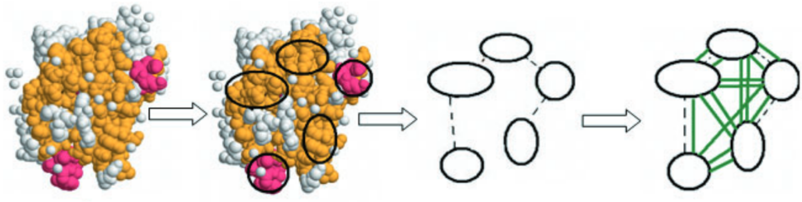
Graph Classification with Spatial GNNs

from How powerful are Graph Neural Networks?

by Lucas Kania

Graph Classification



Outline

- Describe GNNs
- Introduction to the Graph Isomorphism Problem and the WL Test
- Prove that GNN is at most as powerful as the WL Test for differentiating Graphs
- Derive an optimal architecture and approximate it with a NN
- Experiments and extensions

Graph Neural Networks

Given a graph $G = (V, E)$ and $h_v^0 = X_v$

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\})$$

Graph Neural Networks

Given a graph $G = (V, E)$ and $h_v^0 = X_v$

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\})$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

Graph Neural Networks

Given a graph $G = (V, E)$ and $h_v^0 = X_v$

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\})$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

$$h_G = \text{READOUT}(\{h_v^{(K)} | v \in V\})$$

Connection with the Graph Isomorphism Problem

Problem: Given two graphs, is there an isomorphism between them? (i.e. is there a re-labeling that makes them equal?)

Connection with the Graph Isomorphism Problem

Heuristic: Weisfeiler-Lehman Graph Isomorphism Test (WL Test)

function WL(G=(V,E))
  for $v \in V$ **do**
    $h_v = degree(v)$

  while label **did not** convergence **do**
    for $v \in V$ **do**
      $a_v = \text{AGGREGATE}(\{h_u | u \in N(v)\})$
      $h_v = \text{COMBINE}(h_v, a_v)$
return $\{h_v | v \in V\}$

$$\text{WL}(G) \neq \text{WL}(G') \implies G \not\equiv G'$$

where AGGREGATE and COMBINE are injective functions
(i.e. $\forall a, b \in X, \;\; a \neq b \Rightarrow f(a) \neq f(b)$)

GNNs and the WL Test

$$\text{GNN}(G) \neq \text{GNN}(G') \implies \text{WL}(G) \neq \text{WL}(G')$$

There doesn't exist a pair of graph that a GNN can differentiate, and the WL test cannot. Thus, A GNN is at most as powerful as the WL test.

GNNs and the WL Test

$$\mathrm{GNN}(G) \neq \mathrm{GNN}(G') \implies \mathrm{WL}(G) \neq \mathrm{WL}(G')$$

There doesn't exist a pair of graph that a GNN can differentiate, and the WL test cannot. Thus, A GNN is at most as powerful as the WL test.

We would like to have

$$\mathrm{GNN}(G) \neq \mathrm{GNN}(G') \iff \mathrm{WL}(G) \neq \mathrm{WL}(G')$$

GNNs and the WL Test

$$\mathrm{GNN}(G) \neq \mathrm{GNN}(G') \iff \mathrm{WL}(G) \neq \mathrm{WL}(G')$$

It's true if AGGREGATE, COMBINE, and READOUT are injective.

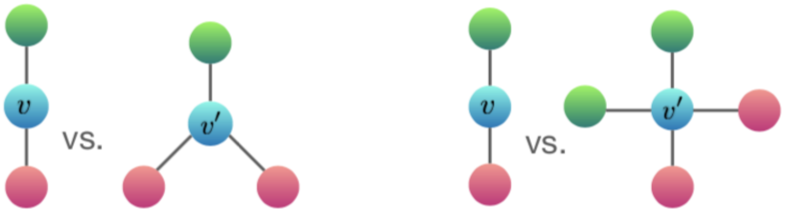
GNN architecture

$$a_v^{(k)} = \mathrm{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\})$$

$$h_v^{(k)} = \mathrm{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

$$h_G = \mathrm{READOUT}(\{h_v^{(K)} | v \in V\})$$

Injectivity in AGGREGATE



Finding the Injective Functions for an optimal GNN

Given that the input space is countable (e.g. one hot encoded categorical data), there exist injective functions ψ and ϕ s.t. g is an injective function over multi-sets

$$g(h_v, \{h_u | u \in N(v)\}) = \phi\left((1+\epsilon)\psi(h_v) + \sum_{u \in N(v)} \psi(h_u)\right)$$

Finding the Injective Functions for an optimal GNN

$$g(h_v, \{h_u | u \in N(v)\}) = \phi\left((1+\epsilon)\psi(h_v) + \sum_{u \in N(v)} \psi(h_u)\right)$$

GNN architecture

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\}) = \sum_{u \in N(v)} \psi(h_u^{(k-1)})$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) = \phi\left((1+\epsilon)\psi(h_v^{(k-1)}) + a_v^{(k)}\right)$$

Finding the Injective Functions for an optimal GNN

$$g(h_v, \{h_u | u \in N(v)\}) = \phi\left((1+\epsilon)\psi(h_v) + \sum_{u \in N(v)} \psi(h_u)\right)$$

GNN architecture

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\}) = \sum_{u \in N(v)} h_u^{(k-1)}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) = \text{MLP}^{(k)}\left((1+\epsilon)h_v^{(k-1)} + a_v^{(k)}\right)$$

Finding the Injective Functions for an optimal GNN

$$g(h_v, \{h_u | u \in N(v)\}) = \phi\left((1+\epsilon)\psi(h_v) + \sum_{u \in N(v)} \psi(h_u)\right)$$

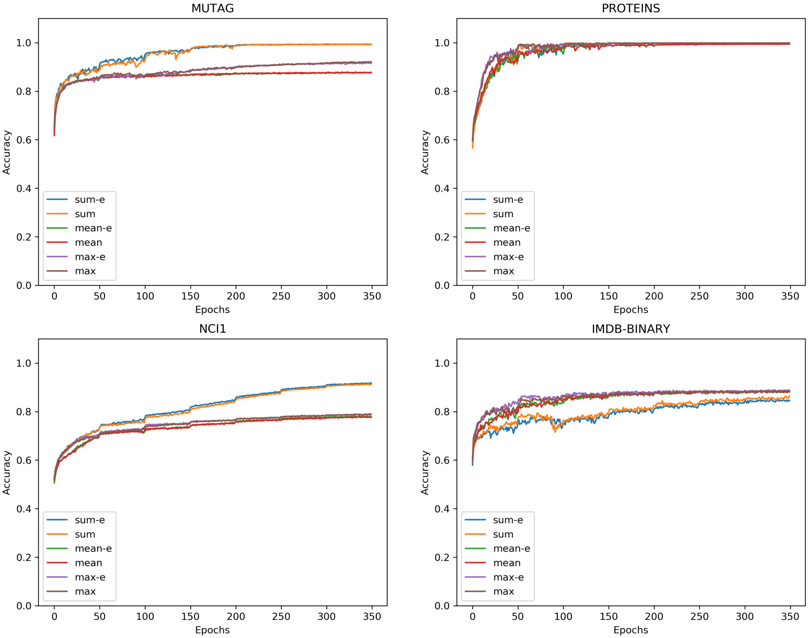
GNN architecture

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} | u \in N(v)\}) = \sum_{u \in N(v)} h_u^{(k-1)}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)}) = \text{MLP}^{(k)}\left((1+\epsilon)h_v^{(k-1)} + a_v^{(k)}\right)$$

$$h_G = \text{READOUT}(\{h_v^{(K)} | v \in V\}) = \sum_{k}^{K} W \sum_{v \in V} h_v^{(k)}$$

Experiments



Experiments

name	MUTAG	PROTEINS	NCI1
sum-e	0.883 ± 0.099	0.661 ± 0.028	0.765 ± 0.023
sum	0.889 ± 0.044	0.660 ± 0.037	0.760 ± 0.022
mean-e	0.835 ± 0.069	0.652 ± 0.027	0.716 ± 0.016
mean	0.852 ± 0.066	0.622 ± 0.066	0.717 ± 0.022
max-e	0.813 ± 0.095	0.648 ± 0.035	0.717 ± 0.014
max	0.814 ± 0.092	0.668 ± 0.035	0.724 ± 0.037

Extensions

- Use an irrational number approximation.
- Derive a different COMBINE function that doesn't require learning an irrational number.
- Force the network to learn injective functions.
- Implement variational inference (Bayesian Graph Neural Network) to obtain a posterior over the weights.

Questions?