Lecture 17: Graph Neural Networks
CS480/680 Intro to Machine Learning

2023-3-14

Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

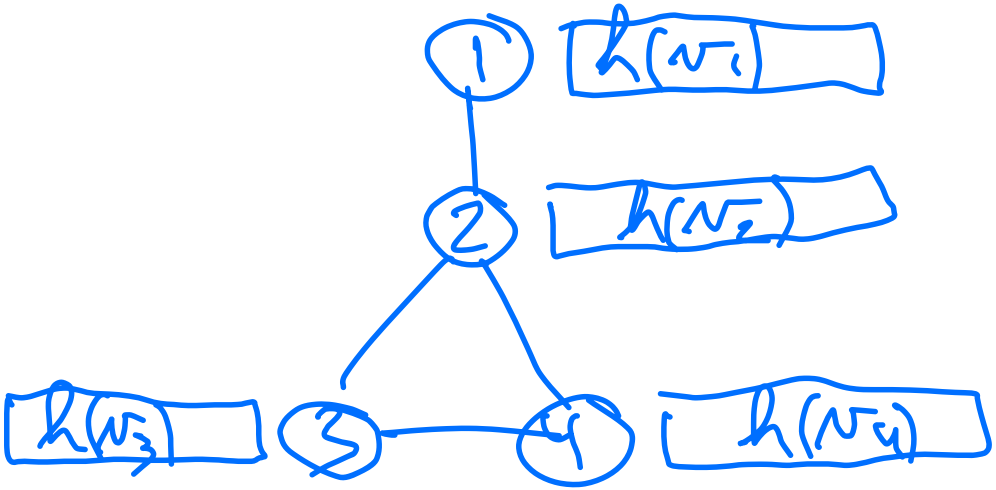
Graph Neural Networks

- Generalization of
  - Convolutional neural networks
  - Transformers

- Applications:
  - Recommender systems
  - Social networks, financial networks
  - Biology, Chemistry and Physics (proteins, moelcules)
  - Combinatorial optimization

Embeddings

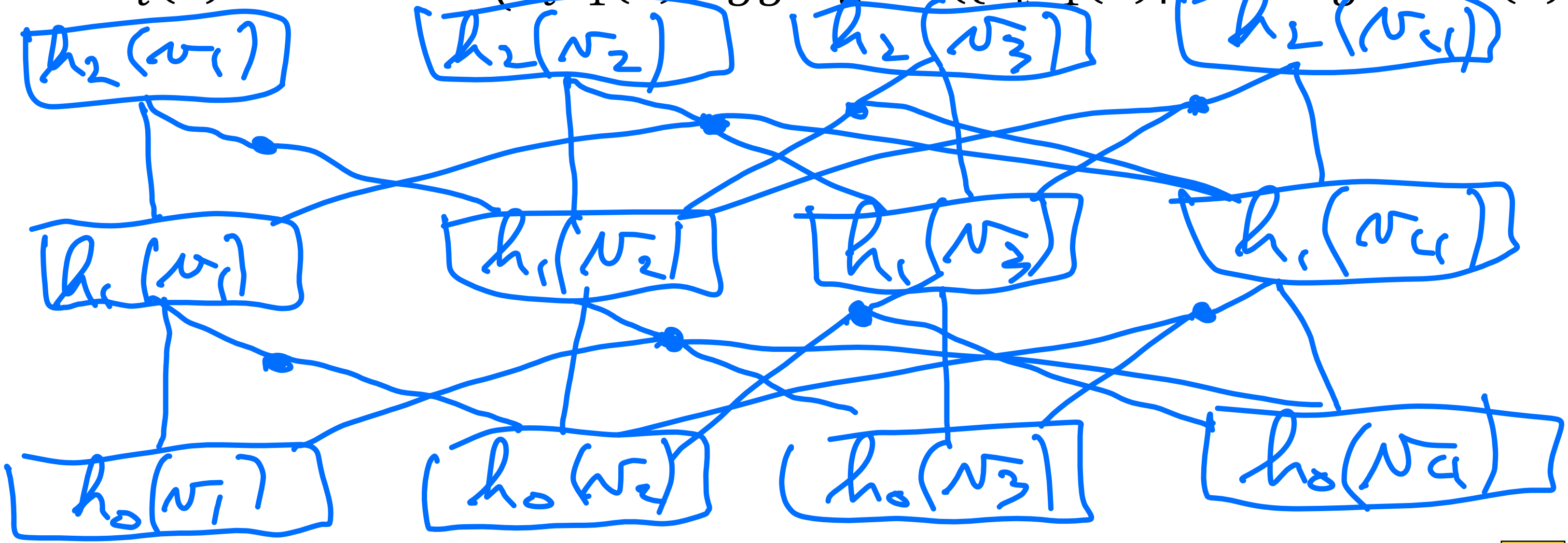
- Neural network that computes embeddings for nodes (and edges) in a graph by passing messages along the edges of the graph
- The embedding of a node captures information about its context



Message Passing

- Graph: Nodes ($V = \{v\}$) and edges ($E = \{e\}$)
  - Initial node embedding: $h_0(v)$
  - Message passing:

$$h_t(v) \leftarrow combine\big(h_{t-1}(v), aggregate(\{h_{t-1}(u) | u \in neighbors(v)\})\big)$$

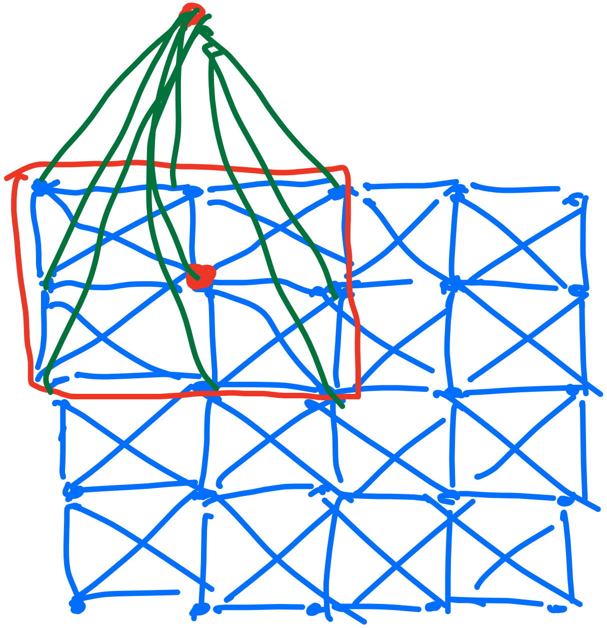


UNIVERSITY OF WATERLOO

Convolutional Neural Network

- CNN that preserves size is a special type of GNN
  - Initial node embedding: $h_0(v)$ = pixel intensities

3x3 filter



Convolutional Neural Network

- CNN that preserves size is a special type of GNN
  - Initial node embedding: $h_0(v) =$ pixel intensities
  - $n \times n$ convolutional layer (stride=1, padding=same):

$$m_{ij} \leftarrow aggregate\left(\left\{h_{t-1}\left(v_{i'j'}\right) \middle| i' \neq i \text{ or } j' \neq j\right\}\right) = \sum_{i' \neq i \text{ or } j' \neq j} w_{i'j'} h_{t-1}(v_{i'j'})$$

$$h_t\left(v_{ij}\right) \leftarrow combine\left(h_{t-1}\left(v_{ij}\right), m_{ij}\right) = \sigma\left(w_{ij} h_{t-1}\left(v_{ij}\right) + m_{ij}\right)$$

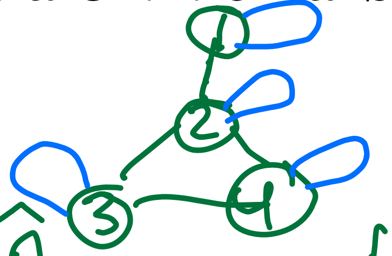
  - $n \times n$ pooling layer (stride=1, padding=same):

$$m_{ij} \leftarrow aggregate\left(\left\{h_{t-1}\left(v_{i'j'}\right) \middle| i' \neq i \text{ or } j' \neq j\right\}\right) = \max_{i' \neq i \text{ or } j' \neq j} h_{t-1}(v_{i'j'})$$

$$h_t\left(v_{ij}\right) \leftarrow combine\left(h_{t-1}\left(v_{ij}\right), m_{ij}\right) = \max\left\{h_{t-1}(v_{ij}), m_{ij}\right\}$$

Graph Convolutional Neural Network

- GNN that does the same operations as a CNN on arbitrary graphs
  - Initial node embedding: $h_0(v)$
  - Message passing: $H_t \leftarrow \sigma\left(\hat{D}^{-1}\hat{A}H_{t-1}W\right)$



$$\overset{H_t}{\begin{bmatrix} h_t(v_1) \\ h_t(v_2) \\ h_t(v_3) \\ h_t(v_4) \end{bmatrix}} \leftarrow \sigma\left(\overset{\hat{D}^{-1}}{\begin{bmatrix} 1/2 & & & 0 \\ & 1/4 & & \\ & & 1/3 & \\ 0 & & & 1/3 \end{bmatrix}} \overset{\hat{A}}{\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{array}} \overset{H_{t-1}}{\begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix}} \overset{W}{\begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}} \right)$$

inverse of the degree of each node

adjacency matrix

weight matrix

UNIVERSITY OF WATERLOO

Transformer

- Transformer is a special type of fully connected GNN
  - Initial node embedding: $h_0(v)$ = ~~pixel intensities~~
  - Message passing:

$$m_v \leftarrow aggregate(\{h_{t-1}(u) | u \in neighbors(v)\}) = \sum_u a_{vu} W_V h_{t-1}(u)$$

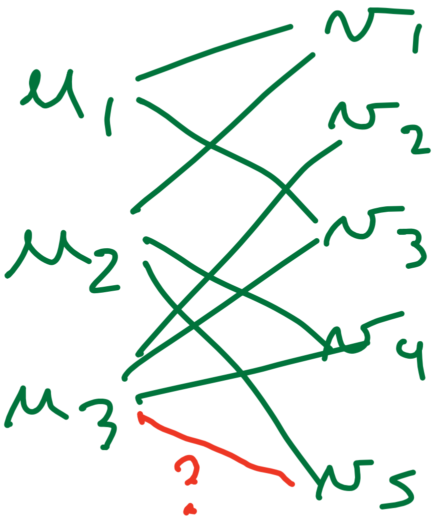
$$h_t(v) \leftarrow combine(h_{t-1}(v), m_v) = norm\left(ff\left(norm(a_{vv} W_V h_{t-1}(v) + m_v)\right)\right)$$

where a_{vu} is the attention weight of u with respect to v

$$a_{vu} = \frac{\exp\left(sim\left(W_Q h(v), W_K h(u)\right)\right)}{\sum_{u'} \exp\left(sim\left(W_Q h(v), W_K h(u')\right)\right)}$$

Recommender System

- Movie recommendation (edge completion)
- Bipartite graph:
  - Nodes: users and movies
  - Edges: e_{uv} = user u watched movie v



Recommender System

- Movie recommendation (edge completion)
- Bipartite graph:
  - Nodes: users and movies
  - Edges: e_{uv} = user u watched movie v
- Messages: $h_t(v) \leftarrow comb_\phi\left(h_{t-1}(v), agg_\theta(\{h_{t-1}(u) | u \in nb(v)\})\right)$
  - Example: $h_t(v) \leftarrow \sigma\left(\phi h_{t-1}(v) + \sum_u \theta\, h_{t-1}(u)\right)$
- Edge prediction: $P(e_{uv}) = f_W(u, v)$
  - Example: $P(e_{uv}) = \sigma\left(h(u)^T W h(v)\right)$
- Train: $\max_{W,\theta,\phi} \sum_{e_{uv} \in E} \log P(e_{uv})$

Boolean Satisfiability $\rightarrow u_1$

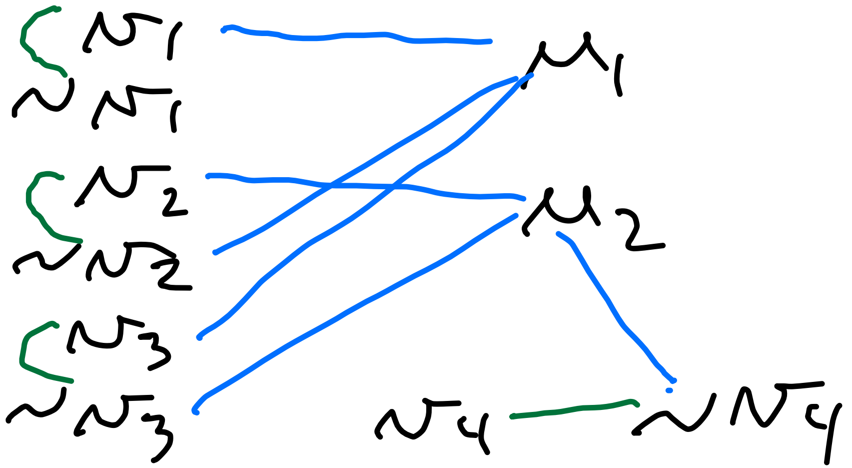
- Example: $\overbrace{(v_1 \vee \sim v_2 \vee v_3)}^{\text{clause}} \wedge \overbrace{(v_2 \vee \sim v_3 \vee \sim v_4)}^{\text{clause}} \rightarrow u_2$

  $v_2 \rightarrow$ positive literal

  $\sim v_4 \rightarrow$ negated literal

- Graph:
  - Nodes: literals $V = \{v\}$ and clauses $U = \{u\}$
  - Edges: e_{vu} = literal v appears in clause u

    $e_{v\sim v}$ = special edge from literal v to its complement $\sim v$



Boolean Satisfiability

- Example: $(v_1 \vee \sim v_2 \vee v_3) \wedge (v_2 \vee \sim v_3 \vee \sim v_4)$
- Graph:
  - Nodes: literals $V = \{v\}$ and clauses $U = \{u\}$
  - Edges: e_{vu} = literal v appears in clause u
    $e_{v\sim v}$ = special edge from literal v to its complement $\sim v$
- Messages: $h_t(v) \leftarrow comb_\phi\left(h_{t-1}(v), agg_\theta(\{h_{t-1}(u) | u \in nb(v)\})\right)$
  - Example: $h_t(v) \leftarrow \sigma\left(\phi h_{t-1}(v) + \sum_u \theta\, h_{t-1}(u)\right)$
- Clause classification: $P(u) = f_w(u)$
  - Example: $P(u) = \sigma(w^T h(u))$
- Train:
  - Satisfiable: $\max_{w,\theta,\phi} \sum_{u \in posClauses} \log P(u)$
  - Unsatisfiable: $\min\limits_{w,\theta,\phi} \sum_{u \in posClauses} \log P(u)$