Expectation-Complete Graph Representations with Homomorphisms




ICML 2023

Pascal Welke[*], Maximilian Thiessen[*], Fabian Jogl, and Thomas Gärtner

TU Wien
Vienna | Austria
Research Unit Machine Learning

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently

TL;DR

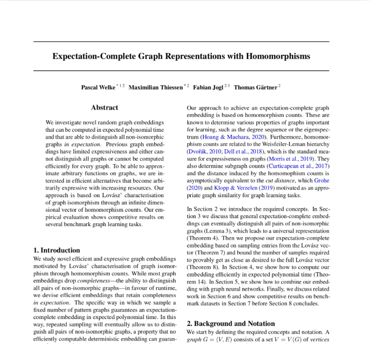
- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets
- We prove lots of nice theorems

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets
- We prove lots of nice theorems



Expectation-Complete Graph Representations with Homomorphisms

Pascal Welke [*1,2] **Maximilian Thiessen** [*2] **Fabian Jogl** [2,3] **Thomas Gärtner** [2]

Abstract

We investigate novel random graph embeddings that can be computed in expected polynomial time and that are able to distinguish all non-isomorphic graphs *in expectation*. Previous graph embeddings have limited expressiveness and either cannot distinguish all graphs or cannot be computed efficiently for every graph. To be able to approximate arbitrary functions on graphs, we are interested in efficient alternatives that become arbitrarily expressive with increasing resources. Our approach is based on Lovász' characterisation of graph isomorphism through an infinite dimensional vector of homomorphism counts. Our empirical evaluation shows competitive results on several benchmark graph learning tasks.

1. Introduction

We study novel efficient and expressive graph embeddings motivated by Lovász' characterisation of graph isomorphism through homomorphism counts. While most graph embeddings drop *completeness*—the ability to distinguish all pairs of non-isomorphic graphs—in favour of runtime, we devise efficient embeddings that retain completeness *in expectation*. The specific way in which we sample a fixed number of pattern graphs guarantees an expectation-complete embedding in expected polynomial time. In this way, repeated sampling will eventually allow us to distinguish all pairs of non-isomorphic graphs, a property that no efficiently computable deterministic embedding can guaran-

Our approach to achieve an expectation-complete graph embedding is based on homomorphism counts. These are known to determine various properties of graphs important for learning, such as the degree sequence or the eigenspectrum (Hoang & Maehara, 2020). Furthermore, homomorphism counts are related to the Weisfeiler-Leman hierarchy (Dvořák, 2010; Dell et al., 2018), which is the standard measure for expressiveness on graphs (Morris et al., 2019). They also determine subgraph counts (Curticapean et al., 2017) and the distance induced by the homomophism counts is asymptotically equivalent to the *cut distance*, which Grohe (2020) and Klopp & Verzelen (2019) motivated as an appropriate graph similarity for graph learning tasks.

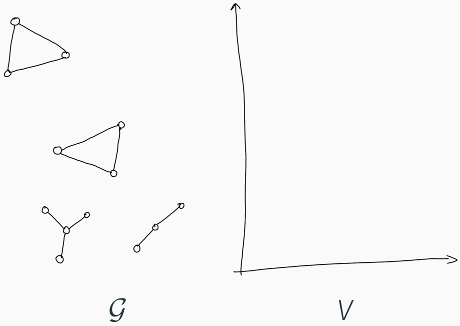
In Section 2 we introduce the required concepts. In Section 3 we discuss that general expectation-complete embeddings can eventually distinguish all pairs of non-isomorphic graphs (Lemma 3), which leads to a universal representation (Theorem 4). Then we propose our expectation-complete embedding based on sampling entries from the Lovász vector (Theorem 7) and bound the number of samples required to provably get as close as desired to the full Lovász vector (Theorem 8). In Section 4, we show how to compute our embedding efficiently in expected polynomial time (Theorem 14). In Section 5, we show how to combine our embedding with graph neural networks. Finally, we discuss related work in Section 6 and show competitive results on benchmark datasets in Section 7 before Section 8 concludes.

2. Background and Notation

We start by defining the required concepts and notation. A *graph* $G = (V, E)$ consists of a set $V = V(G)$ of *vertices*

Complete graph embeddings

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

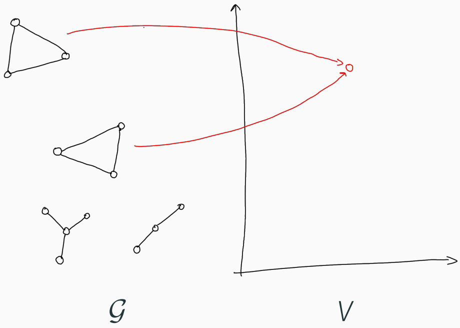


Complete graph embeddings

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

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

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



Complete graph embeddings

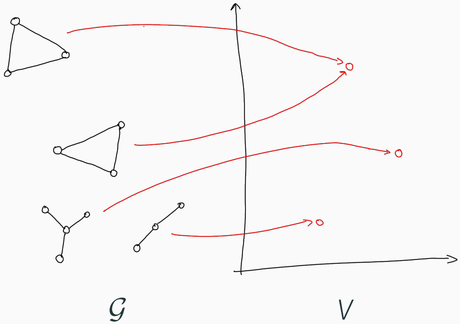
Let $\mathcal{G}_n$ be the set of all graphs up to n vertices, V be a vector space (e.g., $\mathbb{R}^d$)

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

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

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

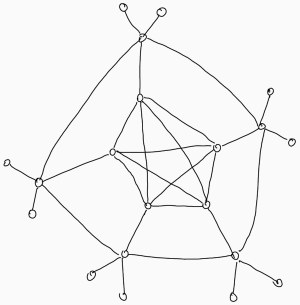
What if we keep completeness …

… but just in expectation

… in polynomial time

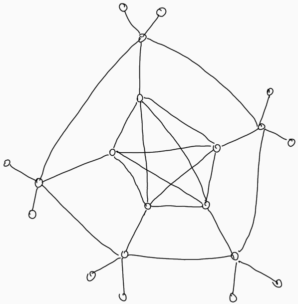
An intractable complete graph embedding

G

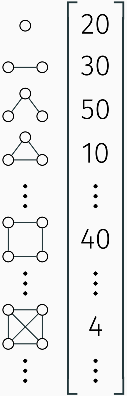


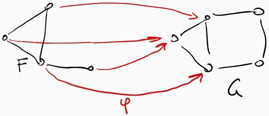
An intractable complete graph embedding

G



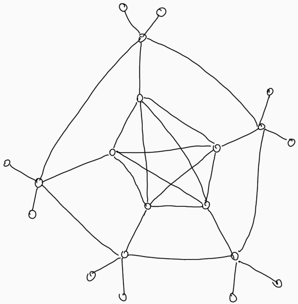
$\varphi_n(G)$



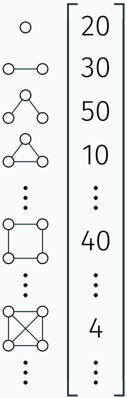


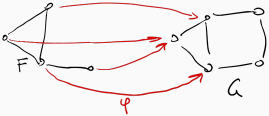
An intractable complete graph embedding

G



$\varphi_n(G)$



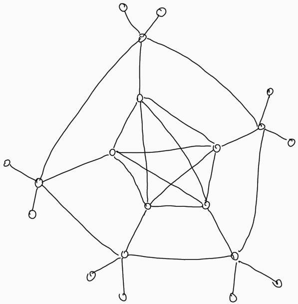


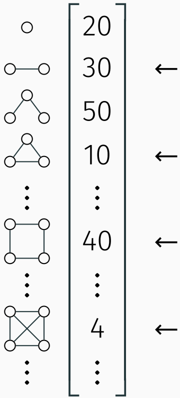
Theorem [Lovász 1967].
Two graphs G and H are isomorphic iff
$\varphi_n(G) = \varphi_n(H)$

An expectation-complete graph embedding

$\varphi_n(G)$

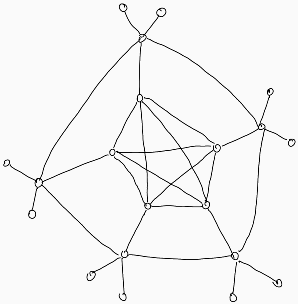
G





An expectation-complete graph embedding

G



$\varphi_n(G)$		$\varphi_{\mathcal{F}}(G)$
20		0
30	←	30
50		0
10	←	10
⋮		0
40	←	40
⋮		0
4	←	4
⋮		0

Efficient and expectation-complete GNNs

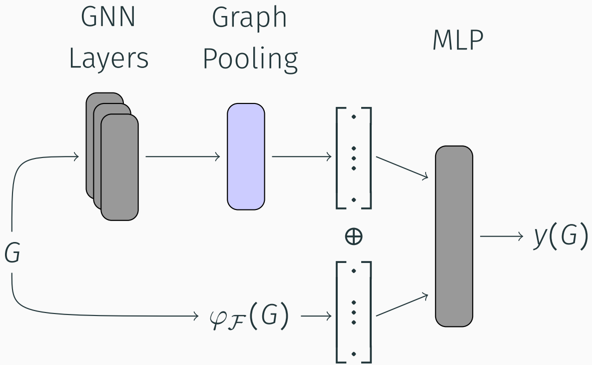
- Homomorphism counting is fixed parameter tractable

Efficient and expectation-complete GNNs

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns

Efficient and expectation-complete GNNs

- Homomorphism counting is fixed parameter tractable
- We design a distribution $\mathcal{D}$ that weights down expensive patterns
- And show how to make any message passing GNN expectation-complete



Expectation-Complete Graph Representations with Homomorphisms




ICML 2023

Pascal Welke*, Maximilian Thiessen*, Fabian Jogl, and Thomas Gärtner

TU Wien
Vienna | Austria
Research Unit Machine Learning

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently

TL;DR

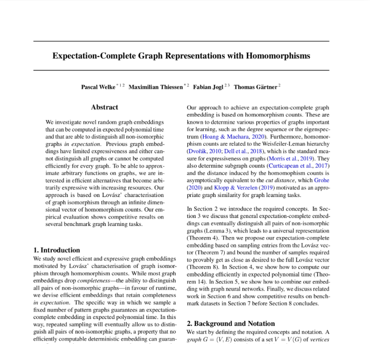
- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets
- We prove lots of nice theorems

TL;DR

- We introduce the concept of expectation-complete graph embeddings (and kernels)
- We present one example of a expectation-complete graph embedding with homomorphisms
- We show how to compute it efficiently
- We train expectation-complete GNNs and show statistically significant performance improvements on ten molecular benchmark datasets
- We prove lots of nice theorems



Expectation-Complete Graph Representations with Homomorphisms

Pascal Welke [*1,2] Maximilian Thiessen [*2] Fabian Jogl [2,3] Thomas Gärtner [2]

Abstract

We investigate novel random graph embeddings that can be computed in expected polynomial time and that are able to distinguish all non-isomorphic graphs *in expectation*. Previous graph embeddings have limited expressiveness and either cannot distinguish all graphs or cannot be computed efficiently for every graph. To be able to approximate arbitrary functions on graphs, we are interested in efficient alternatives that become arbitrarily expressive with increasing resources. Our approach is based on Lovász' characterisation of graph isomorphism through an infinite dimensional vector of homomorphism counts. Our empirical evaluation shows competitive results on several benchmark graph learning tasks.

1. Introduction

We study novel efficient and expressive graph embeddings motivated by Lovász' characterisation of graph isomorphism through homomorphism counts. While most graph embeddings drop *completeness*—the ability to distinguish all pairs of non-isomorphic graphs—in favour of runtime, we devise efficient embeddings that retain completeness *in expectation*. The specific way in which we sample a fixed number of pattern graphs guarantees an expectation-complete embedding in expected polynomial time. In this way, repeated sampling will eventually allow us to distinguish all pairs of non-isomorphic graphs, a property that no efficiently computable deterministic embedding can guaran-

Our approach to achieve an expectation-complete graph embedding is based on homomorphism counts. These are known to determine various properties of graphs important for learning, such as the degree sequence or the eigenspectrum (Hoang & Maehara, 2020). Furthermore, homomorphism counts are related to the Weisfeiler-Leman hierarchy (Dvořák, 2010; Dell et al., 2018), which is the standard measure for expressiveness on graphs (Morris et al., 2019). They also determine subgraph counts (Curticapean et al., 2017) and the distance induced by the homomophism counts is asymptotically equivalent to the *cut distance*, which Grohe (2020) and Klopp & Verzelen (2019) motivated as an appropriate graph similarity for graph learning tasks.

In Section 2 we introduce the required concepts. In Section 3 we discuss that general expectation-complete embeddings can eventually distinguish all pairs of non-isomorphic graphs (Lemma 3), which leads to a universal representation (Theorem 4). Then we propose our expectation-complete embedding based on sampling entries from the Lovász vector (Theorem 7) and bound the number of samples required to provably get as close as desired to the full Lovász vector (Theorem 8). In Section 4, we show how to compute our embedding efficiently in expected polynomial time (Theorem 14). In Section 5, we show how to combine our embedding with graph neural networks. Finally, we discuss related work in Section 6 and show competitive results on benchmark datasets in Section 7 before Section 8 concludes.

2. Background and Notation

We start by defining the required concepts and notation. A *graph* $G = (V, E)$ consists of a set $V = V(G)$ of *vertices*