Graph Neural Networks in Biology: Lecture 3

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 29, 2025

Contents Today

- Reminder: Simple Graph Neural Networks
- Message Passing

Graph Neural Networks: Definition

Graph Neural Networks: Definition

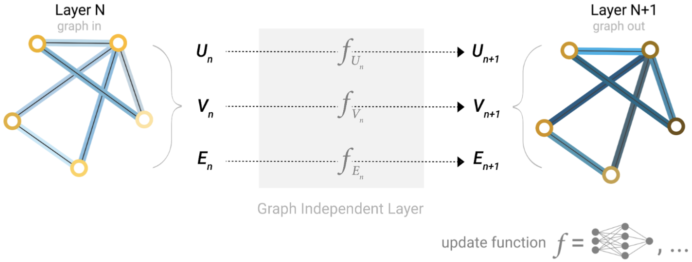
Definition [Graph Neural Network]:
A *graph neural network (GNN)* is an

- optimizable transformation on
- all attributes of the graph (nodes, eges, global) that
- preserves graph symmetries (permutation invariances)

- GNN's adopt a "graph-in, graph-out" architecture:
  - Graph loaded with information accepted as input
  - Embeddings are progressively transformed
  - Connectivity of input graph never changed

Simple Graph Neural Networks

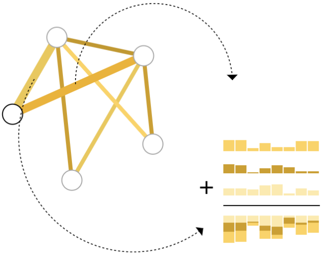
SIMPLE GNN I



U_n, V_n, E_n reflect global, vertex, edge information.
From `https://distill.pub/2021/gnn-intro/`

- Initial embeddings: U_0, V_0, E_0
- $U_n, V_n, E_n, n \geq 0$ iteratively updated to $U_{n+1}, V_{n+1}, E_{n+1}$ ...
- ... using multilayer perceptions (MLP's) $f_{U_n}, f_{V_n}, f_{E_n}$ until ...
- ... final layer is reached, where final embeddings are computed.

Predictions by Pooling

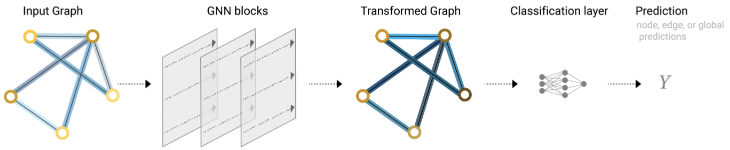


From `https://distill.pub/2021/gnn-intro/`

- May not always be so simple. For example:
  - Would like to raise predictions about nodes
  - But only edge embeddings available
- *Solution:* Aggregate (adjacent) edge embeddings using pooling function

$$\rho_{E_n \to V_n}$$

Predictions by Pooling II



GNN: End-to-end predcition task

From `https://distill.pub/2021/gnn-intro/`

- Classification layer comprises pooling as well, if necessary
- *Remark:* Classification model can be any differentiable model
  - Models other than MLP's conceivable

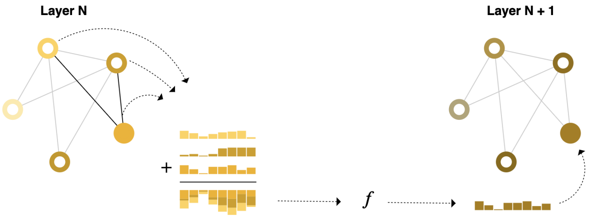
Message Passing

Message Passing: Motivation

- Simple GNN’s so far presented
  - do not pool within the GNN layer
  - have learned embeddings unaware of graph connectivity
- *Goal:* Neighboring nodes and edges
  - exchange information
  - influence each other’s updated embeddings
- *Solution:* Message passing

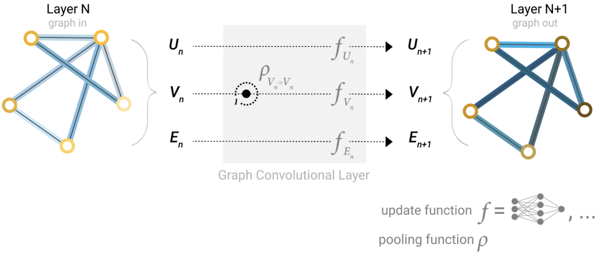
Message Passing: Protocol

1. Each node: gather all embeddings (= *messages*) of neighboring nodes
2. Aggregate all messages using an aggregation function
3. Pooled messages passed through update function (e.g. learned NN)



Message passing: Aggregating information from neighboring nodes
From `https://distill.pub/2021/gnn-intro/`

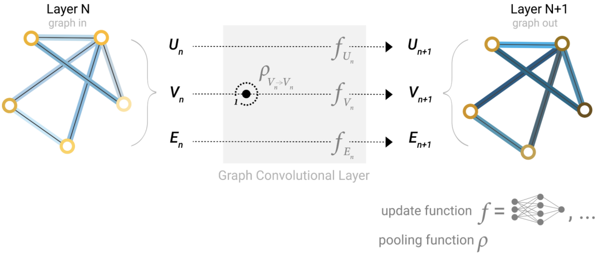
Message Passing and Convolution I



Message passing as convolution on graphs
From `https://distill.pub/2021/gnn-intro/`

- Message passing and convolution are similar in spirit
- *Commonality:* Process element's neighbors to update element
  - *Graphs:* Elements are nodes
  - *Images:* Elements are pixels

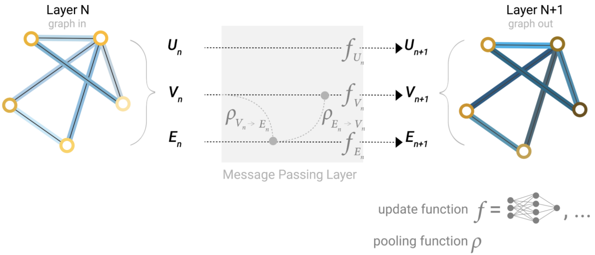
Message Passing and Convolution II



Message passing as convolution on graphs
From `https://distill.pub/2021/gnn-intro/`

- Message passing and convolution are similar in spirit
- *Difference:*
  - *Graphs:* Number of neighbors varies per node
  - *Images:* Number of neighbors constant per pixel

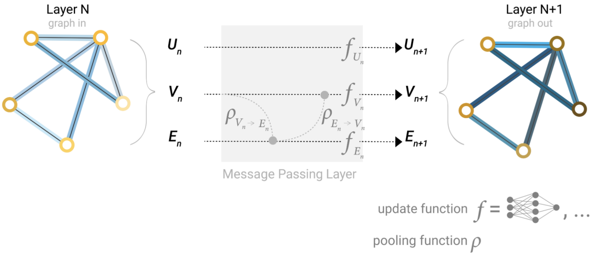
Pooling Within Layers I



Passing messages from edges to nodes within layer
From `https://distill.pub/2021/gnn-intro/`

- *Situation:* Want node from edge information
- *Solution:*
  - Pool information of neighboring edges and transfer to node: $\rho_{E_n \to V_n}$
  - After first iteration: add node information, and transfer pooled information from nodes to edges: $\rho_{V_n \to E_n}$

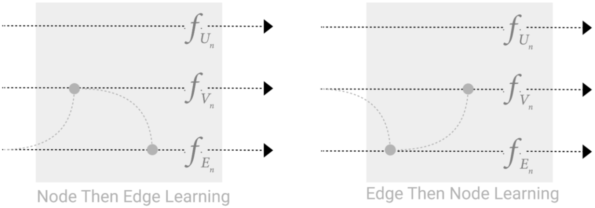
Pooling Within Layers II



Passing messages from edges to nodes within layer
From `https://distill.pub/2021/gnn-intro/`

- *Issue:* Node and edge information may differ in size
- *Solution:* Linear map transforms node into edge information
- And vice versa. Other maps than linear maps conceivable

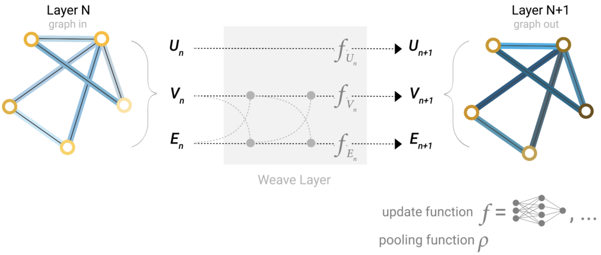
Pooling Within Layers III



Message passing: different modes
From `https://distill.pub/2021/gnn-intro/`

- *Left:* Learning edge information from node information
- *Right:* Learning node information from edge information

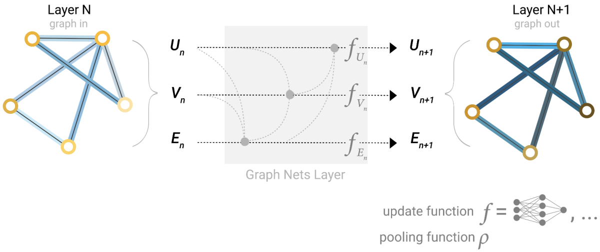
Pooling Within Layers IV



Weave layer: learning node information from edges and learning edge information from nodes
From `https://distill.pub/2021/gnn-intro/`

- f_{V_n} processes node information from edge information and node itself
- f_{E_n} processes edge information from node information and edge itself

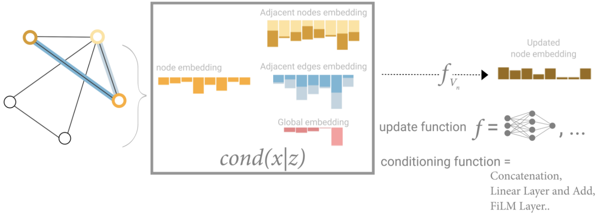
Pooling Within Layers V



Global information: aggregate from nodes and edges
From `https://distill.pub/2021/gnn-intro/`

- *Issue:* After k layers, nodes can reach k-neighborhoods at most
- *Solution:* Consider *master node* or *global context vector*
- Update global context vector by pooling node and/or edge information

Pooling Within Layers VI



Node information from pooling node, edge and global information
From `https://distill.pub/2021/gnn-intro/`

- *Situation:* Want node information based on all information
- *Issue:* Different informations differ in size
- *Solution:* Condition node information x on other information z
- Use *conditioning function:* concatenation, linear etc.

Message Passing and Random Walks

- Let $n := |V|$ be the number of nodes of a graph (V, E)
- Let $A \in \{0, 1\}^{n \times n}$ be its adjacency matrix
- Let m be the length of node information vectors
- Let $X \in \mathbb{R}^{n \times m}$ be the node feature matrix
  - Rows in X are m-dimensional information vectors of nodes

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \ldots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^{n} A_{ik}X_{kj}$$

MESSAGE PASSING AND RANDOM WALKS

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \ldots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^{n} A_{ik}X_{kj}$$

Interpretation:

- Each row B_i reflects a new information vector for node v_i
- B_i again has dimension m
- Each B_{ij} is the aggregation of j-th entries of information vectors of neighbors of v_i
  - ☞ Note that $A_{ik} = 1$ if and only if v_i and v_k are neighbors

UNIVERSITÄT BIELEFELD

Message Passing and Random Walks

Consider

$$B = AX$$

We obtain

$$B_{ij} = A_{i1}X_{1j} + \ldots + A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^{n} A_{ik}X_{kj}$$

Interpretation:

- Replacing A with A^K yields aggregation of information vectors of K-neighbors
  - ☞ $A^K_{ik} = 1$ iff (sic!) v_i and v_k can be connected by path of length K
- This relates to random walks on the graph
  - ☞ Recall the random walk mechanism for computing PageRank

Graph Attention Networks

Motivation:

- When aggregating one would like to consider weighted sums

$$B_{ij} = w_{ij,1}A_{i1}X_{1j} + \ldots + w_{ij,n}A_{in}X_{nj} = \sum_{\substack{k=1 \\ A_{ik}>0}}^{n} w_{ij,k}A_{ik}X_{kj}$$

  ☞ Some neighbors are more important than others

- *Challenge:* How to compute weights in permutation invariant way?

Graph Attention Networks

Solution:

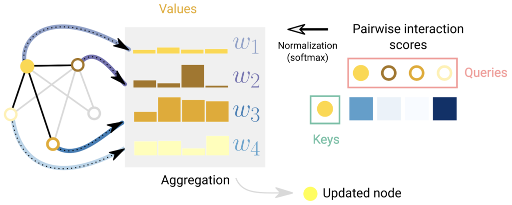
- *Solution:* Base weights on pairs of nodes alone, so

$$w_{ij,k} = f(v_i, v_k)_j$$

  where

  - $f(v_i, v_k)$ is m-dimensional vector that
  - depends on information of nodes v_i and v_k alone
    ☞ Renumbering nodes does not matter

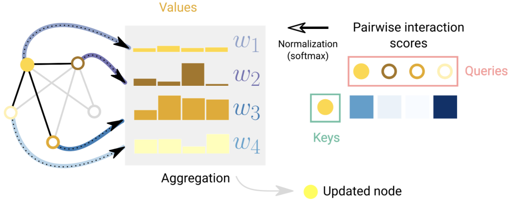
Graph Attention Networks



Graph attention network: mechanism
From `https://distill.pub/2021/gnn-intro/`

- *Attention Networks:* Compute *value* from comparing *key* and *query*
  - *Original motivation:* Compute strength of dependency between words in (or across) sentences
  - *Original application:* Used in language translation for example

Graph Attention Networks



Graph attention network: mechanism

From `https://distill.pub/2021/gnn-intro/`

- *Here:* Compare information vectors of two nodes
    - One node is query, other node is key, weight is value
    - *Example:*

$$f(v_i, v_k) = \langle v_i, v_k \rangle$$

evaluates as scalar product of information vectors of v_i and v_k

Outlook

- Convolution on Graphs
- The Graph Laplacian
- Polynomial Filters

Thanks for your attention!