Graph Neural Networks with Missing Node Features

Emanuele Rossi, Twitter & Imperial College London

March 2022

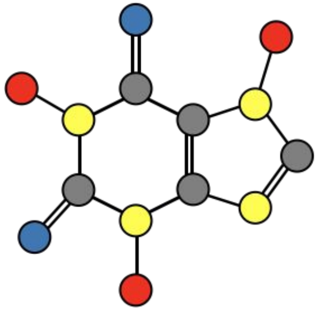
Motivation

Why do we care about graphs and missing node features?

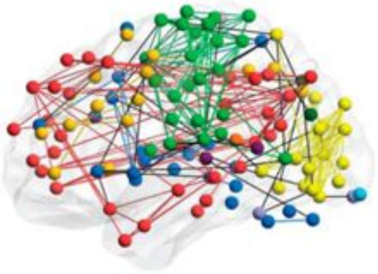
Networks are everywhere

And graphs are a great way to model them

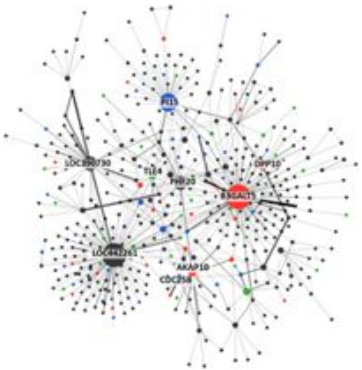
Social
Networks



Molecules

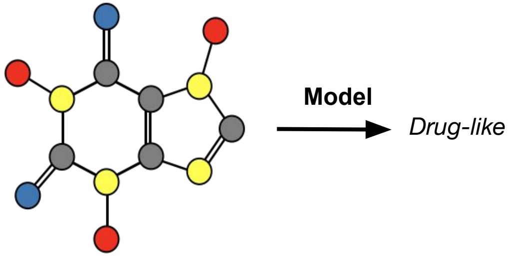


Functional
Networks

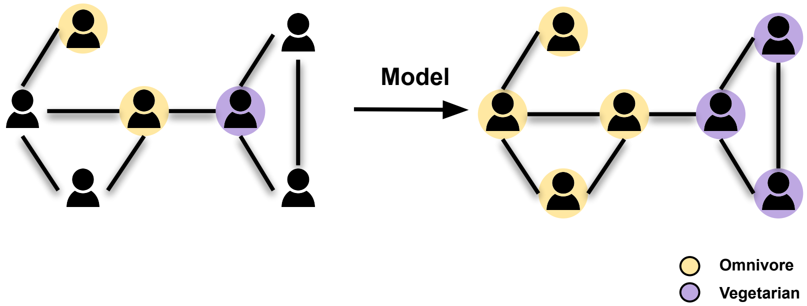


Interaction
Networks

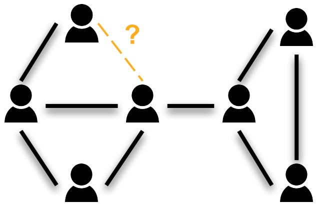
Tasks on Graphs



Graph
Classification



Node
Classification

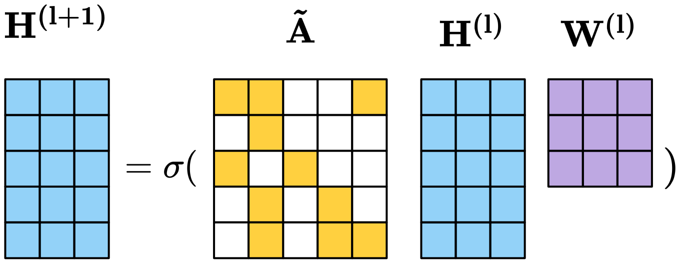


Link
Prediction

Graph Neural Networks (GNNs)

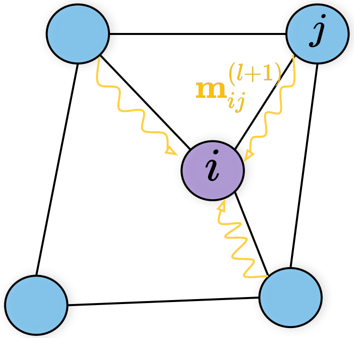
Convolutional GNN

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$
$$\mathbf{H}^{(1)} = \mathbf{X}$$



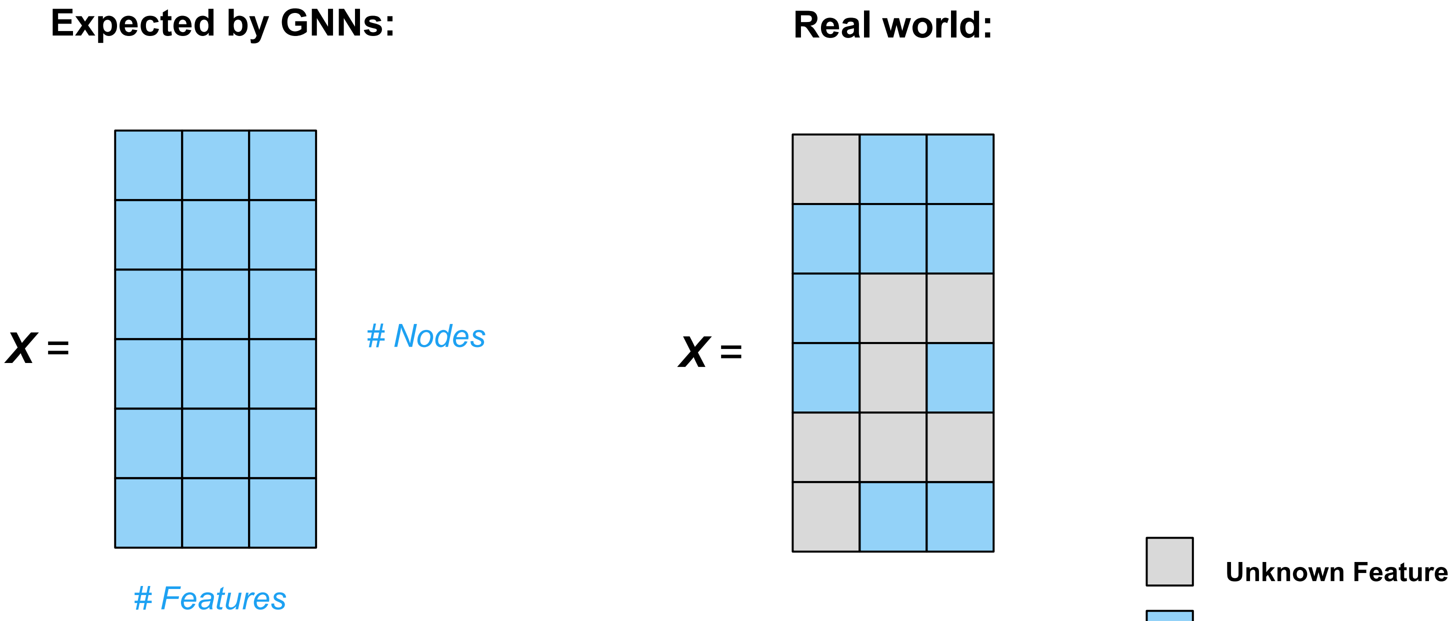
Message-Passing GNN

$$\mathbf{m}_{ij}^{(l+1)} = \mathrm{msg}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}),$$
$$\mathbf{h}_i^{(l+1)} = \sum_{j \in \mathcal{N}_i} f(\mathbf{m}_{ij}^{(l+1)}, \mathbf{h}_i^{(l)})$$
$$\mathbf{h}_i^{(1)} = \mathbf{x}_i$$



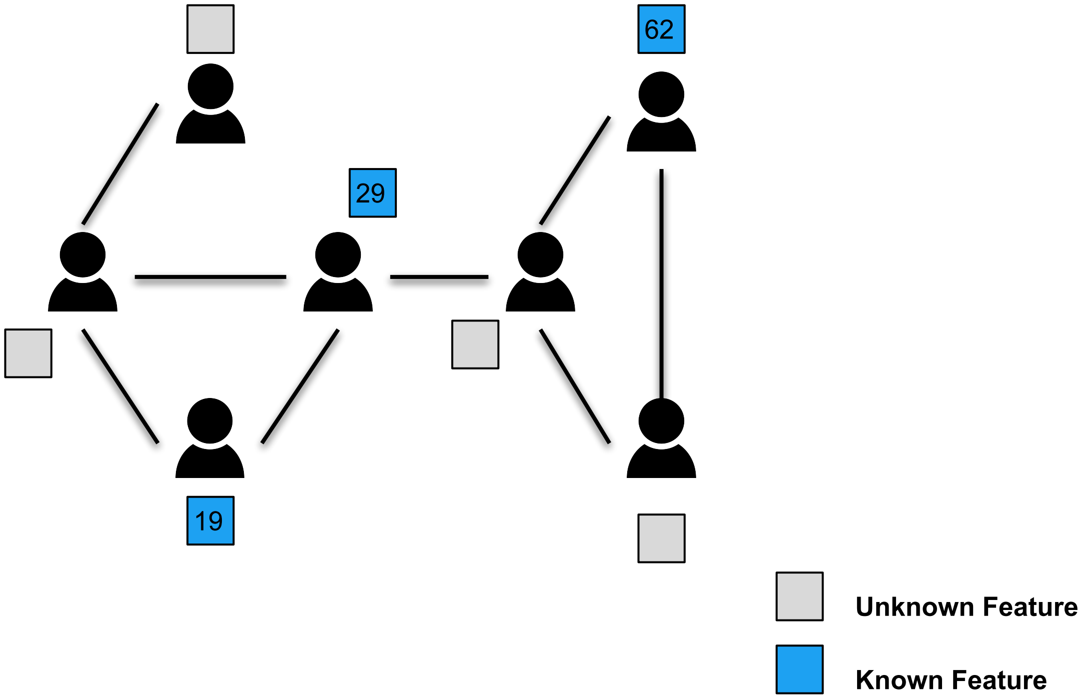
GNNs' Unspoken Assumption

They require a fully observed feature matrix



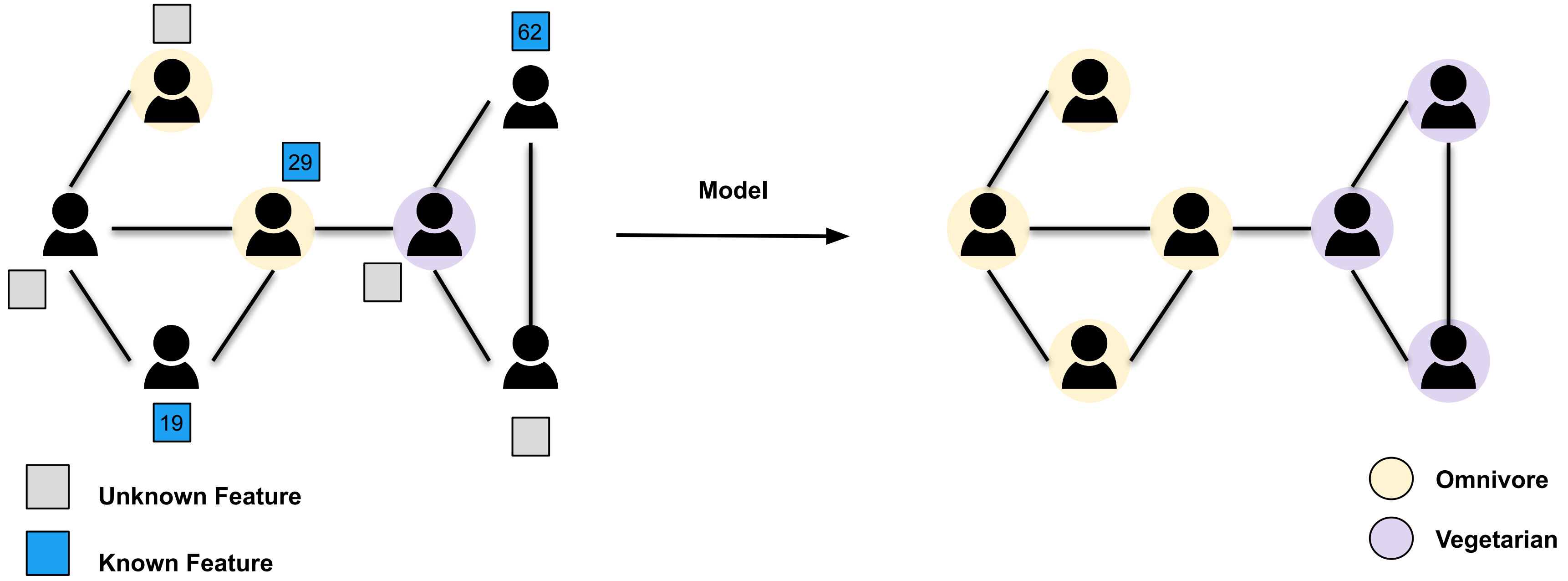
In the real world node features are often missing

Think of user demographics (eg. age) in a social network



Can we learn on graphs with missing node features?

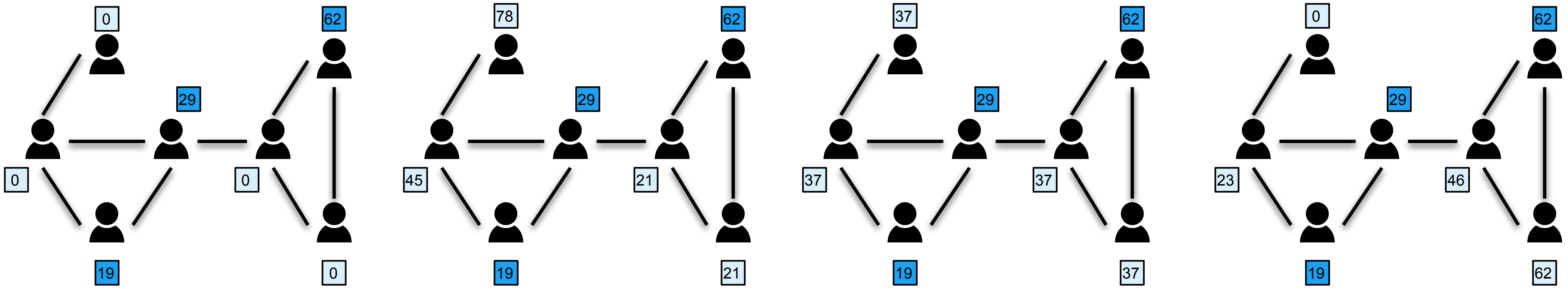
The goal is to solve a downstream task such as node classification



Learning with Missing Node Features

Simplest approach: impute then predict

Imputation step can be task-agnostic



Zero **Random** **Global Mean** **Neighbor Mean**

Previous Work

A largely unexplored problem

GCNMF [1]: Represents the missing data with a Gaussian Mixture Model and computes expected activation for first GCN layer

PaGNN [2]: Partial GCN-like message-passing which only propagates observed features in the first layer

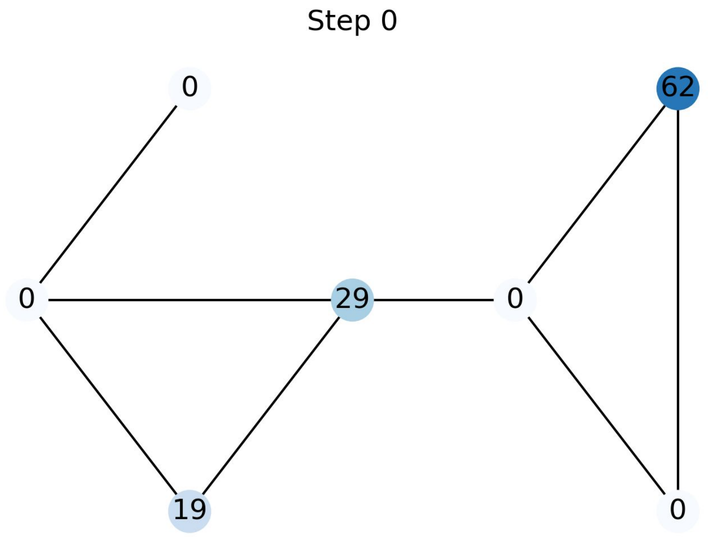
Problems:

- Suffer in regimes on high rates of missing features (>90%)
- Do not scale to large graphs

[1] Taguchi et al., 2020; [2] Jiang and Zhang, 2020

Our Idea: Reconstruction which promotes smoothness on the graph

Homophily assumption (measured through Dirichlet energy)



Some Notation

Node Features: $\mathbf{x} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_u \end{bmatrix} \in \mathbb{R}^n$

Known Features

Unknown Features

Adjacency Matrix: $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{kk} & \mathbf{A}_{ku} \\ \mathbf{A}_{uk} & \mathbf{A}_{uu} \end{bmatrix} \in \{0, 1\}^{n \times n}$

Subgraph between nodes with their feature missing

Degree Matrix: $\mathbf{D} \in \mathbb{R}^{n \times n}, \mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$

Normalized Adjacency: $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$

Laplacian Matrix: $\boldsymbol{\Delta} = \mathbf{I} - \tilde{\mathbf{A}}$

Our Idea: Reconstruction which promotes smoothness on the graph

Homophily assumption (measured through Dirichlet energy)

Dirichlet Energy

$$\ell(\mathbf{x}, G) = \frac{1}{2}\mathbf{x}^\top \boldsymbol{\Delta} \mathbf{x} = \frac{1}{2}\sum_{ij} \tilde{a}_{ij}(x_i - x_j)^2$$

$$\mathbf{x}_u^* = \underset{\mathbf{x}_u}{\operatorname{argmin}}\, \ell$$

$$\mathbf{x}_u^* = -\boldsymbol{\Delta}_{uu}^{-1}\boldsymbol{\Delta}_{ku}^\top \mathbf{x}_k k$$

Closed-form solution for missing features that minimizes the Dirichlet Energy

Scalable minimization with the gradient flow

We can minimize the Dirichlet Energy by doing diffusion on the graph.
Let's look at the unconstrained case first

Gradient of the Dirichlet Energy:

$$\nabla_{\mathbf{x}} \ell(\mathbf{x}, G) = \nabla_{\mathbf{x}} \frac{1}{2} \mathbf{x}^\top \mathbf{\Delta} \mathbf{x} = \mathbf{\Delta} \mathbf{x}$$

Gradient flow:

$$\dot{\mathbf{x}}(t) = -\nabla_{\mathbf{x}} \ell = -\mathbf{\Delta} \mathbf{x}(t)$$

Differential equation whose solution at t->∞ minimizes the Dirichlet Energy

Euler Method Discretization:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \mathbf{\Delta} \mathbf{x}^{(t)}$$
$$= (\mathbf{I} - \mathbf{\Delta}) \mathbf{x}^{(t)}$$
$$= \tilde{\mathbf{A}} \mathbf{x}^{(t)}$$

Solve the above equation by discretizing it

Minimizing the Dirichlet Energy amounts to repeatedly multiplying by normalized adjacency

Scalable minimization with the gradient flow

With boundary conditions (i.e. constraints on the known features)

Known features are left unchanged by diffusion

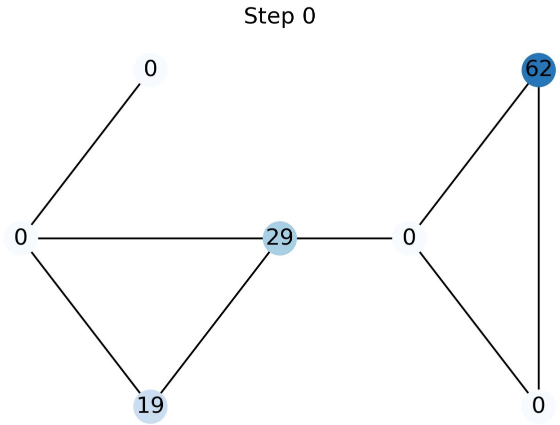
$$\mathbf{x}^{(t+1)} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \tilde{\boldsymbol{A}}_{uk} & \tilde{\boldsymbol{A}}_{uu} \end{bmatrix} \mathbf{x}^{(t)}$$

Diffusion from known features to unknown ones

Diffusion among unknown features

Feature Propagation Algorithm (FP)

Extremely simple and scalable

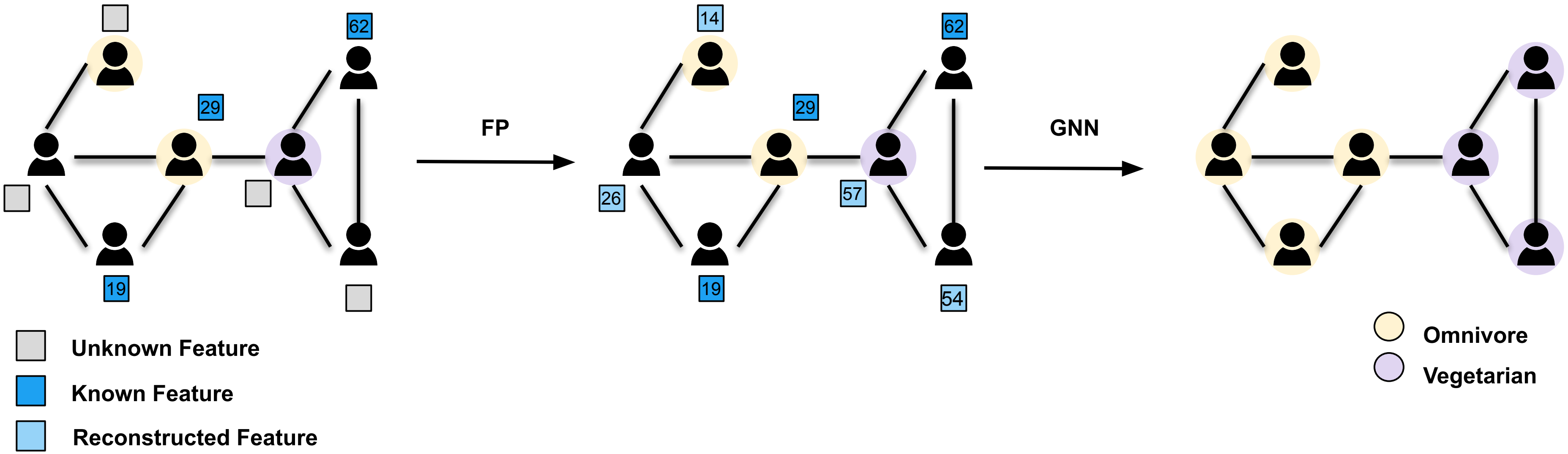


Algorithm 1 Feature Propagation

1: **Input:** feature vector $\mathbf{x}$, diffusion matrix $\tilde{\mathbf{A}}$
2: $\mathbf{y} \leftarrow \mathbf{x}$
3: **while** $\mathbf{x}$ has not converged **do**
4: $\quad \mathbf{x} \leftarrow \tilde{\mathbf{A}}\mathbf{x}$ ▷ Propagate features
5: $\quad \mathbf{x}_k \leftarrow \mathbf{y}_k$ ▷ Reset known features
6: **end while**

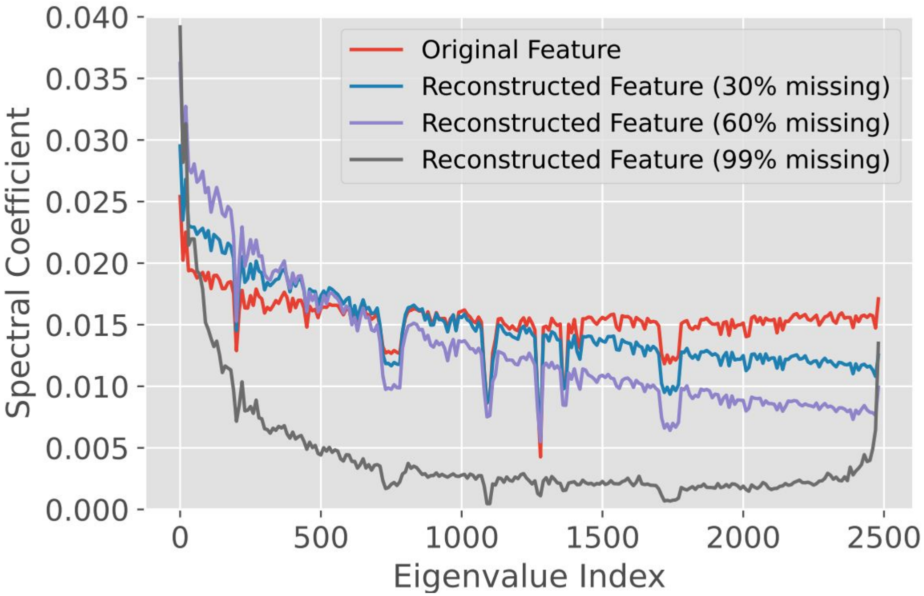
Feature Propagation Algorithm (FP)

Extremely simple and scalable



Intuition Behind FP

It acts as a low pass filter, similarly to most GNNs



Differences with Label Propagation

Algorithmically Similar, but:

Label Propagation:

- Propagates class labels (discrete)
- Prediction is obtained directly from propagating class labels
- Feature-agnostic

Feature Propagation:

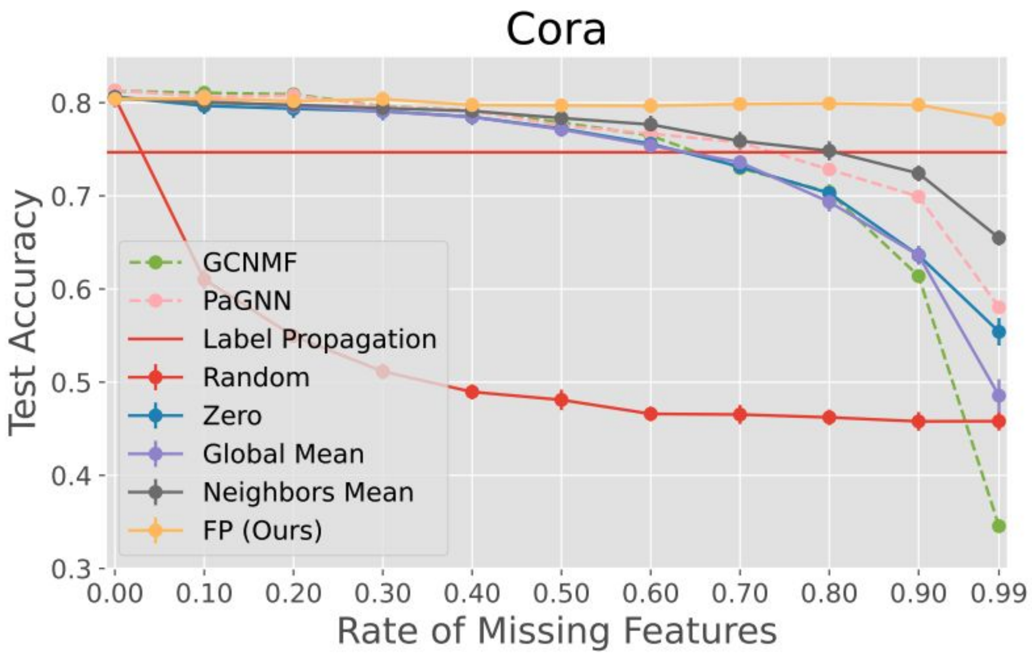
- Propagates features (continuous)
- Prediction is made by a GNN on top of the propagated features
- Uses features, and a low % of them being present is enough for good performance
- Effective solution for missing features problem

Experiments

How well does FP work?

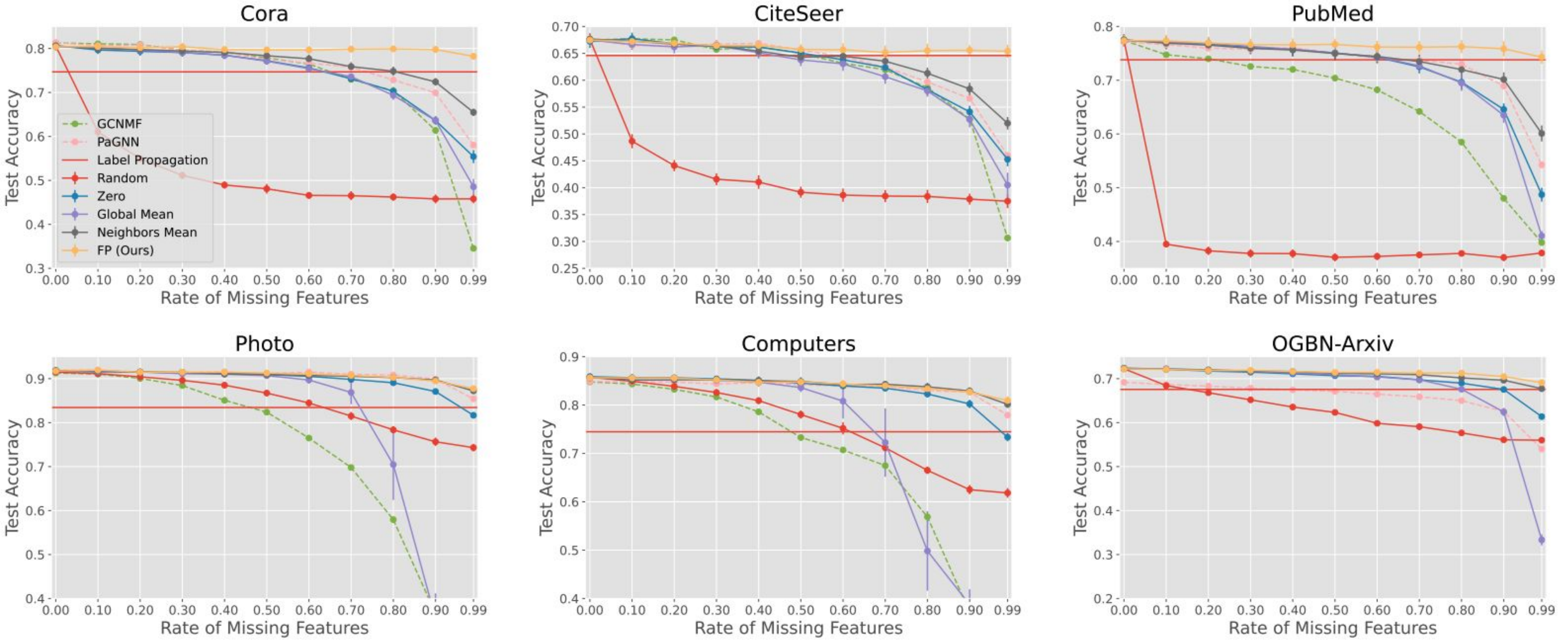
Node Classification Results

Accuracy as a function of the rate of missing features



Node Classification Results

We evaluated on six common benchmarks



Node Classification with 1% of Features

FP can withstand surprisingly high rates of missing features

Dataset	GCNMF	PaGNN	Label Propagation	FP (Ours)
Cora	34.54±2.07	58.03±0.57	74.68±0.36	**78.22**±0.32
CiteSeer	30.65±1.12	46.02±0.58	64.60±0.40	**65.40**±0.54
PubMed	39.80±0.25	54.25±0.70	73.81±0.56	**74.29**±0.55
Photo	29.64±2.78	85.41±0.28	83.45±0.94	**87.73**±0.27
Computers	30.74±1.95	77.91±0.33	74.48±0.61	**80.94**±0.37
OGBN-Arxiv	OOM	53.98±0.08	67.56±0.00	**69.09**±0.06
OGBN-Products	OOM	OOM	74.42±0.00	**74.94**±0.07

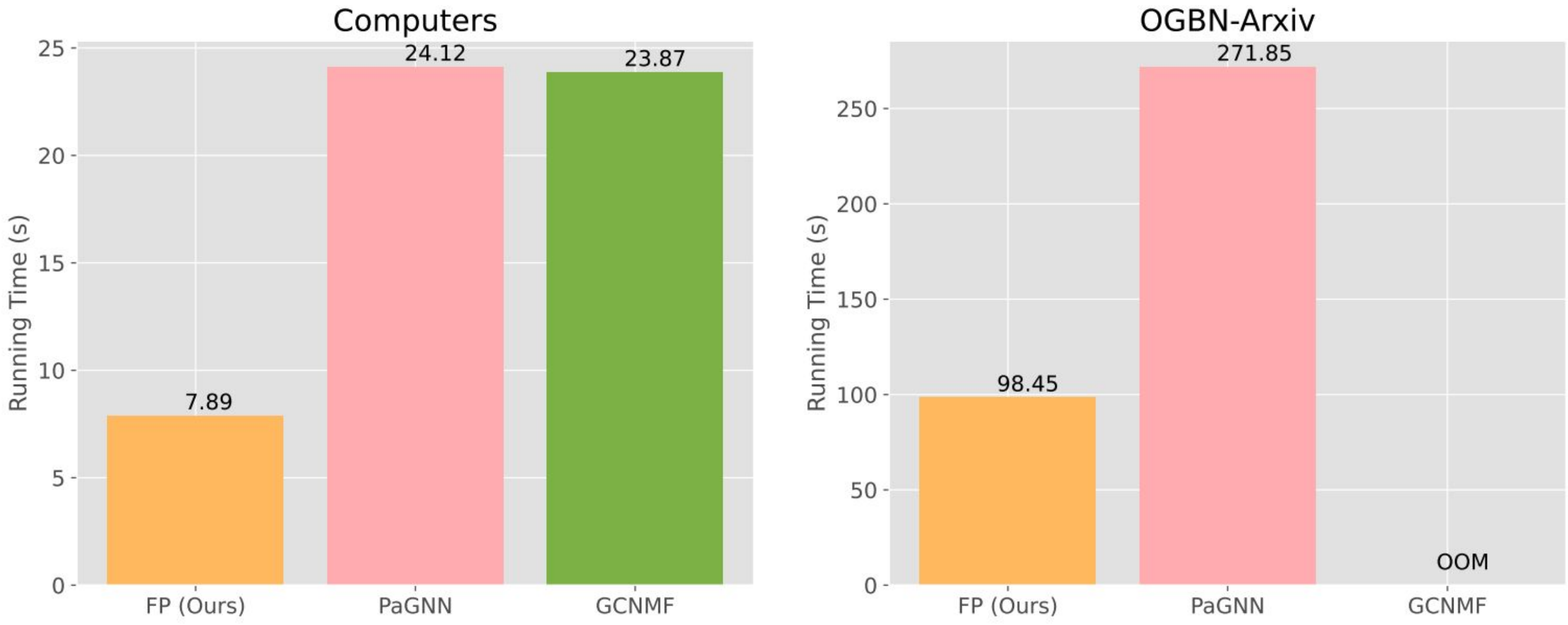
Zooming in to FP

FP only incurs in an average drop of ~4% of relative accuracy when 99% of the features are missing

Dataset	Full Features	50.0% Missing	90.0% Missing	99.0% Missing
Cora	80.39%	79.70%(-0.86%)	79.77%(-0.77%)	78.22%(-2.70%)
CiteSeer	67.48%	65.74%(-2.57%)	65.57%(-2.82%)	65.40%(-3.08%)
PubMed	77.36%	76.68%(-0.89%)	75.85%(-1.96%)	74.29%(-3.97%)
Photo	91.73%	91.29%(-0.48%)	89.48%(-2.46%)	87.73%(-4.36%)
Computers	85.65%	84.77%(-1.04%)	82.71%(-3.43%)	80.94%(-5.51%)
OGBN-Arxiv	72.22%	71.42%(-1.10%)	70.47%(-2.43%)	69.09%(-4.33%)
OGBN-Products	78.70%	77.16%(-1.96%)	75.94%(-3.51%)	74.94%(-4.78%)
Average	79.08%	78.11%(-1.27%)	77.11%(-2.48%)	75.80%(-4.10%)

FP is Fast and Scalable

FP Reconstruction + GNN Training



FP is Fast and Scalable

FP Reconstruction Only

	# Nodes	*# Edges*	*Python*	*BigQuery*
OGBN-Products	~2.5M	~123M	**~10s** (1 GPU)	/
Twitter Internal	~800M	~10B	**~2h** (1 large CPU)	**~45m**

When does FP work?

Spoiler: it does not work well on heterophilic graphs

Future Directions

Some open questions

- End-to-end **learnable diffusion**
- Feature **channel mixing**
- Extension to **heterophilic data**

Conclusions

What you should take away from today

- Missing node features is a **widespread problem**
- **Theoretically motivated approach**
- **Robust** to high rates of missing features (>90%)
- **Scalable and fast**
- **Limitations**: It requires the graph to be homophilous

Questions?

@emaros96
www.emanuelerossi.co.uk