Graph and Subgraph Isomorphism Using GNNs

An overview of the essential concepts in Stanford CS224W (Lectures 9 and 12) with only oversimplified examples

Ng Yen Kaow

Graph Isomorphism

- Complexity of graph isomorphism is unknown
- Weisfeiler-Lehman graph kernel traditionally used to obtain graph-level embedding

WL color-refinement algorithm

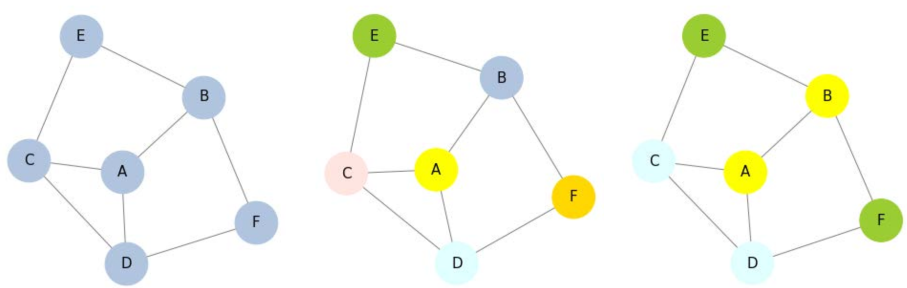
1. Assign initial color $c^{(0)}(v)$ to each node v
2. Iteratively refine node colors by

$$c^{(k+1)}(v) = \mathrm{HASH}\left(c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)}\right)$$

  - HASH function maps input to distinct values (colors)
  - After K steps, $c^{(K)}(v)$ summarizes the structure of the K-hop neighborhood
- To run as GNN, need to implement HASH as AGG
  - Need notion of distinguishing node embeddings

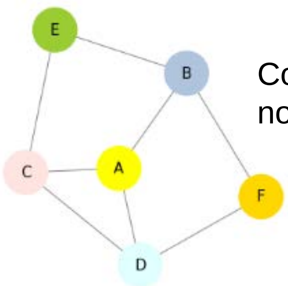
Distinguishing node embeddings

- Consider the task of keeping embeddings feature distinguishable
- Two factors affect embedding
  - (Initial) feature
    - Feature representation will affect whether features can be converted into each other
      - If $\boxed{1 \mid 0}$, $\boxed{0 \mid 1}$ and $\boxed{1 \mid 1}$ represent three distinct features, then the sum of $\boxed{1 \mid 0}$ and $\boxed{0 \mid 1}$ would become $\boxed{1 \mid 1}$, the third feature
    - To simplify this discussion assume that each distinct feature corresponds to one distinct dimension in the feature vector
  - Neighborhood structure
- Nodes with the same (initial) **feature** and **neighborhood structure** should be assigned the same embedding, and vice versa



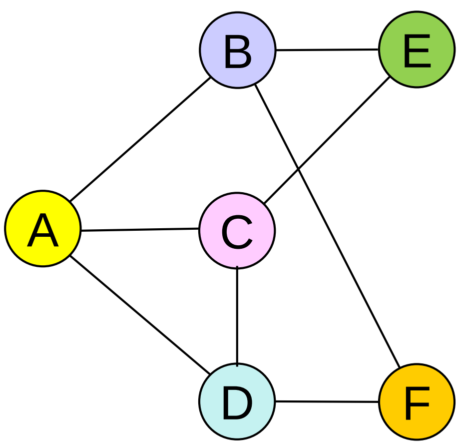
Computing node embeddings

- Recall that given a GCN of 2 layers, the embedding of A is computed through the computation graph as follows

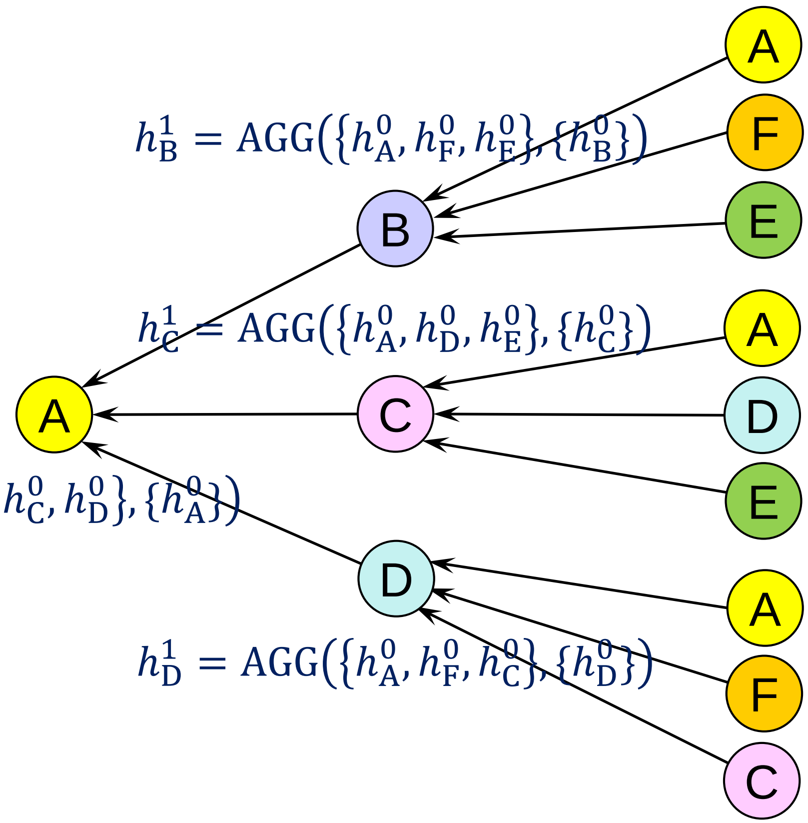


Consider the case where nodes have distinct features

Rearranged w.r.t. distance from A



Computation graph of A’s embedding



$$h_{\mathrm{B}}^{1} = \mathrm{AGG}\left(\{h_{\mathrm{A}}^{0}, h_{\mathrm{F}}^{0}, h_{\mathrm{E}}^{0}\}, \{h_{\mathrm{B}}^{0}\}\right)$$

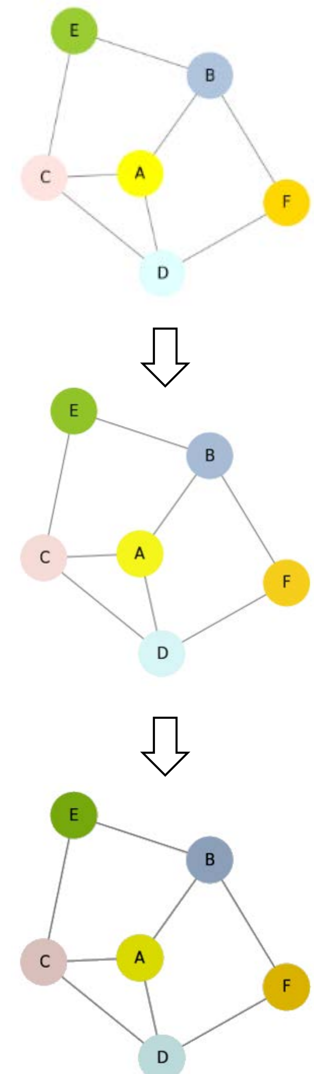
$$h_{\mathrm{C}}^{1} = \mathrm{AGG}\left(\{h_{\mathrm{A}}^{0}, h_{\mathrm{D}}^{0}, h_{\mathrm{E}}^{0}\}, \{h_{\mathrm{C}}^{0}\}\right)$$

$$h_{\mathrm{A}}^{1} = \mathrm{AGG}\left(\{h_{\mathrm{B}}^{0}, h_{\mathrm{C}}^{0}, h_{\mathrm{D}}^{0}\}, \{h_{\mathrm{A}}^{0}\}\right)$$

$$h_{\mathrm{D}}^{1} = \mathrm{AGG}\left(\{h_{\mathrm{A}}^{0}, h_{\mathrm{F}}^{0}, h_{\mathrm{C}}^{0}\}, \{h_{\mathrm{D}}^{0}\}\right)$$

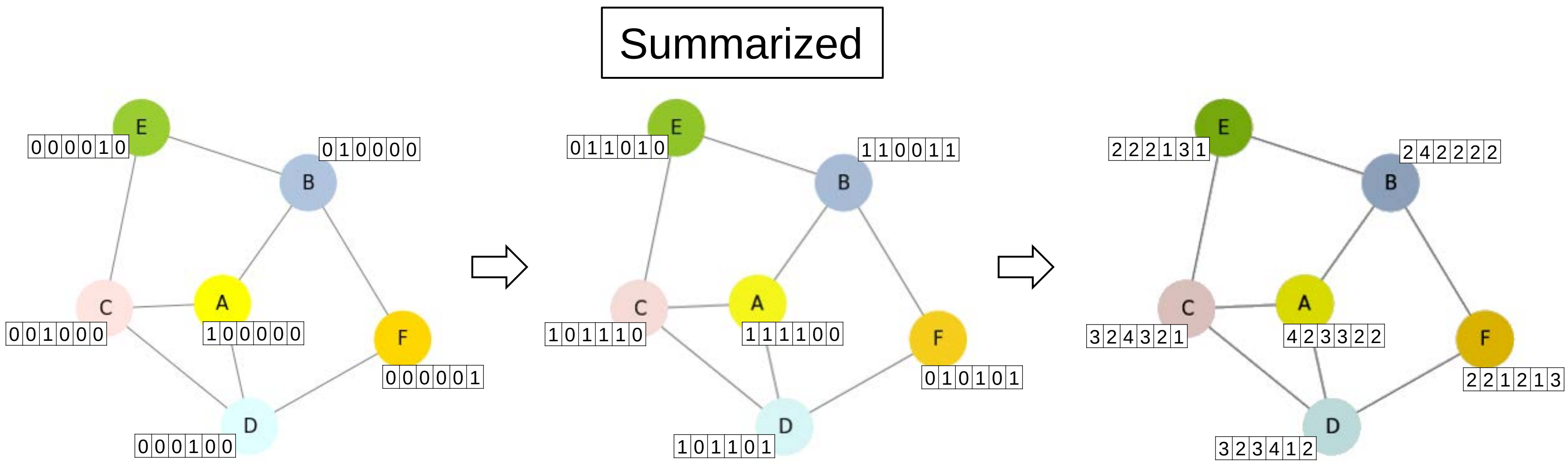
Distinguishability under **sum**

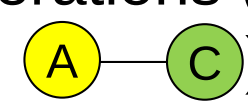
- Let AGG=**sum**, then for the GCN of 2 layers, the embeddings are as follows
  - Let h_{A}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 1&0&0&0&0&0\\\hline\end{array}$, h_{B}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 0&1&0&0&0&0\\\hline\end{array}$, h_{C}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 0&0&1&0&0&0\\\hline\end{array}$, h_{D}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 0&0&0&1&0&0\\\hline\end{array}$, h_{D}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 0&0&0&0&1&0\\\hline\end{array}$, h_{D}^{0}=$\begin{array}{|c|c|c|c|c|c|}\hline 0&0&0&0&0&1\\\hline\end{array}$, and let AGG be **sum**. Then
    - $h_{\mathrm{B}}^{1} = \mathrm{AGG}\left(\left\{h_{\mathrm{A}}^{0}, h_{\mathrm{F}}^{0}, h_{\mathrm{E}}^{0}\right\}, \left\{h_{\mathrm{B}}^{0}\right\}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 1&1&0&0&1&1\\\hline\end{array}$
    - $h_{\mathrm{C}}^{1} = \mathrm{AGG}\left(\left\{h_{\mathrm{A}}^{0}, h_{\mathrm{D}}^{0}, h_{\mathrm{E}}^{0}\right\}, \left\{h_{\mathrm{C}}^{0}\right\}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 1&0&1&1&1&0\\\hline\end{array}$
    - $h_{\mathrm{D}}^{1} = \mathrm{AGG}\left(\left\{h_{\mathrm{A}}^{0}, h_{\mathrm{F}}^{0}, h_{\mathrm{C}}^{0}\right\}, \left\{h_{\mathrm{D}}^{0}\right\}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 1&0&1&1&0&1\\\hline\end{array}$
    - $h_{\mathrm{A}}^{1} = \mathrm{AGG}\left(\left\{h_{\mathrm{B}}^{0}, h_{\mathrm{C}}^{0}, h_{\mathrm{D}}^{0}\right\}, \left\{h_{\mathrm{A}}^{0}\right\}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 1&1&1&1&0&0\\\hline\end{array}$
  - Finally the embedding of A is
    - $h_{\mathrm{A}}^{2} = \mathrm{AGG}\left(\left\{h_{\mathrm{B}}^{1}, h_{\mathrm{C}}^{1}, h_{\mathrm{D}}^{1}\right\}, \left\{h_{\mathrm{A}}^{1}\right\}\right) = \begin{array}{|c|c|c|c|c|c|}\hline 4&2&3&3&2&2\\\hline\end{array}$
  - Similarly, $\left.\begin{array}{l} h_{\mathrm{B}}^{2} = \begin{array}{|c|c|c|c|c|c|}\hline 2&4&2&2&2&2\\\hline\end{array} \\ h_{\mathrm{C}}^{2} = \begin{array}{|c|c|c|c|c|c|}\hline 3&2&4&3&2&1\\\hline\end{array} \\ h_{\mathrm{D}}^{2} = \begin{array}{|c|c|c|c|c|c|}\hline 3&2&3&4&1&2\\\hline\end{array} \\ h_{E}^{2} = \begin{array}{|c|c|c|c|c|c|}\hline 2&2&2&1&3&1\\\hline\end{array} \\ h_{F}^{2} = \begin{array}{|c|c|c|c|c|c|}\hline 2&2&1&2&1&3\\\hline\end{array} \end{array}\right\}$ Distinct embeddings



Distinguishability under **sum**

- Let AGG=**sum**, then for the GCN of 2 layers, the embeddings are as follows

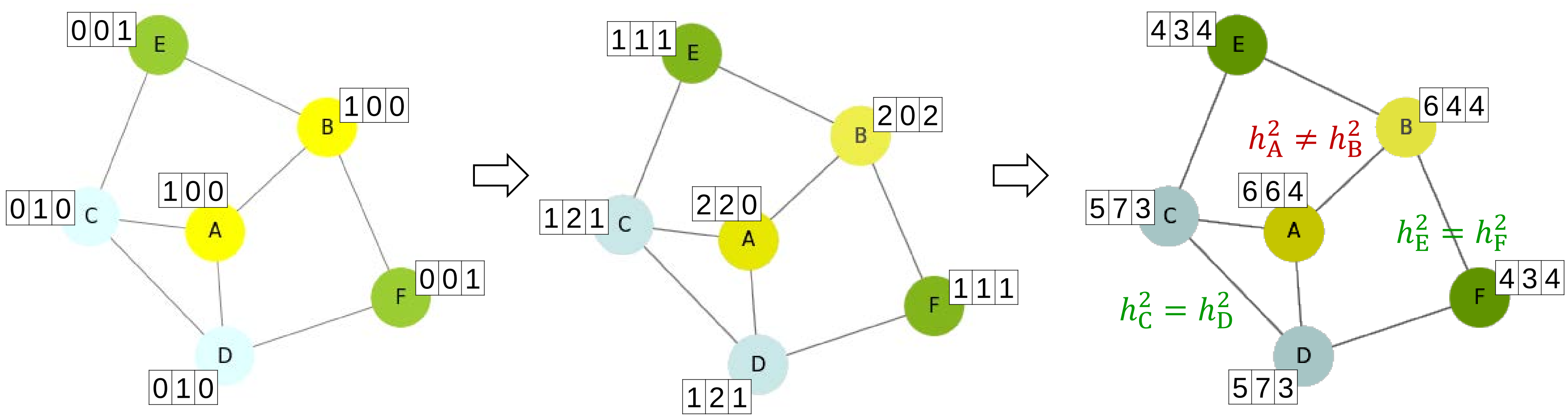


- By induction on the respective distinct feature dimension, the embeddings will be distinct for subsequent iterations
- **If every node has distinct feature, then they have distinct embeddings under sum** regardless of neighborhood structure or iterations (with the exception of the graph of only two nodes 

)

Distinguishability under **sum**

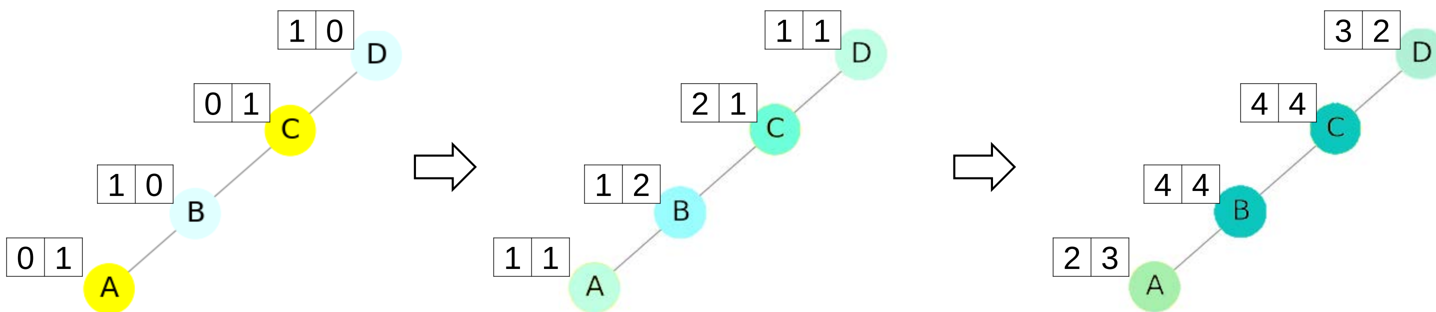
- **If some nodes have the same features?**
  - Let $h_{\mathrm{A}}^{0}=h_{\mathrm{B}}^{0}=\boxed{1\,|\,0\,|\,0}$, $h_{\mathrm{C}}^{0}=h_{\mathrm{D}}^{0}=\boxed{0\,|\,1\,|\,0}$, $h_{\mathrm{E}}^{0}=h_{\mathrm{F}}^{0}=\boxed{0\,|\,0\,|\,1}$, and let AGG be **sum**. Then, as can be seen from the following example



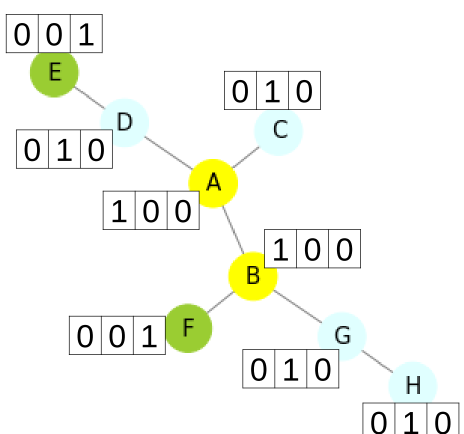
- Two nodes with the **same feature** will have the same embedding under **sum** if they have the same **neighborhood structure**
  - However different features and neighborhood structure cannot guarantee distinct embeddings

Distinguishability under **sum**

- **If some nodes have the same features?**
  - Different features and neighborhood structure cannot guarantee distinct embeddings for various reasons
    - Complementary neighborhood structure



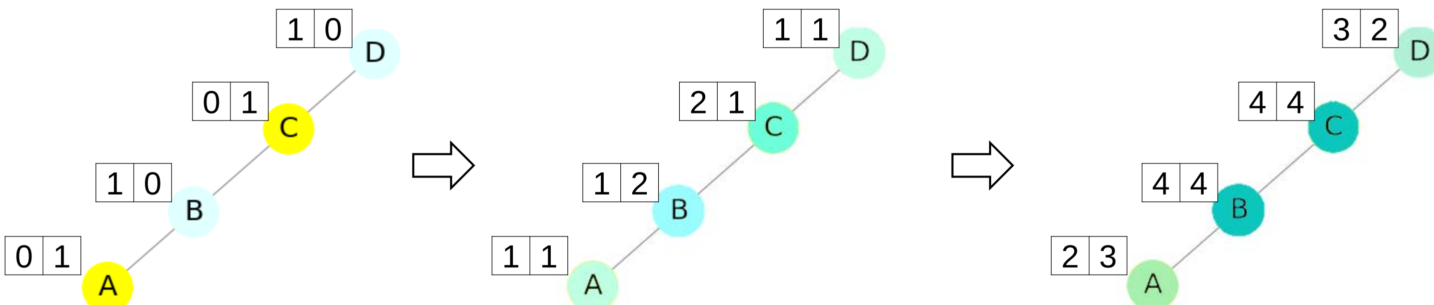
    - Hard-to-predict cases



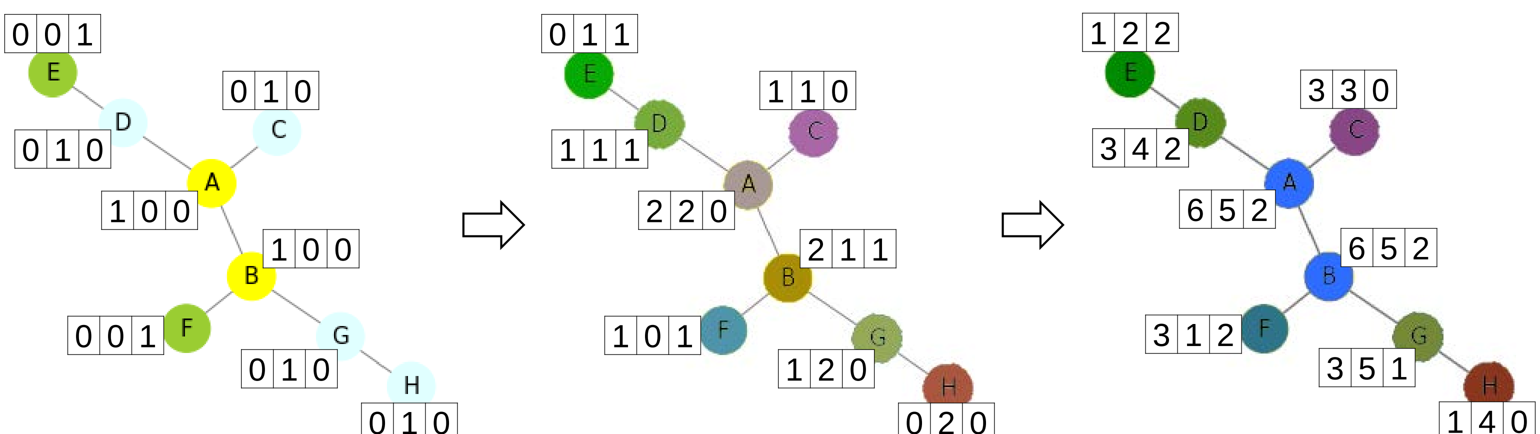
Will A and B ever become the same again after the first step?

Distinguishability under **sum**

- **If some nodes have the same features?**
  - Different features and neighborhood structure cannot guarantee distinct embeddings for various reasons
    - Complementary neighborhood structure

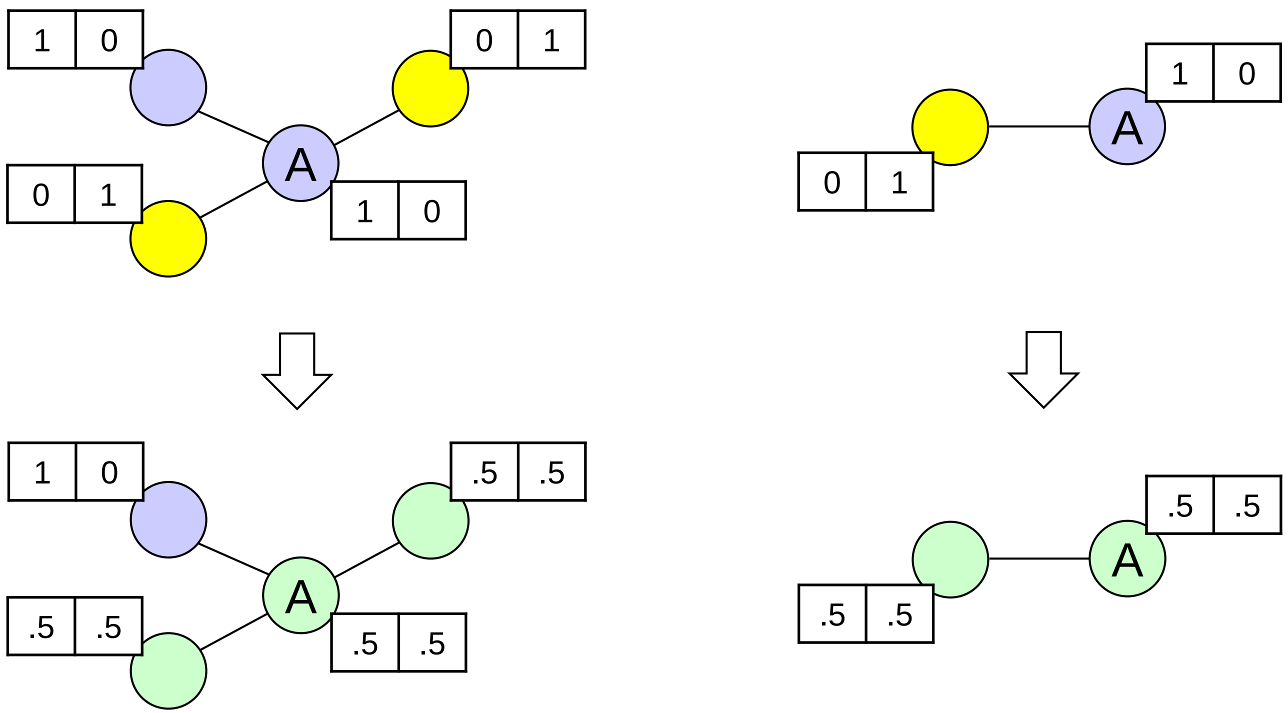


    - Hard-to-predict cases



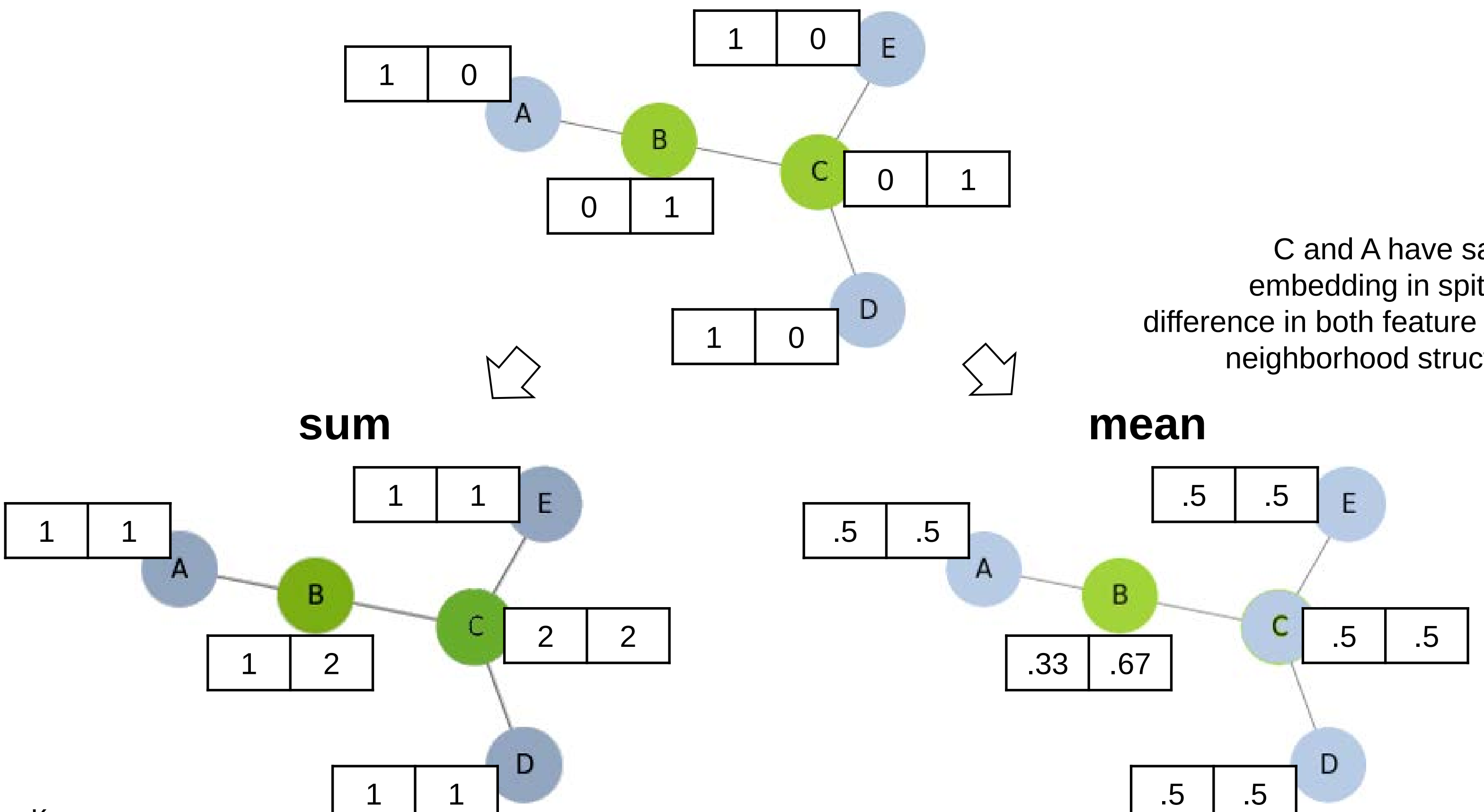
Distinguishability under **mean**

- Using **mean** as AGG results in even less desirable behavior
  - For instance, node A in both graphs below would give the same embedding under **mean** with one iteration



Distinguishability under **mean**

- Using **mean** as AGG results in even less desirable behavior
  - The following example shows **sum** to result in more consistent embeddings than **mean**



C and A have same embedding in spite of difference in both feature and neighborhood structure

Injective function for isomorphism

- An **injective** function will output distinguishable embeddings for nodes of distinct feature and neighborhood structure
  - **sum**, **mean**, and **max** are not injective
- **Theorem** (Xu *et al.* 2019). Any **injective** AGG function can be expressed as $\Phi\left(\sum_{x \in S} f(x)\right)$ for some non-linear Φ and linear f
- Since MLP is able to approximate any function, we can learn Φ and f with non-linear MLP_Φ and linear MLP_f

$$\mathrm{AGG} = \mathrm{MLP}_\Phi\left(\sum_{x \in S} \mathrm{MLP}_f(x)\right)$$

$\Rightarrow$ **Graph Isomorphism Network (GIN)**

Subgraph Isomorphism

- Subgraph isomorphism is NP-complete

  ⇒ Compare **neighborhood around each node**
- The k**-hop neighborhood around node** $u \in Q$ is
  1. All the nodes within k hops from u, and
  2. All the edges in between those nodes
  - Such a neighborhood is a subgraph of Q
    - However, not every subgraph of Q is a neighborhood of some node $u \in Q$
      - At most $k|V|$ neighborhoods for each k
- If P is a subgraph of Q, then every neighborhood of P is a subgraph of some neighborhood of Q

Order embedding space

- Idea: We want an d-dimensional embedding space z such that for every neighborhoods $p \in P$ and $q \in Q$

$$q \subseteq p \Leftrightarrow \forall_{i=1}^{d} z_q[i] \leq z_p[i]$$

- With embedding space, we can test subgraph isomorphism through the following

> For each neighborhood $p \in P$ and $q \in Q$
>
> If $(\exists i)\left[z_q[i] \leq z_p[i]\right]$, return false
>
> Return true

Order embedding space

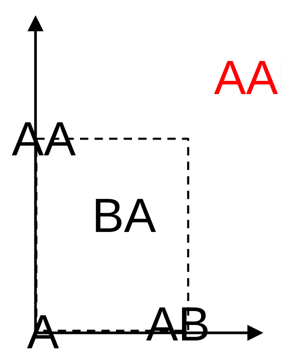
- Idea: We want an d-dimensional embedding space z such that for every neighborhoods $p \in P$ and $q \in Q$

$$q \subseteq p \Leftrightarrow \forall_{i=1}^{d} z_q[i] \leq z_p[i]$$

- Whether such an embedding space exist depends on D and the class of graphs
  - Even for substring relation ($\preccurlyeq$) with only alphabet {A, B}, a 2D embedding space is insufficient for

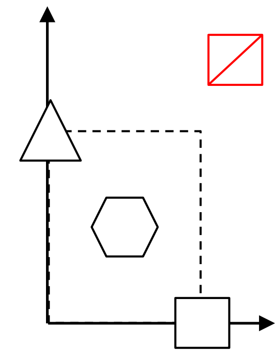
$$q \preccurlyeq p \Leftrightarrow \forall_{i=1}^{2} z_q[i] \leq z_p[i]$$

    - AAB needs to cover the dotted box (since it includes both AA and AB)
    - On the other hand, BA cannot be brought out of the dotted box (otherwise it would cover AA or AB)

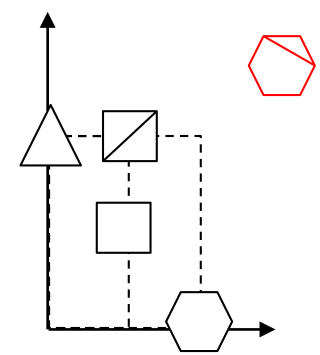


Order embedding space

- 2D embedding space example for graph
  - ⧄ needs to cover the dotted box (since it must cover both triangle and square) but that would cover ⬡
    - On the other hand, ⬡ cannot be brought out of the dotted box (otherwise it would cover △ or □)



  - Toggling hexagon and square will allow ⧄ to be placed, but now ⬡ cannot be placed



- Assume that d is sufficiently large for reasonable embeddings

Training order embedding space

- Use a **node embedding space of** k-hop
- Denote the embedding of a node u as z_u
- Use the loss function

$$\text{loss}(u, v) = \sum_{i=1}^{d} \max(0, z_u[i] - z_v[i])^2$$

  It is clear that
  - $\text{loss}(u, v) = 0$ when $\forall_{i=1}^{d}(z_u[i] \le z_v[i])$
  - $\text{loss}(u, v) > 0$ otherwise
- Generate random pair of graphs P, Q and train GNN such that embeddings of $u_P \in P$ and $u_Q \in Q$ has
  - $\text{loss}(u_Q, u_P) = 0$ when $u_Q \subseteq u_P$, and
  - $\text{loss}(u_Q, u_P) > 0$ otherwise

Mining frequent subgraphs

- **Problem.** Given graph G_T, find r most frequently occurring subgraphs of size k in G_T
- **Solution.** Exhaustively generate all graphs of size k and count the occurrences of each graph in G_T
- Avoid combinatorial explosion by
  - For each node in subgraph, attempt to superpose it to a node in G_T and see if there is a possible match
  - Stop at the first match

Mining frequent subgraphs

- For fast counting
  - Decompose each input G_T into $|G_T|$ subgraphs, each of a k-hop neighborhood around a node $u \in G_T$
    - Embed each subgraph into an order embedding space
  - For each graph of size k, only need to count the number of embeddings of G_T that completely covers its embedding
- Can make use of the order embedding space for efficient enumeration of graphs