Expectation Complete Graph Embeddings Using Graph Homomorphisms



Workshop: Hot Topics in Graph Neural Networks, GAIN Group, Uni Kassel

Maximilian Thiessen, Pascal Welke, and Thomas Gärtner

25.10.2022

TU Wien
Vienna | Austria
Research Unit Machine Learning

UNIVERSITÄT BONN
ML AI lab

Complete graph embeddings

Let $\mathcal{G}$ be the set of all (finite) graphs, V be a vector space (e.g., $\mathbb{R}^d$)

Complete graph embeddings

Let $\mathcal{G}$ be the set of all (finite) graphs, V be a vector space (e.g., $\mathbb{R}^d$)

A graph embedding $\varphi : \mathcal{G} \to V$ is permutation-invariant if

- For all isomorphic graphs $G \simeq H$: $\varphi(G) = \varphi(H)$

Complete graph embeddings

Let $\mathcal{G}$ be the set of all (finite) graphs, V be a vector space (e.g., $\mathbb{R}^d$)

A graph embedding $\varphi : \mathcal{G} \to V$ is permutation-invariant if

- For all isomorphic graphs $G \simeq H$: $\varphi(G) = \varphi(H)$

A permutation-invariant graph embedding φ is complete if

- for all non-isomorphic graphs $G \not\simeq H : \varphi(G) \neq \varphi(H)$

Complete graph embeddings

Originated from complete graph kernels [Gärtner et al., COLT 2003]

- let $\mathcal{H}$ be a dot product space[1]
- graph kernel $k_\varphi(G, H) = \langle \varphi(G), \varphi(H) \rangle_\mathcal{H}$ with $\varphi : \mathcal{G} \to \mathcal{H}$
- k_φ is complete if φ is complete

Complete graph embeddings

Why do we care about complete graph embeddings?

Allow us to learn/approximate any permutation-invariant function!

Complete graph embeddings

Why do we care about complete graph embeddings?

Allow us to learn/approximate any permutation-invariant function!

Unfortunately computing any such embedding (or kernel) is as hard as deciding graph isomorphism

- not known to be NP-hard and not known to be computable in polynomial-time

Complete graph embeddings

Why do we care about complete graph embeddings?

Allow us to learn/approximate any permutation-invariant function!

Unfortunately computing any such embedding (or kernel) is as hard as deciding graph isomorphism

- not known to be NP-hard and not known to be computable in polynomial-time

Typical solution: drop completeness for efficiency

- most practical graph kernels, GNNs, Weisfeiler Leman test, ...

What if we keep completeness ...

... but just in expectation

Expectation complete graph embeddings

Let $\varphi_X : \mathcal{G} \to V$ depend on a random variable X drawn from a distr. $\mathcal{D}$ over a set $\mathcal{X}$[1]

 [1]here assumed to be countable, but any set with a probability distribution would do

Expectation complete graph embeddings

Let $\varphi_X : \mathcal{G} \to V$ depend on a random variable X drawn from a distr. $\mathcal{D}$ over a set $\mathcal{X}$[1]

We call φ_X complete in expectation if the expectation

$$\mathop{\mathbb{E}}_{X\sim\mathcal{D}}[\varphi_X(\cdot)] = \sum_{t\in\mathcal{X}} \Pr(X = t)\varphi_t(\cdot)$$

is a complete graph embedding

 [1]here assumed to be countable, but any set with a probability distribution would do

Expectation complete graph embeddings

Let $\varphi_X : \mathcal{G} \to V$ depend on a random variable X drawn from a distr. $\mathcal{D}$ over a set $\mathcal{X}$[1]

We call φ_X complete in expectation if the expectation

$$\mathop{\mathbb{E}}_{X\sim\mathcal{D}}[\varphi_X(\cdot)] = \sum_{t\in\mathcal{X}} \Pr(X = t)\varphi_t(\cdot)$$

is a complete graph embedding

What is the benefit?

Sampling $X_1, X_2, X_3, \dots$ will eventually make the
joint embedding $(\varphi_{X_1}(G), \varphi_{X_2}(G), \varphi_{X_3}(G), \dots)$ arbitrarily expressive

 [1]here assumed to be countable, but any set with a probability distribution would do

What if we keep completeness …

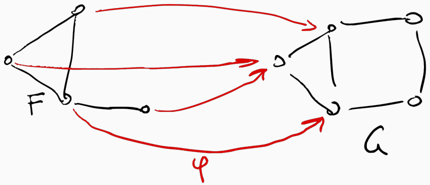
… but just in expectation

… in polynomial time

Graph homomorphisms and Lovász' theorem

Let F, G be graphs. A map $\varphi : V(F) \to V(G)$ is a graph homomorphism if

- φ preserves edges: $\{v, w\} \in E(F)$ implies $\{\varphi(v), \varphi(w)\} \in E(G)$



We denote by $\mathbf{hom}(F, G)$ the number of homomorphisms from F to G

Graph homomorphisms and Lovász' theorem

Let

$$\varphi_\infty(G) = \mathrm{hom}(\mathcal{G}, G) = \left((\mathrm{hom}(F, G)\right)_{F \in \mathcal{G}}$$

denote the countable vector of homomorphism counts indexed by $F \in \mathcal{G}$

Let

$$\varphi_\infty(G) = \hom(\mathcal{G}, G) = ((\hom(F, G))_{F \in \mathcal{G}} = \begin{pmatrix} \vdots \\ \hom(F,G) \\ \vdots \end{pmatrix}_F$$

denote the countable vector of homomorphism counts indexed by $F \in \mathcal{G}$

Graph homomorphisms and Lovász' theorem

Let

$$\varphi_\infty(G) = \mathsf{hom}(\mathcal{G}, G) = ((\mathsf{hom}(F, G))_{F \in \mathcal{G}}$$

denote the countable vector of homomorphism counts indexed by $F \in \mathcal{G}$

Theorem [Lovász 1967]. Two graphs G and H are isomorphic iff $\varphi_\infty(G) = \varphi_\infty(H)$

$\Rightarrow \varphi_\infty(\cdot)$ is complete!

Graph homomorphisms and Lovász' theorem

Let

$$\varphi_\infty(G) = \hom(\mathcal{G}, G) = ((\hom(F, G))_{F \in \mathcal{G}}$$

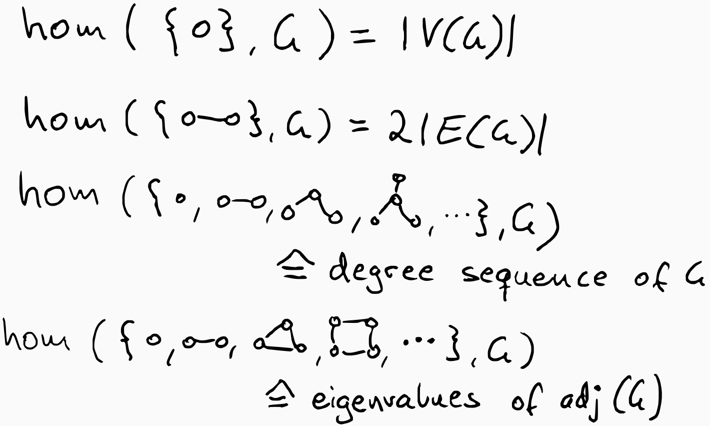
denote the countable vector of homomorphism counts indexed by $F \in \mathcal{G}$

Theorem [Lovász 1967]. Two graphs G and H are isomorphic iff $\varphi_\infty(G) = \varphi_\infty(H)$

$\Rightarrow \varphi_\infty(\cdot)$ is complete!

Our goal: sample from φ_∞ to devise an efficiently computable and expectation complete embedding

They capture important graph properties:



Why graph homomorphisms

They capture aspects important for learning:

$\text{hom}(\{F \mid F \text{ is a tree}\}, G) \mathrel{\hat{=}} 1\text{-WL} \mathrel{\hat{=}} \text{GNNs}$

$\text{hom}(\{F \mid tw(F) \leq k\}, G) \mathrel{\hat{=}} k\text{-WL} \mathrel{\hat{=}} k\text{-GNNs}$

↑ treewidth of F ("tree-likeness")

Why graph homomorphisms

They capture aspects important for learning:

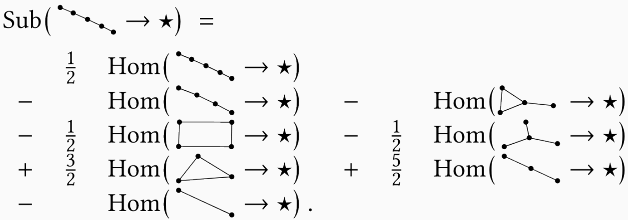
$$\mathrm{hom}(\{F \mid F \text{ is a tree}\}, G) \;\hat{=}\; 1\text{-WL} \;\hat{=}\; \text{GNNs}$$

$$\mathrm{hom}(\{F \mid tw(F) \leq k\}, G) \;\hat{=}\; k\text{-WL} \;\hat{=}\; k\text{-GNNs}$$

↑ treewidth of F ("tree-likeness")

Universality: Any permutation-invariant function $f : \mathcal{G} \to \mathbb{R}^d$ can be approximated arbitrarily well by a polynomial of $\{\mathrm{hom}(F, G) \mid F \in \mathcal{G}\}$ [NT and Maehara, 2020]

They can be used for subgraph counting [Curticapean et al., STOC 2017]



Expectation complete embeddings on $\mathcal{G}_n$

Let

- $\mathcal{G}_n$ be the set of graphs with up to n vertices,
- $\mathcal{D}$ a distribution on $\mathcal{G}_n$ with full support,
- a random pattern $F \sim \mathcal{D}$, and
- $\varphi_n(\cdot) = \mathsf{hom}(\mathcal{G}_n, \cdot)$

Expectation complete embeddings on $\mathcal{G}_n$

Let

- $\mathcal{G}_n$ be the set of graphs with up to n vertices,
- $\mathcal{D}$ a distribution on $\mathcal{G}_n$ with full support,
- a random pattern $F \sim \mathcal{D}$, and
- $\varphi_n(\cdot) = \mathrm{hom}(\mathcal{G}_n, \cdot)$

Define

$$\varphi_F(G) = (\varphi_n(G))_F$$

which samples the 'Fth' entry of φ_n

Expectation complete embeddings on $\mathcal{G}_n$

Let

- $\mathcal{G}_n$ be the set of graphs with up to n vertices,
- $\mathcal{D}$ a distribution on $\mathcal{G}_n$ with full support,
- a random pattern $F \sim \mathcal{D}$, and
- $\varphi_n(\cdot) = \mathrm{hom}(\mathcal{G}_n, \cdot)$

Define

$$\varphi_F(G) = (\varphi_n(G))_F = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathrm{hom}(F,G) \\ 0 \\ \vdots \\ 0 \end{pmatrix} F$$

which samples the 'Fth' entry of φ_n

Expectation complete embeddings on $\mathcal{G}_n$

Let

- $\mathcal{G}_n$ be the set of graphs with up to n vertices,
- $\mathcal{D}$ a distribution on $\mathcal{G}_n$ with full support,
- a random pattern $F \sim \mathcal{D}$, and
- $\varphi_n(\cdot) = \mathbf{hom}(\mathcal{G}_n, \cdot)$

Define

$$\varphi_F(G) = (\varphi_n(G))_F$$

which samples the 'Fth' entry of φ_n

Theorem. φ_F is complete in expectation (on $\mathcal{G}_n$)

Expectation complete embeddings on $\mathcal{G}$?

Can we generalise to all finite graphs $\mathcal{G}$?

Expectation complete embeddings on $\mathcal{G}$?

Can we generalise to all finite graphs $\mathcal{G}$?

Problem: φ_∞ does not yield a norm / dot product

- e.g., $|\varphi_\infty(G)|^2 = \langle \varphi_\infty(G), \varphi_\infty(G) \rangle = \infty$ in most cases

Expectation complete embeddings on $\mathcal{G}$?

Can we generalise to all finite graphs $\mathcal{G}$?

Problem: φ_∞ does not yield a norm / dot product

- e.g., $|\varphi_\infty(G)|^2 = \langle \varphi_\infty(G), \varphi_\infty(G) \rangle = \infty$ in most cases

Solution: only count patterns up to $|V(G)|$:

$\overline{\varphi}_\infty(G) = \left(\mathrm{hom}_{|V(G)|}(F, G)\right)_{F \in \mathcal{G}}$ where

$$\mathrm{hom}_{|V(G)|}(F, G) = \begin{cases} \mathrm{hom}(F, G) & \text{if } |V(F)| \leq |V(G)|\,, \\ 0 & \text{if } |V(F)| > |V(G)|\,. \end{cases}$$

Expectation complete embeddings on $\mathcal{G}$?

Can we generalise to all finite graphs $\mathcal{G}$?

Problem: φ_∞ does not yield a norm / dot product

- e.g., $|\varphi_\infty(G)|^2 = \langle \varphi_\infty(G), \varphi_\infty(G) \rangle = \infty$ in most cases

Solution: only count patterns up to $|V(G)|$:

$\overline{\varphi}_\infty(G) = \left(\mathsf{hom}_{|V(G)|}(F, G)\right)_{F \in \mathcal{G}}$ where

$$\mathsf{hom}_{|V(G)|}(F, G) = \begin{cases} \mathsf{hom}(F, G) & \text{if } |V(F)| \leq |V(G)|\,, \\ 0 & \text{if } |V(F)| > |V(G)|\,. \end{cases}$$

Theorem. $\overline{\varphi}_\infty(\cdot)$ is complete and $k_{\mathsf{min}}(G, H) = \langle \overline{\varphi}_\infty(G), \overline{\varphi}_\infty(H) \rangle$ is a complete graph kernel.

Computational complexity

Computing $\mathsf{hom}(F, G)$ is NP-hard in general.

If we take the treewidth of pattern F into account the runtime is [Díaz et al., 2002]:

$$\mathcal{O}\left(|V(F)||V(G)|^{\mathsf{tw}(F)+1}\right)$$

Computational complexity

Computing $\hom(F, G)$ is NP-hard in general.

If we take the treewidth of pattern F into account the runtime is [Díaz et al., 2002]:

$$\mathcal{O}\left(|V(F)||V(G)|^{\text{tw}(F)+1}\right)$$

Idea: define distribution $\mathcal{D}$ on $\mathcal{G}_n$ s.t. runtime is polynomial in expectation!

Computational complexity

Computing $\textbf{hom}(F, G)$ is NP-hard in general.

If we take the treewidth of pattern F into account the runtime is [Díaz et al., 2002]:

$$\mathcal{O}\left(|V(F)||V(G)|^{\text{tw}(F)+1}\right)$$

Idea: define distribution $\mathcal{D}$ on $\mathcal{G}_n$ s.t. runtime is polynomial in expectation!

General recipe:

1. pick n as the maximum number of vertices in the training set
2. sample treewidth upper bound k
3. sample a maximal graph F' with treewidth k
4. take a random subgraph F of F'

Computational complexity

Computing $\mathbf{hom}(F, G)$ is NP-hard in general.

If we take the treewidth of pattern F into account the runtime is [Díaz et al., 2002]:

$$\mathcal{O}\left(|V(F)||V(G)|^{\mathbf{tw}(F)+1}\right)$$

Idea: define distribution $\mathcal{D}$ on $\mathcal{G}_n$ s.t. runtime is polynomial in expectation!

General recipe:

1. pick n as the maximum number of vertices in the training set
2. sample treewidth upper bound k
3. sample a maximal graph F' with treewidth k
4. take a random subgraph F of F'

E.g., $k \sim \mathrm{Poisson}(\lambda)$ with $\lambda \leq \frac{1+d\log n}{n}$ guarantees runtime $\mathcal{O}\left(|V(G)|^{d+2}\right)$

Practical embedding

Fix $\ell \in \mathbb{N}$, e.g., $\ell = 30$

Sample $F_1, \ldots, F_\ell$ from $\mathcal{D}$, which guarantees completeness and poly-time in expectation

Practical embedding

Fix $\ell \in \mathbb{N}$, e.g., $\ell = 30$

Sample $F_1, \ldots, F_\ell$ from $\mathcal{D}$, which guarantees completeness and poly-time in expectation

Construct

$$\varphi^\ell(G) = \begin{pmatrix} \mathrm{hom}(F_1, G) \\ \vdots \\ \mathrm{hom}(F_\ell, G) \end{pmatrix}$$

Practical embedding

Fix $\ell \in \mathbb{N}$, e.g., $\ell = 30$

Sample $F_1, \ldots, F_\ell$ from $\mathcal{D}$, which guarantees completeness and poly-time in expectation

Construct

$$\varphi^\ell(G) = \begin{pmatrix} \mathsf{hom}(F_1, G) \\ \vdots \\ \mathsf{hom}(F_\ell, G) \end{pmatrix}$$

Theorem. φ^ℓ is complete in expectation and can be computed in polynomial time in expectation.

Experiments

Deterministic embeddings as baseline [NT and Maehara, ICML 2020]

- GHC-tree(6): all tree patterns up to size 6
- GHC-cycle(8): all cycle patterns up to size 8

Additionally:

- graph neural tangent kernel (GNTK) [Du et al., NeurIPS 2019]
- GIN [Xu et al., ICLR 2019]

Experiments

method	MUTAG	IMDB-BIN	IMDB-MULTI	PAULUS25	CSL
GHC-tree(6)	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40	7.14 ± 0.00	10.00 ± 0.00
GHC-cycle(8)	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67	7.14 ± 0.00	100.00 ± 0.00
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56	7.14 ± 0.00	10.00 ± 0.00
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00	7.14 ± 0.00	10.00 ± 0.00
ours (SVM)	87.94 ± 0.01	70.37 ± 0.01	47.34 ± 0.01	100.00 ± 0.00	37.33 ± 0.1
ours (MLP)	88.55 ± 0.01	70.81 ± 0.01	48.29 ± 0.01	40.524 ± 0.00	13.27± 0.01

Experiments

method	MUTAG	IMDB-BIN	IMDB-MULTI	PAULUS25	CSL
GHC-tree(6)	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40	7.14 ± 0.00	10.00 ± 0.00
GHC-cycle(8)	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67	7.14 ± 0.00	100.00 ± 0.00
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56	7.14 ± 0.00	10.00 ± 0.00
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00	7.14 ± 0.00	10.00 ± 0.00
ours (SVM)	87.94 ± 0.01	70.37 ± 0.01	47.34 ± 0.01	100.00 ± 0.00	37.33 ± 0.1
ours (MLP)	88.55 ± 0.01	70.81 ± 0.01	48.29 ± 0.01	40.524 ± 0.00	13.27± 0.01

Experiments

method	MUTAG	IMDB-BIN	IMDB-MULTI	PAULUS25	CSL
GHC-tree(6)	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40	7.14 ± 0.00	10.00 ± 0.00
GHC-cycle(8)	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67	7.14 ± 0.00	100.00 ± 0.00
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56	7.14 ± 0.00	10.00 ± 0.00
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00	7.14 ± 0.00	10.00 ± 0.00
ours (SVM)	87.94 ± 0.01	70.37 ± 0.01	47.34 ± 0.01	100.00 ± 0.00	37.33 ± 0.1
ours (MLP)	88.55 ± 0.01	70.81 ± 0.01	48.29 ± 0.01	40.524 ± 0.00	13.27± 0.01

Experiments

method	MUTAG	IMDB-BIN	IMDB-MULTI	PAULUS25	CSL
GHC-tree(6)	89.28 ± 8.26	72.10 ± 2.62	48.60 ± 4.40	7.14 ± 0.00	10.00 ± 0.00
GHC-cycle(8)	87.81 ± 7.46	70.93 ± 4.54	47.41 ± 3.67	7.14 ± 0.00	100.00 ± 0.00
GNTK	89.46 ± 7.03	75.61 ± 3.98	51.91 ± 3.56	7.14 ± 0.00	10.00 ± 0.00
GIN	89.40 ± 5.60	70.70 ± 1.10	43.20 ± 2.00	7.14 ± 0.00	10.00 ± 0.00
ours (SVM)	87.94 ± 0.01	70.37 ± 0.01	47.34 ± 0.01	100.00 ± 0.00	37.33 ± 0.1
ours (MLP)	88.55 ± 0.01	70.81 ± 0.01	48.29 ± 0.01	40.524 ± 0.00	13.27± 0.01

Research direction

Choose number of patterns ℓ and distribution $\mathcal{D}$ adaptively:

- stop sampling when expressive enough
- pick $\mathcal{D}$ based on the task or a given dataset
- frequent / interesting patterns

Research direction

Choose number of patterns ℓ and distribution $\mathcal{D}$ adaptively:

- stop sampling when expressive enough
- pick $\mathcal{D}$ based on the task or a given dataset
- frequent / interesting patterns

Going beyond expressiveness: similarity!

- if $G \approx H$ then $\varphi(G) \approx \varphi(H)$
- possible solution: cut distance (captures local and global properties)

Research direction

Choose number of patterns ℓ and distribution $\mathcal{D}$ adaptively:

- stop sampling when expressive enough
- pick $\mathcal{D}$ based on the task or a given dataset
- frequent / interesting patterns

Going beyond expressiveness: similarity!

- if $G \approx H$ then $\varphi(G) \approx \varphi(H)$
- possible solution: cut distance (captures local and global properties)

Randomness for powerful graph embeddings

Pointers

Talk mostly based on

- M.T.*, Pascal Welke*, and Thomas Gärtner [GLFrontiers@NeurIPS 2022]

Further related work

- Martin Grohe. *"word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data."* [PoDS 2022]
- Pascal Kühner. Master Thesis: "*Graph Embeddings Based on Homomorphism Counts.*" [2021]
- Pablo Barceló, et al. "*Graph Neural Networks with Local Graph Parameters.*" [NeurIPS 2021]
- Paul Beaujean et al., *"Graph Homomorphism Features: Why Not Sample?"* [GEM@ECMLPKDD 2021]
- Hoang Nguyen and Takanori Maehara. *"Graph homomorphism convolution."* [ICML 2020]
- Lingfei Wu, et al. *"Scalable Global Alignment Graph Kernel Using Random Features: From Node Embedding to Graph Embedding."* [KDD 2019]
- Till Schulz, et al. *"Mining Tree Patterns with Partially Injective Homomorphisms"* [ECMLPKDD 2018]