RWTH AACHEN UNIVERSITY

word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data

Martin Grohe
RWTH Aachen

Vector Embeddings

Represent objects from arbitrary class $\mathcal{X}$ by vectors in some vector space $\mathbb{H}$



space $\mathcal{X}$ of objects

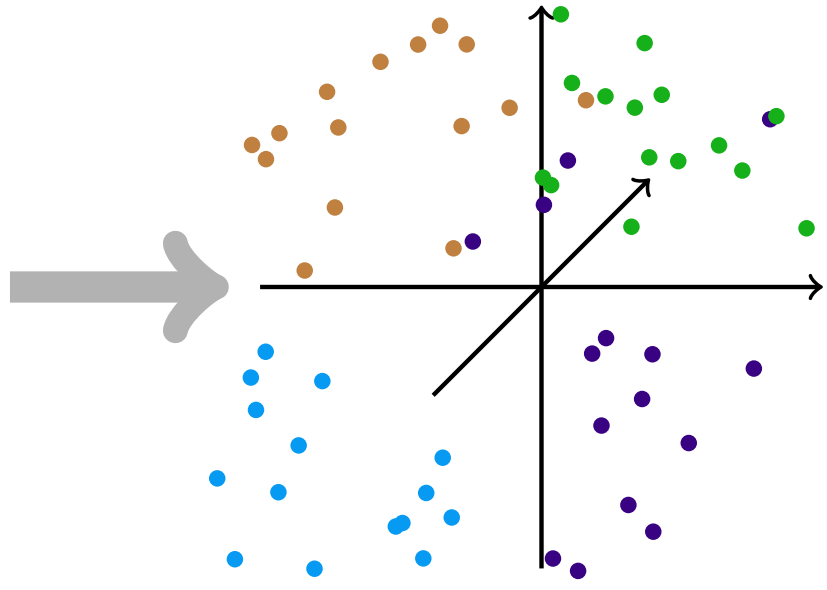
vector space $\mathbb{H}$

Vector Embeddings

Represent objects from arbitrary class $\mathcal{X}$ by vectors in some vector space $\mathbb{H}$



space $\mathcal{X}$ of objects



vector space $\mathbb{H}$

Example

Word embeddings in natural language processing: word2vec



Motivation and Ideas

How?

- embed objects into vector space (Hilbert space) in such a way that semantic relationships between objects are reflected in geometric relationships of their images



Motivation and Ideas

How?

- embed objects into vector space (Hilbert space) in such a way that semantic relationships between objects are reflected in geometric relationships of their images
- most importantly: “semantic distance” between objects correlated to geometric distance



Motivation and Ideas

How?

- embed objects into vector space (Hilbert space) in such a way that semantic relationships between objects are reflected in geometric relationships of their images
- most importantly: "semantic distance" between objects correlated to geometric distance

Why?

- standard machine learning algorithms operate on vector representations of data



Motivation and Ideas

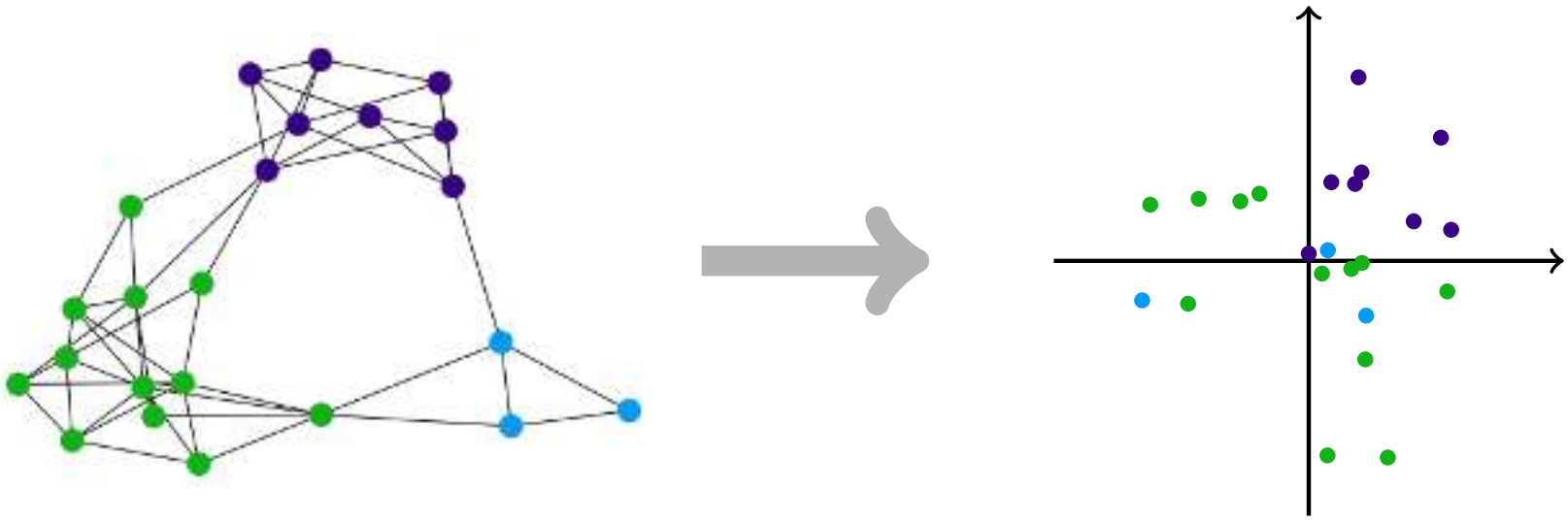
How?

- embed objects into vector space (Hilbert space) in such a way that semantic relationships between objects are reflected in geometric relationships of their images
- most importantly: "semantic distance" between objects correlated to geometric distance

Why?

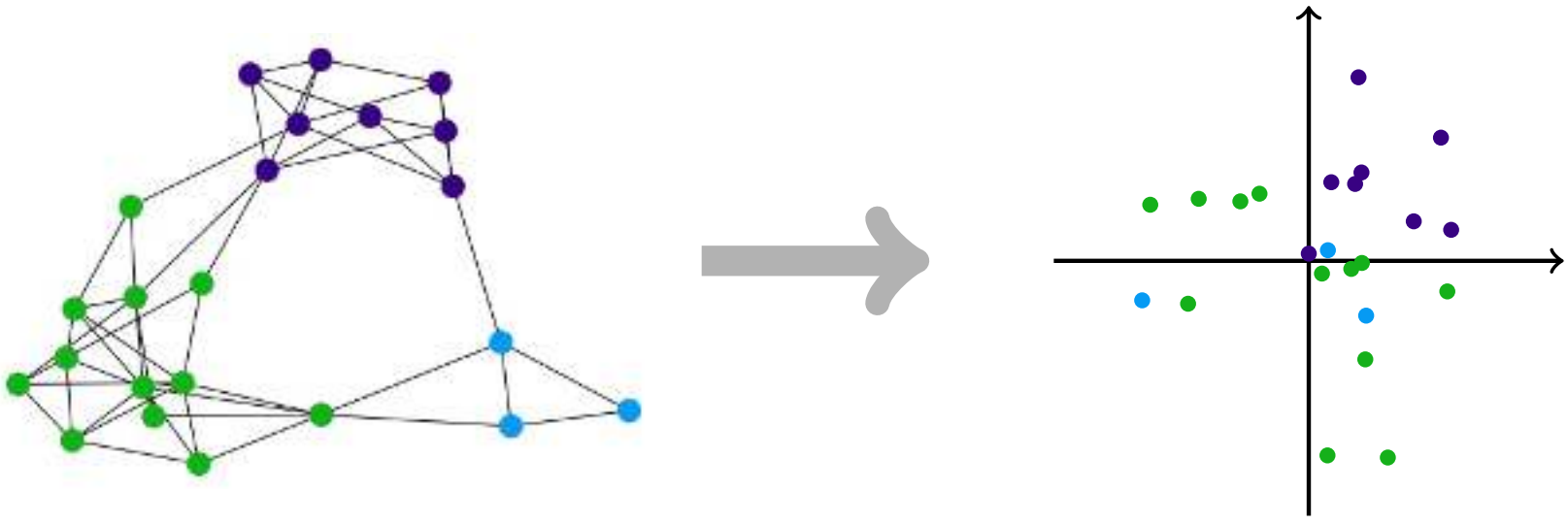
- standard machine learning algorithms operate on vector representations of data
- enable a quantitative analysis using mathematical techniques ranging from linear algebra to functional analysis

Node Embeddings



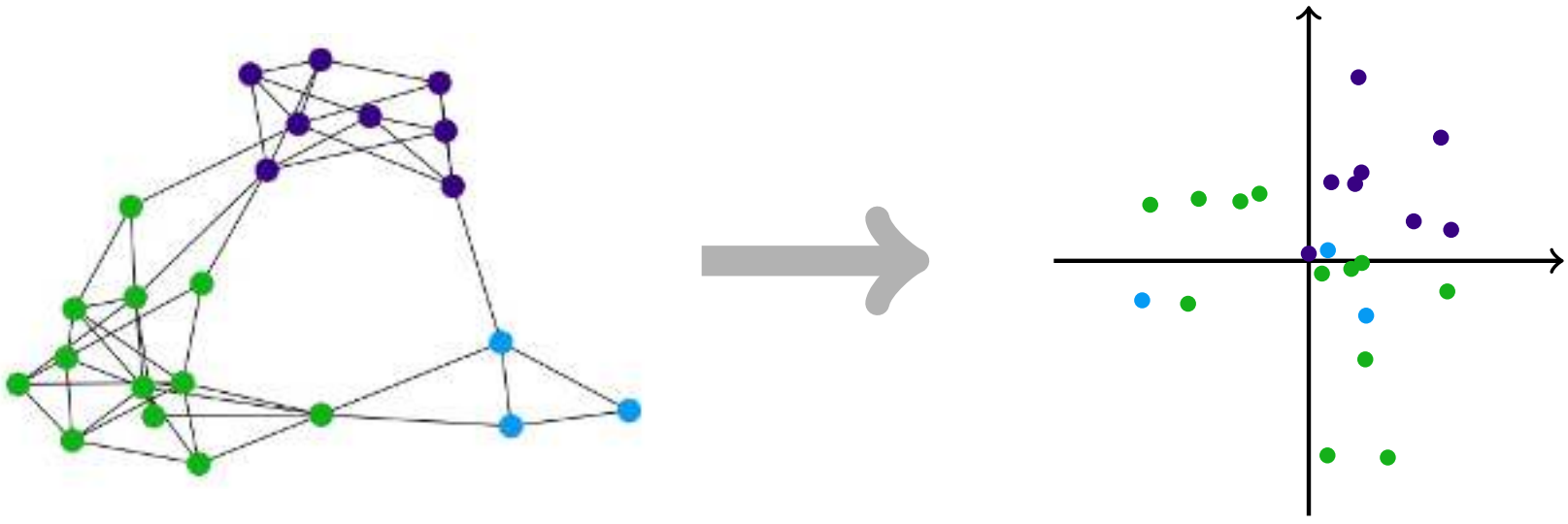
- objects that are embedded are the vertices of a graph

Node Embeddings



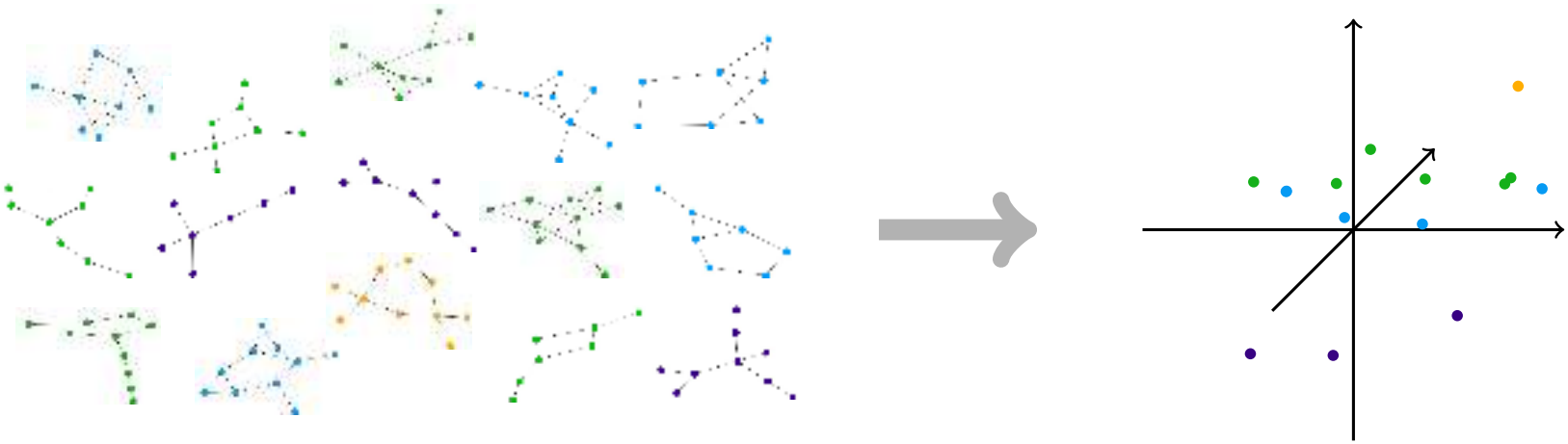
- objects that are embedded are the vertices of a graph
- graph can be directed, labelled, weighted, et cetera

Node Embeddings



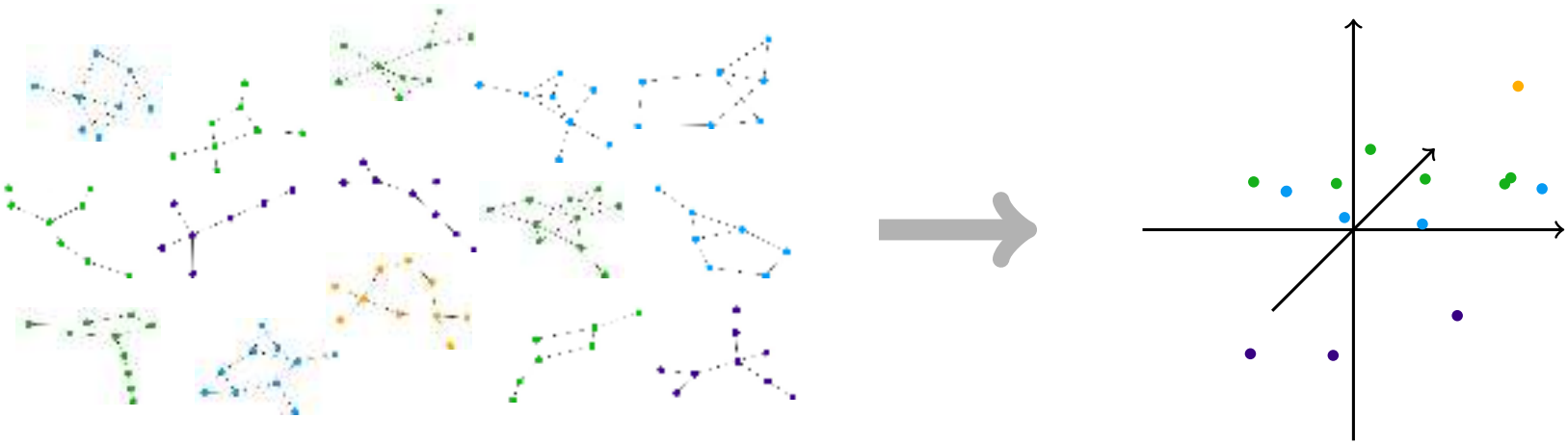
- objects that are embedded are the vertices of a graph
- graph can be directed, labelled, weighted, et cetera
- typical applications: node classification in social networks, link prediction in knowledge graphs

Graph Embeddings



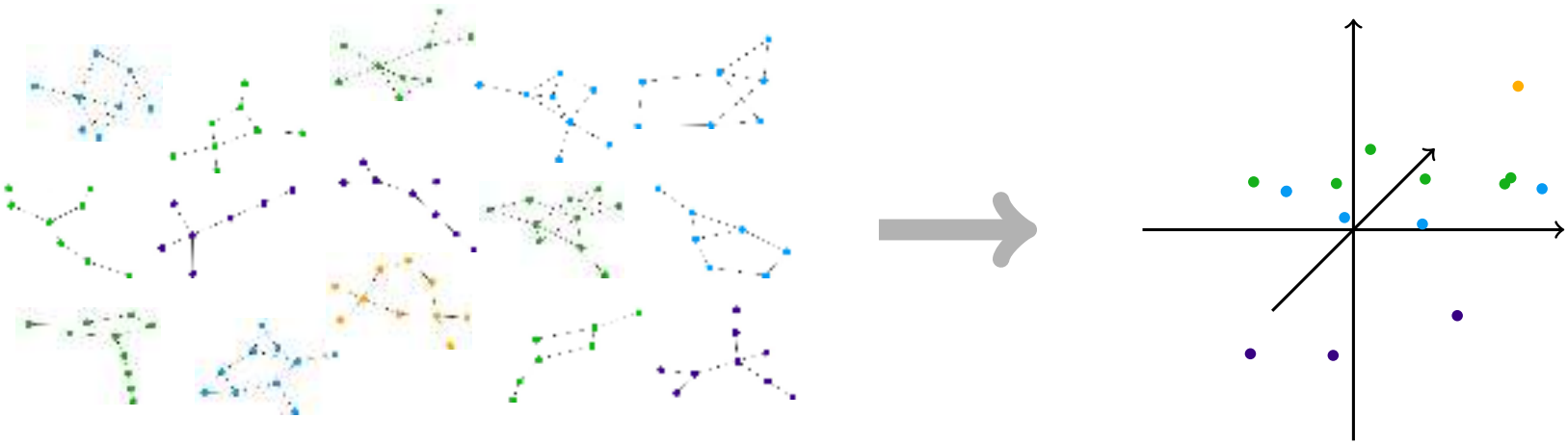
- objects that are embedded are graphs

Graph Embeddings



- objects that are embedded are graphs
- graphs can be directed, labelled, weighted, et cetera

Graph Embeddings



- objects that are embedded are graphs
- graphs can be directed, labelled, weighted, et cetera
- typical applications: classification and regression on chemical molecules

Node Embeddings vs Graph Embeddings

Commonality

- same structure

Node Embeddings vs Graph Embeddings

Commonality

- same structure
- graph embedding applied to neighbourhoods of nodes yields node embedding

Node Embeddings vs Graph Embeddings

Commonality

- same structure
- graph embedding applied to neighbourhoods of nodes yields node embedding

Important differences

- nodes a of a single graph have explicit relation(s) between them and a clearly defined distance, but different graphs are only “semantically” related

Node Embeddings vs Graph Embeddings

Commonality

- same structure
- graph embedding applied to neighbourhoods of nodes yields node embedding

Important differences

- nodes a of a single graph have explicit relation(s) between them and a clearly defined distance, but different graphs are only "semantically" related
- node embeddings can (often) be trained for fixed graphs, but we rarely have to deal with a finite set of graphs fixed in advance

Towards a Theory of Vector Embeddings

- Vector embeddings allow for a quantitative analysis of discrete data using standard machine learning algorithms and mathematical techniques from linear algebra to functional analysis.

Towards a Theory of Vector Embeddings

- Vector embeddings allow for a quantitative analysis of discrete data using standard machine learning algorithms and mathematical techniques from linear algebra to functional analysis.
- Vector embeddings allow it to integrate data from different sources to a common framework

Towards a Theory of Vector Embeddings

- Vector embeddings allow for a quantitative analysis of discrete data using standard machine learning algorithms and mathematical techniques from linear algebra to functional analysis.
- Vector embeddings allow it to integrate data from different sources to a common framework
- Node and graph embeddings have been studied in machine learning for a long time.
  The existing work focusses on graphs and binary structures, not much is known about embedding arbitrary relations.

Towards a Theory of Vector Embeddings

- Vector embeddings allow for a quantitative analysis of discrete data using standard machine learning algorithms and mathematical techniques from linear algebra to functional analysis.
- Vector embeddings allow it to integrate data from different sources to a common framework
- Node and graph embeddings have been studied in machine learning for a long time.
  The existing work focusses on graphs and binary structures, not much is known about embedding arbitrary relations.
- Except for the mathematical foundations, there is little theory.
  Vector embeddings deserve a deeper study using the tools and methods of TCS (analysis of algorithms and complexity, logic, structural graph theory, et cetera)

Key Questions

Algorithms and Complexity

Can we compute the embedding efficiently? Can we invert it efficiently (i.e., reconstruct graphs from their image)?

Key Questions

Algorithms and Complexity

Can we compute the embedding efficiently? Can we invert it efficiently (i.e., reconstruct graphs from their image)?

Logic and Semantics

What is the semantical meaning of the embeddings? Which properties are preserved? Can we answer queries on the embedded data?

Key Questions

Algorithms and Complexity

Can we compute the embedding efficiently? Can we invert it efficiently (i.e., reconstruct graphs from their image)?

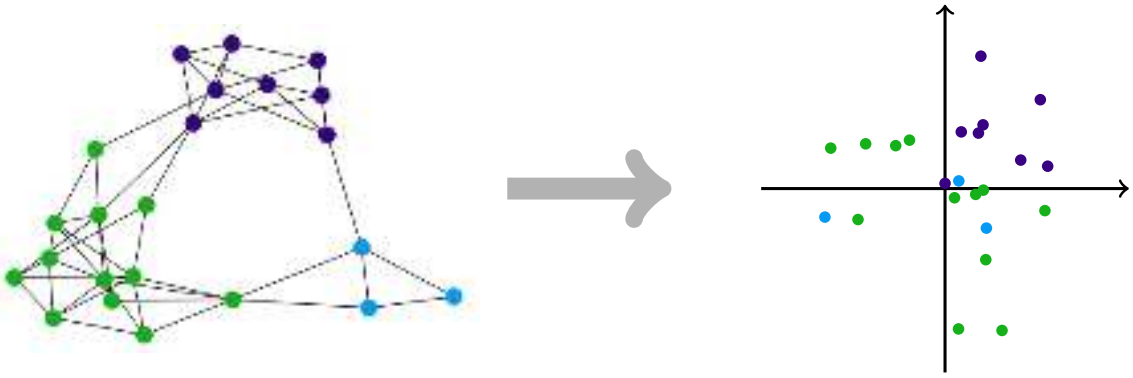
Logic and Semantics

What is the semantical meaning of the embeddings? Which properties are preserved? Can we answer queries on the embedded data?

Comparison

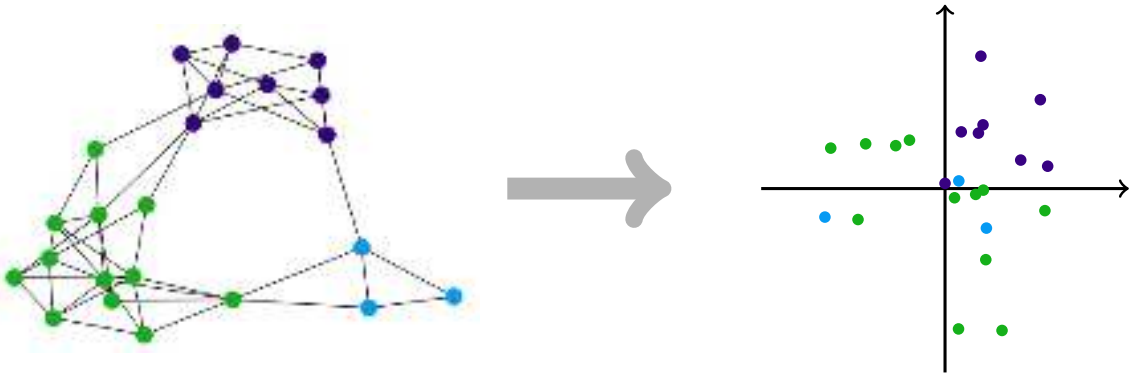
How do different embeddings compare? Do they induce the same metric, the same topology?

Embeddings Techniques



Matrix Factorisation

Classical approach to defining node embeddings based on graph metric.



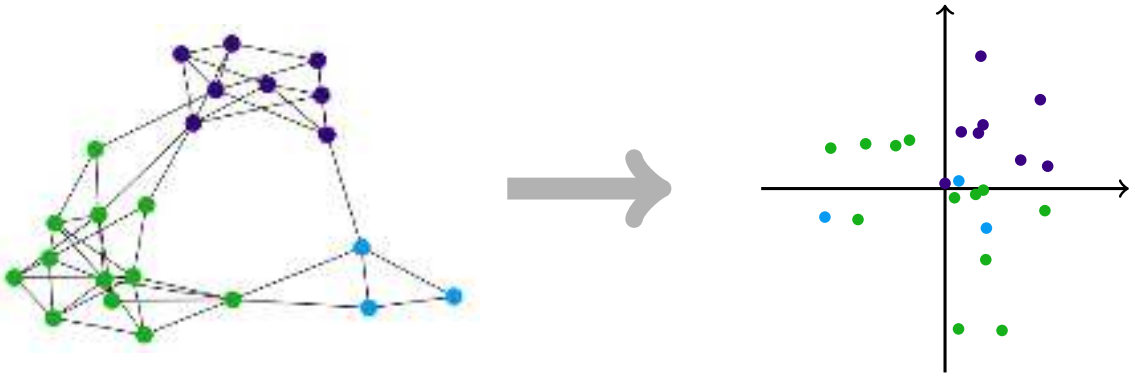
Matrix Factorisation

Classical approach to defining node embeddings based on graph metric.

Graph $G = (V, E)$

- Define similarity matrix $S \in \mathbb{R}^{V \times V}$, e.g. by

$$S_{vw} := \exp\big(-\mathrm{dist}_G(v, w)\big)$$



Matrix Factorisation

Classical approach to defining node embeddings based on graph metric.

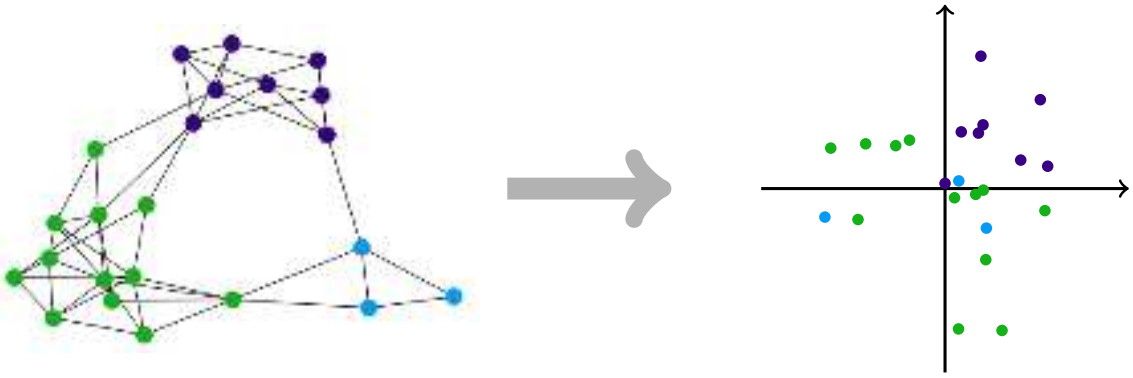
Graph $G = (V, E)$

- Define similarity matrix $S \in \mathbb{R}^{V \times V}$, e.g. by

$$S_{vw} := \exp\big(-\operatorname{dist}_G(v, w)\big)$$

- Compute low dimensional factorisation, e.g. matrix $A \in \mathbb{R}^{V \times k}$ minimising

$$\|AA^\top - S\|$$



Matrix Factorisation

Classical approach to defining node embeddings based on graph metric.

Graph $G = (V, E)$

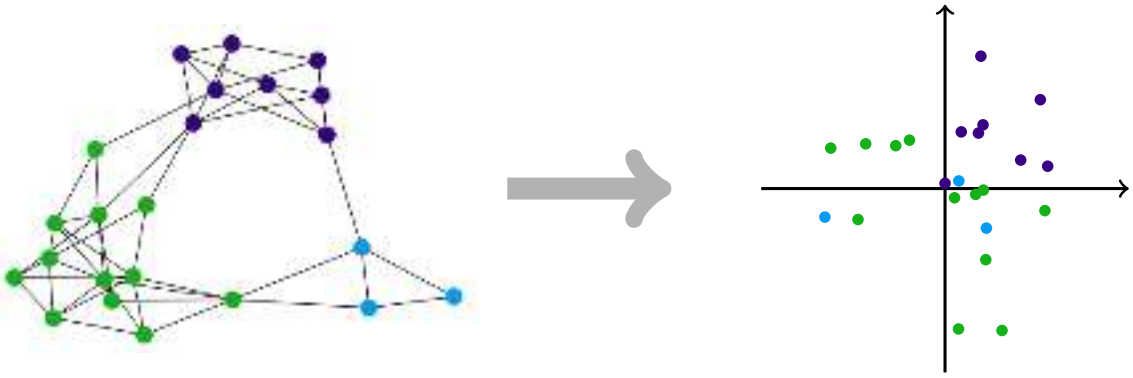
- Define similarity matrix $S \in \mathbb{R}^{V \times V}$, e.g. by

$$S_{vw} := \exp\big(-\operatorname{dist}_G(v, w)\big)$$

- Compute low dimensional factorisation, e.g. matrix $A \in \mathbb{R}^{V \times k}$ minimising

$$\|AA^\top - S\|$$

- Rows $\boldsymbol{a}_v$ of matrix A define embedding $V \to \mathbb{R}^k$.



Matrix Factorisation

Classical approach to defining node embeddings based on graph metric.

Graph $G = (V, E)$

- Define similarity matrix $S \in \mathbb{R}^{V \times V}$, e.g. by

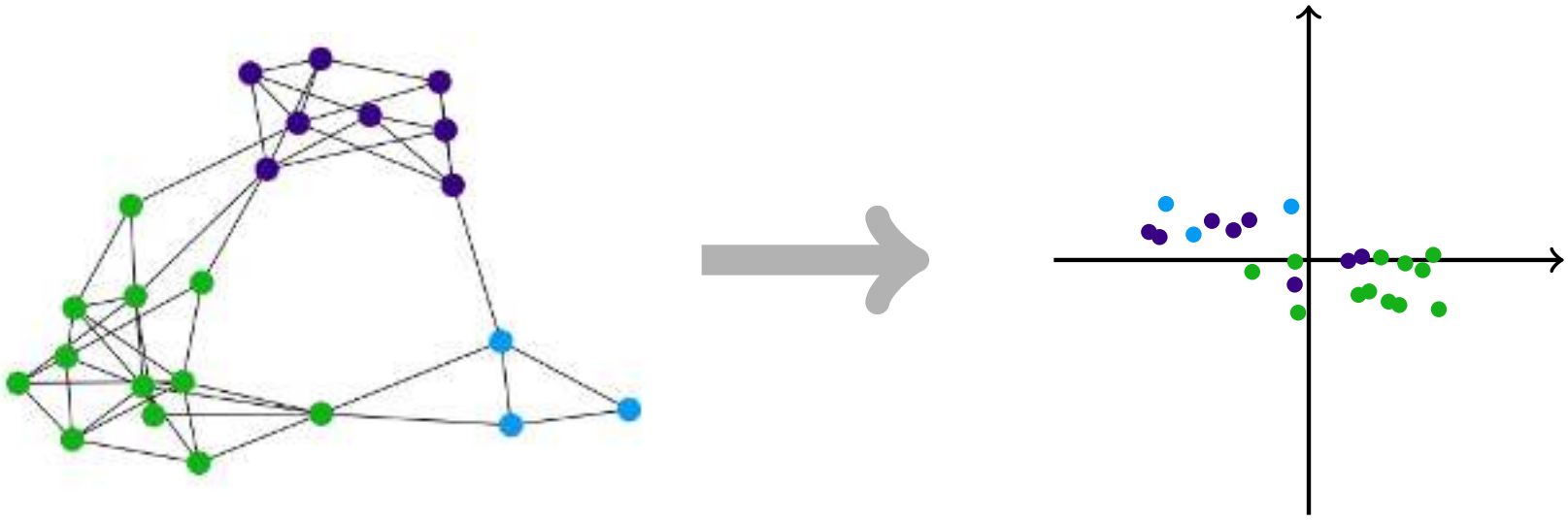
$$S_{vw} := \exp\big(-\operatorname{dist}_G(v, w)\big)$$

- Compute low dimensional factorisation, e.g. matrix $A \in \mathbb{R}^{V \times k}$ minimising

$$\|AA^\top - S\|$$

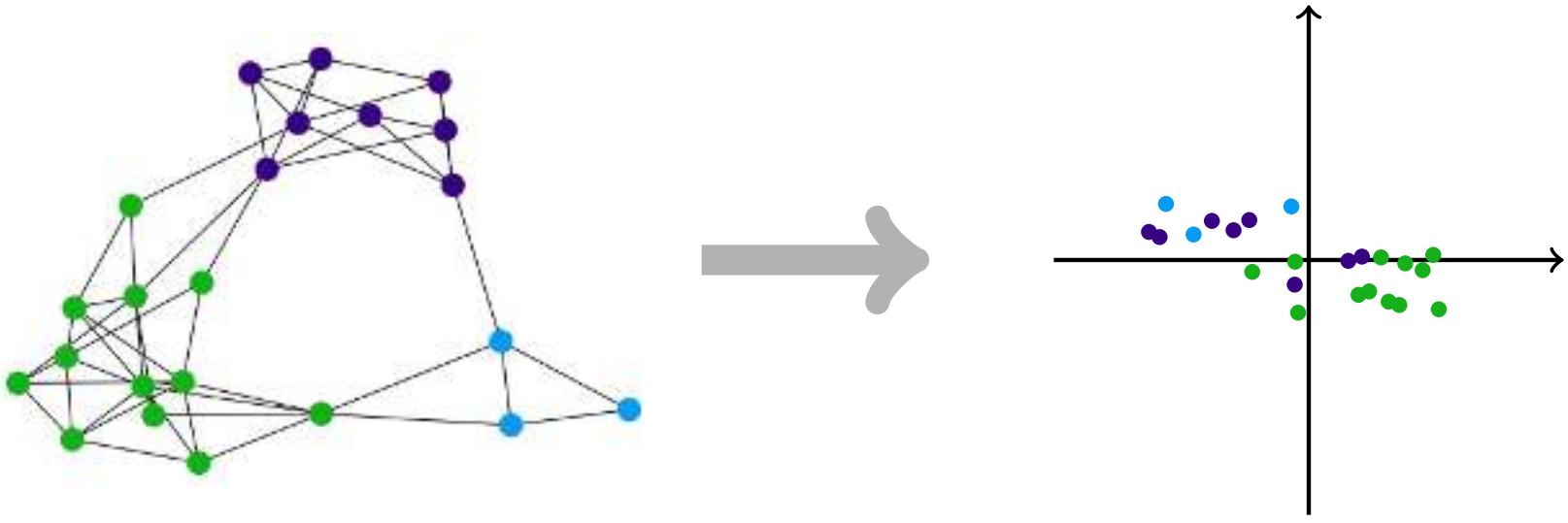
- Rows $\boldsymbol{a}_v$ of matrix A define embedding $V \to \mathbb{R}^k$.
  Inner product $\langle \boldsymbol{a}_v, \boldsymbol{a}_w \rangle$ approximates similarity S_{vw}.

Random Walks



DeepWalk and node2vec are node embedding algorithms based on the following idea:

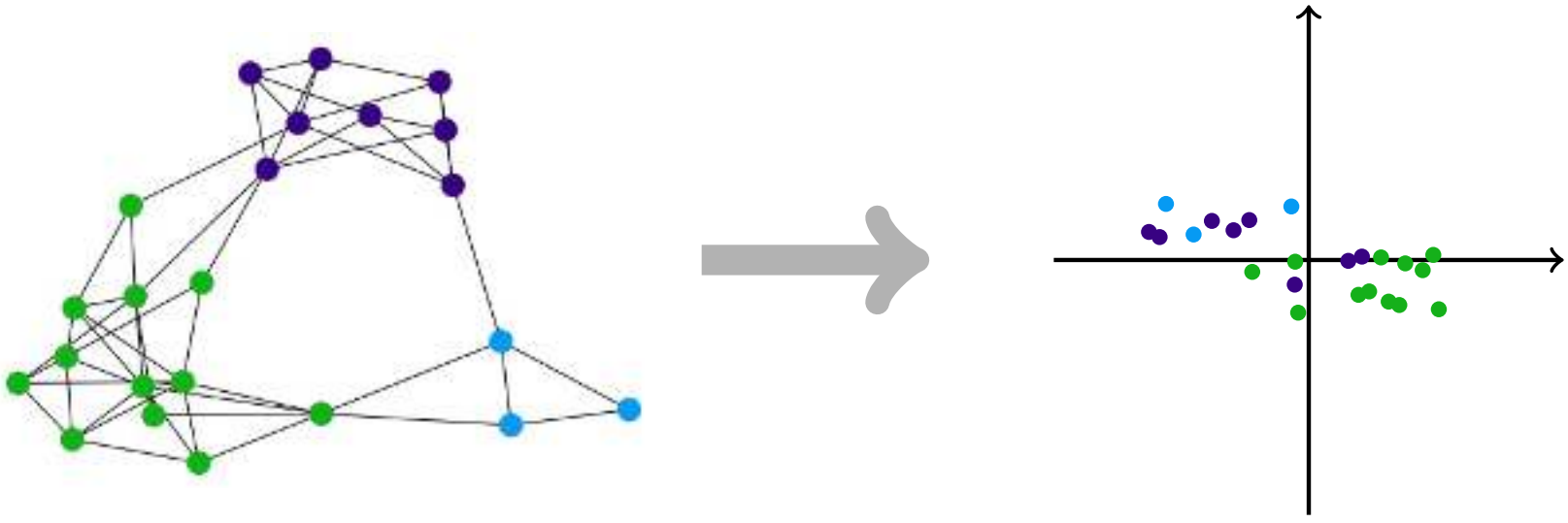
Random Walks



DeepWalk and node2vec are node embedding algorithms based on the following idea:

- Compile a list of short random walks in the graph.

Random Walks



DeepWalk and node2vec are node embedding algorithms based on the following idea:

- Compile a list of short random walks in the graph.
- Treat them like sentences in in natural language, with the nodes of the graph as words, and use word embedding techniques (the skip-gram technique of word2vec) to compute embedding.

(Perozzi, Al-Rfou, Skiena 2014, Grover Leskovec 2016)

Graph Neural Networks

Graph Neural Networks (GNNs) are a deep learning framework for graphs:

- learned message passing network on the input graph

Graph Neural Networks

Graph Neural Networks (GNNs) are a deep learning framework for graphs:

- learned message passing network on the input graph
- each node v has a vector-valued state $\boldsymbol{x}_v^{(t)} \in \mathbb{R}^k$ at each point t in time

Graph Neural Networks

Graph Neural Networks (GNNs) are a deep learning framework for graphs:

- learned message passing network on the input graph
- each node v has a vector-valued state $\boldsymbol{x}_v^{(t)} \in \mathbb{R}^k$ at each point t in time
- states are updated based on the states of the neighbours, e.g.

$$\textit{Aggregate}: \qquad \boldsymbol{a}_v^{(t+1)} \leftarrow \sum_{w \in N(v)} W_{\text{agg}} \cdot \boldsymbol{x}_w^{(t)},$$

$$\textit{Update}: \qquad \boldsymbol{x}_v^{(t+1)} \leftarrow \sigma\left(W_{\text{up}} \cdot \begin{pmatrix} \boldsymbol{x}_v^{(t)} \\ \boldsymbol{a}_v^{(t+1)} \end{pmatrix}\right)$$

with learned parameter matrices $W_{\text{agg}}, W_{\text{up}}$ and a nonlinear "activation function" σ.

Graph Neural Networks

Graph Neural Networks (GNNs) are a deep learning framework for graphs:

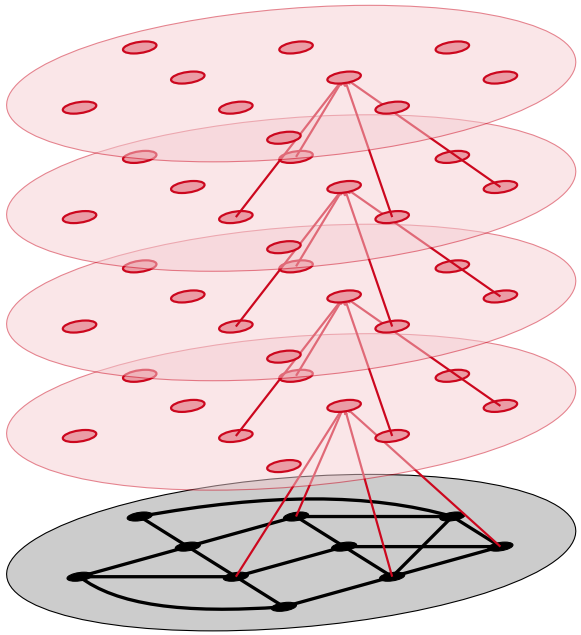
- learned message passing network on the input graph
- each node v has a vector-valued state $\boldsymbol{x}_v^{(t)} \in \mathbb{R}^k$ at each point t in time
- states are updated based on the states of the neighbours, e.g.

$$\textit{Aggregate}: \qquad \boldsymbol{a}_v^{(t+1)} \leftarrow \sum_{w \in N(v)} W_{\text{agg}} \cdot \boldsymbol{x}_w^{(t)},$$

$$\textit{Update}: \qquad \boldsymbol{x}_v^{(t+1)} \leftarrow \sigma\left(W_{\text{up}} \cdot \begin{pmatrix} \boldsymbol{x}_v^{(t)} \\ \boldsymbol{a}_v^{(t+1)} \end{pmatrix}\right)$$

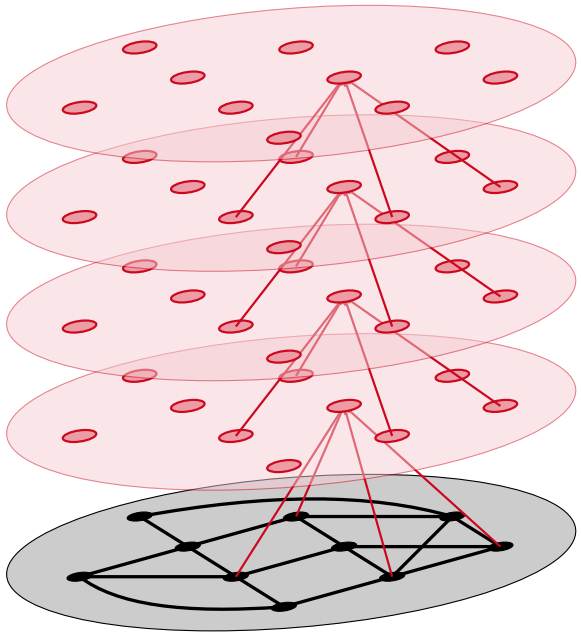
  with learned parameter matrices $W_{\text{agg}}, W_{\text{up}}$ and a nonlinear "activation function" σ.
- iterated for a fixed number s of rounds, $v \mapsto \boldsymbol{x}_v^{(s)}$ is the resulting node embedding

Graph Neural Networks (cont'd)



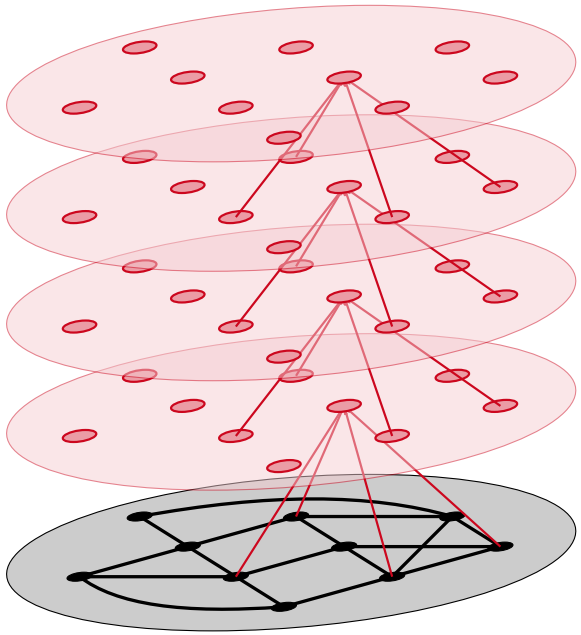
- a GNN encodes the embedding function and not just a list of vectors for the nodes of a fixed graph (as the previous techniques do)

Graph Neural Networks (cont'd)



- a GNN encodes the embedding function and not just a list of vectors for the nodes of a fixed graph (as the previous techniques do)
- once trained, a GNN can still be applied if the graph changes, or even to an entirely different graph

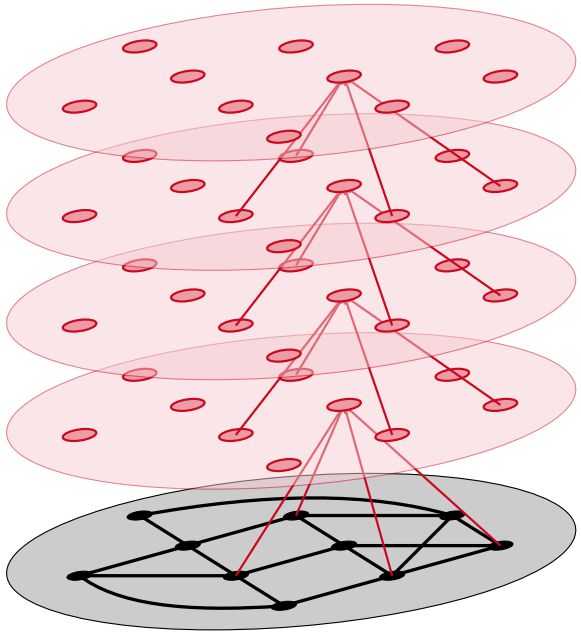
Graph Neural Networks (cont'd)



- a GNN encodes the embedding function and not just a list of vectors for the nodes of a fixed graph (as the previous techniques do)
- once trained, a GNN can still be applied if the graph changes, or even to an entirely different graph

- we say that GNNs provide an inductive node-embedding method, whereas matrix factorisation techniques and random walk techniques are transductive

Graph Neural Networks (cont'd)

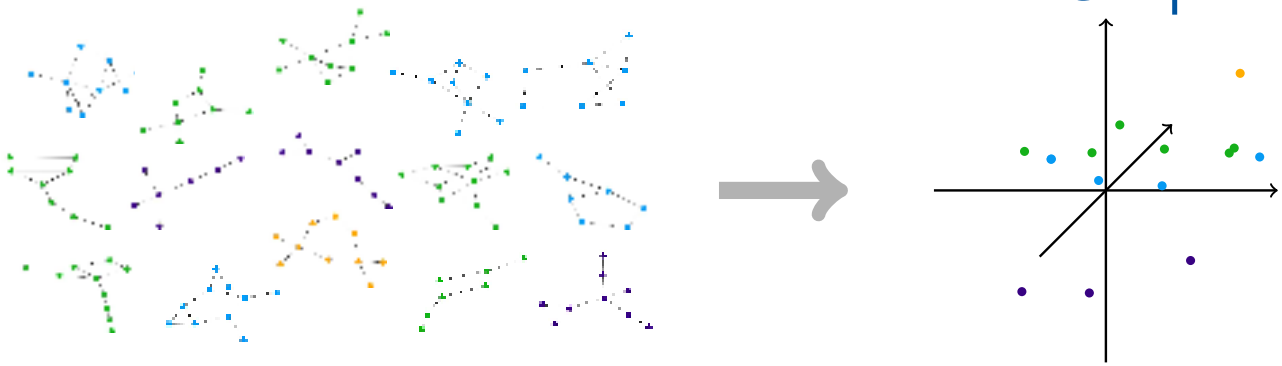


- a GNN encodes the embedding function and not just a list of vectors for the nodes of a fixed graph (as the previous techniques do)
- once trained, a GNN can still be applied if the graph changes, or even to an entirely different graph

- we say that GNNs provide an inductive node-embedding method, whereas matrix factorisation techniques and random walk techniques are transductive
- by aggregating the vectors of all nodes, we can also use GNNs to compute graph embeddings.

(Hamilton, Ying, Leskovec 2017)

Graph Kernels

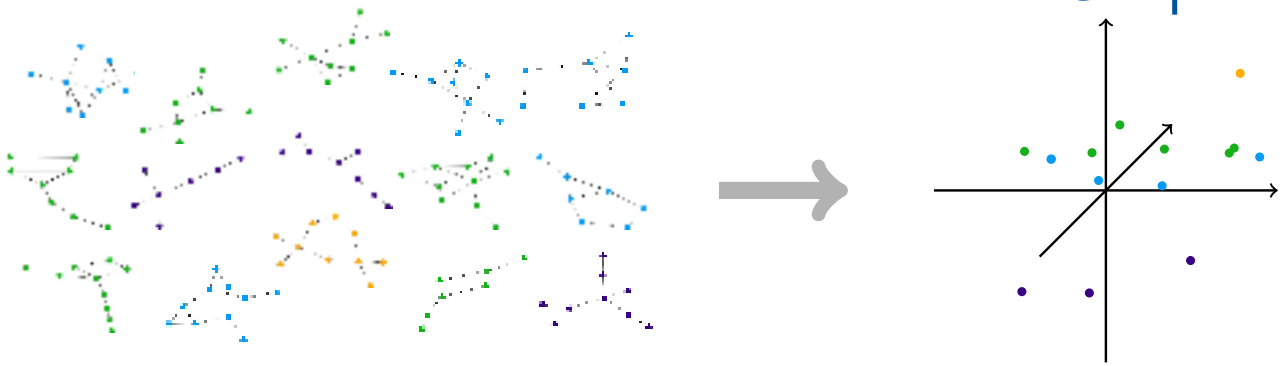


Graph kernels are the inner product functions of graph embeddings, that is, functions

$$K(G, H) = \langle \Phi(G), \Phi(H) \rangle$$

for some graph embedding Φ.

Graph Kernels



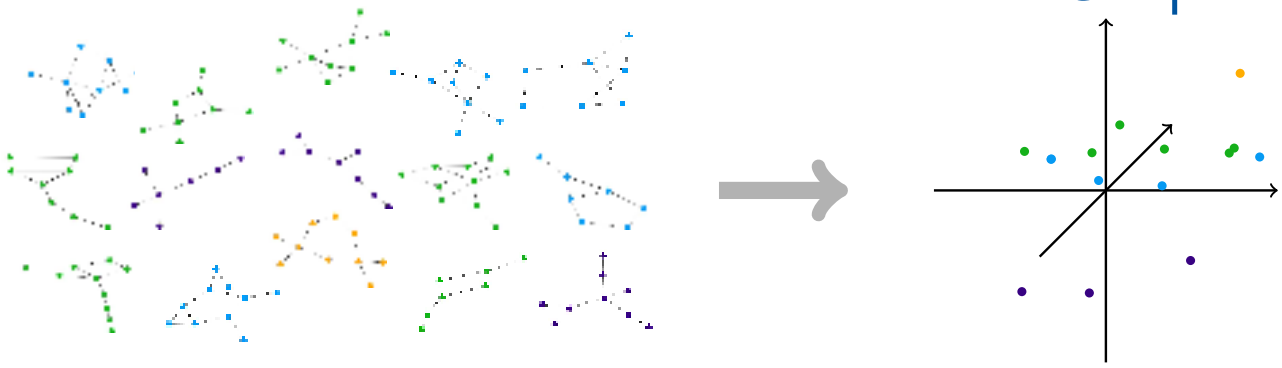
Graph kernels are the inner product functions of graph embeddings, that is, functions

$$K(G, H) = \langle \Phi(G), \Phi(H) \rangle$$

for some graph embedding Φ.

- Kernel trick: Compute and use $K(G, H)$ without ever computing $\Phi(G), \Phi(H)$.

Graph Kernels



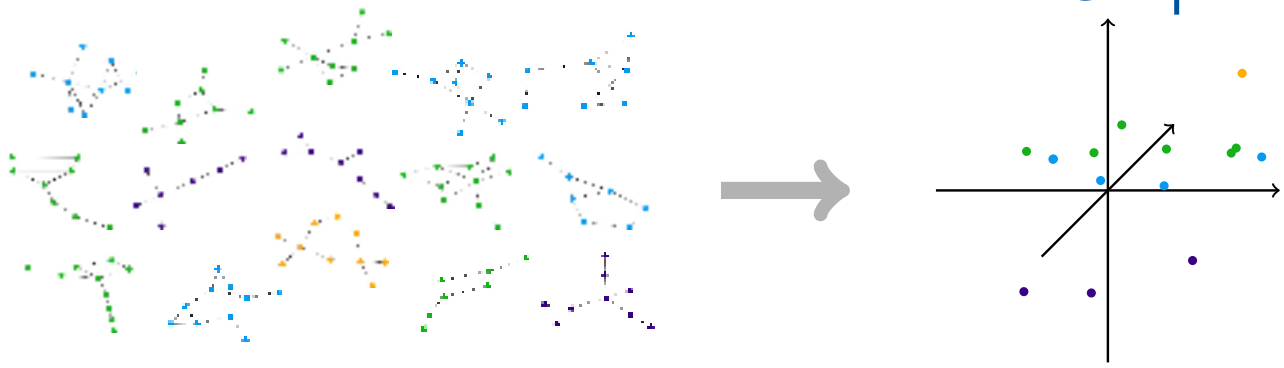
Graph kernels are the inner product functions of graph embeddings, that is, functions

$$K(G, H) = \langle \Phi(G), \Phi(H) \rangle$$

for some graph embedding Φ.

- Kernel trick: Compute and use $K(G, H)$ without ever computing $\Phi(G), \Phi(H)$.
- graph kernels are widely used for ML problems on graphs

Graph Kernels



Graph kernels are the inner product functions of graph embeddings, that is, functions

$$K(G, H) = \langle \Phi(G), \Phi(H) \rangle$$

for some graph embedding Φ.

- Kernel trick: Compute and use $K(G, H)$ without ever computing $\Phi(G), \Phi(H)$.
- graph kernels are widely used for ML problems on graphs
- Common graph kernels are based on comparing random walks, counting small subgraphs, and the Weisfeiler-Leman algorithm

(Kashima et al. 2003, Gärtner et al 2003, Shervashidze et al. 2009,...)

The Weisfeiler-Leman Algorithm

A Simple Combinatorial Algorithm

The 1-dimensional Weisfeiler-Leman algorithm (1-WL) (a.k.a. colour refinement) iteratively computes a colouring of the vertices of graph G.

A Simple Combinatorial Algorithm

The 1-dimensional Weisfeiler-Leman algorithm (1-WL) (a.k.a. colour refinement) iteratively computes a colouring of the vertices of graph G.

Initialisation All vertices get the same colour.

A Simple Combinatorial Algorithm

The 1-dimensional Weisfeiler-Leman algorithm (1-WL) (a.k.a. colour refinement) iteratively computes a colouring of the vertices of graph G.

Initialisation All vertices get the same colour.

Refinement Step Two nodes v, w get different colours if there is some colour c such that v and w have different numbers of neighbours of colour c.

A Simple Combinatorial Algorithm

The 1-dimensional Weisfeiler-Leman algorithm (1-WL) (a.k.a. colour refinement) iteratively computes a colouring of the vertices of graph G.

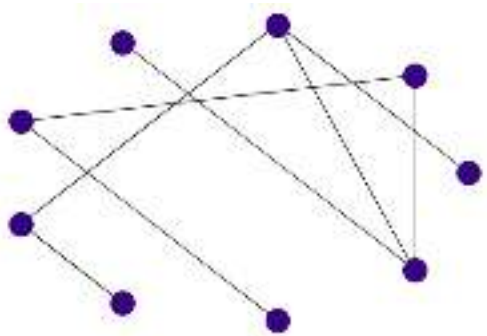
Initialisation All vertices get the same colour.

Refinement Step Two nodes v, w get different colours if there is some colour c such that v and w have different numbers of neighbours of colour c.

Refinement is repeated until colouring stays stable.

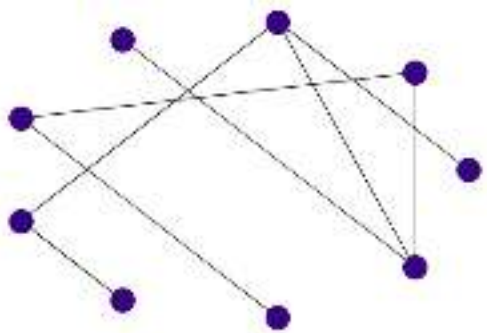
Run 1-WL (Demo by Erkal Selman

Example

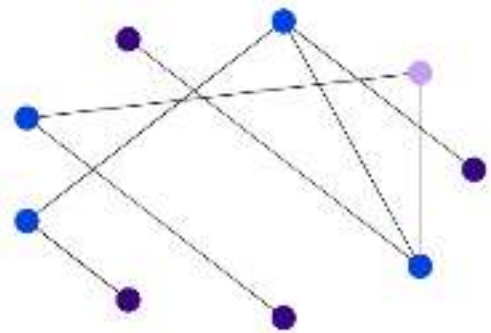


initial graph

Example

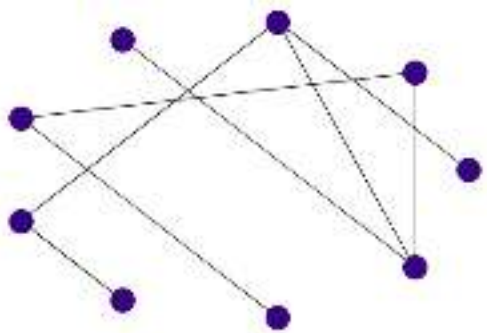


initial graph

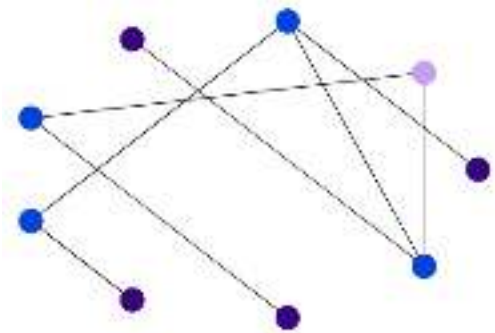


colouring after round 1

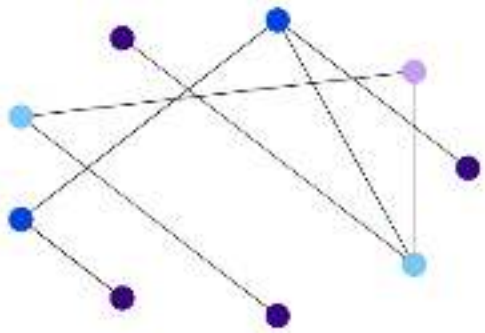
Example



initial graph

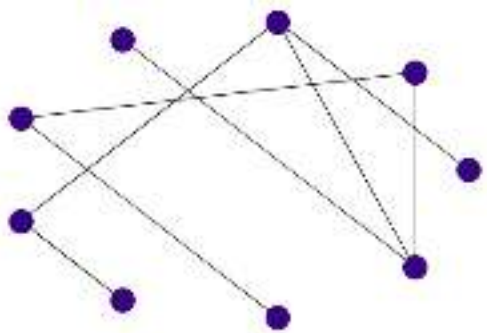


colouring after round 1

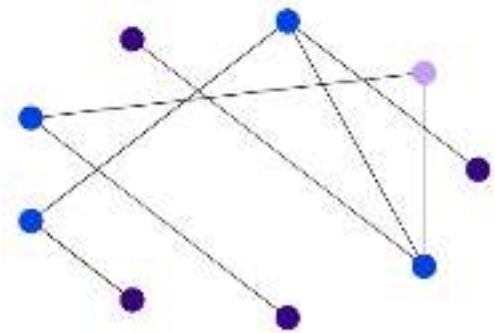


colouring after round 2

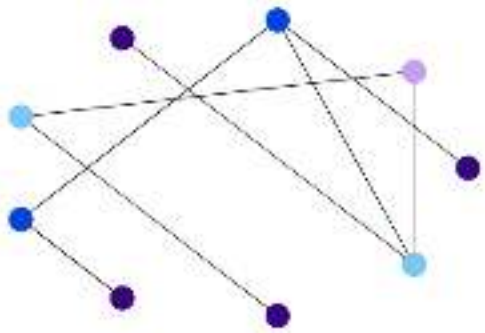
Example



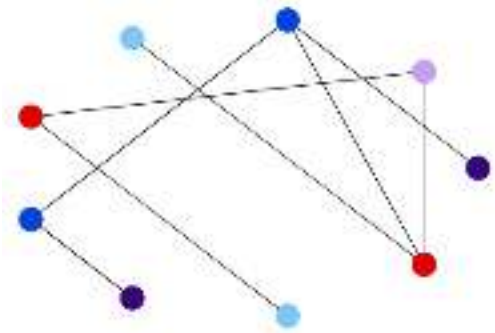
initial graph



colouring after round 1



colouring after round 2



stable colouring after round 3

Theorem (Cardon, Crochemore, 1982, Paige, Tarjan, 1987)

The stable colouring of a given graph with n vertices and m edges can be computed in time $O((n+m)\log n)$.

Theorem (Cardon, Crochemore, 1982, Paige, Tarjan, 1987)

The stable colouring of a given graph with n vertices and m edges can be computed in time $O((n+m)\log n)$.

This is optimal for a fairly general class of partitioning algorithms (Berkholz, Bonsma, G. 2017).

1-WL as an Isomorphism Test

1-WL distinguishes two graphs G, H if their colour histograms differ, that is, some colour appears a different number of times in G and H.

1-WL as an Isomorphism Test

1-WL distinguishes two graphs G, H if their colour histograms differ, that is, some colour appears a different number of times in G and H.

Thus 1-WL can be used as an incomplete isomorphism test.

1-WL as an Isomorphism Test

1-WL distinguishes two graphs G, H if their colour histograms differ, that is, some colour appears a different number of times in G and H.

Thus 1-WL can be used as an incomplete isomorphism test.

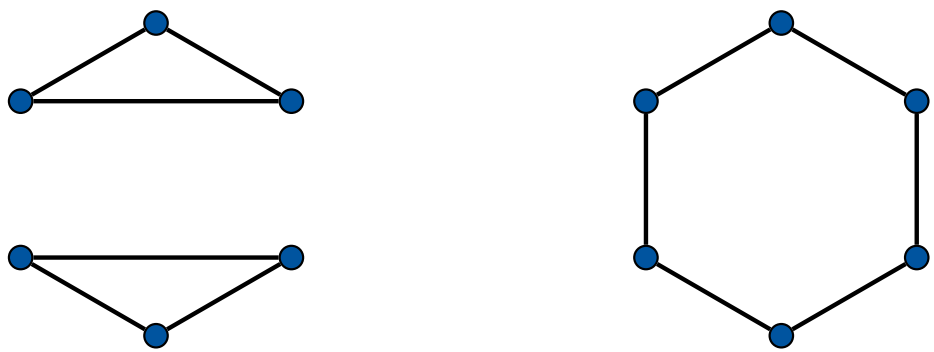
- works on almost all graphs (Babai, Erdös, Selkow 1980)

1-WL as an Isomorphism Test

1-WL distinguishes two graphs G, H if their colour histograms differ, that is, some colour appears a different number of times in G and H.

Thus 1-WL can be used as an incomplete isomorphism test.

- works on almost all graphs (Babai, Erdös, Selkow 1980)
- fails on some very simple graphs:



Colours and Trees

Λ_r = set of all colours used by 1-WL in the first r rounds

Colours and Trees

Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

Colours and Trees

Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

We can think of the colours in Λ_r as rooted trees of height r.

Colours and Trees

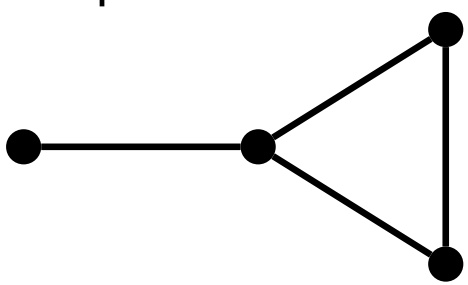
Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

We can think of the colours in Λ_r as rooted trees of height r.

Example

Graph G



Colours and Trees

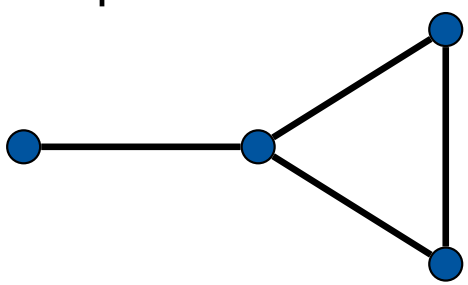
Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

We can think of the colours in Λ_r as rooted trees of height r.

Example

Graph G



Round 0



Colours and Trees

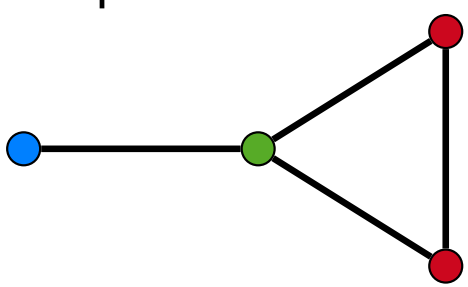
Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

We can think of the colours in Λ_r as rooted trees of height r.

Example

Graph G



Round 0



Round 1



Colours and Trees

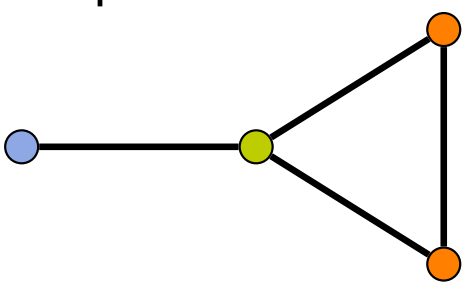
Λ_r = set of all colours used by 1-WL in the first r rounds

$\Lambda = \bigcup_{r \geq 0} \Lambda_r$ = set of all colours used

We can think of the colours in Λ_r as rooted trees of height r.

Example

Graph G



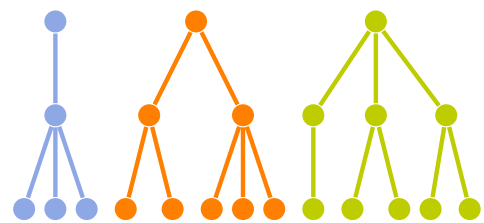
Round 0



Round 1



Round 2



Weisfeiler-Leman Graph Kernels

For $\lambda \in \Lambda$: $\mathsf{wl}(\lambda, G) =$ number of vertices of G of colour λ

For $\lambda \in \Lambda$: $\mathsf{wl}(\lambda, G) =$ number of vertices of G of colour λ

WL Graph Embeddings

Graph embeddings WL_r for every $r \geq 0$ defined by:

$$\mathsf{WL}_r(G) = \big(\mathsf{wl}(\lambda, G) \mid \lambda \in \Lambda_r\big) \in \mathbb{R}^{\Lambda_r}.$$

Weisfeiler-Leman Graph Kernels

For $\lambda \in \Lambda$: $\mathsf{wl}(\lambda, G) =$ number of vertices of G of colour λ

WL Graph Embeddings

Graph embeddings WL_r for every $r \geq 0$ defined by:

$$\mathsf{WL}_r(G) = \big(\mathsf{wl}(\lambda, G) \mid \lambda \in \Lambda_r\big) \in \mathbb{R}^{\Lambda_r}.$$

WL Graph Kernels

The r-round WL-kernel is the mapping $K_{WL}^{(r)}$ defined by

$$K_{WL}^{(r)}(G, H) := \sum_{\ell=0}^{r} \sum_{\lambda \in \Lambda_\ell} \mathsf{wl}(\lambda, G) \cdot \mathsf{wl}(\lambda, H).$$

(Shervashidze, Schweitzer, van Leeuwen, Mehlhorn, Borgwardt 2011)

Weisfeiler-Leman Graph Kernels

For $\lambda \in \Lambda$: $\mathsf{wl}(\lambda, G) =$ number of vertices of G of colour λ

WL Graph Embeddings

Graph embeddings WL_r for every $r \geq 0$ defined by:

$$\mathsf{WL}_r(G) = \big(\mathsf{wl}(\lambda, G) \mid \lambda \in \Lambda_r\big) \in \mathbb{R}^{\Lambda_r}.$$

WL Graph Kernels

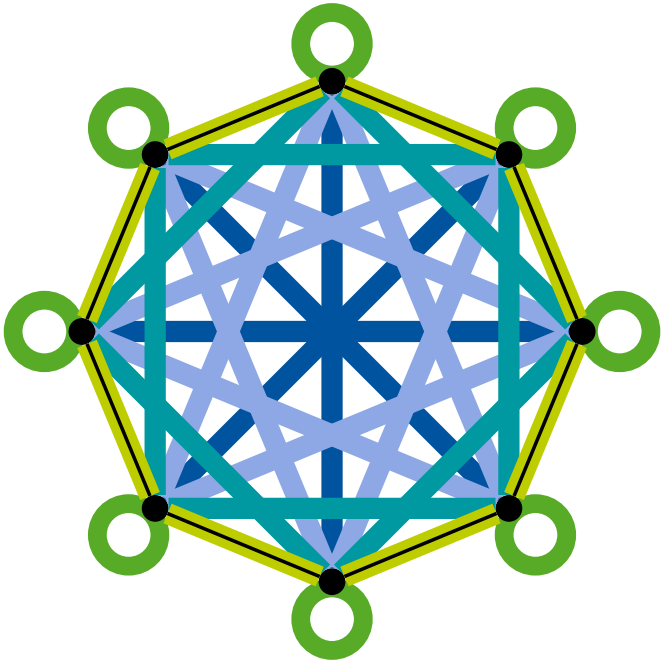
The r-round WL-kernel is the mapping $K^{(r)}_{WL}$ defined by

$$K^{(r)}_{WL}(G, H) := \sum_{\ell=0}^{r} \sum_{\lambda \in \Lambda_\ell} \mathsf{wl}(\lambda, G) \cdot \mathsf{wl}(\lambda, H).$$

(Shervashidze, Schweitzer, van Leeuwen, Mehlhorn, Borgwardt 2011)

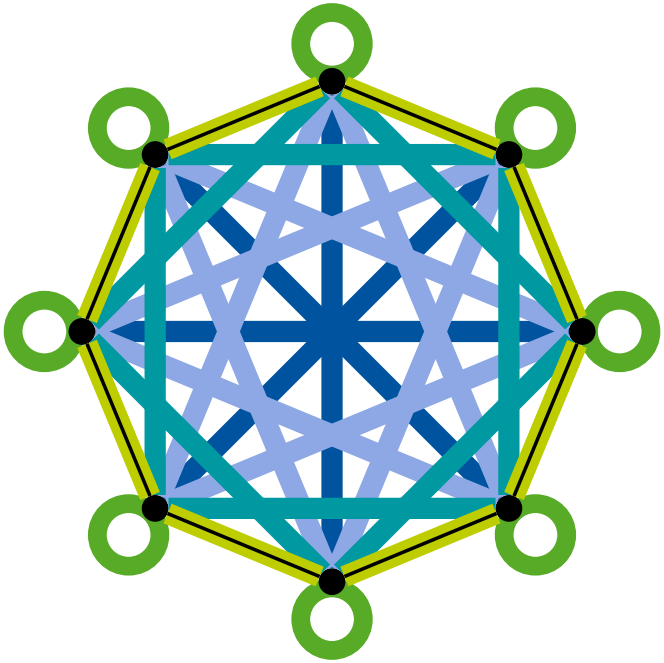
We can also define an infinite-dimensional version based on the embedding $\mathsf{WL}(G) = \big(\mathsf{wl}(\lambda, G) \mid \lambda \in \Lambda\big) \in \mathbb{R}^{\Lambda}$.

Higher-Dimensional Weisfeiler-Leman



The k-dimensional Weisfeiler-Leman algorithm (k-WL) iteratively colours k-tuples of vertices (Weisfeiler and Leman 1968, Babai $\sim$1980)

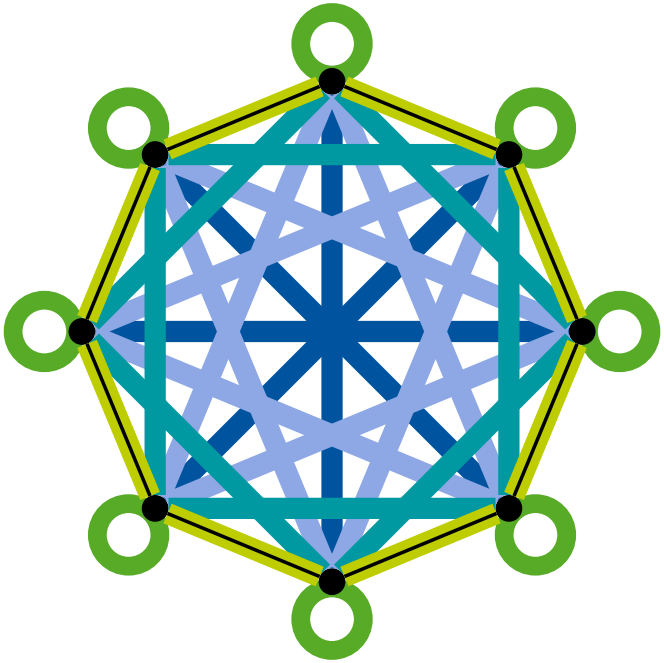
Higher-Dimensional Weisfeiler-Leman



The k-dimensional Weisfeiler-Leman algorithm (k-WL) iteratively colours k-tuples of vertices (Weisfeiler and Leman 1968, Babai $\sim$1980)

Running time: $O(n^k \log n)$

Higher-Dimensional Weisfeiler-Leman



The k-dimensional Weisfeiler-Leman algorithm (k-WL) iteratively colours k-tuples of vertices (Weisfeiler and Leman 1968, Babai $\sim$1980)

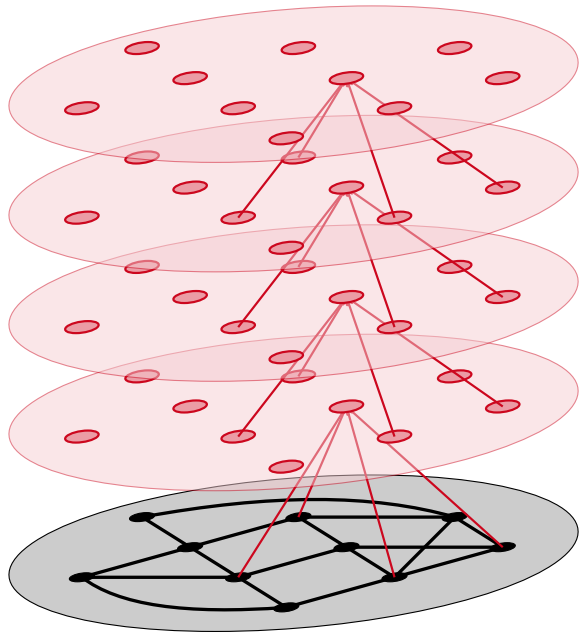
Running time: $O(n^k \log n)$

k-WL is much more powerful than 1-WL, but still not a complete isomorphism test.

Theorem (Cai, Fürer, Immerman 1992)

For every k there are non-isomorphic graphs G_k, H_k not distinguished by k-WL.

Weisfeiler and Leman go Neural



Theorem (Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, G. 2019, Xu, Hu, Leskovec, Jegelka 2019)

GNNs can extract exactly the same information from a graph as 1-WL.

Digression: GNNs in Chemical Engineering

Higher Dimensional GNNs

Based on the higher dimensional WL algorithm (Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, G. 2019).

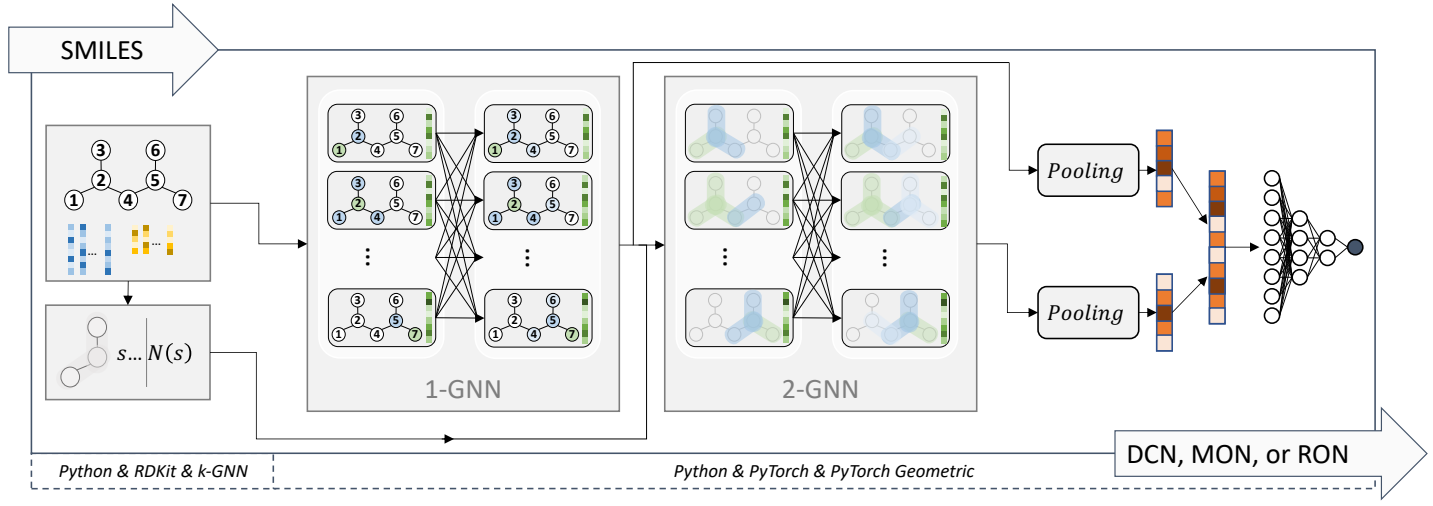
Digression: GNNs in Chemical Engineering

Higher Dimensional GNNs

Based on the higher dimensional WL algorithm (Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, G. 2019).

Predicting Fuel Ignition Quality

Prediction model for combustion-related properties of hydrocarbons based on 2-dimensional GNNs



(Schweidtmann, Rittig, König, G., Mitsos, and Dahmen, 2020)

Logical Characterisation

C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Logical Characterisation

C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Theorem (Cai, Fürer, Immerman 1992)

For every $k \geq 1$ and all graphs G, H, the following are equivalent.

1. *k-WL does not distinguish G and H.*
2. *G and H satisfy the same sentences of the logic C with at most $(k+1)$ variables.*

Fractional Isomorphisms

G, H graphs with vertex sets V, W and adjacency matrices A, B

Fractional Isomorphisms

G, H graphs with vertex sets V, W and adjacency matrices A, B

System $L(G, H)$ of linear equations in variables X_{vw}:

$$\sum_{v' \in V} A_{vv'} X_{v'w} = \sum_{w' \in W} X_{vw'} B_{w'w} \qquad \text{for all } v \in V, w \in W,$$

$$\sum_{v' \in V} X_{v'w} = \sum_{w' \in W} X_{vw'} = 1 \qquad \text{for all } v \in V, w \in W.$$

Fractional Isomorphisms

G, H graphs with vertex sets V, W and adjacency matrices A, B

System $L(G, H)$ of linear equations in variables X_{vw}:

$$\sum_{v' \in V} A_{vv'} X_{v'w} = \sum_{w' \in W} X_{vw'} B_{w'w} \qquad \text{for all } v \in V, w \in W,$$

$$\sum_{v' \in V} X_{v'w} = \sum_{w' \in W} X_{vw'} = 1 \qquad \text{for all } v \in V, w \in W.$$

Observation

$L(G, H)$ has a nonnegative integer solution iff G and H are isomorphic.

Fractional Isomorphisms

G, H graphs with vertex sets V, W and adjacency matrices A, B

System $L(G, H)$ of linear equations in variables X_{vw}:

$$\sum_{v' \in V} A_{vv'} X_{v'w} = \sum_{w' \in W} X_{vw'} B_{w'w} \qquad \text{for all } v \in V, w \in W,$$

$$\sum_{v' \in V} X_{v'w} = \sum_{w' \in W} X_{vw'} = 1 \qquad \text{for all } v \in V, w \in W.$$

Observation

$L(G, H)$ has a nonnegative integer solution iff G and H are isomorphic.

Theorem (Tinhofer 1990)

$L(G, H)$ has a nonnegative rational solution (called a fractional isomorphism) if and only if 1-WL does not distinguish G, H.

Higher Dimensional Version

Atserias and Maneva 2013

- There is a close correspondence between k-WL and the kth level of the Sherali-Adams hierarchy over $L(G, H)$ (together with the inequalities $X_{vw} \geq 0$).

Higher Dimensional Version

Atserias and Maneva 2013

- There is a close correspondence between k-WL and the kth level of the Sherali-Adams hierarchy over $L(G, H)$ (together with the inequalities $X_{vw} \geq 0$).
- Exact correspondence by G. and Otto 2015.

Higher Dimensional Version

Atserias and Maneva 2013

- There is a close correspondence between k-WL and the kth level of the Sherali-Adams hierarchy over $L(G, H)$ (together with the inequalities $X_{vw} \geq 0$).
- Exact correspondence by G. and Otto 2015.

We denote the system of linear equalities whose nonnegative rational solutions characterise k-WL by $L^{(k)}(G, H)$.

Homomorphism Vectors

Homomorphism Counts

$\hom(F, G) :=$ number of homomorphisms from F to G

Counting Graph Homomorphisms

Homomorphism Counts

$$\hom(F, G) := \text{number of homomorphisms from } F \text{ to } G$$

Homomorphism Graph Embeddings

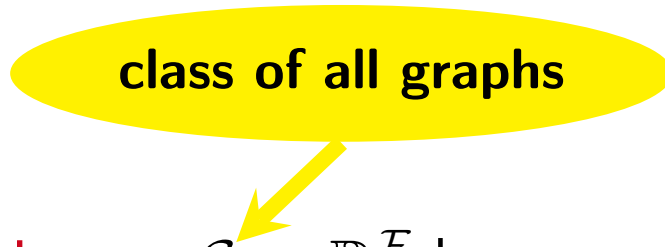
For every class $\mathcal{F}$ of graphs, we define $\mathrm{Hom}_{\mathcal{F}} : \mathcal{G} \to \mathbb{R}^{\mathcal{F}}$ by

$$\mathrm{Hom}_{\mathcal{F}}(G) := \Big(\hom(F, G) \;\Big|\; F \in \mathcal{F} \Big).$$

Counting Graph Homomorphisms

Homomorphism Counts

$$\hom(F, G) := \text{number of homomorphisms from } F \text{ to } G$$



Homomorphism Graph Embeddings

For every class $\mathcal{F}$ of graphs, we define $\mathrm{Hom}_{\mathcal{F}} : \mathcal{G} \to \mathbb{R}^{\mathcal{F}}$ by

$$\mathrm{Hom}_{\mathcal{F}}(G) := \Big(\hom(F, G) \;\Big|\; F \in \mathcal{F} \Big).$$

Homomorphism Counts

$$\hom(F, G) := \text{number of homomorphisms from } F \text{ to } G$$

Homomorphism Graph Embeddings

For every class $\mathcal{F}$ of graphs, we define $\mathrm{Hom}_{\mathcal{F}} : \mathcal{G} \to \mathbb{R}^{\mathcal{F}}$ by

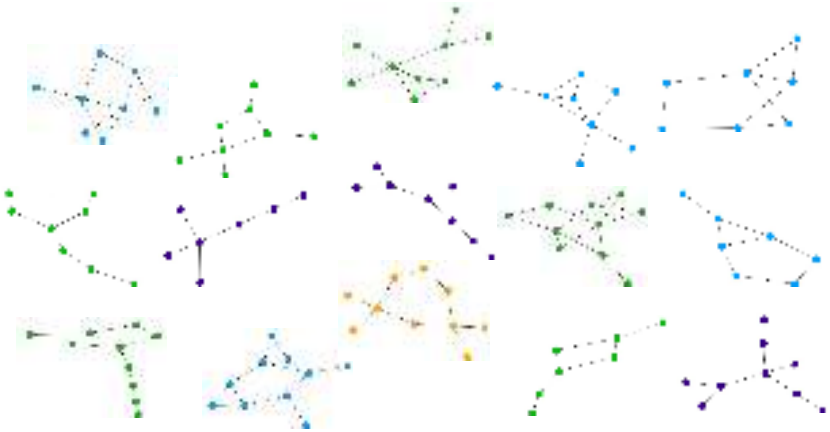
$$\mathrm{Hom}_{\mathcal{F}}(G) := \Big(\hom(F, G) \;\Big|\; F \in \mathcal{F} \Big).$$

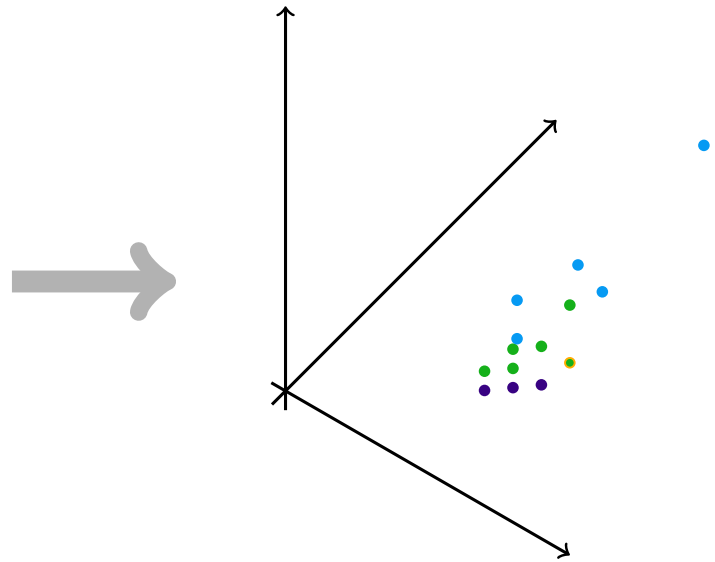
We can define an inner product on $\mathrm{rg}(\mathrm{Hom}_{\mathcal{F}})$ by

$$\Big\langle \mathrm{Hom}_{\mathcal{F}}(G), \mathrm{Hom}_{\mathcal{F}}(H) \Big\rangle = \sum_{k \geq 1} \frac{1}{2^k \cdot |\mathcal{F}_k|} \sum_{F \in \mathcal{F}_k} \hom(F, G) \cdot \hom(F, H),$$

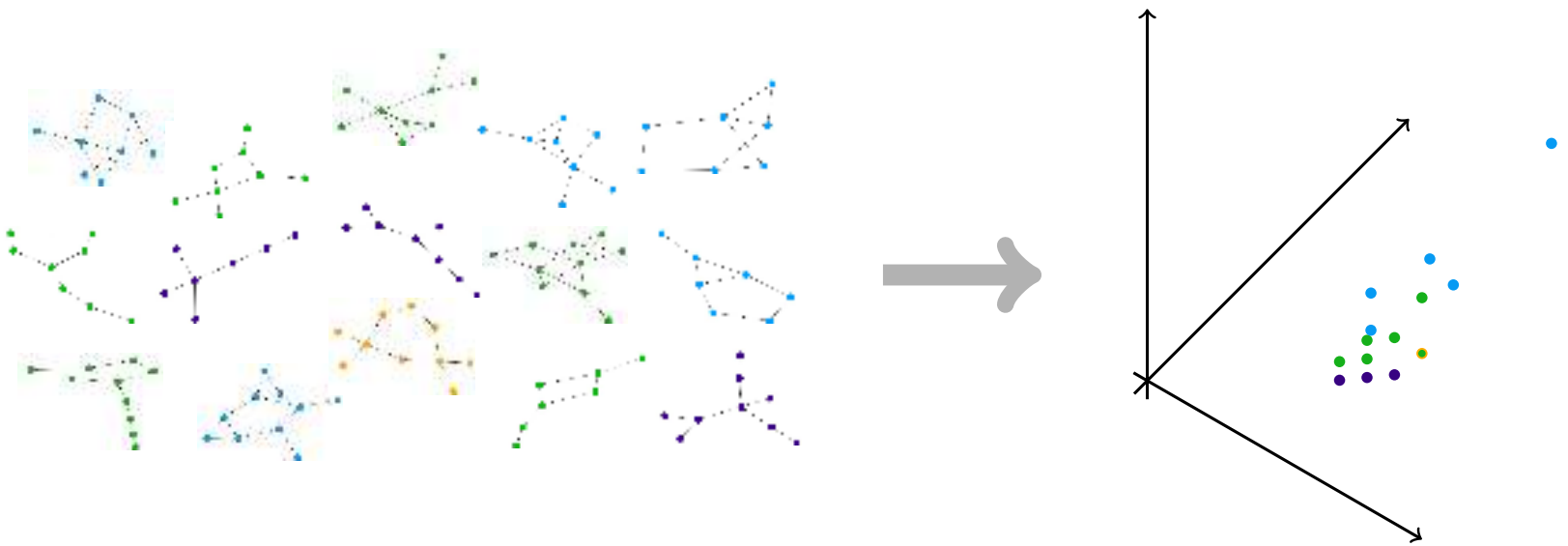
where $\mathcal{F}_k := \big\{ F \in \mathcal{F} \;\big|\; |F| = k \big\}$.

Example



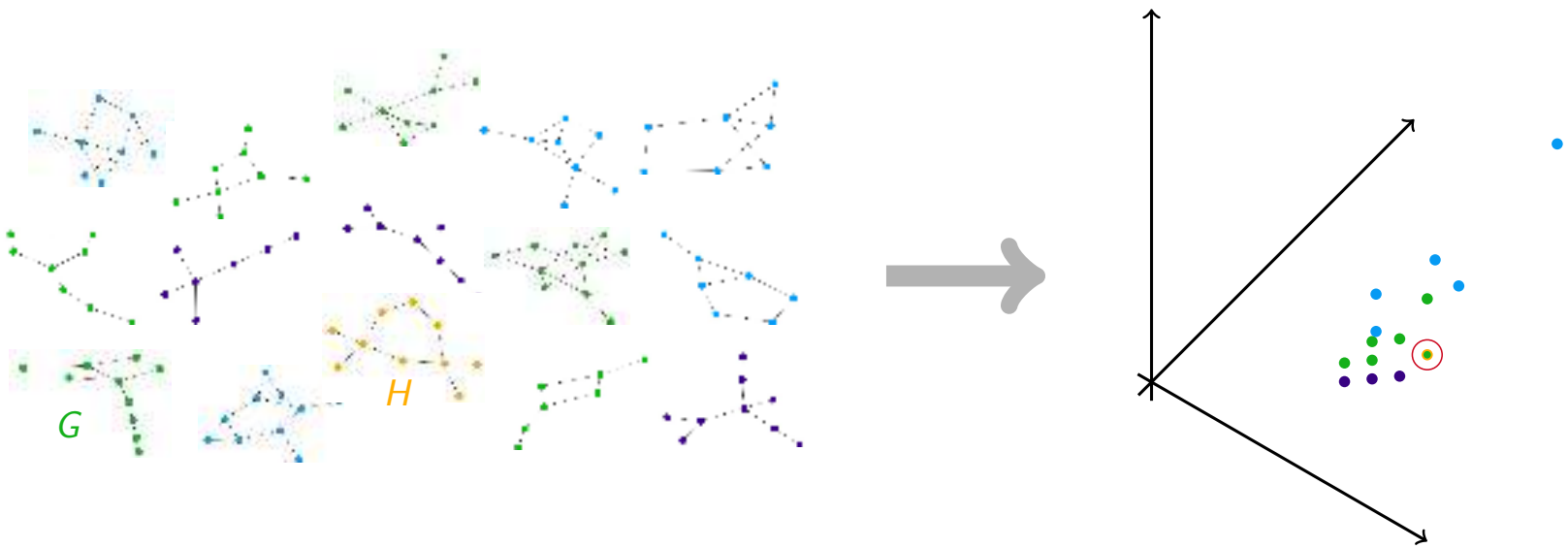


Example



The embedding $\mathrm{Hom}_{\mathcal{F}}$ for $\mathcal{F} = \{\bullet, \bullet\!\!-\!\!\bullet, \triangle\}$

Example



The embedding $\mathrm{Hom}_{\mathcal{F}}$ for $\mathcal{F} = \{\bullet, \bullet\!-\!\bullet, \triangle\}$

The graphs G and H are mapped to the same vector:

$$\mathrm{Hom}_{\mathcal{F}}(G) = \mathrm{Hom}_{\mathcal{F}}(H) = (10, 20, 0).$$

Homomorphism Indistinguishability

Graph G and H are homomorphism indistinguishable over a class $\mathcal{F}$ of graphs if $\mathrm{Hom}_{\mathcal{F}}(G) = \mathrm{Hom}_{\mathcal{F}}(H)$.

Homomorphism Indistinguishability

Graph G and H are homomorphism indistinguishable over a class $\mathcal{F}$ of graphs if $\mathrm{Hom}_{\mathcal{F}}(G) = \mathrm{Hom}_{\mathcal{F}}(H)$.

Theorem (Lovász 1967)

For all graphs G, H, the following are equivalent.

1. *G and H are homomorphism indistinguishable over the class of all graphs.*
2. *G and H are isomorphic.*

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.
Row and column indices are ordered by increasing size.

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.
Row and column indices are ordered by increasing size.

Observation 1

I is upper triangular with positive diagonal entries and S is lower triangular with positive diagonal entries.

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.
Row and column indices are ordered by increasing size.

Observation 1

I is upper triangular with positive diagonal entries and S is lower triangular with positive diagonal entries.

Observation 2

$\mathrm{hom}(F, H) = \sum_G S(F, G) \cdot \frac{1}{\mathrm{aut}(G)} \cdot I(G, H)$.

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.
Row and column indices are ordered by increasing size.

Observation 1

I is upper triangular with positive diagonal entries and S is lower triangular with positive diagonal entries.

Observation 2

$\mathrm{hom}(F, H) = \sum_G S(F, G) \cdot \frac{1}{\mathrm{aut}(G)} \cdot I(G, H)$.

That is, $\mathrm{hom} = S \cdot D \cdot I$ for a diagonal matrix D with positive diagonal entries.

Proof

$\mathrm{aut}(G) :=$ number of automorphisms of G

$I(F, G) :=$ number of injective homomorphisms from F to G

$S(F, G) :=$ number of surjective homomorphisms from F to G

We view hom, I, and S as infinite matrices in $\mathbb{R}^{\mathcal{G} \times \mathcal{G}}$.
Row and column indices are ordered by increasing size.

Observation 1

I is upper triangular with positive diagonal entries and S is lower triangular with positive diagonal entries.

Observation 2

$\hom(F, H) = \sum_G S(F, G) \cdot \frac{1}{\mathrm{aut}(G)} \cdot I(G, H)$.

That is, $\hom = S \cdot D \cdot I$ for a diagonal matrix D with positive diagonal entries.

Hence the columns of hom are linearly independent and thus mutually distinct.

Theorem (Dvorák 2010)

For all graphs G, H, the following are equivalent.

1. *G and H are homomorphism indistinguishable over the class of all graphs of tree width at most k.*
2. *G and H are not distinguished by the k-dimensional Weisfeiler-Leman graph isomorphism algorithm.*

Theorem (Dvorák 2010)

For all graphs G, H, the following are equivalent.

1. *G and H are homomorphism indistinguishable over the class of all graphs of tree width at most k.*
2. *G and H are not distinguished by the k-dimensional Weisfeiler-Leman graph isomorphism algorithm.*

Corollary

For all graphs G, H, the following are equivalent.

1. *G and H are homomorphism indistinguishable over the class of all graphs of tree width at most k*
2. *$L^{(k)}(G, H)$ has a nonnegative rational solution.*

Theorem (Dell, G., Rattan 2018)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all paths.*
2. *$L(G, H)$ has a rational solution.*

Paths, Cycles, and Planar Graphs

Theorem (Dell, G., Rattan 2018)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all paths.*
2. *$L(G, H)$ has a rational solution.*

Theorem (Folklore)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all cycles.*
2. *G and H are co-spectral.*

Paths, Cycles, and Planar Graphs

Theorem (Dell, G., Rattan 2018)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all paths.*
2. *$L(G, H)$ has a rational solution.*

Theorem (Folklore)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all cycles.*
2. *G and H are co-spectral.*

Theorem (Mančinska and Roberson 2019)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all planar graphs.*
2. *G and H are quantum isomorphic.*

Logical Characterisations

Recall: C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Logical Characterisations

Recall: C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Corollary (Dvorák 2010)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all graphs of tree width at most k.*
2. *G and H satisfy the same sentences of the logic C with at most $k+1$ variables.*

Logical Characterisations

Recall: C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Corollary (Dvorák 2010)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all graphs of tree width at most k.*
2. *G and H satisfy the same sentences of the logic C with at most $k+1$ variables.*

Theorem (G. 2020)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all graphs of tree depth at most k.*
2. *G and H satisfy the same sentences of the logic C of quantifier rank at most k.*

Logical Characterisations

Recall: C is the (syntactical) extension of first-order predicate logic with counting quantifiers $\exists^{\geq i} x$.

Corollary (Dvorák 2010)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all graphs of tree width at most k.*
2. *G and H satisfy the same sentences of the logic C with at most $k+1$ variables.*

Theorem (G. 2020)

For all graphs G, H, the following are equivalent.

1. *G and H are hom. ind. over the class of all graphs of tree depth at most k.*
2. *G and H satisfy the same sentences of the logic C of quantifier rank at most k.*

Remarks

- Lovász's deep theory of Graph Limits is based on the metric induced by $\mathrm{Hom}_{\mathcal{G}}$.

Remarks

- Lovász's deep theory of Graph Limits is based on the metric induced by $\mathrm{Hom}_{\mathcal{G}}$.
- Homomorphism embeddings can be extended to arbitrary relational structures and to weighted graphs and structures; many of the results are preserved.

Remarks

- Lovász's deep theory of Graph Limits is based on the metric induced by $\mathrm{Hom}_{\mathcal{G}}$.
- Homomorphism embeddings can be extended to arbitrary relational structures and to weighted graphs and structures; many of the results are preserved.
- There are also homomorphism node embeddings based on homomorphism counts of rooted graphs.

Concluding Remarks

Research Directions

- Study relations between graph metrics, for example, the measure induced by WL-Graph Kernels and the homomorphism vectors over trees.

Research Directions

- Study relations between graph metrics, for example, the measure induced by WL-Graph Kernels and the homomorphism vectors over trees.
- Design algorithms for reconstructing graphs from vectors.

Research Directions

- Study relations between graph metrics, for example, the measure induced by WL-Graph Kernels and the homomorphism vectors over trees.
- Design algorithms for reconstructing graphs from vectors.
- Design vector embeddings for more complex objects like relational structures, processes, dynamical systems.

Research Directions

- Study relations between graph metrics, for example, the measure induced by WL-Graph Kernels and the homomorphism vectors over trees.
- Design algorithms for reconstructing graphs from vectors.
- Design vector embeddings for more complex objects like relational structures, processes, dynamical systems.
- Develop a better understanding of the semantics of vector embeddings and answer queries directly on the embedded data.