Graph Neural Networks

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

\[ \phi_\Theta : \mathbb{R}^n \rightarrow \mathbb{R}^j \]

\( \Theta \) 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
- transportation networks ✔

This is a sentence
What is special about graphs?

- represent relations between entities

**DIFFICULTY** nodes are typically **UNORDERED**

So 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
- Is for each node

**Edge level**
- identify friend relationship
- Is 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 $x_1, \ldots, x_k \in \mathbb{R}^n$

We want to predict some function $F: \mathbb{R}^{m \times k} \rightarrow \mathbb{R}$

We know that $F$ does NOT depend on order.

We already lost when we wrote $x_1, \ldots, x_k$: arbitrary order!

**Idea:**

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

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

$\sum$ is order invariant!

(Btw, so is max and mean)

Now we have a model that only represents unordered functions!
A step back: permutation invariance

We knew that

\[ F(Px) = F(x) \quad \text{target function is permutation invariant} \]

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

\[ f(Px) = f(x) \]

Invariance for graphs

Key difference: permuting vertices affects edges

\[ F(PV, PEP^T) = F(V, E) \quad \text{invariant} \]
\[ F(PV, PEP^T) = P^o F(V, E) \quad \text{equivariant} \]
For a node $v$:

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

neighborhood

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

multiset

$$g(F_v, F_{N_v}, \ldots)$$

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

$g$ is invariant under permutation of neighbors

(we'll talk about $g$ soon)

Slightly more general: output node features using node + edge features, etc...
GNN: the full model

- Embed
- Node / edge features
- Task-specific output

Stack a number of GNN layers

Constructing the local function \( g \)

\( g \) is called "diffusion" / "propagation" / "message passing"

General definition: **message-passing style**

\[
g(F_v, F_N) = \phi_\theta \left( F_v, \bigoplus_{u \in N_v} \psi_\theta(F_u, F_v) \right)
\]

\( \phi_\theta, \psi_\theta \) are MLPs

\( \bigoplus \) is sum / mean / max
Less general: \( g(F_v, F_{N_v}) = \Phi_{\theta} \left( F_v, \bigoplus_{v'\in N_v} a(v,v') F_{v'} \right) \)

Even less: \( g(F_v, F_{N_v}) = \Phi_{\theta} \left( F_v, \bigoplus_{v'\in N_v} c(v,v') F_{v'} \right) \)

Package:
- Pytorch Geometric

References:
- Distill article (Google)
- ICLR'21 invited talk by Bronstein