Position-aware Graph Neural Networks

[ICML 2019 oral]

Seungryong Yoo
Seoul National University

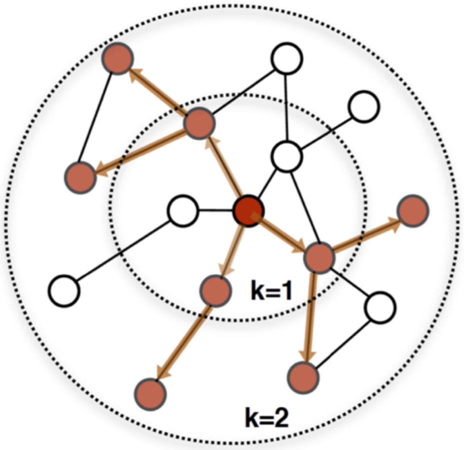
Position-aware Graph Neural Networks

Motivation

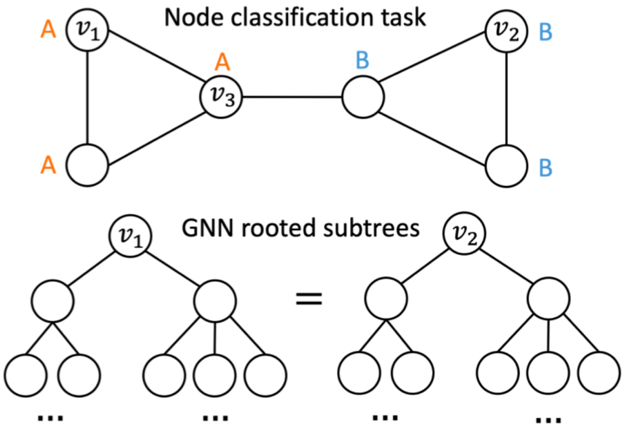
- **Current GNN-based node embedding methods only focus on local neighborhood structure**

  (e.g. q - hop neighbor)

- **Without node features, GNN cannot distinguish between two nodes at different location, but having the same local structure**



from [GraphSAGE 2017]



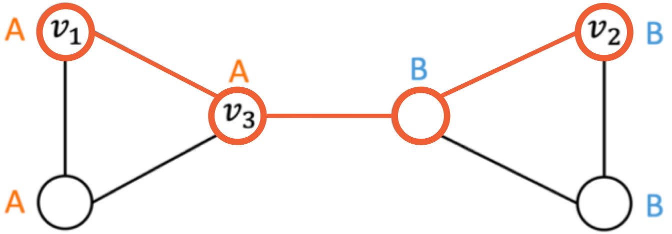
Position-aware Graph Neural Networks

Why node position information is important?

- How can we decide node position?
  → compare distance from common reference

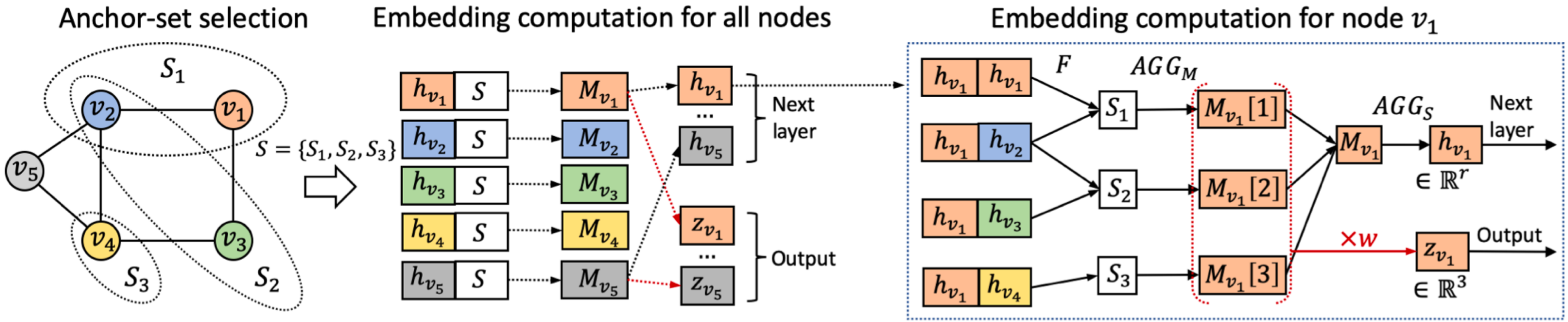
- d_{sp} : shortest path distance between two nodes
  $d_{sp}(v_1, v_3) = 1,\ d_{sp}(v_2, v_3) = 2$
  → now v_1 and v_2 are distinguishable

- call these reference as anchor-set



Position-aware Graph Neural Networks

Method Overview



h_{v_i} : node feature

S : set of anchor-sets

M_{v_i} : set(matrix) of aggregated messages to v_i from each anchor-set in S

z_{v_i} : computed distance vector

F : message computation function between two nodes

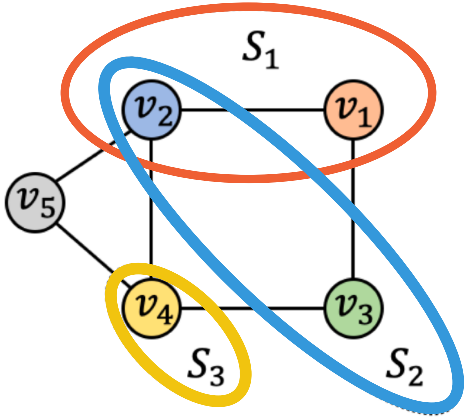
AGG_M : message aggregation function within each anchor-set

AGG_S : message aggregation function across all anchor-sets

w : vector which projects M_{v_i} to low-dimension vector

Position-aware Graph Neural Networks

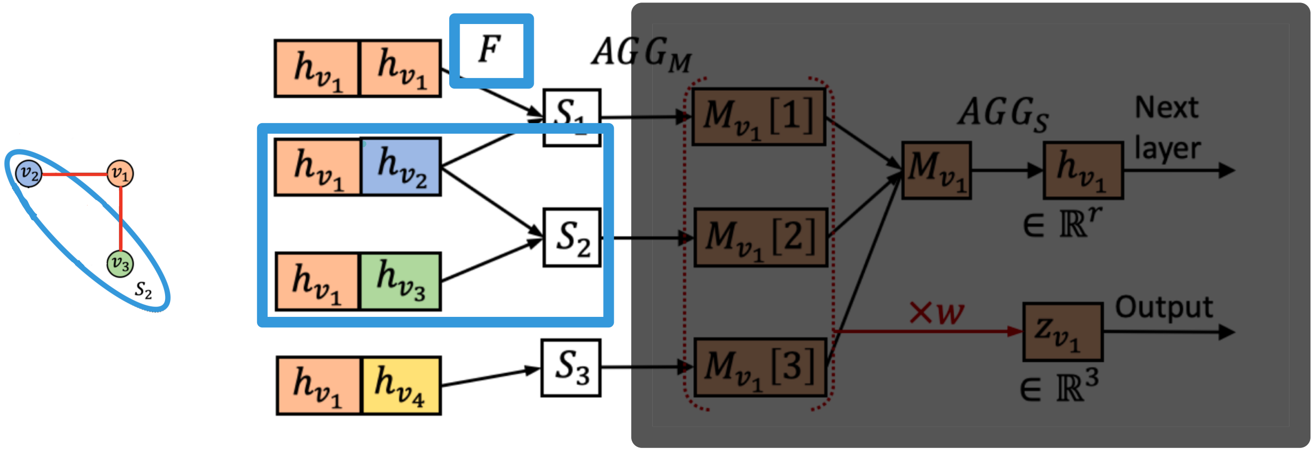
Method Overview : Anchor-set Selection



- randomly select k anchor-sets **for each forward pass(i.e. each layer in network)**
  (sampling guided by *Bourgain Theorem*)

Position-aware Graph Neural Networks

Method Overview : Message Computation



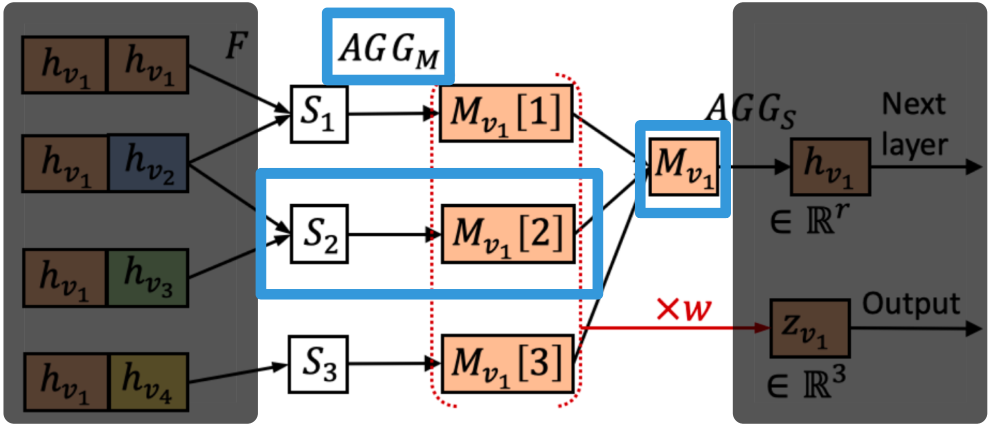
$$F\left(v_i, v_j, h_{v_i}, h_{v_j}\right) = s\left(v_i, v_j\right) \cdot \mathrm{CONCAT}\left(h_{v_i}, h_{v_j}\right)$$

$$s\left(v_i, v_j\right) = \frac{1}{d_{sp}^q\left(v_i, v_j\right) + 1}$$

- compute messages between query node and each node in each anchor-set (considering both position similarity and node features)

Position-aware Graph Neural Networks

Method Overview : Message Aggregation



$for\ anchor\ set\ S_t$

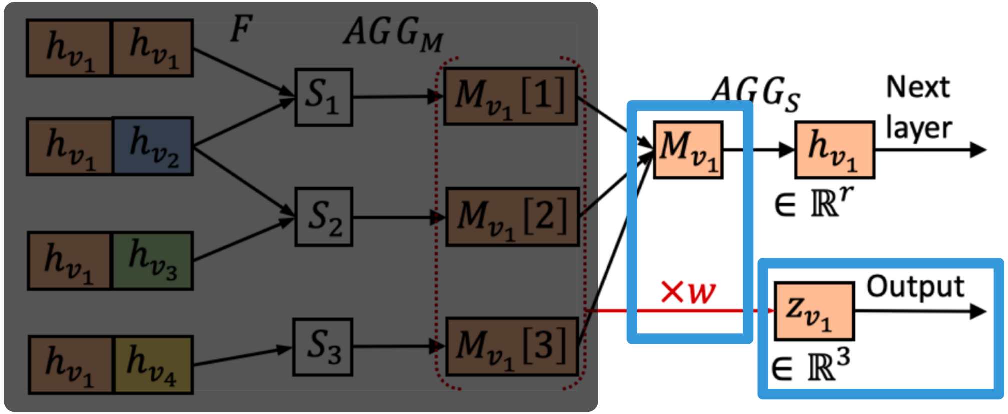
$$M_{v_i}[t] = AGG_M\left(\left\{F\left(v_i, v_j, h_{v_i}, h_{v_j}\right), v_j \in S_t\right\}\right)$$

$$M_{v_i} \in \mathcal{R}^{k \times m}$$

- aggregate messages within each anchor-set
  → output a matrix, in which each row is the information from each anchor-set

Position-aware Graph Neural Networks

Method Overview : Message Computation



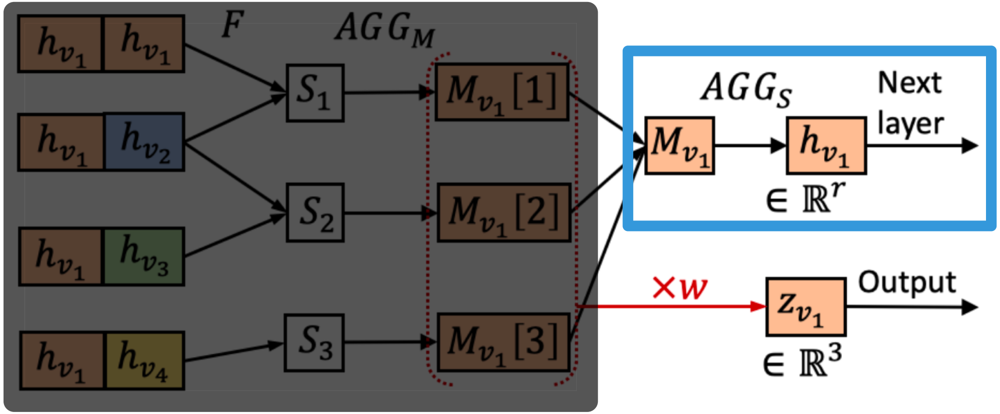
$$w \in \mathcal{R}^{m\times 1}$$

$$z_{v_i} = \sigma(M_{v_i} w) \in \mathcal{R}^{k\times 1}$$

- **project output matrix to low-dimension(=number of anchor-sets) vector**
- **each element of the low-dimension vector encodes the distance information for each anchor-sets, therefore structurally equivalent nodes are distinguishable**

Position-aware Graph Neural Networks

Method Overview : Message Computation



$$h_{v_i} = AGG_S(M_{v_i})$$

- aggregate messages across all the anchor-sets
  → computed new node feature passed to next layer

Position-aware Graph Neural Networks

Proposed Algorithm

Algorithm 1 The framework of P-GNNs

Input: Graph $G = (\mathcal{V}, \mathcal{E})$; Set S of k anchor-sets $\{S_i\}$; Node input features $\{\mathbf{x}_v\}$; Message computation function F that outputs an r dimensional message; Message aggregation functions $\textsc{Agg}_M, \textsc{Agg}_S$; Trainable weight vector $\mathbf{w} \in \mathbb{R}^r$; Non-linearity σ; Layer $l \in [1, L]$
Output: Position-aware embedding $\mathbf{z}_v$ for every node v

$\mathbf{h}_v \leftarrow \mathbf{x}_v$
for $l = 1, \dots, L$ **do**
  $S_i \sim \mathcal{V}$ for $i = 1, \dots, k$ → anchor-set selection
  for $v \in \mathcal{V}$ **do**
    $\mathbf{M}_v = \mathbf{0} \in \mathbb{R}^{k \times r}$
    for $i = 1 \dots, k$ **do**
      $\mathcal{M}_i \leftarrow \{F(v, u, \mathbf{h}_v, \mathbf{h}_u), \forall u \in S_i\}$ → message computation between nodes
      $\mathbf{M}_v[i] \leftarrow \textsc{Agg}_M(\mathcal{M}_i)$ → message aggregation within anchor-set
    end for
    $\mathbf{z}_v \leftarrow \sigma(\mathbf{M}_v \cdot \mathbf{w})$ → project to low-dimension distance vector
    $\mathbf{h}_v \leftarrow \textsc{Agg}_S(\{\mathbf{M}_v[i], \forall i \in [1, k]\})$ → message aggregation across all the anchor-sets
  end for
end for
$\mathbf{z}_v \in \mathbb{R}^k, \forall v \in \mathcal{V}$

Thank you