principled simplicial neural networks for trajectory prediction

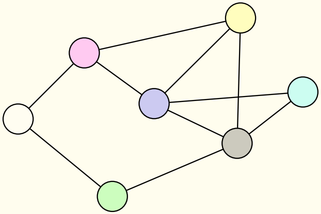
T. Mitchell Roddenberry,* Nicholas Glaze,* Santiago Segarra
mitch@rice.edu
mitch.roddenberry.xyz

Electrical and Computer Engineering
Rice University
ICML 2021

*equal contribution

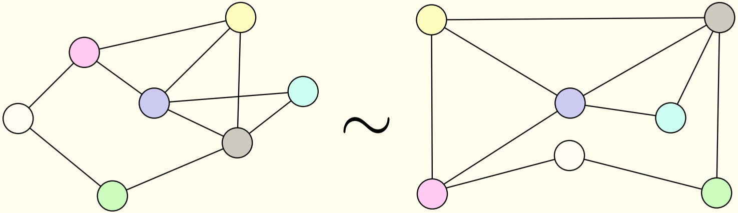
graph neural networks

- Sequence of local aggregations and activation functions



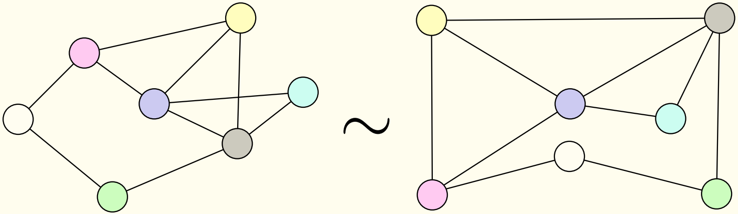
graph neural networks

- Sequence of local aggregations and activation functions
- ***Permutation-equivariance***



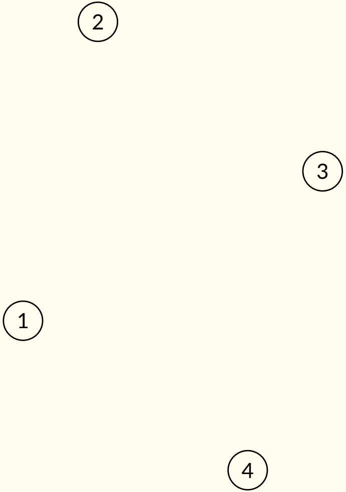
graph neural networks

- Sequence of local aggregations and activation functions
- ***Permutation-equivariance***



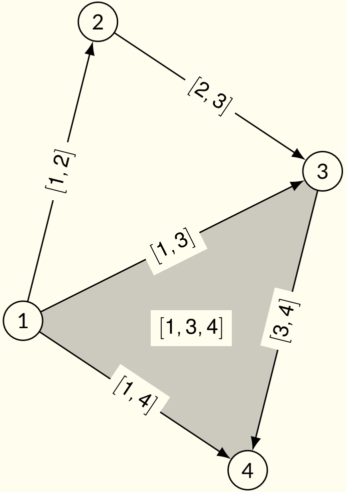
Enforcing appropriate ***symmetries*** helps with ***generalization***

abstract simplicial complexes



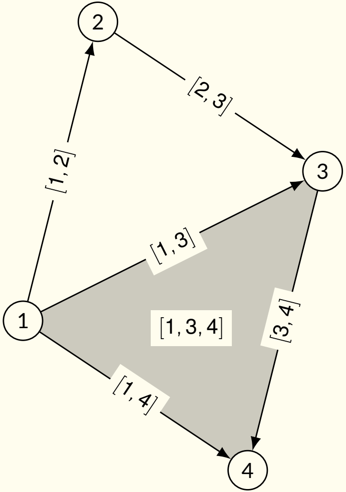
- Start with a set of nodes X_0

abstract simplicial complexes



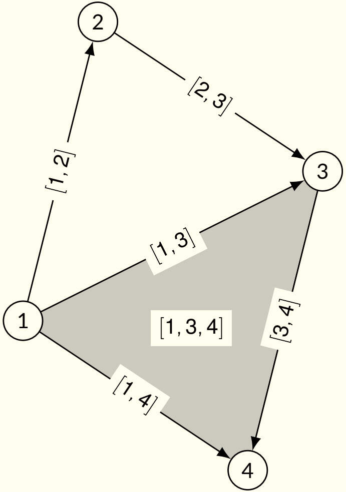
- Start with a set of nodes X_0
- Take finite subsets of X_0

abstract simplicial complexes



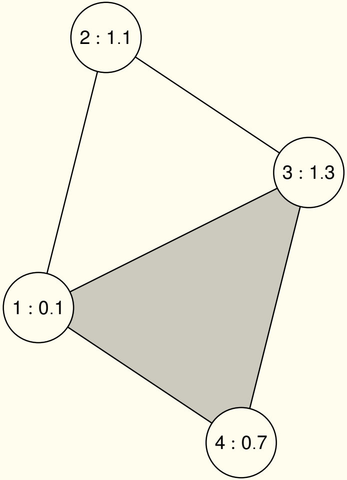
- Start with a set of nodes X_0
- Take finite subsets of X_0
- ***Abstract simplicial complex***: a collection X of finite subsets of X_0 that is closed under restriction

abstract simplicial complexes



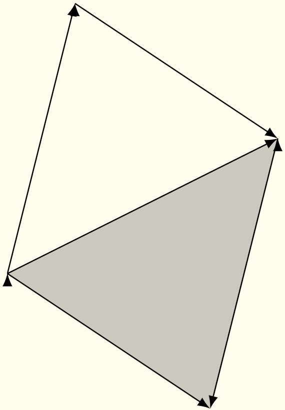
- Start with a set of nodes X_0
- Take finite subsets of X_0
- ***Abstract simplicial complex***: a collection X of finite subsets of X_0 that is closed under restriction
- X_k: collection of elements of X with cardinality $k+1$

chains on simplicial complexes



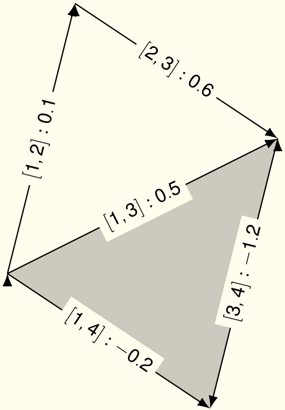
- Graph neural networks act on ***graph signals***: nodal data

chains on simplicial complexes



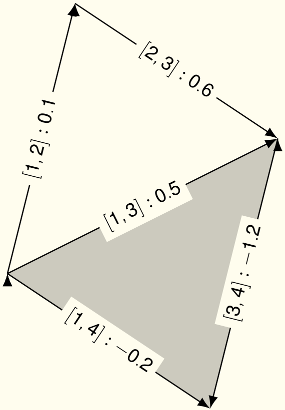
- Graph neural networks act on ***graph signals***: nodal data
- What kind of data lives on a simplicial complex?

chains on simplicial complexes



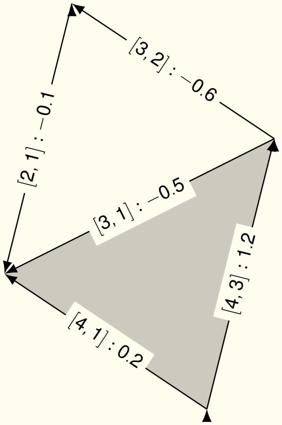
- Graph neural networks act on ***graph signals***: nodal data
- What kind of data lives on a simplicial complex?
- k-chains are elements of the vector space $\mathcal{C}_k$

chains on simplicial complexes



- Graph neural networks act on ***graph signals***: nodal data
- What kind of data lives on a simplicial complex?
- k-chains are elements of the vector space $\mathcal{C}_k$
- Attach real numbers to the ***oriented k-simplices***

chains on simplicial complexes



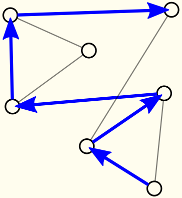
- Graph neural networks act on ***graph signals***: nodal data
- What kind of data lives on a simplicial complex?
- k-chains are elements of the vector space $\mathcal{C}_k$
- Attach real numbers to the ***oriented k-simplices***
- ***Enforce skew-symmetry:*** $[i_0, i_1] = -[i_1, i_0]$

example: flows on a graph

- Consider a ***flow*** of resources over a graph ($\mathcal{C}_1$)

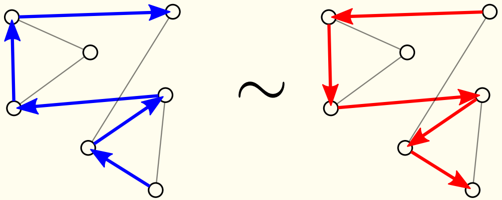
example: flows on a graph

- Consider a ***flow*** of resources over a graph ($\mathcal{C}_1$)
- Blue indicates positive flow ($+1$)



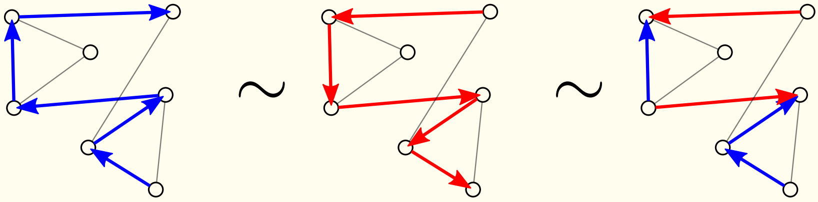
example: flows on a graph

- Consider a ***flow*** of resources over a graph ($\mathcal{C}_1$)
- Blue indicates positive flow ($+1$)
- Red indicates negative flow (-1)



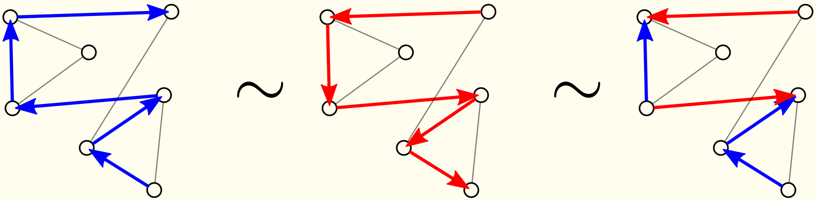
example: flows on a graph

- Consider a ***flow*** of resources over a graph ($\mathcal{C}_1$)
- Blue indicates positive flow ($+1$)
- Red indicates negative flow (-1)



example: flows on a graph

- Consider a ***flow*** of resources over a graph ($\mathcal{C}_1$)
- Blue indicates positive flow ($+1$)
- Red indicates negative flow (-1)



Skew-symmetry: flow does not depend on ***chosen orientation***

goal of this talk

- ***Graph neural networks*** are maps: $\mathcal{C}_0 \to \mathcal{C}_0$

goal of this talk

- ***Graph neural networks*** are maps: $\mathcal{C}_0 \to \mathcal{C}_0$
- We want to understand fancier maps: $\mathcal{C}_k \to \mathcal{C}_\ell$

goal of this talk

- ***Graph neural networks*** are maps: $\mathcal{C}_0 \to \mathcal{C}_0$
- We want to understand fancier maps: $\mathcal{C}_k \to \mathcal{C}_\ell$
- Guiding questions:

goal of this talk

- ***Graph neural networks*** are maps: $\mathcal{C}_0 \to \mathcal{C}_0$
- We want to understand fancier maps: $\mathcal{C}_k \to \mathcal{C}_\ell$
- Guiding questions:
  - Symmetries/invariances?
  - Natural operations available to us?
  - Fully leverage the simplicial structure?

a suitable operator for graphs

- **Q:** What is the most important operator on a graph?

a suitable operator for graphs

- ***Q:*** What is the most important operator on a graph?
- ***A:*** The Laplacian!

a suitable operator for graphs

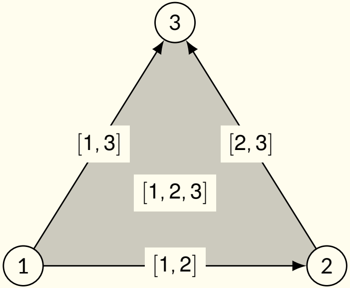
- ***Q:*** What is the most important operator on a graph?
- ***A:*** The Laplacian!
- $\mathbf{L} = \mathbf{D} - \mathbf{A}$

a suitable operator for graphs

- ***Q:*** What is the most important operator on a graph?
- ***A:*** The Laplacian!
- $\mathbf{L} = \mathbf{D} - \mathbf{A}$
- Or even better: $\mathbf{L} = \mathbf{B}_1 \mathbf{B}_1^\top$

incidence for simplicial complexes

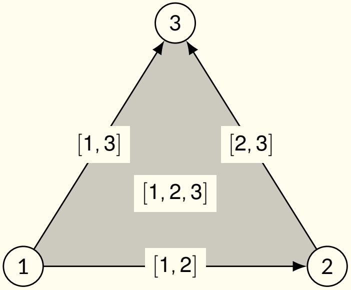
- For a simplicial complex, what are the **B** matrices?



incidence for simplicial complexes

- For a simplicial complex, what are the **B** matrices?

- $\mathbf{B}_1$ is as usual:
  node/edge incidence
  $\mathbf{B}_1[1,2] = [2] - [1]$

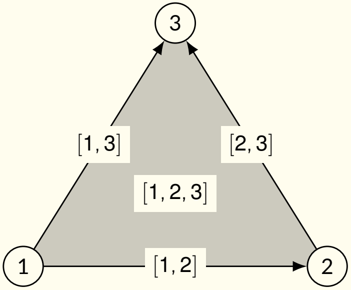


incidence for simplicial complexes

- For a simplicial complex, what are the $\mathbf{B}$ matrices?

- $\mathbf{B}_1$ is as usual:
  node/edge incidence
  $\mathbf{B}_1[1,2] = [2] - [1]$

- $\mathbf{B}_2$ is not too hard:
  edge/triangle incidence
  $\mathbf{B}_2[1,2,3] = [1,2] + [2,3] - [1,3]$



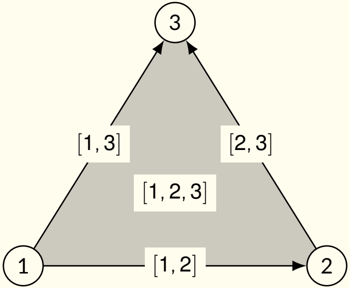
incidence for simplicial complexes

- For a simplicial complex, what are the **B** matrices?

- $\mathbf{B}_1$ is as usual:
  node/edge incidence
  $\mathbf{B}_1[1,2] = [2] - [1]$

- $\mathbf{B}_2$ is not too hard:
  edge/triangle incidence
  $\mathbf{B}_2[1,2,3] = [1,2] + [2,3] - [1,3]$

- $\mathbf{B}_k$ follows:
  k-simplex/$k+1$-simplex incidence



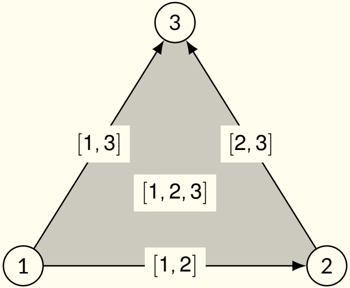
incidence for simplicial complexes

- For a simplicial complex, what are the $\mathbf{B}$ matrices?

- $\mathbf{B}_1$ is as usual:
  node/edge incidence
  $\mathbf{B}_1[1,2] = [2] - [1]$

- $\mathbf{B}_2$ is not too hard:
  edge/triangle incidence
  $\mathbf{B}_2[1,2,3] = [1,2] + [2,3] - [1,3]$

- $\mathbf{B}_k$ follows:
  k-simplex/$k+1$-simplex incidence



$$\mathbf{B}_k \mathbf{B}_{k+1} = 0$$

admissible maps

We ask that a neural network satisfy...

admissible maps

We ask that a neural network satisfy...

1. ***Permutation equivariance***: ordering of simplices ***doesn't*** matter

admissible maps

We ask that a neural network satisfy...

1. ***Permutation equivariance***: ordering of simplices ***doesn't*** matter
2. ***Orientation equivariance***: orientation of simplices ***doesn't*** matter

admissible maps

We ask that a neural network satisfy...

1. ***Permutation equivariance***: ordering of simplices ***doesn't*** matter
2. ***Orientation equivariance***: orientation of simplices ***doesn't*** matter
3. ***Simplicial awareness***: total simplicial structure ***does*** matter

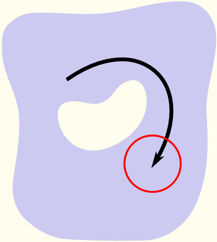
admissible maps

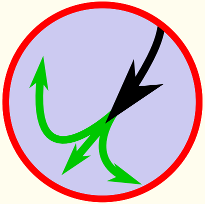
We ask that a neural network satisfy...

1. ***Permutation equivariance***: ordering of simplices ***doesn't*** matter
2. ***Orientation equivariance***: orientation of simplices ***doesn't*** matter
3. ***Simplicial awareness***: total simplicial structure ***does*** matter

A map $\mathcal{C}_k \to \mathcal{C}_\ell$ that satisfies all of these is what we call ***admissible***

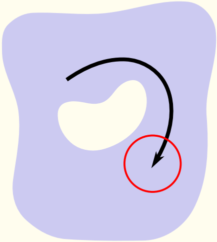
trajectory prediction

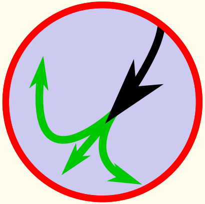




- Observe an agent moving along a ***path***

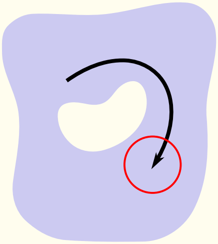
trajectory prediction

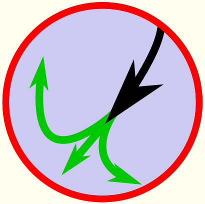




- Observe an agent moving along a ***path***
- Predict next step ***locally***

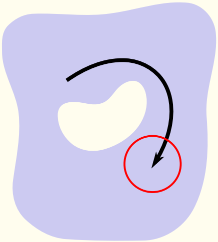
trajectory prediction

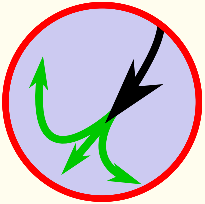




- Observe an agent moving along a ***path***
- Predict next step ***locally***
- ***Discrete space***: a 2-dim. simplicial complex

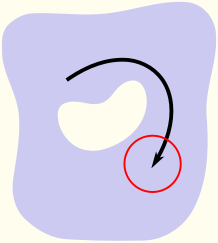
trajectory prediction

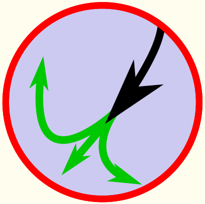




- Observe an agent moving along a ***path***
- Predict next step ***locally***
- ***Discrete space***: a 2-dim. simplicial complex
- ***Trajectory***: a 1-chain

trajectory prediction



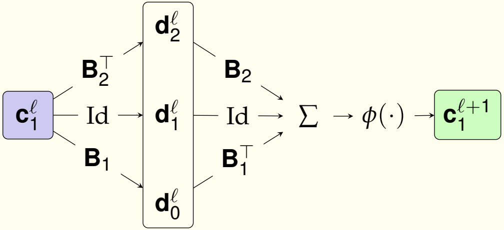


- Observe an agent moving along a ***path***
- Predict next step ***locally***
- ***Discrete space***: a 2-dim. simplicial complex
- ***Trajectory***: a 1-chain
- ***Predict***: a node (0-simplex)

SCoNe: simplicial complex net

- ***SCoNe***: simplicial complex net

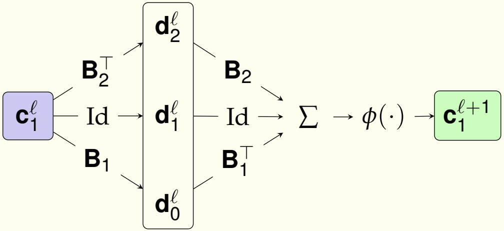
$$
\begin{aligned}
\mathbf{c}_1^{\ell+1} \leftarrow \phi(&\mathbf{B}_2\mathbf{B}_2^\top \mathbf{c}_1^\ell \mathbf{W}_\ell^2 \\
&+ \mathbf{c}_1^\ell \mathbf{W}_\ell^1 \\
&+ \mathbf{B}_1^\top \mathbf{B}_1 \mathbf{c}_1^\ell \mathbf{W}_\ell^0)
\end{aligned}
$$



SCoNe: simplicial complex net

- ***SCoNe***: simplicial complex net

$$\mathbf{c}_1^{\ell+1} \leftarrow \phi\big(\mathbf{B}_2\mathbf{B}_2^\top\mathbf{c}_1^\ell\mathbf{W}_\ell^2 \\ + \mathbf{c}_1^\ell\mathbf{W}_\ell^1 \\ + \mathbf{B}_1^\top\mathbf{B}_1\mathbf{c}_1^\ell\mathbf{W}_\ell^0\big)$$

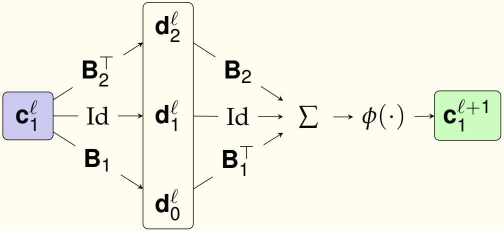


- After L such layers, ***project***:
  $\mathbf{c}_0^{L+1} = \mathbf{B}_1\mathbf{c}_1^L\mathbf{W}_L^0$

SCoNe: simplicial complex net

- ***SCoNe***: simplicial complex net

$$\mathbf{c}_1^{\ell+1} \leftarrow \phi(\mathbf{B}_2\mathbf{B}_2^\top \mathbf{c}_1^\ell \mathbf{W}_\ell^2 + \mathbf{c}_1^\ell \mathbf{W}_\ell^1 + \mathbf{B}_1^\top \mathbf{B}_1 \mathbf{c}_1^\ell \mathbf{W}_\ell^0)$$



- After L such layers, ***project***: $\mathbf{c}_0^{L+1} = \mathbf{B}_1 \mathbf{c}_1^L \mathbf{W}_L^0$

- Softmax over candidate nodes

permutation equivariance

$$\mathcal{P}\,\mathrm{SCN}_{\mathbf{W}}(\mathbf{x};\{\mathbf{B}_j\}) = \mathrm{SCN}_{\mathbf{W}}(\mathcal{P}\mathbf{x};\{\mathcal{P}\mathbf{B}_j\})$$

- Similar lines to graph neural networks
- Ordering the nodes, edges, triangles, etc. does not affect the output
- Composition of permutation equivariant operations:
    - Boundary maps
    - ***Elementwise*** activation function

orientation equivariance

$$\mathcal{D}\,\mathrm{SCN}_{\mathbf{W}}(\mathbf{x}; \{\mathbf{B}_j\}) = \mathrm{SCN}_{\mathbf{W}}(\mathcal{D}\mathbf{x}; \{\mathcal{D}\mathbf{B}_j\})$$

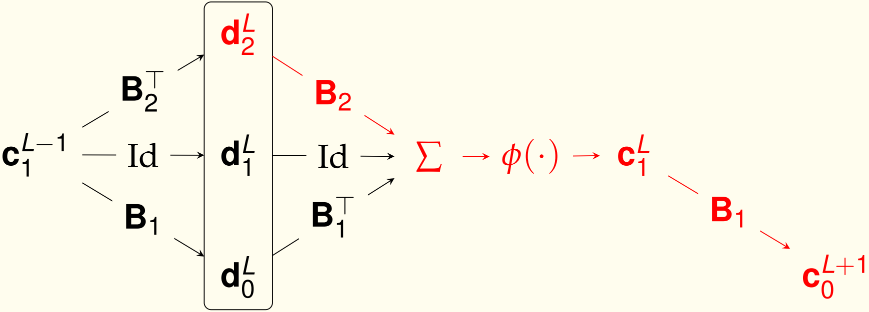
- One step higher: orientation of all simplices
- Reorienting the edges, triangles, etc. does not affect the output
- Not a concern for GNNs: nodes only have one orientation
- Composition of orientation equivariant operations:
    - Boundary maps
    - ***Odd*** activation function

simplicial awareness

$$\mathrm{SCN}_{\mathbf{W}}(\mathbf{x}; \{\mathbf{B}_1, \ldots, \mathbf{B}_K\}) \neq \mathrm{SCN}_{\mathbf{W}}(\mathbf{x}; \{\mathbf{B}_1, \ldots, \mathbf{B}'_\ell, \ldots, \mathbf{B}_K\})$$

- There ***exists*** an input $\mathbf{x}$, weights $\mathbf{W}$, and alternative boundary map $\mathbf{B}'_\ell$ such that the two maps are not the same
- Sensitivity to structure of ℓ-simplices
- For *SCoNe* : $\mathcal{C}_1 \to \mathcal{C}_0$, we must kill homology!
  - ***Recall***: $\mathbf{B_1 B_2} = 0$
  - Get around this with ***nonlinear*** activation function
  - $\mathbf{B_1} \circ \phi \circ \mathbf{B_2} \neq 0$

simplicial awareness



$$\mathbf{B_1 B_2} = 0$$
$$\mathbf{B_1} \circ \phi \circ \mathbf{B_2} \neq 0.$$

that is to say...

Assume the activation function ϕ is ***continuous*** and ***elementwise***.

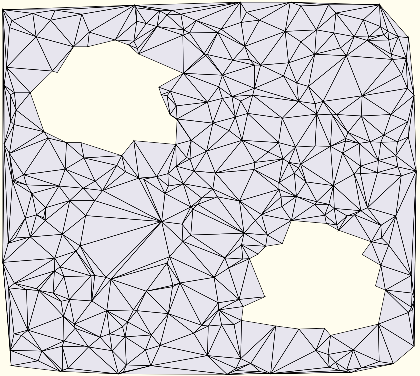
that is to say...

Assume the activation function ϕ is ***continuous*** and ***elementwise***.

SCoNe is admissible only if ϕ is ***odd*** and ***nonlinear***.

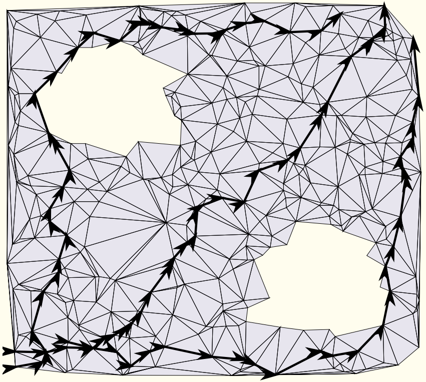
a synthetic example

- ***Synthetic dataset***: random edge orientation



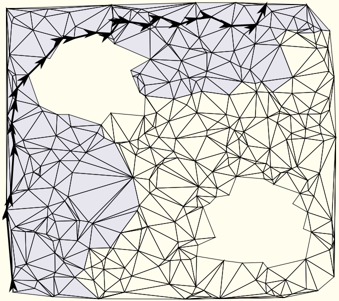
a synthetic example

- ***Synthetic dataset***: random edge orientation
- Trajectories: walks between corners

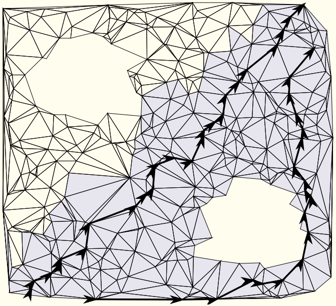


results - disjoint regions

	SCoNe tanh	SCoNe tanh, no tri.	SCoNe ReLU	SCoNe sigm.	SCoNe Id	Ebli et. al. (2020)	Bunch et. al. (2020)
Test Acc.	***0.61***	0.58	0.56	0.53	0.44	0.42	0.57



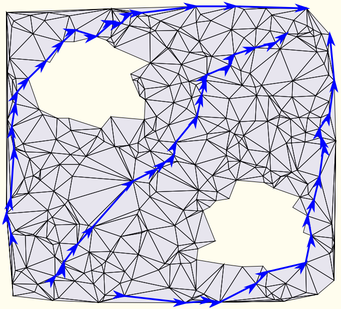
Training set



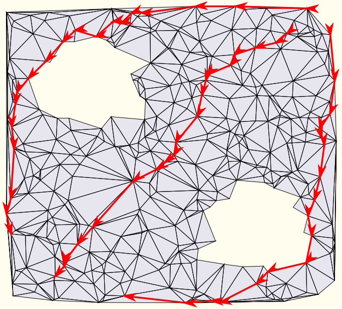
Testing set

results - orientation (in)sensitivity

	SCoNe tanh	SCoNe ReLU	SCoNe sigm.	SCoNe Id
Train Acc.	0.65	0.65	***0.66***	0.27



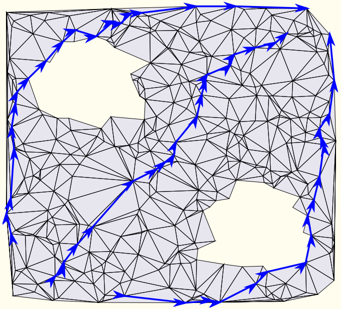
Training set



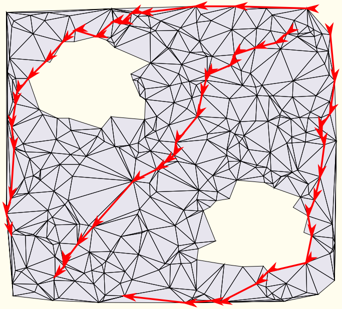
Testing set

results - orientation (in)sensitivity

	SCoNe tanh	SCoNe ReLU	SCoNe sigm.	SCoNe Id
Train Acc.	0.65	0.65	***0.66***	0.27
Test Acc.	***0.63***	0.24	0.10	0.31



Training set



Testing set

conclusion

- Graph neural networks process nodal data using intrinsic graph operators

conclusion

- Graph neural networks process nodal data using intrinsic graph operators
- Use operators on simplicial complexes to extend this to k-chain maps

conclusion

- Graph neural networks process nodal data using intrinsic graph operators
- Use operators on simplicial complexes to extend this to k-chain maps
- Demand that the architecture obeys symmetries of the system

conclusion

- Graph neural networks process nodal data using intrinsic graph operators
- Use operators on simplicial complexes to extend this to k-chain maps
- Demand that the architecture obeys symmetries of the system
- Yields better generalization