Weisfeiler-Lehman use Simplicial Sets

PseudoTop Vertex Neural Network

Abdullah Naeem Malik

49th Annual New York State Regional
Graduate Mathematics Conference
Syracuse University

March XXIII, 2024

Outline

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

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

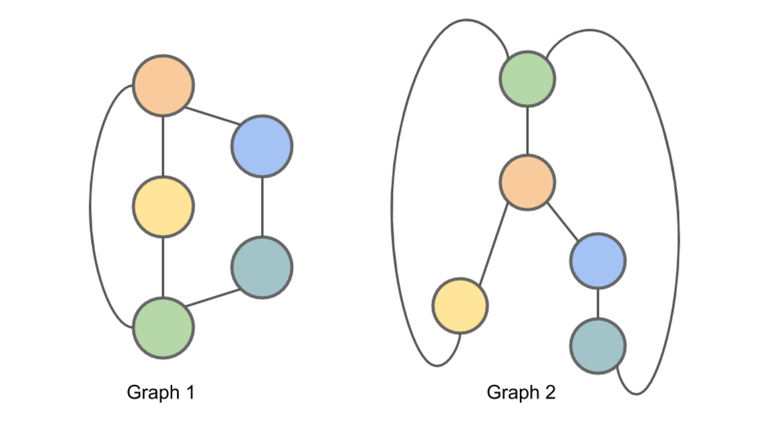
A **coloring** of a graph $G = (V, E)$ is a function $c : V \longrightarrow \mathbb{N}$.

Definition

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

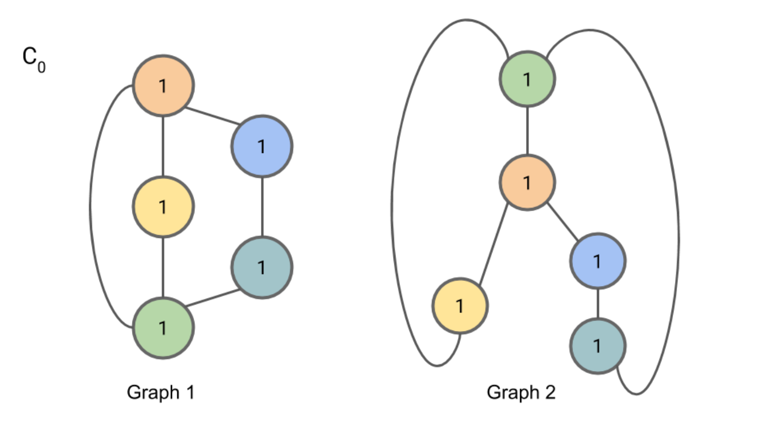
Basic idea: start with $c(v) = c^{(0)}(v)$, and
$c^{(t+1)}(v) = \text{hash}\left(c^{(t)}(v), \{\{c^{(t)}(w) : w \in N(v)\}\}\right)$[WL68]

Weisfeiler-Lehman Algorithm



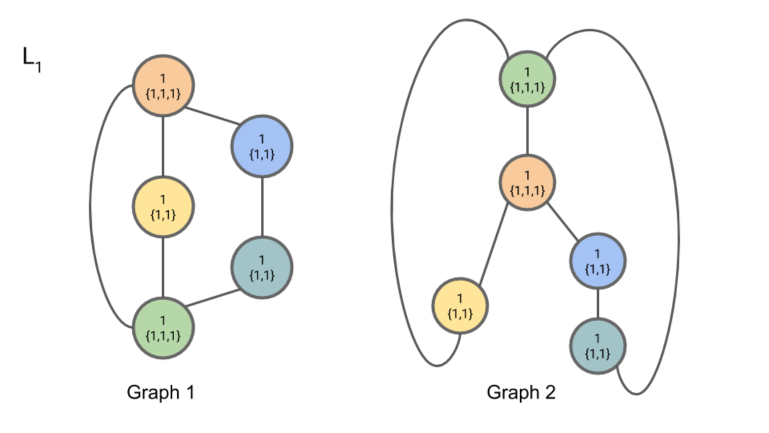
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



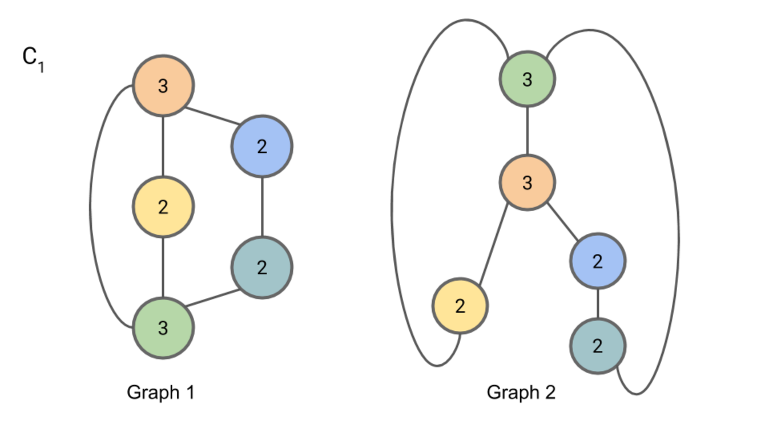
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



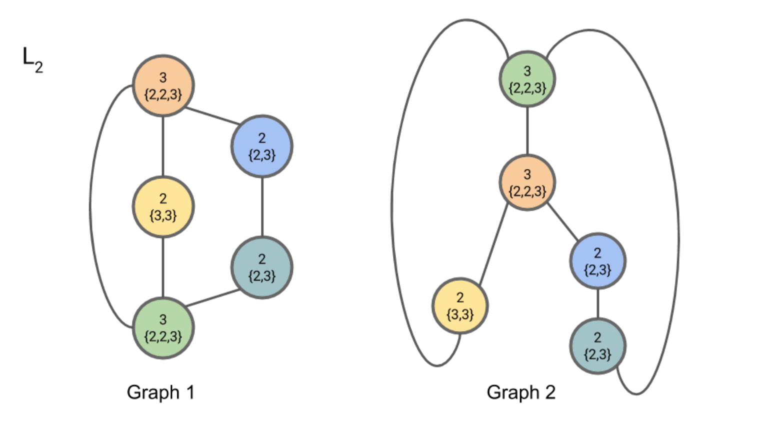
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



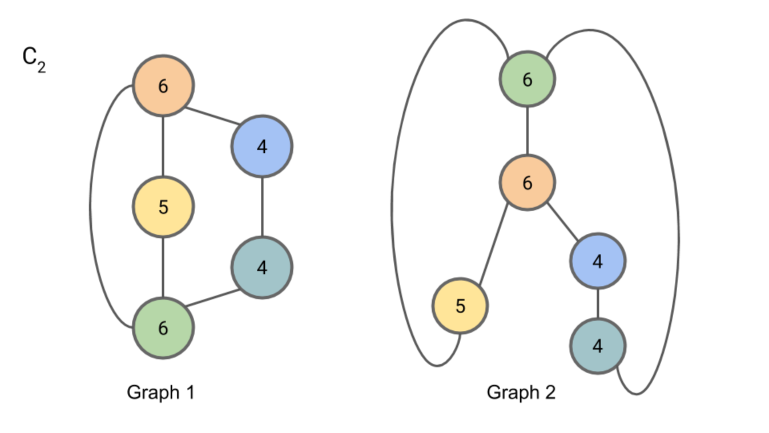
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



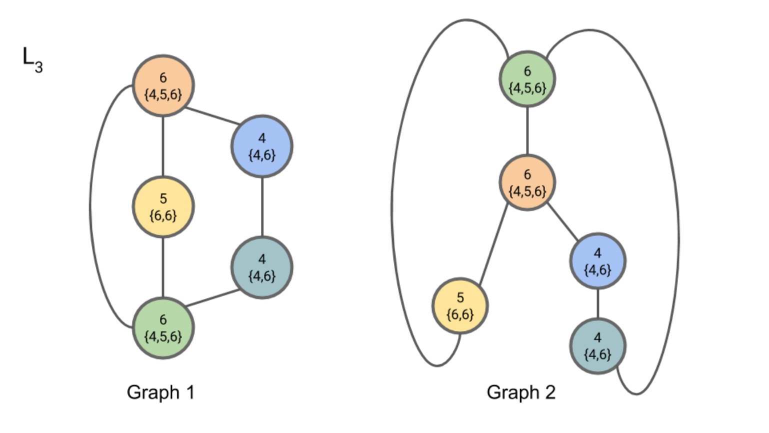
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Weisfeiler-Lehman Algorithm



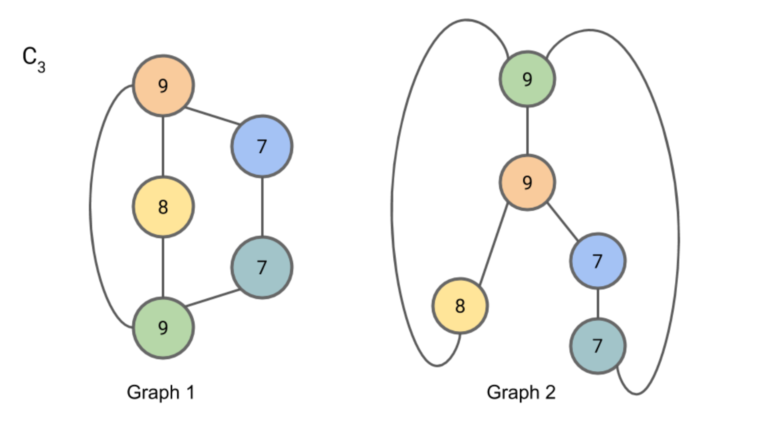
Source:
https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

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

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**
    $c^{(t+1)}(v) \longleftarrow \text{hash}\big(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\big)$
end while
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: $c^{(T)}(v)\ \forall v \in V$

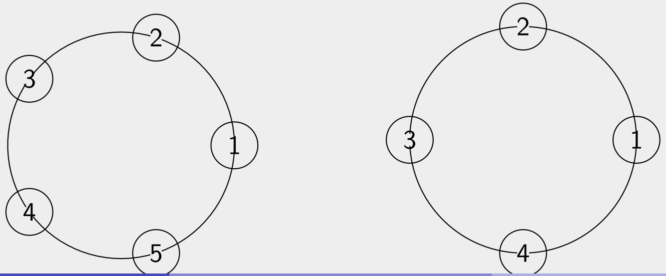
If two graphs have different colorings, then the graphs are not isomorphic. But the converse is not true[MBHSL19]

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**
    $c^{(t+1)}(v) \longleftarrow \text{hash}\big(c^{(t)}(v), \{\!\{c^{(t)}(w) : w \in N(v)\}\!\}\big)$
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 $f(h_v) = x_v$

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

$$x_v^{(k+1)} = COMBINE^{(k+1)}\left(x_v^{(k)}, a_v^{(k+1)}\right)$$

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..

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

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].

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 (undirected) 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].

DL $\sqsubset$ 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]

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**
    $c_i^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \{\!\{c^{(t)}\left(\overrightarrow{w}\right) : w \in N_i\left(\overrightarrow{v}\right)\}\!\}\ \forall \overrightarrow{v} \in V^k$
    $c^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \text{hash}\left(c^{(t)}\left(\overrightarrow{v}\right), c_1^{(t+1)}\left(\overrightarrow{v}\right), ..., c_k^{(t+1)}\left(\overrightarrow{v}\right)\right)\ \forall \overrightarrow{v} \in V^k$
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 $N_i\left(\overrightarrow{v}\right) = \{(v_1, ..., v_{i-1}, u, v_{i+1}, ..., v_k) : u \in V\}$

k-Weisfeiler-Lehman Algorithm

Algorithm 2 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**

  $c_i^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \{\{c^{(t)}\left(\overrightarrow{w}\right) : w \in N_i\left(\overrightarrow{v}\right)\}\} \; \forall \overrightarrow{v} \in V^k$

  $c^{(t+1)}\left(\overrightarrow{v}\right) \longleftarrow \text{hash}\left(c^{(t)}\left(\overrightarrow{v}\right), c_1^{(t+1)}\left(\overrightarrow{v}\right), ..., c_k^{(t+1)}\left(\overrightarrow{v}\right)\right) \; \forall \overrightarrow{v} \in V^k$

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 $N_i\left(\overrightarrow{v}\right) = \{(v_1, ..., v_{i-1}, u, v_{i+1}, ..., v_k) : u \in V\}$

May be used to make GNN kernels

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 $N_i\left(\overrightarrow{v}\right) = \{(v_1, ..., v_{i-1}, u, v_{i+1}, ..., v_k) : u \in V\}$

May be used to make GNN kernels

$(k+1)$-WL $\sqsubset$ k-WL[HV21]

The case for higher order relations

Clique complexes of graphs to the rescue!

Simplicial WL [BFW+21] uses

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

The case for higher order relations

Clique complexes of graphs to the rescue!

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]

The case for higher order relations

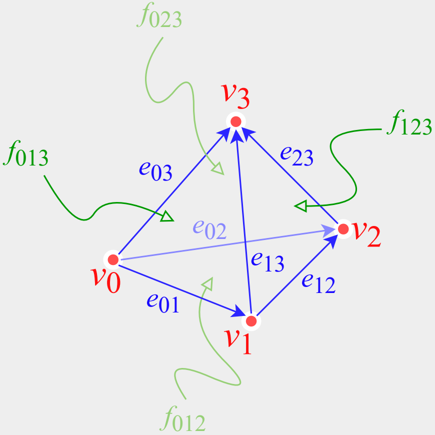
Clique complexes of graphs to the rescue!

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]

What if we go for cliques in directed graphs?



Modelling higher relations

Theorem

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

Modelling higher relations

Theorem

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

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

Modelling higher relations

Theorem

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

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

In summary:

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

Top Vertices

Definition

A vertex $v \in G_0$ is said to be a **top vertex** of dimension k if there is a simplex $x \in X_\bullet$ of dimension k such that $d_0^{(1)} d_0^{(2)} \ldots d_0^{(k-1)} d_0^{(k)} x = v$, where $d_0^{(j)} : X_j \longrightarrow X_{j-1}$ is the 0-th face map for $0 \leq j \leq j$

Top Vertices

Definition

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

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

Top Vertices

Definition

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

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

Lemma

Let G be any directed graph. Then the i, j entry of $\widetilde{A} \odot \widetilde{A^{\bullet 2}} \odot ... \odot \widetilde{A^{\bullet k}}$, 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*

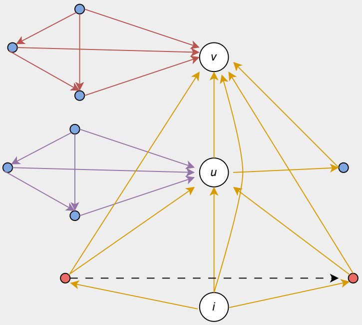
Lemma

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 $k \geq 1$, v 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 4 Required pre-processing

1: Find pseudotop vertices
2: Label every vertex v with its (pure) maximum dimension
3: **while** Refinement Stabilizes **do**
4: Form partition vector $R(v) = ([v], [w_1], ..., [w_n])$, where $w_i \in N_{in}(v)$
5: If $R(v) = R(u)$, then $[u] = [v]$.
6: **end while**

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.

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.

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.

References II

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).