Weisfeiler-Lehman use Simplicial Sets

PseudoTop Vertex Neural Network

Abdullah Naeem Malik

Department of Mathematics
College of Arts and Sciences
Florida State University

October XXVII, 2023

Outline

- Weisfeiler-Lehman Algorithm and graph neural networks
- The case for higher order relations
- Kan Extensions and Indexing Categories
- Top vertices and pseudotop vertices
- Pseudotop Vertex Neural Network
- Implementation

Weisfeiler-Lehman Algorithm

Definition

A **coloring** of a graph $G = (V, E)$ is a function $c : V \longrightarrow \mathbb{N}$.

Weisfeiler-Lehman Algorithm

Definition

A **coloring** of a graph $G = (V, E)$ is a function $c : V \longrightarrow \mathbb{N}$.

Definition

A (perfect) **hashing** is any injective function.

Weisfeiler-Lehman Algorithm

Definition

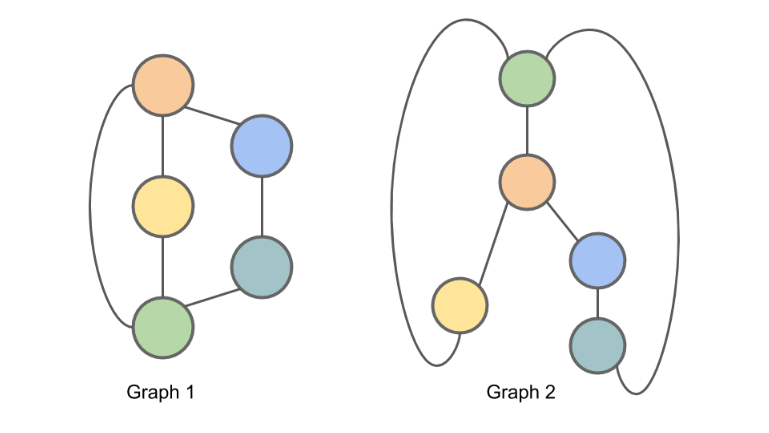
A **coloring** of a graph $G = (V, E)$ is a function $c : V \longrightarrow \mathbb{N}$.

Definition

A (perfect) **hashing** is any injective function.

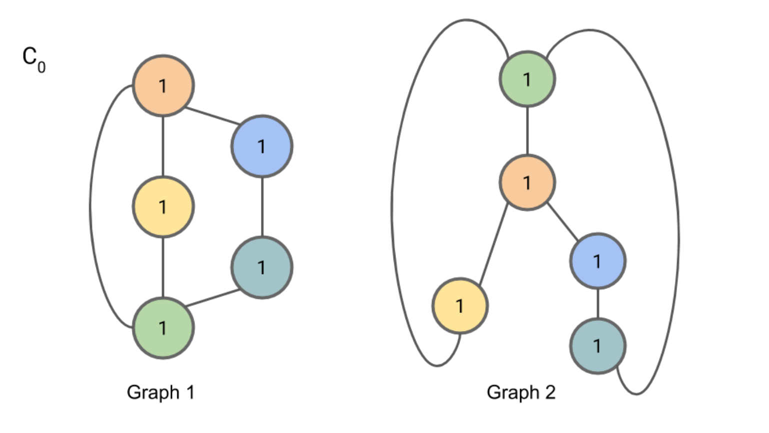
Basic idea: start with $c(v) = c^{(0)}(v)$, and
$c^{(t+1)}(v) = \text{hash}\left(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\right)$[WL68]

Weisfeiler-Lehman Algorithm



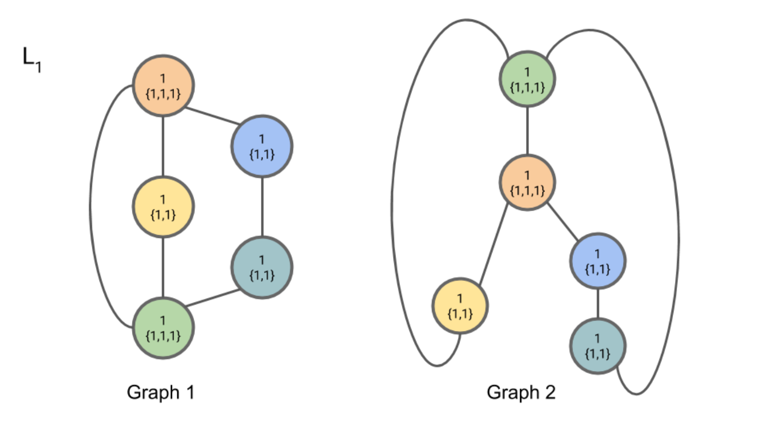
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



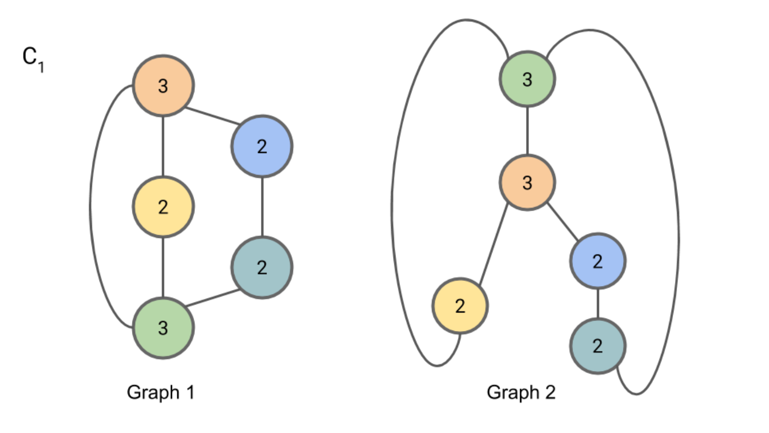
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



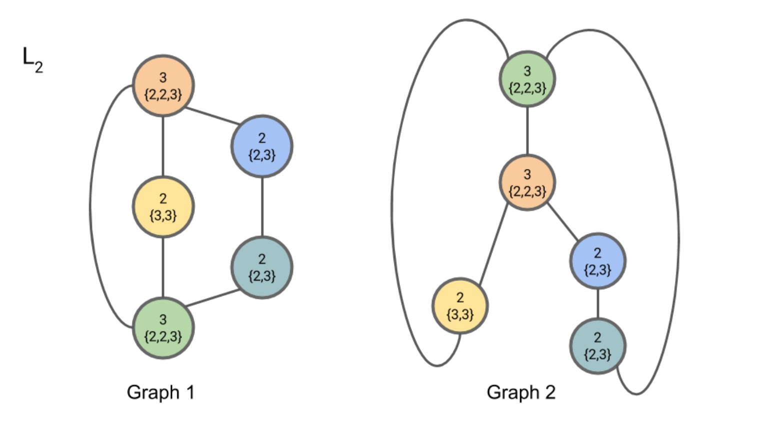
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



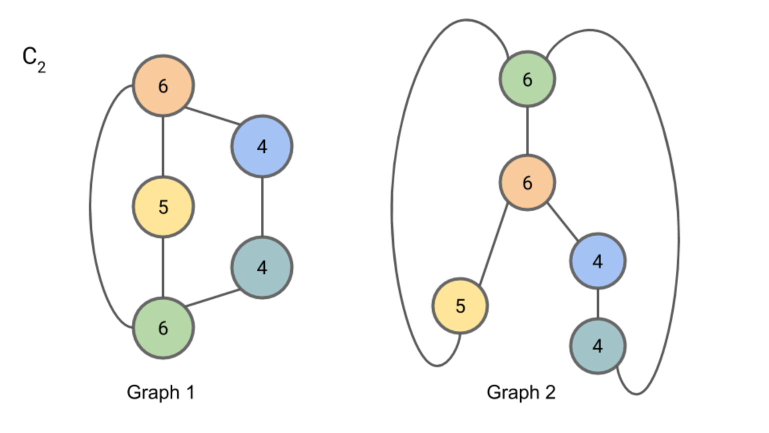
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



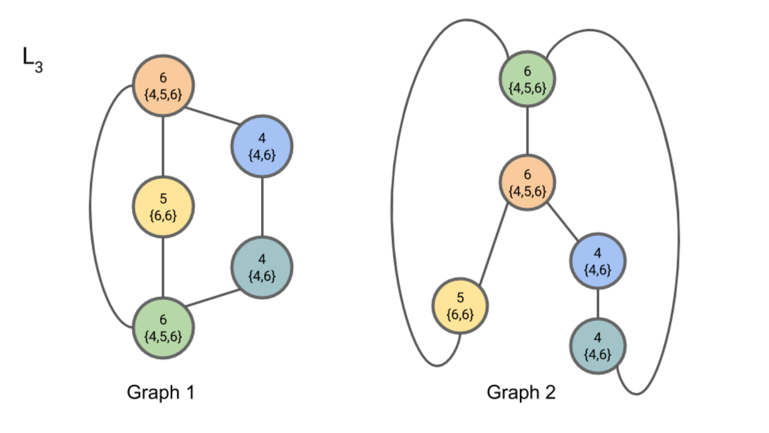
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



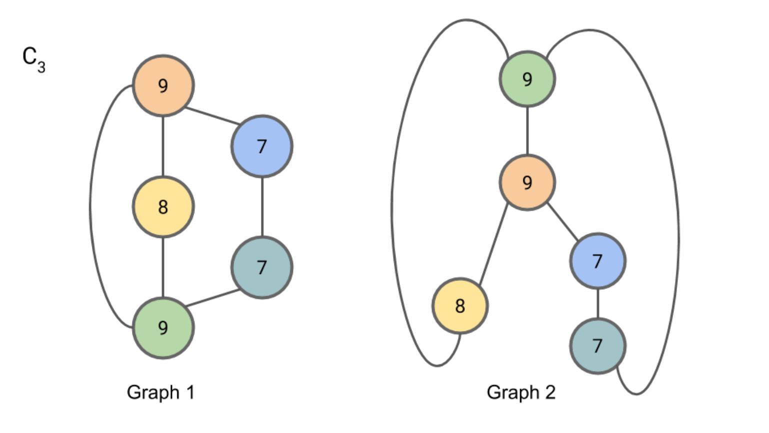
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm

Algorithm 1 Weisfeiler-Lehman (WL) or Naive vertex refinement

1: **Input**: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$
2: $c(v) = c^{(0)}(v) \longleftarrow hash(x_v)$
3: **while** $c^{(t)}(v) = c^{(t+1)}(v)\ \forall v \in V$ **do**
4: $\quad c^{(t+1)}(v) \longleftarrow \text{hash}\big(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\big)$
5: **end while**
6: **Output:** $c^{(T)}(v)\ \forall v \in V$

Weisfeiler-Lehman Algorithm

Algorithm 1 Weisfeiler-Lehman (WL) or Naive vertex refinement

Input: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$

$c(v) = c^{(0)}(v) \longleftarrow hash(x_v)$

while $c^{(t)}(v) = c^{(t+1)}(v)\ \forall v \in V$ **do**

  $c^{(t+1)}(v) \longleftarrow \text{hash}\left(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\right)$

end while

Output: $c^{(T)}(v)\ \forall v \in V$

If two graphs have different colorings, then the graphs are not isomorphic. But the converse is not true[MBHSL19]

Weisfeiler-Lehman Algorithm

Algorithm 1 Weisfeiler-Lehman (WL) or Naive vertex refinement

Input: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$

$c(v) = c^{(0)}(v) \longleftarrow hash(x_v)$

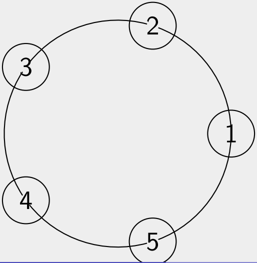
while $c^{(t)}(v) = c^{(t+1)}(v)\ \forall v \in V$ **do**

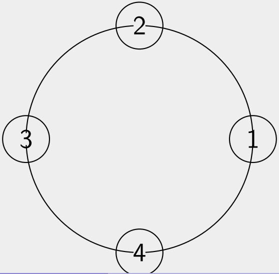
  $c^{(t+1)}(v) \longleftarrow \text{hash}\left(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\right)$

end while

Output: $c^{(T)}(v)\ \forall v \in V$

If two graphs have different colorings, then the graphs are not isomorphic. But the converse is not true[MBHSL19]





Graph Neural Networks

How powerful are graph neural networks?[MBHSL19, XHLJ18]

Graph Neural Networks

How powerful are graph neural networks?[MBHSL19, XHLJ18]

For node classification: Given (V, E, X_V), find $h_v = x_v^{(K)}$ and f such that $f(h_v) = x_v$

Graph Neural Networks

How powerful are graph neural networks?[MBHSL19, XHLJ18]

For node classification: Given (V, E, X_V), find $h_v = x_v^{(K)}$ and f such that $f(h_v) = x_v$

$$
\begin{aligned}
a_v^{(k+1)} &= AGGREGATE^{(k+1)}\left(\left\{x_u^{(k)} : u \in N(v)\right\}\right), \\
x_v^{(k+1)} &= COMBINE^{(k+1)}\left(x_v^{(k)}, a_v^{(k+1)}\right)
\end{aligned}
$$

Graph Neural Networks

How powerful are graph neural networks?[MBHSL19, XHLJ18]

For node classification: Given (V, E, X_V), find $h_v = x_v^{(K)}$ and f such that $f(h_v) = x_v$

$$\begin{aligned} a_v^{(k+1)} &= AGGREGATE^{(k+1)}\left(\left\{x_u^{(k)} : u \in N(v)\right\}\right), \\ x_v^{(k+1)} &= COMBINE^{(k+1)}\left(x_v^{(k)}, a_v^{(k+1)}\right) \end{aligned}$$

AGGREGATE	COMBINE	Ref
$\text{MAX}\left(\left\{\sigma\left(W_1.x_u^{(k)}\right)\right\}, u \in N(v)\right)$	$W_2.\left[x_v^{(k)}, a_v^{(k+1)}\right]$	GraphSAGE
$W_1.\text{MEAN}\left(x_u^{(k)}, u \in N(v) \cup \{v\}\right)$	$\sigma\left(\left\{W_2.a_v^{(k+1)}\right\}\right)$	GCN

Graph Neural Networks

or..

$$x_v^{(k+1)} = \text{COMBINE}\left(x_v^{(k)}, \text{AGGREGATE}^{(k+1)}\left(\left\{x_u^{(k)} : u \in N\left(v\right)\right\}\right)\right)$$

Graph Neural Networks

or..

$$x_v^{(k+1)} = \text{COMBINE}\left(x_v^{(k)}, \text{AGGREGATE}^{(k+1)}\left(\left\{x_u^{(k)} : u \in N(v)\right\}\right)\right)$$

Compare with

$$c^{(t+1)}(v) = \text{hash}\left(c^{(t)}(v), \{\{c^{(t)}(w) : w \in N(v)\}\}\right)$$

Graph Neural Networks

or..
$x_v^{(k+1)} = \text{COMBINE}\left(x_v^{(k)}, \text{AGGREGATE}^{(k+1)}\left(\left\{x_u^{(k)} : u \in N(v)\right\}\right)\right)$
Compare with
$c^{(t+1)}(v) = \text{hash}\left(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\right)$

Theorem

Let G_1 and G_2 be any two non-isomorphic graphs. If a graph neural network $\mathcal{A} : \mathcal{G} \longrightarrow \mathbb{R}^d$ maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic. Converse holds if COMBINE and AGGREGATE are injective[XHLJ18].

k-Weisfeiler-Lehman Algorithm

Algorithm 1 k-Weisfeiler-Lehman (k-WL)

Input: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$
$c\left(\overrightarrow{v}\right) = c^{(0)}\left(\overrightarrow{v}\right) \longleftarrow hash\left(x_{\overrightarrow{v}}\right)$
while $c^{(t)}\left(\overrightarrow{v}\right) = c^{(t+1)}\left(\overrightarrow{v}\right) \; \forall \overrightarrow{v} \in V^k$ **do**
 $c_i^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \{\!\{c^{(t)}\left(\overrightarrow{w}\right) : w \in N_i\left(\overrightarrow{v}\right)\}\!\} \; \forall \overrightarrow{v} \in V^k$
 $c^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \text{hash}\left(c^{(t)}\left(\overrightarrow{v}\right), c_1^{(t+1)}\left(\overrightarrow{v}\right), \ldots, c_k^{(t+1)}\left(\overrightarrow{v}\right)\right) \; \forall \overrightarrow{v} \in V^k$
end while
Output: $c^{(T)}\left(\overrightarrow{v}\right) \; \forall \overrightarrow{v} \in V^k$

Here, $hash\left(x_{\overrightarrow{v}}\right) = hash\left(x_{\overrightarrow{w}}\right)$ iff $x_{v_i} = x_{w_i}$ and if $(v_i, v_j) \in E$ iff $(w_i, w_j) \in E$ and $N_i\left(\overrightarrow{v}\right) = \{(v_1, \ldots, v_{i-1}, u, v_{i+1}, \ldots, v_k) : u \in V\}$

k-Weisfeiler-Lehman Algorithm

Algorithm 2 k-Weisfeiler-Lehman (k-WL)

1: **Input**: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$
2: $c\left(\overrightarrow{v}\right) = c^{(0)}\left(\overrightarrow{v}\right) \longleftarrow hash\left(x_{\overrightarrow{v}}\right)$
3: **while** $c^{(t)}\left(\overrightarrow{v}\right) = c^{(t+1)}\left(\overrightarrow{v}\right)\ \forall \overrightarrow{v} \in V^k$ **do**
4: $\quad c_i^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \{\!\{c^{(t)}\left(\overrightarrow{w}\right) : w \in N_i\left(\overrightarrow{v}\right)\}\!\}\ \forall \overrightarrow{v} \in V^k$
5: $\quad c^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \text{hash}\left(c^{(t)}\left(\overrightarrow{v}\right), c_1^{(t+1)}\left(\overrightarrow{v}\right), ..., c_k^{(t+1)}\left(\overrightarrow{v}\right)\right)\ \forall \overrightarrow{v} \in V^k$
6: **end while**
7: **Output:** $c^{(T)}\left(\overrightarrow{v}\right)\ \forall \overrightarrow{v} \in V^k$

Here, $hash\left(x_{\overrightarrow{v}}\right) = hash\left(x_{\overrightarrow{w}}\right)$ iff $x_{v_i} = x_{w_i}$ and if $(v_i, v_j) \in E$ iff $(w_i, w_j) \in E$ and
$N_i\left(\overrightarrow{v}\right) = \{(v_1, ..., v_{i-1}, u, v_{i+1}, ..., v_k) : u \in V\}$
$N_i\left(\overrightarrow{v}\right) = \{(i, v_2 ..., v_k), (v_1, i, ..., v_k), ..., (v_1, ..., v_{k-1}, i) : i \in V\}$ for Folklore k-WL

k-Weisfeiler-Lehman Algorithm

Algorithm 3 k-Weisfeiler-Lehman (k-WL)

1: **Input**: (V, E, X_V) $\{\triangleright x_v \in \mathbb{Z}_2^d\}$
2: $c\left(\overrightarrow{v}\right) = c^{(0)}\left(\overrightarrow{v}\right) \longleftarrow hash\left(x_{\overrightarrow{v}}\right)$
3: **while** $c^{(t)}\left(\overrightarrow{v}\right) = c^{(t+1)}\left(\overrightarrow{v}\right)\ \forall \overrightarrow{v} \in V^k$ **do**
4: $\quad c_i^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \{\!\{c^{(t)}\left(\overrightarrow{w}\right) : w \in N_i\left(\overrightarrow{v}\right)\}\!\}\ \forall \overrightarrow{v} \in V^k$
5: $\quad c^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \text{hash}\left(c^{(t)}\left(\overrightarrow{v}\right), c_1^{(t+1)}\left(\overrightarrow{v}\right), ..., c_k^{(t+1)}\left(\overrightarrow{v}\right)\right)\ \forall \overrightarrow{v} \in V^k$
6: **end while**
7: **Output:** $c^{(T)}\left(\overrightarrow{v}\right)\ \forall \overrightarrow{v} \in V^k$

Here, $hash\left(x_{\overrightarrow{v}}\right) = hash\left(x_{\overrightarrow{w}}\right)$ iff $x_{v_i} = x_{w_i}$ and if $(v_i, v_j) \in E$ iff $(w_i, w_j) \in E$ and
$N_i\left(\overrightarrow{v}\right) = \{(v_1, ..., v_{i-1}, u, v_{i+1}, ..., v_k) : u \in V\}$
$N_i\left(\overrightarrow{v}\right) = \{(i, v_2 ..., v_k), (v_1, i, ..., v_k), ..., (v_1, ..., v_{k-1}, i) : i \in V\}$ for Folklore k-WL
For directed graphs, use
$c^{t+1}(v) \longleftarrow \text{hash}\left(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N_{in}(v)\}\!\}, \{\!\{c^{(t)}(w) : w \in N_{out}(v)\}\!\}\right)$
[MG21]

WL-kernels

Can be used to make different neural networks.

WL-kernels

Can be used to make different neural networks.

$$
\begin{aligned}
x_S^{(t)} &= \sigma\left(x_S^{(t-1)}.W_1^{(t)} + \sum_{u \in N_L(S) \cup N_G(S)} x_u^{(t-1)}.W_2^{(t)}\right) \text{ (global)} \\
x_S^{(t)} &= \sigma\left(x_S^{(t-1)}.W_1^{(t)} + \sum_{u \in N_L(S)} x_u^{(t-1)}.W_2^{(t)}\right) \text{ (local)}
\end{aligned}
$$

where $S = (v_1, ..., v_k)$,
$N_L(S) = \left\{T \in V^k : |S \cap T| = k-1, (v, w) \in E \text{ for some unique } v, w \in S \setminus T\right\}$
$N(S) = \left\{T \in V^k : |S \cap T| = k-1\right\}$, $N_G(S) = N(S) \setminus N_L(S)$[MRF+19].

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]
- k-WL is strictly weaker than $(k+1)$-WL[HV21] ($(k+1)$-WL $\sqsubset$ k-WL)

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]
- k-WL is strictly weaker than $(k+1)$-WL[HV21] ($(k+1)$-WL $\sqsubset$ k-WL)
- Therefore, the architecture of [MRF$^+$19] performs better than simple message passing.

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]
- k-WL is strictly weaker than $(k+1)$-WL[HV21] ($(k+1)$-WL $\sqsubset$ k-WL)
- Therefore, the architecture of [MRF$^+$19] performs better than simple message passing.
- $(k+1)$-WL $\equiv$ k-FWL[HV21]

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]
- k-WL is strictly weaker than $(k+1)$-WL[HV21] ($(k+1)$-WL $\sqsubset$ k-WL)
- Therefore, the architecture of [MRF$^+$19] performs better than simple message passing.
- $(k+1)$-WL $\equiv$ k-FWL[HV21]
- For every k, there is an infinite family of graphs for which the k-WL test fails[CFI92]

Strength of WL Tests

- 2-WL $\equiv$ WL[MBHSL19]
- k-WL is strictly weaker than $(k+1)$-WL[HV21] ($(k+1)$-WL $\sqsubset$ k-WL)
- Therefore, the architecture of [MRF+19] performs better than simple message passing.
- $(k+1)$-WL $\equiv$ k-FWL[HV21]
- For every k, there is an infinite family of graphs for which the k-WL test fails[CFI92]
- Note: DWL $\sqsubseteq$ WL, so GNN with directed edges are more powerful[RCDG+23]
  $a_v^{(k+1)} = \text{AGGREGATE}^{(k+1)}\left(\left\{x_u^{(k)}, x_v^{(k)} : (u,v) \in E\right\}\right)$

The case for higher order relations

Simplicial WL [BFW+21] uses

$$c^{t+1}(\sigma) \longleftarrow \mathrm{hash}\Big(c^{t}(\sigma), c^{t}_{\mathcal{B}}(\sigma), c^{t}_{\mathcal{C}}(\sigma), c^{t}_{\downarrow}(\sigma), c^{t}_{\uparrow}(\sigma)\Big)$$

The case for higher order relations

Simplicial WL [BFW+21] uses

$c^{t+1}(\sigma) \longleftarrow \mathrm{hash}\Big(c^{t}(\sigma), c^{t}_{\mathcal{B}}(\sigma), c^{t}_{\mathcal{C}}(\sigma), c^{t}_{\downarrow}(\sigma), c^{t}_{\uparrow}(\sigma)\Big)$

Simplicial WL is more powerful than 3-WL[BFW+21]

The case for higher order relations

Simplicial WL [BFW+21] uses

$c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^{t}(\sigma), c^{t}_{\mathcal{B}}(\sigma), c^{t}_{\mathcal{C}}(\sigma), c^{t}_{\downarrow}(\sigma), c^{t}_{\uparrow}(\sigma)\Big)$

Simplicial WL is more powerful than 3-WL[BFW+21]

Conjecture

For each $n \leq k \leq |V(G)|$ and $a : V(G)^{n} \longrightarrow \mathbb{N}$ and $c : S_n \longrightarrow \mathbb{N}$ we have $c \sqsubseteq a|_{S_n}$, where S_n is the collection of directed n-simplices.

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.
- A directed graph DG is a functor $DG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$. Here, $[0], [1] \in Obj(\Delta_{\leq 1})$, and $Hom_{\Delta_{\leq 1}}([0], [1]) = \{\sigma, \tau\}$ and $Hom_{\Delta_{\leq 1}}([1], [0]) = \{\delta\}$.

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.
- A directed graph DG is a functor $DG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$. Here, $[0], [1] \in Obj(\Delta_{\leq 1})$, and $Hom_{\Delta_{\leq 1}}([0], [1]) = \{\sigma, \tau\}$ and $Hom_{\Delta_{\leq 1}}([1], [0]) = \{\delta\}$.
- Thus, DG is given by the data $DG([0]) = DG_0$, $DG([1]) = DG_1$, $s, t : E \longrightarrow V$ and $d : V \longrightarrow E$.

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.
- A directed graph DG is a functor $DG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$. Here, $[0], [1] \in Obj(\Delta_{\leq 1})$, and $Hom_{\Delta_{\leq 1}}([0], [1]) = \{\sigma, \tau\}$ and $Hom_{\Delta_{\leq 1}}([1], [0]) = \{\delta\}$.
- Thus, DG is given by the data $DG([0]) = DG_0$, $DG([1]) = DG_1$, $s, t : E \longrightarrow V$ and $d : V \longrightarrow E$.
- That is, a graph is a 1d simplicial set[Lur23].

Modelling relations

- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.
- A directed graph DG is a functor $DG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$. Here, $[0], [1] \in Obj(\Delta_{\leq 1})$, and $Hom_{\Delta_{\leq 1}}([0], [1]) = \{\sigma, \tau\}$ and $Hom_{\Delta_{\leq 1}}([1], [0]) = \{\delta\}$.
- Thus, DG is given by the data $DG([0]) = DG_0$, $DG([1]) = DG_1$, $s, t : E \longrightarrow V$ and $d : V \longrightarrow E$.
- That is, a graph is a 1d simplicial set[Lur23].
- An undirected graph is a 1d symmetric simplicial set: a functor $UG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$ such that $t_i^n : UG_i \longrightarrow UG_i$ is a bijection for $i = 0, ..., n-1$

Modelling relations

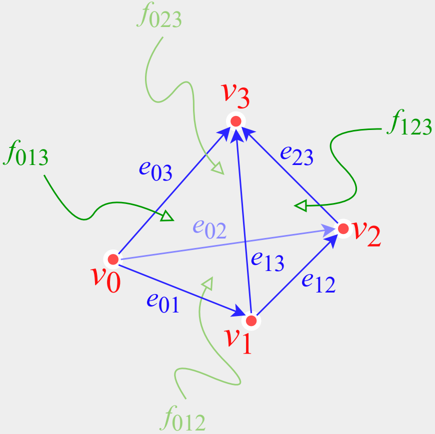
- Binary relations $E \subset V \times V$ a.k.a directed graphs!
- Undirected graphs: symmetric E.
- A directed graph DG is a functor $DG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$. Here, $[0], [1] \in Obj(\Delta_{\leq 1})$, and $Hom_{\Delta_{\leq 1}}([0], [1]) = \{\sigma, \tau\}$ and $Hom_{\Delta_{\leq 1}}([1], [0]) = \{\delta\}$.
- Thus, DG is given by the data $DG([0]) = DG_0$, $DG([1]) = DG_1$, $s, t : E \longrightarrow V$ and $d : V \longrightarrow E$.
- That is, a graph is a 1d simplicial set[Lur23].
- An undirected graph is a 1d symmetric simplicial set: a functor $UG : \Delta_{\leq 1}^{op} \longrightarrow \mathcal{S}et$ such that $t_i^n : UG_i \longrightarrow UG_i$ is a bijection for $i = 0, ..., n-1$
- Therefore, DL $\sqsubseteq$ WL

Modelling higher relations

Kan extension of DG along $i : \Delta_{\leq 1} \longrightarrow \Delta$ produces the functor $Ran_{\imath}- := \imath_* : \mathcal{DG} \longrightarrow \mathcal{SS}ets$ and a natural bijection $\mathrm{Hom}_{\mathcal{S}Set}(X, \imath_*(G)) \cong \mathrm{Hom}_{\mathcal{G}}(\imath^*(X), G)$

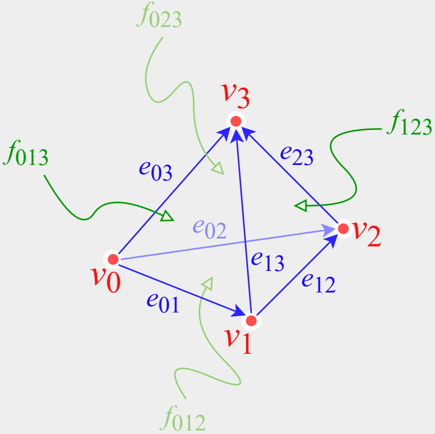
Modelling higher relations

Kan extension of DG along $i : \Delta_{\leq 1} \longrightarrow \Delta$ produces the functor $Ran_{\imath} - := \imath_* : \mathcal{DG} \longrightarrow \mathcal{SS}ets$ and a natural bijection $\mathrm{Hom}_{\mathcal{S}Set}(X, \imath_*(G)) \cong \mathrm{Hom}_{\mathcal{G}}(\imath^*(X), G)$



Modelling higher relations

Kan extension of DG along $i : \Delta_{\leq 1} \longrightarrow \Delta$ produces the functor $Ran_{\imath} - := \imath_* : \mathcal{DG} \longrightarrow \mathcal{SS}ets$ and a natural bijection $\mathrm{Hom}_{\mathcal{S}Set}(X, \imath_*(G)) \cong \mathrm{Hom}_{\mathcal{G}}(\imath^*(X), G)$



Clique complexes of graphs are given by Kan Extensions of UG along i

Modelling higher relations

Lemma

Simplicial Set WL $\sqsubseteq$ *Simplicial WL* $\sqsubset$ *WL*

Modelling higher relations

Lemma

Simplicial Set WL $\sqsubseteq$ *Simplicial WL* $\sqsubset$ *WL*

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^{t}(\sigma), c^{t}_{\mathcal{B}_i}(\sigma), c^{t}_{\mathcal{C}_i}(\sigma), c^{t}_{\downarrow,i}(\sigma), c^{t}_{\uparrow,i}(\sigma)\Big)$

Modelling higher relations

Lemma

Simplicial Set WL $\sqsubseteq$ *Simplicial WL* $\sqsubset$ *WL*

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma), c^t_{\downarrow,i}(\sigma), c^t_{\uparrow,i}(\sigma)\Big)$
- $\equiv \text{hash}\big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma)\big)$

Modelling higher relations

Lemma

Simplicial Set WL$\sqsubseteq$Simplicial WL$\sqsubset$ WL

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma), c^t_{\downarrow,i}(\sigma), c^t_{\uparrow,i}(\sigma)\Big)$
- $\equiv \text{hash}\big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma)\big)$
- A hypergraph is a presheaf of the category $[0] \longrightarrow [i]$ for $i = 1, 2, ...$

Modelling higher relations

Lemma

Simplicial Set WL$\sqsubseteq$Simplicial WL$\sqsubset$ WL

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma), c^t_{\downarrow,i}(\sigma), c^t_{\uparrow,i}(\sigma)\Big)$
- $\equiv \text{hash}\big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma)\big)$
- A hypergraph is a presheaf of the category $[0] \longrightarrow [i]$ for $i = 1, 2, \ldots.$
- So, Simplicial Set WL$\sqsubseteq$Hypergraph WL

Modelling higher relations

Lemma

Simplicial Set WL$\sqsubseteq$Simplicial WL$\sqsubset$ WL

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma), c^t_{\downarrow,i}(\sigma), c^t_{\uparrow,i}(\sigma)\Big)$
- $\equiv \text{hash}\big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma)\big)$
- A hypergraph is a presheaf of the category $[0] \longrightarrow [i]$ for $i = 1, 2, ...$
- So, Simplicial Set WL$\sqsubseteq$Hypergraph WL

Modelling higher relations

Lemma

Simplicial Set WL$\sqsubseteq$Simplicial WL$\sqsubset$ WL

- Simplicial Set WL:
  $c^{t+1}(\sigma) \longleftarrow \text{hash}\Big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma), c^t_{\downarrow,i}(\sigma), c^t_{\uparrow,i}(\sigma)\Big)$
- $\equiv \text{hash}\big(c^t(\sigma), c^t_{\mathcal{B}_i}(\sigma), c^t_{\mathcal{C}_i}(\sigma)\big)$
- A hypergraph is a presheaf of the category $[0] \longrightarrow [i]$ for $i = 1, 2, \ldots$.
- So, Simplicial Set WL$\sqsubseteq$Hypergraph WL

In summary:

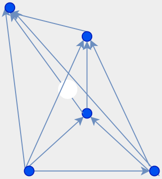
$$\begin{array}{ccc} DWL & \sqsubseteq & WL \\ \sqcup & & \sqcup \\ 3DWL & \sqsubseteq & 3\text{-}WL \\ \sqcup & & \sqcup \\ SSWL & \sqsubseteq & SWL \end{array}$$

Top Vertices

Recall simplicial set $\Delta[n]_\bullet$

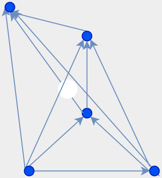
Top Vertices

Recall simplicial set $\Delta[n]_\bullet$



Top Vertices

Recall simplicial set $\Delta[n]_\bullet$



Algorithm 6 Creating 1-skeleton of the geometric realization of standard n-simplex

Input n
for i from 1 to n **do**
  for j from 1 to n **do**
    if $i < j$ **then**
      `src` $\leftarrow$ `src` $+ [i]$
      `dst` $\leftarrow$ `dst` $+ [j]$
      `edges` $\leftarrow$ `edges` $+ [(i, j)]$
    end if
  end for
end for
Output List of directed edges `edges`, source vertices `src` and target vertices

Top Vertices

Definition

A vertex $v \in G_0$ is said to be a **top vertex** of dimension k if there is a simplicial set x of dimension k such that $d_0^{(1)} d_0^{(2)} ... d_0^{(k-1)} d_0^{(k)} x = v$, where $d_0^{(k)} : X_k \longrightarrow X_{k-1}$ is the 0-th face map.

Top Vertices

Definition

A vertex $v \in G_0$ is said to be a **top vertex** of dimension k if there is a simplicial set x of dimension k such that $d_0^{(1)} d_0^{(2)} \dots d_0^{(k-1)} d_0^{(k)} x = v$, where $d_0^{(k)} : X_k \longrightarrow X_{k-1}$ is the 0-th face map.

Lemma

There is a bijective correspondence between top vertices and standard simplices.

Top Vertices

Definition

A vertex $v \in G_0$ is said to be a **top vertex** of dimension k if there is a simplicial set x of dimension k such that $d_0^{(1)} d_0^{(2)} ... d_0^{(k-1)} d_0^{(k)} x = v$, where $d_0^{(k)} : X_k \longrightarrow X_{k-1}$ is the 0-th face map.

Lemma

There is a bijective correspondence between top vertices and standard simplices.

Lemma

If v is a top vertex of dimension k, then $d_{in}(v) \geq k$.

Top Vertices

Definition

A vertex $v \in G_0$ is said to be a **top vertex** of dimension k if there is a simplicial set x of dimension k such that $d_0^{(1)} d_0^{(2)} ... d_0^{(k-1)} d_0^{(k)} x = v$, where $d_0^{(k)} : X_k \longrightarrow X_{k-1}$ is the 0-th face map.

Lemma

There is a bijective correspondence between top vertices and standard simplices.

Lemma

If v is a top vertex of dimension k, then $d_{in}(v) \geq k$.

Lemma

Let A be the adjacency matrix of G, and $\widetilde{A} = A - diag(A)$. If v is a top vertex for a k-simplex x, then the v-th column of $A \circledcirc \left(\widetilde{A} + \widetilde{A}^2 + ... + \widetilde{A}^k\right)$ is a decreasing sequence (possibly after a permutation), starting with $\sum_{n=0}^{k} \binom{k}{n}$.

Top Vertices

Lemma

Let $v \in G_0$*, and* A *be the adjacency matrix associated with the graph* G*. If*

$$\left(A + A^T\right)_{vv}^{k+1} \geq 4\sum_{n=0}^{k} \binom{k}{n}$$

and $\left|N_1^{in}(u) \cap N_1^{in}(v)\right| \geq k - 1$*, then the following are equivalent:*

1. $\exists u \in N_1^{in}(v)$ *such that* u *is a top vertex for a* $(k-1)$*-simplex*
2. v *is a top vertex for a* k*-simplex*

Top Vertices

Lemma

Let $v \in G_0$, and A be the adjacency matrix associated with the graph G. If

$$\left(A + A^T\right)_{vv}^{k+1} \geq 4\sum_{n=0}^{k} \binom{k}{n}$$

and $\left|N_1^{in}(u) \cap N_1^{in}(v)\right| \geq k-1$, then the following are equivalent:

1. *$\exists u \in N_1^{in}(v)$ such that u is a top vertex for a $(k-1)$-simplex*
2. *v is a top vertex for a k-simplex*

Way around: $(A \bullet B)_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$

Top Vertices

Lemma

Let $v \in G_0$, and A be the adjacency matrix associated with the graph G. If

$$\left(A + A^T\right)_{vv}^{k+1} \geq 4\sum_{n=0}^{k} \binom{k}{n}$$

and $\left|N_1^{in}(u) \cap N_1^{in}(v)\right| \geq k-1$, then the following are equivalent:

1. *$\exists u \in N_1^{in}(v)$ such that u is a top vertex for a $(k-1)$-simplex*
2. *v is a top vertex for a k-simplex*

Way around: $(A \bullet B)_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$

Lemma

Let G be any directed graph. Then the i,j entry of $\widetilde{A} \odot \widetilde{A^{\bullet 2}} \odot \ldots \odot \widetilde{A^{\bullet k}}$, denoted by $\widetilde{a}_{ij}^{(k)}$, nonzero if and only if there is a path of length 1, length 2, ..., length k from vertex i to vertex j without repeating any vertices.

Here, $\widetilde{X} := X \oplus diag(X)$

Top Vertices

Lemma

Let $v \in G_0$. Then $\widetilde{a}_{iv}^{(2)}$ is nonzero if and only if v is a top 2 vertex.

Top Vertices

Lemma

Let $v \in G_0$. Then $\widetilde{a}_{iv}^{(2)}$ is nonzero if and only if v is a top 2 vertex.

Lemma

If $\widetilde{a}_{iu}^{(2)} \neq 0$, $\widetilde{a}_{iv}^{(3)} \neq 0$, $u \in \widetilde{N}_{in}^{1}(v)$ (i.e., u and v are top 2-vertices) and $\widetilde{N}_{out}^{1}(i) \cap \widetilde{N}_{in}^{1}(u) \cap \widetilde{N}_{in}^{1}(v)$ is nonempty, then v is a top 3-vertex

Top Vertices

Lemma

Let $v \in G_0$. Then $\widetilde{a}_{iv}^{(2)}$ is nonzero if and only if v is a top 2 vertex.

Lemma

If $\widetilde{a}_{iu}^{(2)} \neq 0$, $\widetilde{a}_{iv}^{(3)} \neq 0$, $u \in \widetilde{N}_{in}^{1}(v)$ (i.e., u and v are top 2-vertices) and $\widetilde{N}_{out}^{1}(i) \cap \widetilde{N}_{in}^{1}(u) \cap \widetilde{N}_{in}^{1}(v)$ is nonempty, then v is a top 3-vertex

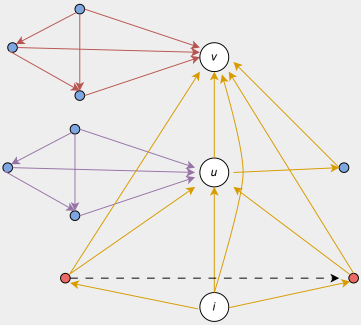
Lemma

For three vertices u, v, w with $u \in \widetilde{N}_{in}^{1}(v)$ and $w \in \widetilde{N}_{in}^{1}(v) \cap \widetilde{N}_{in}^{1}(u)$, if $\widetilde{a}_{iv}^{(4)}$ and $\widetilde{a}_{iu}^{(3)}$ and $\widetilde{a}_{iw}^{(2)}$ are nonzero, and if $\widetilde{N}_{out}^{1}(i) \cap \widetilde{N}_{in}^{1}(v) \cap \widetilde{N}_{in}^{1}(u)$ is nonempty, then $[i, x, w, u, v]$ is a 4 simplex for all $x \in \widetilde{N}_{out}^{1}(i) \cap \widetilde{N}_{in}^{1}(v) \cap \widetilde{N}_{in}^{1}(u) \cap \widetilde{N}_{in}^{1}(w)$

Pseudo Top Vertices

Definition

For $v \in G_0$, and any integer $k \geq 1$, u is said to be a k-pseudotop vertex if $\exists u \in \widetilde{N}^1_{in}(v)$ that is a $(k-1)$-semitop vertex such that $\left|\widetilde{N}^1_{out}(i) \cap \widetilde{N}^1_{in}(u) \cap \widetilde{N}^1_{in}(v)\right| > k-3$, where i is a vertex such that $\widetilde{a}^{(k-1)}_{iu}$ and $\widetilde{a}^{(k)}_{iv}$ are nonzero. For $k = 0$, all vertices are defined to be 0-semitop vertices. The integer k is said to be the dimension of the pseudotop vertex.



Pseudotop Vertex Neural Network

Algorithm 7 Required pre-processing for PTVNN

1: Find pseudotop vertices
2: Label every vertex v with its maximum dimension
3: Partition vertices based on dimension
4: Form partition vector $R(v) = ([v], [w_1], ..., [w_n])$, where $w_i \in N_{in}(v)$ {Refine partition}
5: If $R(v) = R(u)$, then $[u] = [v]$.
6: Repeat until refinement stabilizes

$$
\begin{aligned}
\mathbf{a}_v^{t+1} &= \text{AGG}\left(\mathbf{x}_v^t, \mathbf{x}_u^t : u \in N_{in}(v)\right) \\
\mathbf{x}_v^{t+1} &= \text{COMBINE}\left(\mathbf{x}_v^t, \mathbf{a}_v^{t+1}\right)
\end{aligned}
$$

Pseudotop Vertex Neural Network

Architecture 1

- Make one hot encoding of partition p after refinement
- If x_v is feature vector of v, then form $[x_v.p]$

Pseudotop Vertex Neural Network

Architecture 1

- Make one hot encoding of partition p after refinement
- If x_v is feature vector of v, then form $[x_v.p]$

Architecture 2

- Make one hot encoding of vertex dimension d
- Make one hot encoding of refinement index r
- If x_v is feature vector of v, then form $[x_v.d.r]$

Pseudotop Vertex Neural Network

Architecture 1

- Make one hot encoding of partition p after refinement
- If x_v is feature vector of v, then form $[x_v.p]$

Architecture 2

- Make one hot encoding of vertex dimension d
- Make one hot encoding of refinement index r
- If x_v is feature vector of v, then form $[x_v.d.r]$

Architecture 3

- Make multi hot encoding of vertex dimension d
- Make multi hot encoding of partition indices k
- If x_v is feature vector of v, then form $[x_v.k.d]$

Implementation

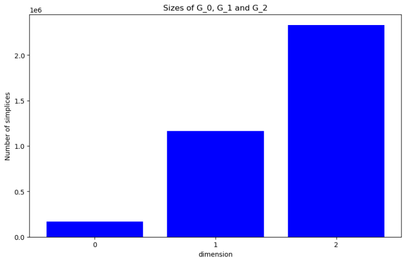
Dataset: Directed citation network[HFZ+20] with 40 subject areas.
Task: subject area classification.
Code: https://abdullahnaeemmalik.github.io/portfolio/ptvnn/

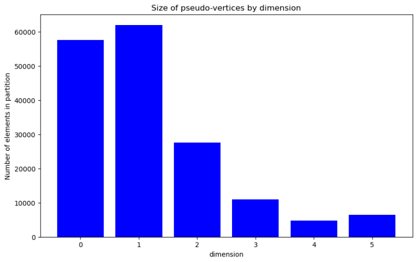
Implementation

Dataset: Directed citation network[HFZ+20] with 40 subject areas.
Task: subject area classification.
Code: https://abdullahnaeemmalik.github.io/portfolio/ptvnn/
The training is performed on the papers published until 2017, validated on those published in 2018, and tested on those published since 2019.

Implementation

Dataset: Directed citation network[HFZ+20] with 40 subject areas.
Task: subject area classification.
Code: https://abdullahnaeemmalik.github.io/portfolio/ptvnn/
The training is performed on the papers published until 2017, validated on those published in 2018, and tested on those published since 2019.
Number of nodes=169343. Number of edges=1166243. Number of 2-simplices=2332322





References I

- Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein, *Weisfeiler and lehman go topological: Message passing simplicial networks*, International Conference on Machine Learning, PMLR, 2021, pp. 1026–1037.
- Jin-Yi Cai, Martin Fürer, and Neil Immerman, *An optimal lower bound on the number of variables for graph identification*, Combinatorica **12** (1992), no. 4, 389–410.
- Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec, *Open graph benchmark: Datasets for machine learning on graphs*, arXiv preprint arXiv:2005.00687 (2020).
- Ningyuan Teresa Huang and Soledad Villar, *A short tutorial on the weisfeiler-lehman test and its variants*, ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 8533–8537.
- Jacob Lurie, *Kerodon*, `https://kerodon.net`, 2023.

References II

- Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman, *Provably powerful graph networks*, Advances in neural information processing systems **32** (2019).
- Martin Mladenov Pascal Schweitzer Martin Grohe, Kristian Kersting, *Color refinement and its applications*, An Introduction to Lifted Probabilistic Inference (Sriraam Natarajan Davide Poole Guy Van den Broeck, Kristian Kersting, ed.), MIT Press, 2021.
- Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe, *Weisfeiler and leman go neural: Higher-order graph neural networks*, Proceedings of the AAAI conference on artificial intelligence, vol. 33, 2019, pp. 4602–4609.
- Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael Bronstein, *Edge directionality improves learning on heterophilic graphs*, arXiv preprint arXiv:2305.10498 (2023).

References III

- Boris Weisfeiler and Andrei Leman, *The reduction of a graph to canonical form and the algebra which appears therein*, nti, Series **2** (1968), no. 9, 12–16.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, *How powerful are graph neural networks?*, arXiv preprint arXiv:1810.00826 (2018).