v, F_{N_v}) = \phi_\theta\left(F_v, \bigoplus_{v' \in N_v} \psi_{\theta'}(F_v, F_{v'})\right)$$

$\phi_\theta, \psi_{\theta'}$ are MLPs

$\bigoplus$ is sum / mean / max

Less general: attentional GNNs

learnable weights

$$g(F_v, F_{N_v}) = \phi_\theta\left(F_v, \bigoplus_{v' \in N_v} a(v,v')\, F_{v'}\right)$$

Even less: Convolutional GNNs

fixed weights

$$g(F_v, F_{N_v}) = \phi_\theta\left(F_v, \bigoplus_{v' \in N_v} c(v,v')\, F_{v'}\right)$$

Packages:

Pytorch Geometric

References:

Distill article (Google)

ICLR'21 invited talk by Bronstein