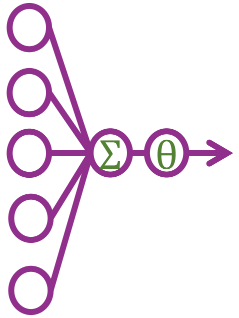
Inter-University Research Institute Corporation
Research Organization of Information and Systems
National Institute of Informatics

From Basics to GNNs

An Overview of Graph Neural Networks

Presented by PÂRȚACHI Profir-Petru
profir@nii.ac.jp
Supervised by SUGIYAMA Mahito
mahito@nii.ac.jp

Single-layered Perceptron



The basic formulation of a single-layered perceptron is

$$f(\boldsymbol{x}) = \theta(\boldsymbol{wx} + \boldsymbol{b})$$

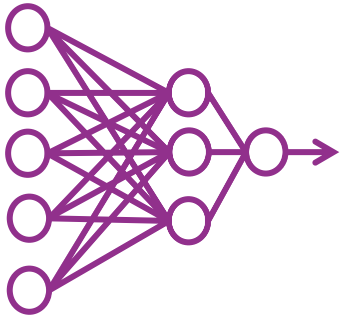
Where $\boldsymbol{x}$ is the input, $\boldsymbol{w}$ is the learnt weights and $\boldsymbol{b}$ is the bias.

θ is a non-linearity.

In the original formulation of the perceptron, θ is a step-function.

Representational Power: can learn any linearly separable collection of **N** binary labels provided that the size of the input $\boldsymbol{x}$ is greater than **N**.

Multi-layered Perceptron (MLP)



Starting from the multi-classification perceptron, chain perceptrons layer-wise.

Adjust the weights using backpropagation.

Historically MLP tackled the XOR learning problem which the single-layered perceptron cannot handle.

It is often a building block in modern Graph Neural Networks.

Fixed Assumptions in MLP

There are no intra-layer connections,
instead all connections are inter-layer.

The aggregation of the signal happens via summation.

Each neuron holds only a scalar value.

Each update only depends on the neurons in the following layer.

Relaxing the Assumptions

Allow arbitrary edges.

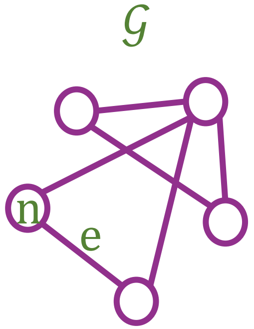
Allow signal aggregation to happen via other methods:

maximum, minimum, summation, median, average.

Allow each neuron/node to have a vector representation.

Consider the current state of a node when updating the representation.

Graph Prediction Tasks

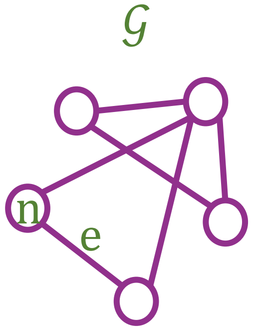


- Node level
  - Community detection, Node classification, Node representation learning.
- Edge level
  - Relationship learning, operator learning:

    e.g. collaboration/co-author prediction, scene narrative graph
- Graph level
  - Graph classification:

    e.g. drug prediction/discovery

A Simple Graph Neural Network



Consider a graph $\mathcal{G}(\mathcal{N},\mathcal{E})$, where at each node $\mathrm{n} \in \mathcal{N}$ and each edge e $\in \mathcal{E}$, we have an associated vector v_n, respectively v_e, representing the node (respectively edge).

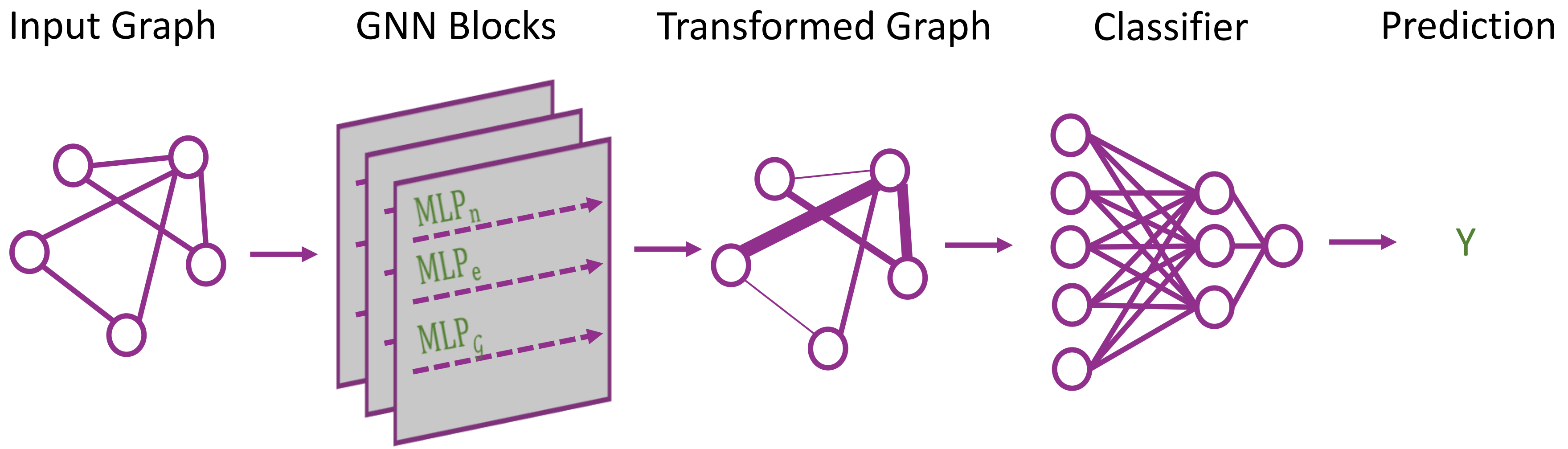
A simple Graph Neural Network, takes the graph $\mathcal{G}$ and associates with each node/edge an MLP whose output is then used for node/edge/graph classification tasks.

$$\mathrm{MLP_n} := \mathrm{v_n} \rightarrow \mathrm{f_n}$$

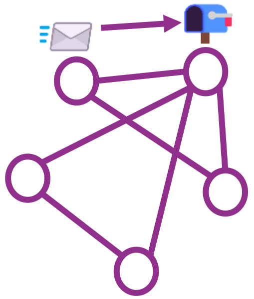
$$\mathrm{MLP_e} := \mathrm{v_e} \rightarrow \mathrm{f_e}$$

$$\mathrm{MLP}_{\mathcal{G}} := (\mathrm{f_n},\mathrm{f_e}) \rightarrow \mathrm{f}_{\mathcal{G}}$$

End-to-End Example of a GNN



Message Passing



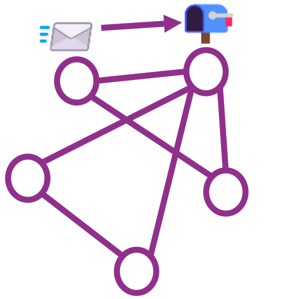
Let's add graph structure into the learning algorithm.

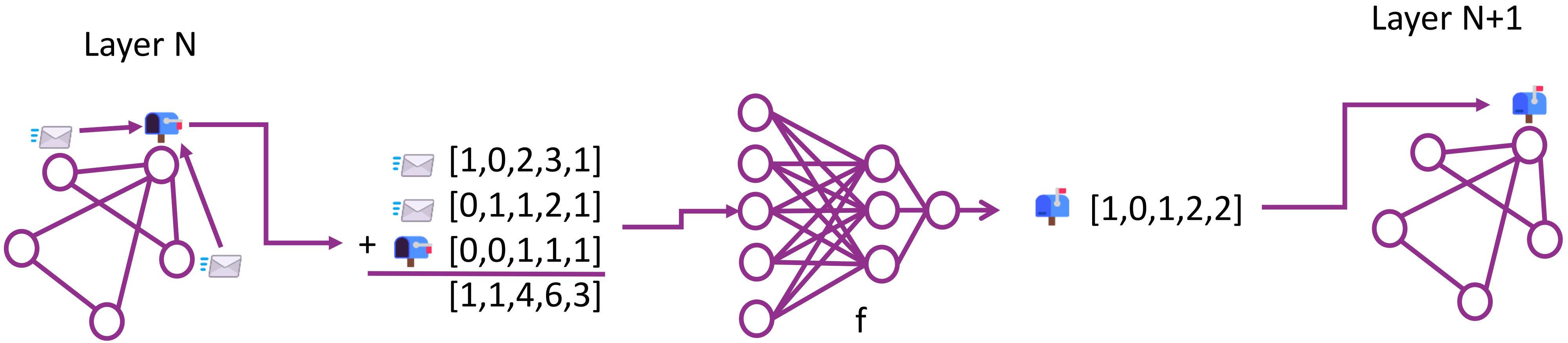
We do so in the following steps:

1. For each node $n \in \mathcal{G}$ where $N(n)$ are the nodes neighbouring n, form the set $U = \{v_{n'} \mid n' \in N(n)\}$;
2. Apply an aggregation function that reduces the set U to a single vector v'_n;
3. Apply an update function $f: (v_n, v'_n) \rightarrow v_n$.

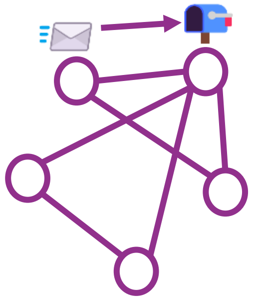
(For edges, consider the dual graph and apply the above)

Message Passing





Representational Power

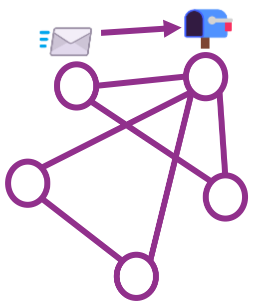


Theorem 1 †(R. Sato, 2020): For any message passing GNN and for any graphs $\mathcal{G}$ and $\mathcal{H}$, if the 1-WL algorithm outputs that $\mathcal{G}$ and $\mathcal{H}$ are "possibly isomorphic", the vector representations associated with each node and edge in $\mathcal{G}$ and $\mathcal{H}$ are the same.

Note: k-WL first "gathers" the labels of all nodes in a neighbourhood, and then hashes this set to obtain a new label, it outputs "possibly isomorphic" if after k steps, if the set of all neighbourhood label sets are the same.

†R. Sato, A survey on the expressive power of graph neural networks (2020). arXiv: 2003.04078. Uses random features for positional encoding.

Short-comings and Current Research

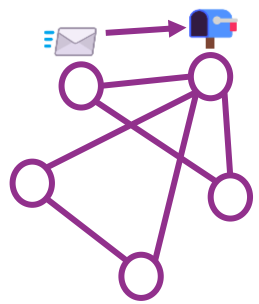


Focusing primarily on Message Passing GNNs, there are two main short-comings associated:

Oversmoothing: As the number of GNN layers is increased, the vector representation of nodes and edges tends to a uniform distribution.

Oversquashing: Under certain (often common) graph configurations, information loss occurs as a single node has to encode updates from an exponentially large neighbourhood.

Common Methods to Alleviate Oversmoothing and Oversquashing



For oversmoothing:

- Regularise and add noise to node level features
- Residual connections
- Reduce model depth

For oversquashing:

- Increase model depth
- Master node/Fully connected graph
- Graph rewiring
- Hierarchical message passing/Overlay networks

Fin

Questions?