Anonymous Walk Embedding

ICML'18, Sergey Ivanov & Evgeny Burnaev

Hoang NT

Murata Laboratory
Tokyo Tech

2019/01/10

Overview

Graphs Distance

Why do we need graph distance?

Given a network modeled as a graph, there are many questions to ask. For example:

Why do we need graph distance?

Given a network modeled as a graph, there are many questions to ask. For example:

- Is the graph connected?
- How to maximize vertices independent set?
- Where is the best graph cut?
- How to describe the graph mathematically?
- How to do all above if the graph is "large"?

Why do we need graph distance?

Given a network modeled as a graph, there are many questions to ask. For example:

- Is the graph connected?
- How to maximize vertices independent set?
- Where is the best graph cut?
- How to describe the graph mathematically?
- How to do all above if the graph is "large"?

How to answer (some of) the questions above?

Defining a scalable graph distance is a key challenge that would give us an advantage in analysis.

Example problems

Graph classification

Using measure distance between graphs, we can have better machinery to understand new graph designs.

Example problems

Graph classification

Using measure distance between graphs, we can have better machinery to understand new graph designs.

Graph modeling

Combining random graph modelings with generative neural networks, we can generate new designs for materials, drugs, and complex systems.

Molecules and nano-structures are often modeled as graphs:

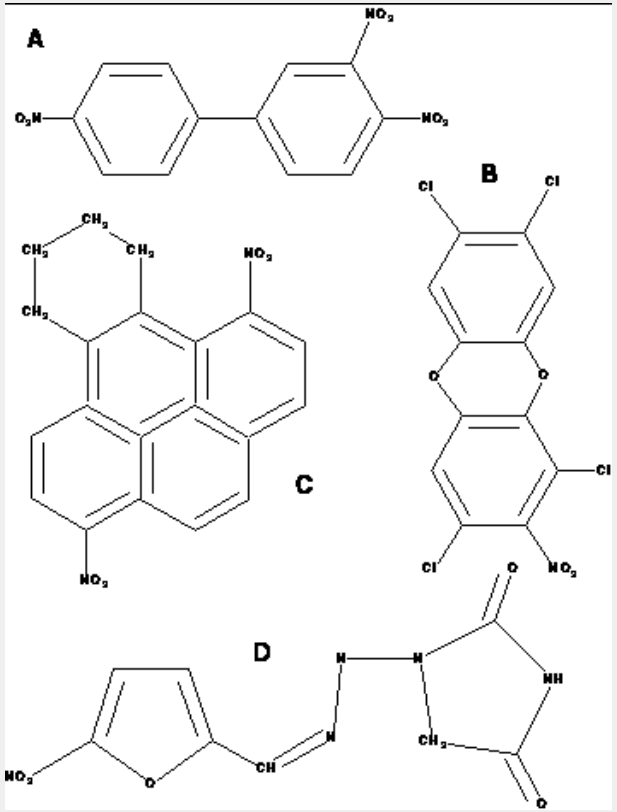

Figure: How do we compare these molecules?

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$
   - Frieze-Kannan cut-norm

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$
   - Frieze-Kannan cut-norm
   - etc.
2. Other (approximate) distances:

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$
   - Frieze-Kannan cut-norm
   - etc.
2. Other (approximate) distances:
   - Spectral gap and graph spectrum

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$
   - Frieze-Kannan cut-norm
   - etc.
2. Other (approximate) distances:
   - Spectral gap and graph spectrum
   - Graph features-based distance (L1,2 of features)

Types of graph distance

There are many definitions for graph distances depending on the specific problem.

1. Exact topological distances:
   - Edit distance (Hamming distance)
   - Homomophism count for some smaller graph $\mathcal{F}$
   - Frieze-Kannan cut-norm
   - etc.
2. Other (approximate) distances:
   - Spectral gap and graph spectrum
   - Graph features-based distance (L1,2 of features)
   - etc.

This paper

In this paper: "Anonymous Walk Embedding", the authors addressed the graph distance design by an modified version of approximate homomomophism counting.

This paper

In this paper: "Anonymous Walk Embedding", the authors addressed the graph distance design by an modified version of approximate homomomophism counting.

Main experiment

Input: A set of graphs, each associated with a label.
Output: A set of (task-agnostic) vector representations for each graph in the dataset. These representations later used as feature vectors for the graph classification task.

Embedding Methods

Anonymous Walk definition

Definition 1: Anonymous Walk

An Anonymous Walk is a random walk where vectices are replaced by their *appearance orders in the walk*.
(Def. 2 in the paper) If $\omega = (v_1, v_2, ..., v_k)$ is a random walk, then its corresponding Anonymous Walk is a sequence of integers $a = (f(v_1), f(v_2), ..., f(v_k))$ where the integer $f(v_i) = \min pos(v_i, w)$.

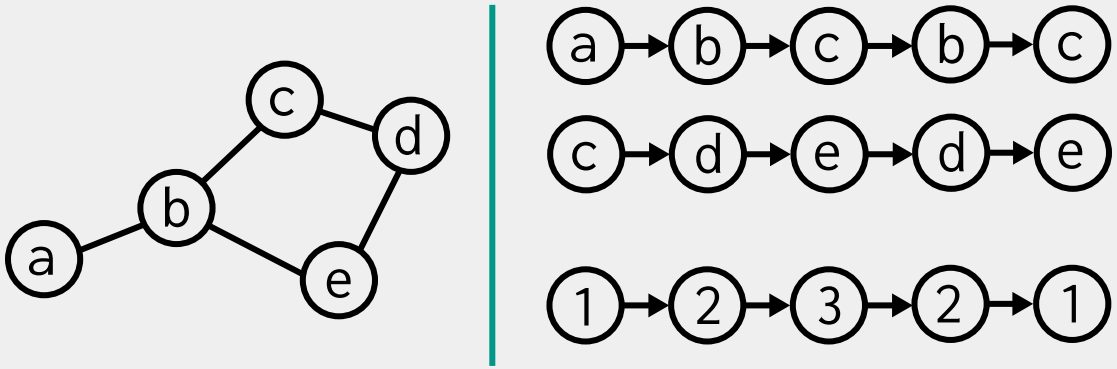

Figure: Anonymous Walk Example.

Assumtions and intuition

This paper was built on the results (Theorem 1 and 1') of [2]:

Theorem 1 (Micali et al., 2016)

Let n be a number of vectices in $B(v, r)$ and m is the number of edges. One can reconstruct $B(v, r)$ in time $O(n^2)$ with $O(n^2)$ oracale access to $(\mathcal{D}_1, ..., \mathcal{D}_l)$, where $l = O(m)$. Moreover, the reconstruction algorithm only makes membership queries to $\text{supp}(\mathcal{D}_i)$ for $i \in [l]$.

The author of AWE assumes that if one could reconstruct the topological ball $B(v, r)$ around vertex v by the distribution of Anonymous Walks, one could approximately say the samething about the whole graph. (?)

Method 1: Features Based

Definition 2: AW-FB

(Def. 3 in the paper) Feature Based Anonymous Walk: Let $\mathcal{A}_l = (a_1, a_2, ..., a_\eta)$ be the set of all possible anonymous walk of length l. *Anonymous Walk Embedding* of a graph G is the vector f_G of size η, whose i-th component corresponds to a probability $p(a_i)$, of having anonymous walk a_i in the graph G:

$$f_G = (p(a_1), p(a_2), ..., p(a_\eta))$$

The probability $p(a_i)$ is empirically estimated using random sampling. The confident intervals for m samples is given by a similar work using graphlet kernel for graph comparisons [3].

Method 2: Data Driven

Definition 3: AW-DD

Data Driven Anonymous Walk: Let an Anonymous Walk starting from a vertex u be analogous to a "word" and the whole graph be analoguous to a "document". The authros now use document embedding in the NLP setting to obtain the graph vector by maximizing the average log-probability:

$$\frac{1}{T}\sum_{t=\Delta}^{T-\Delta}\log p(w_t|\Delta(w_t),d)$$

In here, a neighboorhood Δw_t is the set of Anonymous Walks rooted at the same vertex.

Method 2: Data Driven

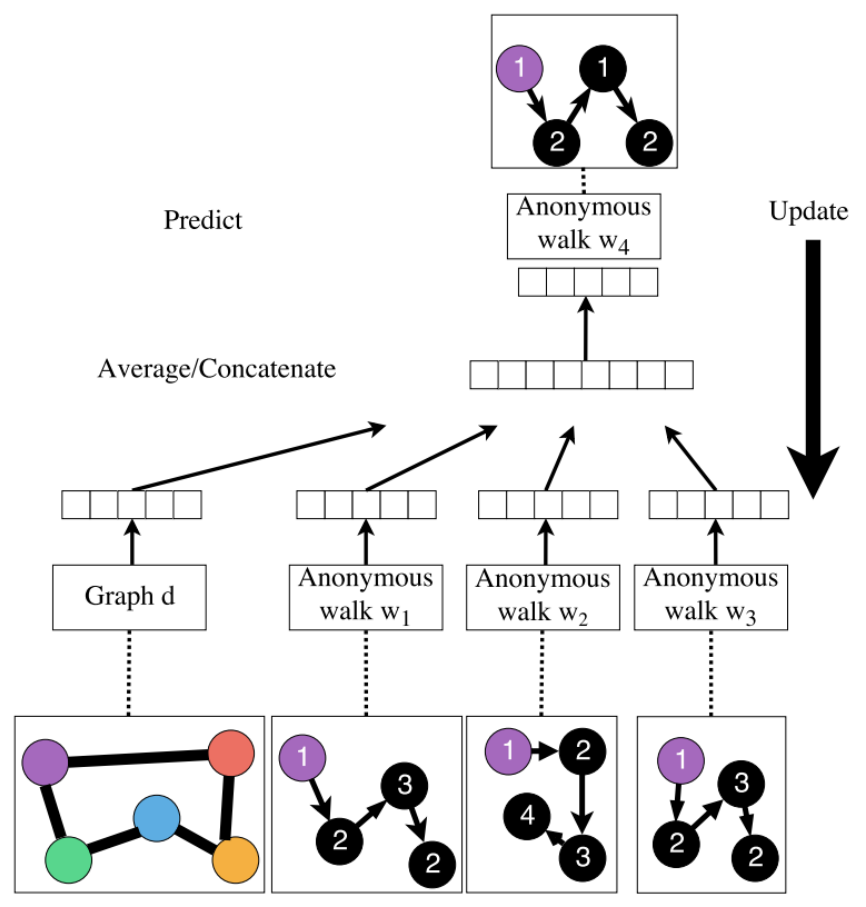

Figure: Data Driven Framework. [1]

Method 2: Data Driven

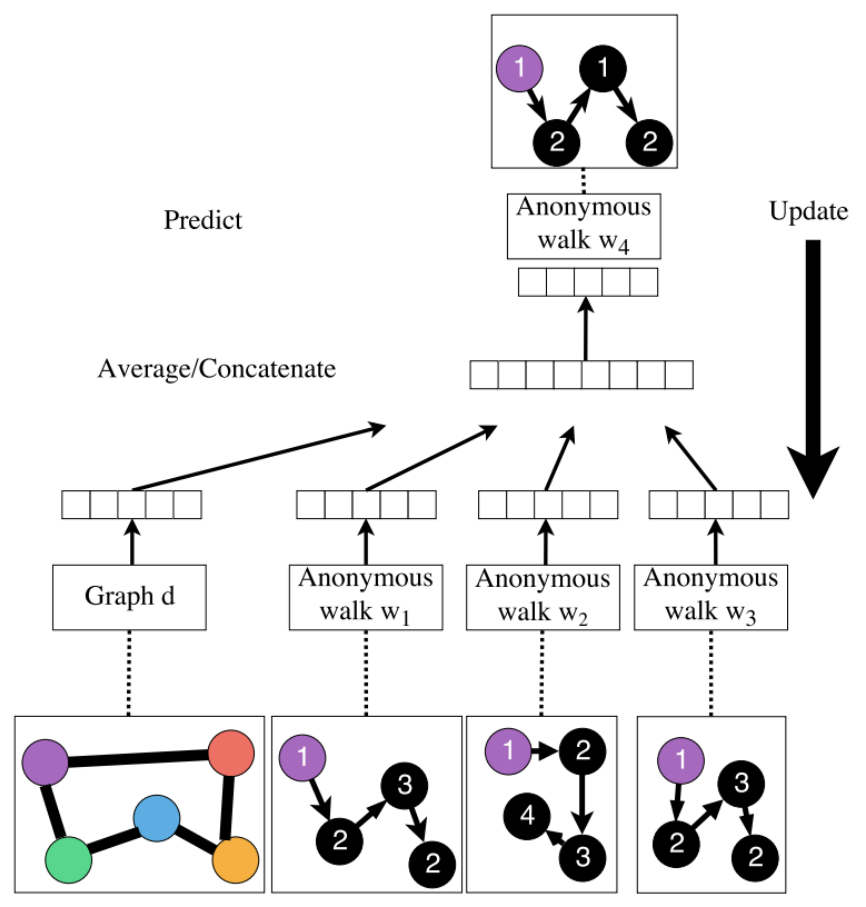

Figure: Data Driven Framework. [1]

Step 1: Create data

Starting at each vertex, perform T anonymous walks of length l.

Method 2: Data Driven

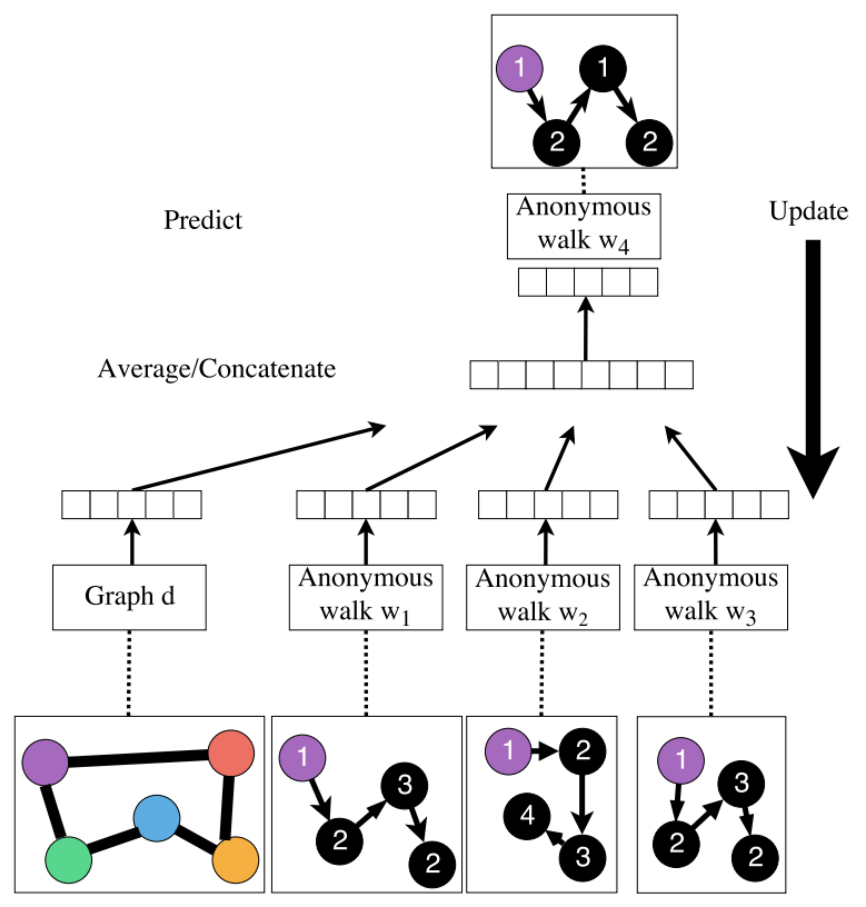

Figure: Data Driven Framework. [1]

Step 1: Create data

Starting at each vertex, perform T anonymous walks of length l.

Step 2: Doc2Vec

Training a doc2vec model with anonymous walks as words and the graph as the document. Local context is the set T walks of each vertex.

Experiments & Results

Datasets

Datasets	Source	#Graphs	Classes (max)	N/E Avg.
COLLAB	Social	5000	3 (2600)	74.49 / 4914.99
IMDB-B	Social	1000	2 (500)	19.77 / 193.06
IMDB-M	Social	1500	3 (500)	13 / 131.87
RE-B	Social	2000	2 (1000)	429.61 / 995.50
RE-M5K	Social	4999	5 (1000)	508.5 / 1189.74
RE-M12K	Social	12000	11 (2592)	391.4 / 913.78
Enzymes	Bio	600	6 (100)	32.6 / 124.3
DD	Bio	1178	2 (691)	284.31 / 715.65
Mutag	Bio	188	2 (125)	17.93 / 19.79

Table: Datasets used in the paper.

10-fold cross validation is used for each dataset and each parameter setting of their methods.

Competitors

There are two groups of other methods:

Competitors

There are two groups of other methods:

1. Data Driven:

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.
   - **DGK**: Deep Graph Kernel (Yanardag et al., 2015) - Learn a positive semidefinite matrix to weight the relationship between substructures.

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.
   - **DGK**: Deep Graph Kernel (Yanardag et al., 2015) - Learn a positive semidefinite matrix to weight the relationship between substructures.
2. Features Based:

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.
   - **DGK**: Deep Graph Kernel (Yanardag et al., 2015) - Learn a positive semidefinite matrix to weight the relationship between substructures.
2. Features Based:
   - **WL**: Weisfeiler-Lehman Kernel (Shervashidze et al., 2011) - Construct graph features from the subtree patterns of a graph.

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.
   - **DGK**: Deep Graph Kernel (Yanardag et al., 2015) - Learn a positive semidefinite matrix to weight the relationship between substructures.
2. Features Based:
   - **WL**: Weisfeiler-Lehman Kernel (Shervashidze et al., 2011) - Construct graph features from the subtree patterns of a graph.
   - **GK**: Graphlet Kernel (Shervashidze et al., 2009) - Construct graph features from graphlets (m̃otifs).

Competitors

There are two groups of other methods:

1. Data Driven:
   - **PSCN**: Patchy-San (Niepert et al., 2016) - Trains a convolutional neural network from generated graph neighboorhood.
   - **DGK**: Deep Graph Kernel (Yanardag et al., 2015) - Learn a positive semidefinite matrix to weight the relationship between substructures.
2. Features Based:
   - **WL**: Weisfeiler-Lehman Kernel (Shervashidze et al., 2011) - Construct graph features from the subtree patterns of a graph.
   - **GK**: Graphlet Kernel (Shervashidze et al., 2009) - Construct graph features from graphlets (m̃otifs).
   - **ER**: Exponential Random Walk Kernel (Gartner et al., 2003).
   - **kR**: k-step Random Walk Kernel (Sugiyama et al., 2015).

Results: Social Networks

	Algorithm	IMDB-M	IMDB-B	COLLAB	RE-B	RE-M5K	RE-M12K
DD	AWE (DD)	**51.54 ± 3.61**	**74.45 ± 5.83**	**73.93 ± 1.94**	**87.89 ± 2.53**	**50.46 ± 1.91**	**39.20 ± 2.09**
	PSCN	45.23 ± 2.84	71.00 ± 2.29	72.60 ± 2.15	**86.30 ± 1.58**	49.10 ± 0.70	41.32 ± 0.32
	DGK	44.55 ± 0.52	66.96 ± 0.56	73.09 ± 0.25	78.04 ± 0.39	41.27 ± 0.18	32.22 ± 0.10
FB	AWE (FB)	**51.58 ± 4.66**	73.13 ± 3.28	70.99 ± 1.49	82.97 ± 2.86	**54.74 ± 2.93**	**41.51 ± 1.98**
	WL	49.33 ± 4.75	**73.4 ± 4.63**	**79.02 ± 1.77**	81.1 ± 1.9	49.44 ± 2.36	38.18 ± 1.3
	GK	43.89 ± 0.38	65.87 ± 0.98	72.84 ± 0.28	65.87 ± 0.98	41.01 ± 0.17	31.82 ± 0.08
	ER	OOM	64.00 ± 4.93	OOM	OOM	OOM	OOM
	kR	34.47 ± 2.42	45.8 ± 3.45	OOM	OOM	OOM	OOM

Figure: Classification accuracy for social networks (Table 2 [1])

Results: Bio Datasets

Algorithm	Enzymes	DD	Mutag
AWE	35.77 ± 5.93	71.51 ± 4.02	87.87 ± 9.76
PSCN	–	77.12 ± 2.41	92.63 ± 4.21
DGK	27.08 ± 0.79	–	82.66 ± 1.45
WL	53.15 ± 1.14	77.95 ± 0.70	80.72 ± 3.00
GK	32.70 ± 1.20	78.45 ± 0.26	81.58 ± 2.11
ER	14.97 ± 0.28	OOM	71.89 ± 0.66
kR	30.01 ± 1.01	OOM	80.05 ± 1.64

Figure: Classification accuracy for Bio datasets (Table 4 [1])

Conclusion

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.
- Open to many different ML tasks.

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.
- Open to many different ML tasks.
- The algorithm is highly customizable to learn node and subgraph representations.

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.
- Open to many different ML tasks.
- The algorithm is highly customizable to learn node and subgraph representations.

Disadvantage:

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.
- Open to many different ML tasks.
- The algorithm is highly customizable to learn node and subgraph representations.

Disadvantage:

- Doc2Vec algorithm takes very long time to run (2.7hrs for 100 epochs of Mutag on my machine).

Conclusion

The authors used **anonymous walks** to create embedding vector to a given graph. There are two ways to create such vector:

1. Count the number of unique anonymous walks.
2. Use `doc2vec` algorithm to learn the embedding vector.

Advantage:

- Scalable whole graph embedding algorithm.
- Open to many different ML tasks.
- The algorithm is highly customizable to learn node and subgraph representations.

Disadvantage:

- Doc2Vec algorithm takes very long time to run (2.7hrs for 100 epochs of Mutag on my machine).
- Theorem 1 is trivial. There is no guarantee for the assumption that similar graphs exhibit similar distribution of anonymous walk.

Thanks for listening!

References

Sergey Ivanov and Evgeny Burnaev.
Anonymous walk embeddings.
In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2191–2200, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Silvio Micali and Zeyuan Allen Zhu.
Reconstructing markov processes from independent and anonymous experiments.
Discrete Applied Mathematics, 200:108–122, 2016.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt.
Efficient graphlet kernels for large graph comparison.
In *Artificial Intelligence and Statistics*, pages 488–495, 2009.