Graph Neural Networks and Arithmetic Circuits

Laura Strieker
Paper by:
T. Barlag, V. Holzapfel, L. Strieker, J. Virtema, H. Vollmer

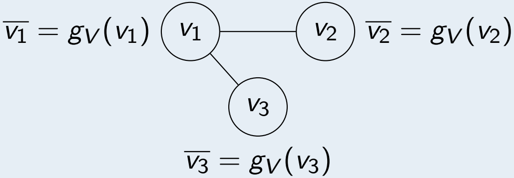
Institut für Theoretische Informatik
Leibniz Universität Hannover

03.03.25

Preliminaries

Labeled graph

- Undirected Graph $\mathfrak{G} = (V, E, g_V)$ with labeling function $g_V : V \mapsto \mathbb{R}^k$
- Vectors belonging to nodes, called *feature vectors*



Graph Neural Networks - AC-GNN

Definition

An L layer *aggregate combine graph neural network* (AC-GNN) is a tuple $\mathcal{D} = (\{\text{AGG}^{(i)}\}_{i=1}^{L}, \{\text{COM}^{(i)}\}_{i=1}^{L}, \{\sigma^{(i)}\}_{i=1}^{L}, \text{CLS})$, where

- $\{\text{AGG}^{(i)}\}_{i=1}^{L}$: aggregate functions
- $\{\text{COM}^{(i)}\}_{i=1}^{L}$: combine functions
- $\{\sigma^{(i)}\}_{i=1}^{L}$: activation functions
- CLS: $\mathbb{R}^k \to \{0, 1\}$: classification function

Graph Neural Networks - AC-GNN

Definition

An L layer *aggregate combine graph neural network* (AC-GNN) is a tuple $\mathcal{D} = (\{\mathrm{AGG}^{(i)}\}_{i=1}^{L}, \{\mathrm{COM}^{(i)}\}_{i=1}^{L}, \{\sigma^{(i)}\}_{i=1}^{L}, \mathrm{CLS})$, where

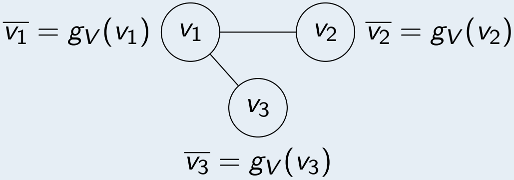
- $\{\mathrm{AGG}^{(i)}\}_{i=1}^{L}$: aggregate functions
- $\{\mathrm{COM}^{(i)}\}_{i=1}^{L}$: combine functions
- $\{\sigma^{(i)}\}_{i=1}^{L}$: activation functions
- CLS: $\mathbb{R}^k \to \{0, 1\}$: classification function

Updating vectors in every layer $i \leq L$ as follows:

- initial feature vector of v: $\overline{v}^{(0)} = g_V(v)$
- for $1 \leq i \leq L$:
  - $\overline{y} = \mathrm{AGG}^{(i)}\left(\{\!\{\overline{u}^{(i-1)} \mid u \in N_{\mathfrak{G}}(v)\}\!\}\right)$
  - $\overline{v}^{(i)} = \sigma^{(i)}\left(\mathrm{COM}^{(i)}\left(\overline{v}^{(i-1)}, \overline{y}\right)\right)$

Labeled graph

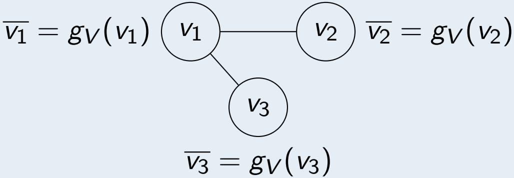
- Undirected Graph $\mathfrak{G} = (V, E, g_V)$ with labeling function $g_V : V \mapsto \mathbb{R}^k$
- Vectors belonging to nodes, called *feature vectors*



Graph Neural Networks

Labeled graph

- Undirected Graph $\mathfrak{G} = (V, E, g_V)$ with labeling function $g_V : V \mapsto \mathbb{R}^k$
- Vectors belonging to nodes, called *feature vectors*

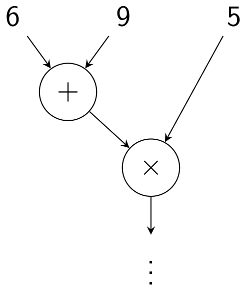


$$\overline{v_1}^{(i)} = \sigma^{(i)}\left(\mathrm{COM}^{(i)}\left(\overline{v_1}^{(i-1)}, \mathrm{AGG}^{(i)}\left(\{\!\{\overline{v_2}^{(i-1)}, \overline{v_3}^{(i-1)}\}\!\}\right)\right)\right)$$

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



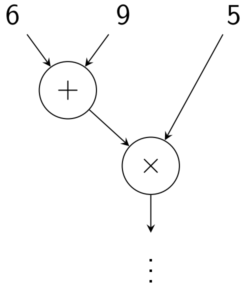
Circuit Function Classes

- $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



But we want to compare arithmetic circuits to Graph Neural Networks.
How to deal with activation functions?

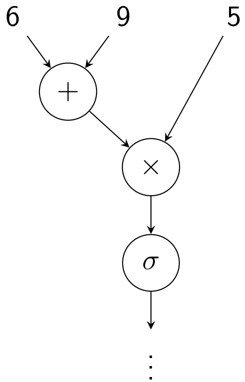
Circuit Function Classes

- $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



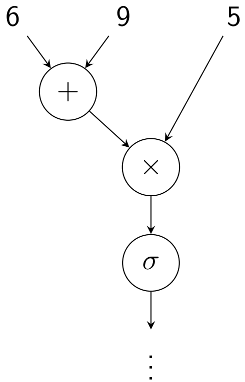
Circuit Function Classes

- $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



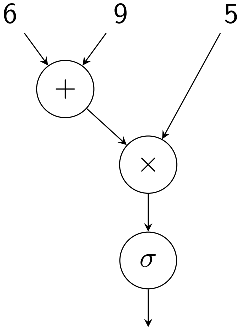
Circuit Function Classes

- $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size
- $\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$: additional function gates

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



Circuit Function Classes

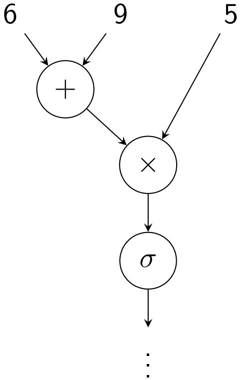
- $\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size
- $\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$: additional function gates

Nodes in a GNN aggregate the values of their neighbors, regardless of their order. Our circuits should mimic this behavior.

$\mathbb{R}^k$-Arithmetic Circuits

Definition

Directed acyclic graph with gates that perform arithmetic operations over $\mathbb{R}^k$.



Circuit Function Classes

- $t\mathrm{FAC}^0_{\mathbb{R}^k}$: constant depth & polynomial size
- $t\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$: additional function gates

t = *tailsymmetric*

Graph Neural Networks using Circuits

Model of Computation: Circuit-GNN

- instead of aggregate/combine functions: circuit families of a specific circuit function class $\mathfrak{F}$
- defined set of activation functions $\mathcal{A}$
- a function assigning a circuit family and activation function for every layer
- "a GNN with circuits in its nodes"
- $(\mathfrak{F}, \mathcal{A})$-GNN, e.g. $(\mathrm{FAC}^0_{\mathbb{R}}, \{\mathit{id}\})$-GNN

Simulating C-GNNs with Arithmetic Circuits

Theorem

Let $\mathcal{N}$ be a $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \{\mathrm{id}\} \cup \mathcal{A}\right)$-GNN. Then there exists an $\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-circuit family $\mathcal{C}$, such that for all labeled graphs $\mathfrak{G}$ the circuit family computes the same feature vectors as the C-GNN.

- idea: roll out the circuits to one big one
- simulate the circuits of each layer
- use function gates for the activation functions in $\mathcal{A}$

Simulating Arithmetic Circuits with C-GNNs

Goal

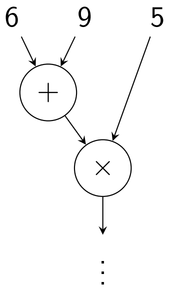
Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a circuit family of some circuit function class. Show there exists a C-GNN $\mathcal{N}$ that has the output of the circuit family among its feature vectors.

- number of layers in $\mathcal{N}$ = depth of $\mathcal{C}$.
- use the graph structure of the circuit as the input graph of the C-GNN
- simulate the computation of the circuit layerwise in the C-GNN and store the intermediate results in the feature vectors

Circuits without additional function gates

Theorem

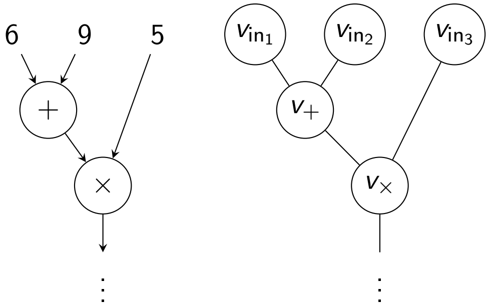
$\mathrm{FAC}^0_{\mathbb{R}^k}$-*circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \{\mathrm{id}\}\right)$-*GNN*



Circuits without additional function gates

Theorem

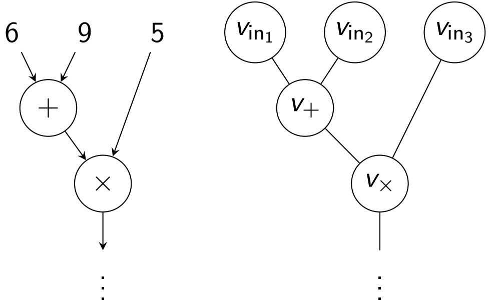
$\mathrm{FAC}^0_{\mathbb{R}^k}$*-circuit family* $\rightarrow$ $(t\mathrm{FAC}^0_{\mathbb{R}^k}, \{\mathrm{id}\})$*-GNN*



Circuits without additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}$*-circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \{\mathrm{id}\}\right)$*-GNN*



- no additional function gates on the circuit side $\Longrightarrow$ no activation functions needed in the C-GNN (besides identity)

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}], \{\mathrm{id}\}\right)$-*GNN*

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-*GNN*

- more natural to have $\mathcal{A}$ as activation functions than in the nodes

Circuits with additional function gates

Theorem

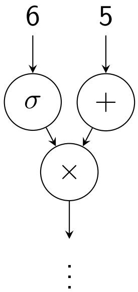
$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\to$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-*GNN*

- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-*GNN*

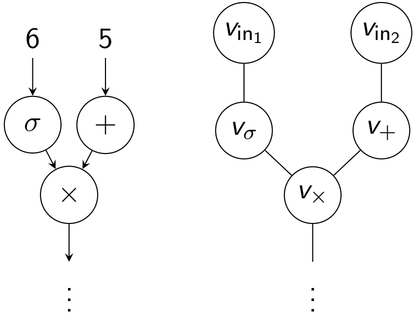


- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-*GNN*

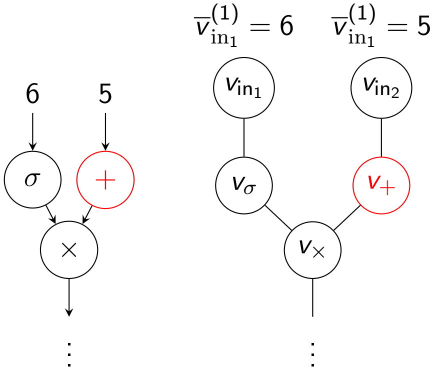


- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$*-circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$*-GNN*

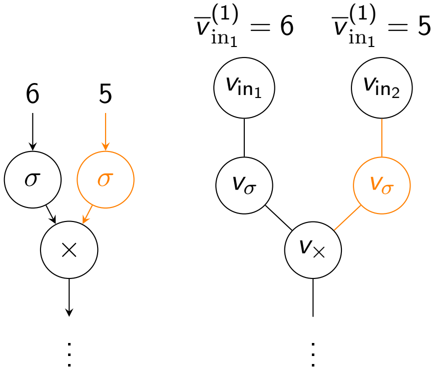


- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector

Circuits with additional function gates

Theorem

$\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$-*circuit family* $\to$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$-*GNN*

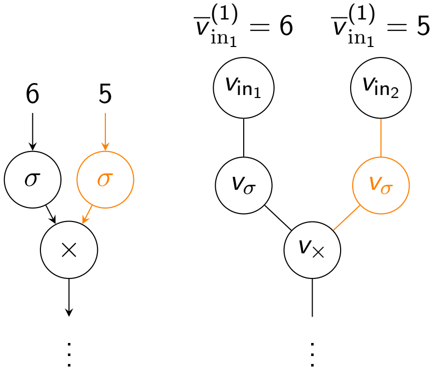


- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector
- we can only simulate circuits in a function layer form

Circuits with additional function gates

Theorem

$f\mathrm{FAC}^0_{\mathbb{R}^k}[\mathcal{A}]$*-circuit family* $\rightarrow$ $\left(t\mathrm{FAC}^0_{\mathbb{R}^k}, \mathcal{A} \cup \{\mathrm{id}\}\right)$*-GNN*



- more natural to have $\mathcal{A}$ as activation functions than in the nodes
- but the function now has to be applied to every feature vector
- we can only simulate circuits in a function layer form

Conclusion and Open Questions

Conclusion and Open Questions

- correspondence between arithmetic circuits and a generalization of graph neural networks using circuits
  - Can the imposed restrictions be made on "both sides of the simulation"?

Conclusion and Open Questions

- correspondence between arithmetic circuits and a generalization of graph neural networks using circuits
  - Can the imposed restrictions be made on "both sides of the simulation"?
- circuit function classes that characterize the computational power of a C-GNN
  - complexity of these classes
  - Which functions are "more" complex than others?

Conclusion and Open Questions

- correspondence between arithmetic circuits and a generalization of graph neural networks using circuits
  - Can the imposed restrictions be made on "both sides of the simulation"?
- circuit function classes that characterize the computational power of a C-GNN
  - complexity of these classes
  - Which functions are "more" complex than others?

Thank You