Anonymous Walk Embeddings

Sergey Ivanov [1 2] **Evgeny Burnaev** [1]

Abstract

The task of representing entire graphs has seen a surge of prominent results, mainly due to learning convolutional neural networks (CNNs) on graph-structured data. While CNNs demonstrate state-of-the-art performance in graph classification task, such methods are supervised and therefore steer away from the original problem of network representation in task-agnostic manner. Here, we coherently propose an approach for embedding entire graphs and show that our feature representations with SVM classifier increase classification accuracy of CNN algorithms and traditional graph kernels. For this we describe a recently discovered graph object, *anonymous walk*, on which we design task-independent algorithms for learning graph representations in explicit and distributed way. Overall, our work represents a new scalable unsupervised learning of state-of-the-art representations of entire graphs.

1. Introduction

A wide range of real world applications deal with network analysis and classification tasks. An ease of representing data with graphs makes them very valuable asset in any data mining toolbox; however, the complexity of working with graphs led researchers to seek for new ways of representing and analyzing graphs, of which network embeddings have become broadly popular due to their success in several machine learning areas such as graph classification (Cai et al., 2017), visualization (Cao et al., 2016), and pattern recognition (Monti et al., 2017).

Essentially, network embeddings are vector representations of graphs that capture local and global traits and, as a consequence, are more suitable for standard machine learning techniques such as SVM that works on numerical vectors rather than graph structures. Ideally, a practitioner would like to have a *polynomial*-time algorithm that can convert different graphs into different feature vectors. However, such algorithm would be capable of deciding whether two graphs are isomorphic (Gärtner et al., 2003), for which currently only quasipolynomial-time algorithm exists (Babai, 2016). Hence, there are fundamental challenges in the design of polynomial-time algorithm for network-to-vector conversion. Instead, a lot of research was devoted to the question of designing network embedding models that are computationally efficient *and* preserve similarity between graphs.

Broadly speaking, network embeddings come from one of the two buckets, either based on engineered graph features or driven by training on graph data. Feature-based methods traditionally appeared in graph kernel setting (Vishwanathan et al., 2010), where each graph is decomposed into discrete components, distribution of which is used as a vector representation of a graph (Haussler, 1999). Importantly, general concept of feature-based methods implies ad-hoc knowledge about the data at hand. For example, Random Walk kernel (Vishwanathan et al., 2010) assumes that graph realization originates from the types of random walks a graph has, whereas for Weisfeiler-Lehman (WL) kernel (Shervashidze et al., 2011) the insight is in subtree patterns of a graph. For high-dimensional graph embeddings feature-based methods produce sparse solution as only few substructures are common across graphs. This is known as *diagonal dominance* (Yanardag & Vishwanathan, 2015), a situation when a graph representation is only similar to itself, but not to any other graph.

On the other hand, data-driven approach learns network embeddings by optimizing some form of objective function defined on graph data. Deep Graph Kernels (DGK) (Yanardag & Vishwanathan, 2015), for example, learns a positive semidefinite matrix that weights the relationship between graph substructures, while Patchy-San (PSCN) (Niepert et al., 2016) constructs locally connected neighborhoods for training a convolutional neural network on. Data-driven approach implies learning *distributed* graph representations that have demonstrated promising classification results (Niepert et al., 2016; Tixier et al., 2017).

Our approach. We propose to use a natural graph object

[1] Skolkovo Institute of Science and Technology, Moscow, Russia [2] Criteo Research, Paris, France. Correspondence to: Sergey Ivanov <sergei.ivanov@skolkovotech.ru>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

named *anonymous walk* as a base for learning feature-based and data-driven network embeddings. Recent discovery (Micali & Allen Zhu, 2016) has shown that anonymous walks provide characteristic graph traits and are capable to reconstruct network proximity of a node *exactly*. In particular, distribution of anonymous walks starting at node u is sufficient for reconstruction of a subgraph induced by all vertices within a fixed distance from u; and such distribution uniquely determines underlying Markov processes from u, i.e. no two different subgraphs exist having the same distribution of anonymous walks. This implies that two graphs with similar distributions of anonymous walks should be topologically similar. We therefore define feature-based network embeddings on distribution of anonymous walks and show an efficient sampling approach that approximates distributions for large networks.

To overcome sparsity of feature-based methods, we design a data-driven approach that learns distributed representations on the generated corpus of anonymous walks via backpropagation, in the same vein as neural models in NLP (Le & Mikolov, 2014; Bengio et al., 2003). Considering anonymous walks for the same source node as co-occurring words in the sentence and graph as a collection of such sentences, the hope is that by predicting a target word in a given context of words and a document, the proposed algorithm learns semantic meaning of words and a document.

To the best of our knowledge, we are the first to introduce anonymous walks in the context of learning network representations and we highlight the following contributions:

- Based on the notion of anonymous walk, we propose feature-based network embeddings, for which we describe an efficient sampling procedure to alleviate time complexity of exact computation.
- By maximizing the likelihood of preserving network proximity of anonymous walks, we propose a scalable algorithm to learn data-driven network embeddings.
- On widely-used real datasets, we demonstrate that our network embeddings achieve state-of-the-art performance in comparison with other graph kernels and neural networks in graph classification task.

2. Anonymous Walks

Random walks are the sequences of nodes, where each new node is selected independently from the set of neighbors of the last node in the sequence. Normally states in a random walk correspond to a label or a global name of a node; however, for reasons described below such states could be unavailable. Yet, recently it has been shown that anonymized version of a random walk can provide a flexible way to reconstruct a network even when global names are absent (Micali & Allen Zhu, 2016). We next define a notion of anonymous walk.

→ AW

Definition 1. Let $s = (u_1, u_2, \dots, u_k)$ be an ordered list of elements $u_i \in V$. We define the positional function pos: $(s, u_i) \mapsto q$ such that for any ordered list $s = (u_1, u_2, \dots, u_k)$ and an element $u_i \in V$ it returns a list $q = (p_1, p_2, \dots, p_l)$ of all positions $p_j \in \mathbb{N}$ of u_i occurrences in a list s.

Natural no. → list

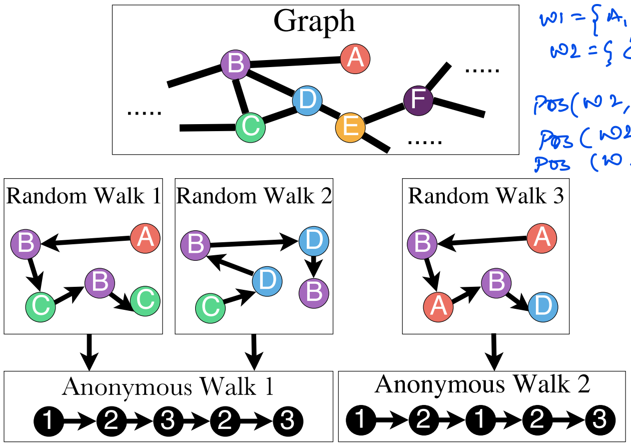
For example, if $s = (a, b, c, b, c)$, then $pos(s, a) = (1)$ as element a appears only on the first position and $pos(s, b) = (2, 4)$.

Definition 2 (Anonymous Walk)**.** If $w = (v_1, v_2, \dots, v_k)$ is a random walk, then its corresponding *anonymous walk* is the sequence of integers $a = (f(v_1), f(v_2), \dots, f(v_k))$, where integer $f(v_i) = \min\limits_{p_j \in pos(w, v_i)} pos(w, v_i)$.

$a = (1, 2, 3, 2, 3)$

We denote mapping of a random walk w to anonymous walk a by $w \mapsto a$.

Pos(w1, A) = {1}
Pos(w1, B) = {2, 4}
Pos(w1, C) = {3, 5}



1 2 3 4 5
w1 = {A, B, C, B, C}
1 2 3 4 5
w2 = {C, D, B, D, B}

Pos(w2, C) = {1}
Pos(w2, D) = {2, 4}
Pos (w2, B) = {3, 5}

Figure 1. An example demonstrating the concept of anonymous walk. Two different random walks 1 and 2 of the graph correspond to the *same* anonymous walk 1. A random walk 3 corresponds to *another* anonymous walk 2.

For instance, in the graph of Fig. 1 a random walk $a \to b \to c \to b \to c$ matches anonymous walk $1 \to 2 \to 3 \to 2 \to 3$. Likewise, another random walk $c \to d \to b \to d \to b$ also corresponds to anonymous walk $1 \to 2 \to 3 \to 2 \to 3$. Conversely, another random walk $a \to b \to a \to b \to d$ corresponds to a different anonymous walk $1 \to 2 \to 1 \to 2 \to 3$.

Intuitively, states in anonymous walk correspond to the first position of the node in a random walk and their total number equals to the number of distinct nodes in a random walk. Particular name of the state does not matter (so, for example, anonymous walk $1 \to 2 \to 3$ would be the same as anonymous walk $3 \to 1 \to 2$); however, by agreement, anonymous walks start from 1 and continue to name new states by incrementing the current maximum state in an anonymous walk.

Rationale. From the perspective of a single node, in the position of an observer, global topology of the network may be hidden deliberately (e.g. social networks often restrict outsiders to examine your friendships) or otherwise (e.g. newly created links in the world wide web may be yet unknown to the search engine). Nevertheless, an observer can, on his own, experiment with the network by starting a random walk from itself, passing the process to its neighbors and recording the observed states in a random walk. As global names of the nodes are not available to an observer, one way to record the states *anonymously* is by describing them by the first occurrence of a node in a random walk. Not only are such records succinct, but it is common to have privacy constraints (Abraham, 2012) that would not allow to record a full description of nodes.

Somewhat remarkably, (Micali & Allen Zhu, 2016) show that for a single node u in a graph G, a known distribution $\mathfrak{D}_l$ over anonymous walks of length l is sufficient to reconstruct topology of the ball $B(u, r)$ with the center at u and radius r, i.e. the subgraph of graph G induced by all vertices distanced at most r hops from u. For the task of learning embeddings, the topology of network is available and thus distribution of anonymous walks $\mathfrak{D}_l$ can be computed precisely. As no two different subgraphs can have the same distribution $\mathfrak{D}_l$, it is useful to generalize distribution of anonymous walks from a single node to the whole network and use it as a feature representation of a graph. This idea paves the way to our feature-based network embeddings.

3. Algorithms

We start from discussion of leveraging anonymous walks for learning network embeddings in a feature-based manner. Inspired by empirical results we train an objective function on local neighborhoods of anonymous walks, which further improves results of classification.

3.1. AWE: Feature-Based model

By definition, a weighted directed graph is a tuple $G = (V, E, \Omega)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices, $E \subseteq V \times V$ is a set of edges, and $\Omega \subset \mathbb{R}$ is a set of edge weights. Given graph G we construct a *random walk graph* $R = (V, E, P)$ such that every edge $e = (u, v)$ has a weight p_e equals to $\omega_e / \sum\limits_{v \in N_{out}(u)} \omega_{(u,v)}$, where $N_{out}(u)$ is the set of out-neighbors of u and $\omega_e \in \Omega$. A random walk w with length l on graph R is a sequence of nodes $u_1, u_2, \ldots, u_{l+1}$, where $u_i \in V$, such that a pair (u_i, u_{i+1}) is selected with a probability $p_{(u_i, u_{i+1})}$ in a random walk graph R. A probability $p(w)$ of having a random walk w is the total probability of choosing the edges in a random walk, i.e. $p(w) = \prod\limits_{e \in w} p_e$.

According to the Definition 1, anonymous walk is a random walk, where each state is recorded by its first occurrence index in the random walk. The number of all possible anonymous walks of length l in an arbitrary graph grows exponentially with l (Figure 2). Consider an initial node u and a set of all different random walks W_l^u that start from u and have length l. These random walks correspond to a set of η different anonymous walks $\mathcal{A}_l^u = (a_1^u, a_2^u, \ldots, a_\eta^u)$. A probability of seeing anonymous walk a_i^u of length l for a node u is $p(a_i^u) = \sum\limits_{\substack{w \in W_l^u \\ w \mapsto a_i}} p(w)$. Aggregating probabilities across all vertices in a graph and normalizing them by the total number of nodes N, we get the probability of choosing anonymous walk a_i in graph G:

$$p(a_i) = \frac{1}{N} \sum_{u \in G} p(a_i^u) = \frac{1}{N} \sum_{u \in G} \sum_{\substack{w \in W_l^u \\ w \mapsto a_i}} p(w).$$

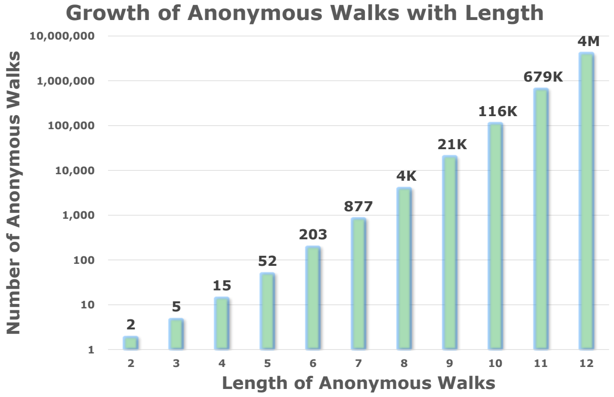


Figure 2. Y-axis is in log scale. The number of different anonymous walks increases exponentially with length of walks l.

We are now ready to define network embeddings that we name feature-based anonymous walk embeddings (AWE).

Definition 3 (feature-based AWE)**.** Let $\mathcal{A}_l = (a_1, a_2, \ldots, a_\eta)$ be the set of all possible anonymous walks 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 a graph G:

$$f_G = (p(a_1), p(a_2), \ldots, p(a_\eta)). \tag{1}$$

Direct computation of AWE relies on the enumeration of all different random walks in graph G, which is shown below to grow exponentially with the number of steps l.

Theorem 1. The running time of Anonymous Walk Embeddings (eq. 1) is $\mathcal{O}(nl(d_{in}^{max}(v) \cdot d_{out}^{max}(v))^{l/2})$, where $d_{in/out}^{max}$ is the maximum in/out degree in graph G with n vertices.

Proof. Let k_l be the number of random walks of length l in a directed graph. According to (Tubig, 2012) k_l can be bounded by the powers of in- and out-degrees of nodes in G:

$$k_l^2 \leq (\sum_{v \in G} d_{in}^l(v))(\sum_{v \in G} d_{out}^l(v)).$$

Hence, the number of random walks in a graph is at most $n(d_{in}^{max}(v) \cdot d_{out}^{max}(v))^{l/2}$, where $d_{in/out}^{max}$ is the maximum in/out degree. As it requires $\mathcal{O}(l)$ operations to map one random walk of length l to anonymous walk, the theorem follows. □

Sampling. As complete counting of all anonymous walks in a large graph may be infeasible, we describe a sampling approach to approximate the true distribution. In this fashion, we draw independently a set of m random walks and calculate its corresponding empirical distribution of anonymous walks. To guarantee that empirical and actual distributions are close with a given confidence, we set the number m of random walks sufficiently large.

More formally, let $\mathcal{A}_l = (a_1, a_2, \dots, a_\eta)$ be the set of all possible anonymous walks of length l. For two discrete probability distributions P and Q on set $\mathcal{A}_l$, define L_1 distance as:

$$\|P - Q\|_1 = \sum_{a_i \in \mathcal{A}} |P(a_i) - Q(a_i)|$$

For a graph G let $\mathfrak{D}_l$ be the actual distribution of anonymous walks $\mathcal{A}_l$ of length l and let $X^m = (X_1, X_2, \dots, X_m)$ be i.i.d. random variables drawn from $\mathfrak{D}_l$. The empirical distribution $\mathfrak{D}^m$ of the original distribution $\mathfrak{D}_l$ is defined as:

$$\mathfrak{D}^m(i) = \frac{1}{m} \sum_{X_j \in X^m} [\![X_j = a_i]\!],$$

where $[\![x]\!] = 1$ if x is true and 0 otherwise.

Then, for all $\varepsilon > 0$ and $\delta \in [0, 1]$ the number of samples m to satisfy $P\{\|\mathfrak{D}^m - \mathfrak{D}\|_1 \geq \varepsilon\} \leq \delta$ equals to (from (Shervashidze et al., 2009)):

$$m = \left\lceil \frac{2}{\varepsilon^2}(\log(2^\eta - 2) - \log(\delta)) \right\rceil. \tag{2}$$

For example, there are $\eta = 877$ possible anonymous walks with length $l = 7$ (Figure 2). If we set $\varepsilon = 0.5$ and $\delta = 0.05$, then $m = 4888$. If we decrease $\varepsilon = 0.1$ and $\delta = 0.01$, then the number of samples will increase to 122500.

As transition probabilities for random walks can be preprocessed, sampling of a node in a random walk of length l can be done in $\mathcal{O}(1)$ via alias method. Hence, the overall running time of sampling approach to compute feature-based anonymous walk embeddings is $\mathcal{O}(ml)$.

Our experimental study shows state-of-the-art classification accuracy of feature-based AWE on real datasets. We continue to design data-driven approach that eliminates the sparsity of feature-based embeddings.

3.2. AWE: data-driven model

Our approach for learning network embeddings is analogous to methods for learning paragraph vectors in a text corpus (Le & Mikolov, 2014). In our case, an anonymous walk is a word, a randomly sampled set of anonymous walks starting from the same node is a set of co-occurring words, and a graph is a document.

Neighborhoods of anonymous walks. To leverage the analogy from NLP, we first need to generate a corpus of co-occurring anonymous walks in a graph G. We define a neighborhood between two anonymous walks of length l if they share the same source node. This is similar to other methods such as shortest-paths co-occurrence in DGK (Yanardag & Vishwanathan, 2015) and rooted subgraphs neighborhood in graph2vec (Narayanan et al., 2017), which proved to be successful in empirical studies. Therefore, we iterate over each vertex u in a graph G, sampling T random walks $(w_1^u, w_2^u, \dots, w_T^u)$ that start at node u and map to a sequence of co-occurred anonymous walks $s^u = (a_1^u, a_2^u, \dots, a_T^u)$, i.e. $w_i^u \mapsto a_i^u$. A collection of all s^u for all vertices $u \in G$ is a corpus of co-occurred anonymous walks in a graph and is analogous to a collection of sentences in a document.

Training. In this framework, we learn representation vector d of a graph and anonymous walks matrix W (see Figure 3). Vector d has $1 \times d_g$ size, where d_g is embedding size of a graph. Matrix W has $\eta \times d_a$ size, where η is the number of all possible anonymous walks of length l and d_a is embedding size of anonymous walk. For convenience, we call d as a document vector and W as a word matrix. Each graph corresponds to its vector d and an anonymous walk corresponds to a row in a matrix W. The model tries to predict a target anonymous walk given co-occurring context anonymous walks and a graph.

Formally, a sequence of co-occurred anonymous walks $s = (a_1, a_2, \dots, a_T)$ corresponds to vectors $\mathrm{w}_1, \mathrm{w}_2, \dots, \mathrm{w}_T$ of matrix W, and a graph G corresponds to vector d. We aim to maximize the average log probability:

$$\frac{1}{T} \sum_{t=\Delta}^{T-\Delta} \log p(\mathrm{w}_t | \mathrm{w}_{t-\Delta}, \dots, \mathrm{w}_{t+\Delta}, \mathrm{d}), \tag{3}$$

where Δ is a window size, i.e. number of context words for each target word. Probability in objective (3) is defined via softmax function:

$$p(\mathrm{w}_t|\mathrm{w}_{t-\Delta},\dots,\mathrm{w}_{t+\Delta},\mathrm{d}) = \frac{e^{y(\mathrm{w}_t)}}{\sum\limits_{i=1}^{\eta} e^{y(\mathrm{w}_i)}} \tag{4}$$

Each $y(\mathrm{w}_t)$ is unnormalized log probability for output word i:

$$y(\mathrm{w}_t) = b + Uh(\mathrm{w}_{t-\Delta},\dots,\mathrm{w}_{t+\Delta},\mathrm{d})$$

where $b \in \mathbb{R}$ and $U \in \mathbb{R}^{d_a+d_g}$ are softmax parameters. Vector h is constructed by first averaging walk vectors $\mathrm{w}_{t-\Delta},\dots,\mathrm{w}_{t+\Delta}$ and then concatenating with a graph vector d. The reason is that since anonymous walks are randomly sampled, we average vectors $\mathrm{w}_{t-\Delta},\dots,\mathrm{w}_{t+\Delta}$ to compensate for the lack of knowledge on the order of walks; and at the same time, the graph vector d is shared among multiple (context, target) pairs.

To avoid computation of the sum in softmax equation (4), which becomes impractical for large sets of anonymous walks, one can use Hierarchical softmax (Mikolov et al., 2013b) or NCE loss functions (Gutmann & Hyvärinen, 2010) to speed up training. In our work, we use sampled softmax (Jean et al., 2015) that for each training example picks only a fraction of vocabulary according to a chosen sampling function. One can measure distribution of anonymous walks in a graph via means of definition 1 and decide on a corresponding sampling function.

At every step of the model, we sample context and target anonymous walks from a graph and compute the gradient error from prediction of target walk and update vectors of context walks and a graph via gradient backpropagation. When given several networks to embed, one can reuse word matrix W across graphs, thereby sharing previously learned embeddings of walks.

Summarizing, after initialization of matrix W for all anonymous walks of length l and a graph vector d, the model repeats the following two steps for all nodes in a graph: 1) for sampled co-occurred anonymous walks the model calculates a loss (Eq. 3) of predicting a target walk (one of the sampled anonymous walks) by considering all context walks and a graph; 2) the model updates the vectors of context walks in matrix W and graph vector d via gradient backpropagation. One step of the model is depicted in Figure 3. After using up all sampled corpus, a learned graph vector d is called *anonymous walk embedding*.

Definition 4 (data-driven AWE)**.** *Anonymous walk embedding* of a graph G is a vector representation d learned on a corpus of sampled anonymous walks from a graph G.

So despite the fact that graph and walk vectors are initialized randomly, as an indirect result of predicting a walk in the context of other walks and a graph the model also learns feature representations of networks. Intuitively, a graph vector can be thought as a word with a special meaning: it serves as an overall summary for all anonymous walks in the graph.

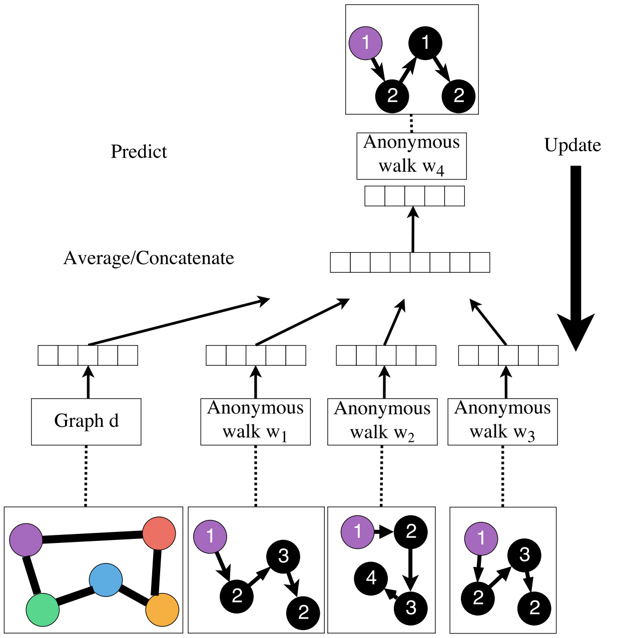

1
1
2
2
Predict
Anonymous walk w_4
Update
Average/Concatenate
Graph d
Anonymous walk w_1
Anonymous walk w_2
Anonymous walk w_3
1
3
2
2
1
2
4
3
1
3
2
2

Figure 3. A framework for learning data-driven anonymous walk embeddings. Graph is represented by a vector d and anonymous walks are represented by rows of matrix W. All co-occurring anonymous walks start from the same node in a graph. The goal is to predict a target walk w_4 by its surrounding context walks $(\mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3)$ and a graph vector d. We average embeddings of context walks and then concatenate with a graph vector to predict a target vector. Vectors are updated using stochastic gradient descent on a corpus of sampled anonymous walks.

In our experiments, we show how anonymous walk network embeddings can be used in graph classification problem, demonstrating state-of-the-art performance in classification accuracy.

4. Graph Classification

Graph classification is a task to predict a class label of a whole graph and it has found applications in bioinformatics (Nikolentzos et al., 2017) and malware detection (Narayanan et al., 2017). In this task, given a series of N graphs $\{G_i\}_{i=1}^N$ and their corresponding labels $\{L_i\}_{i=1}^N$, we are asked to train a model m: $G \mapsto L$ that would efficiently classify new graphs. Two typical approaches to graph classification problem are (1) supervised learning classification algorithms such as PSCN algorithm (Niepert et al., 2016) and (2) graph kernel methods such as WL kernel (Shervashidze et al., 2011). As we are interested in designing task-agnostic network embeddings that do not require labeled data during training, we show how to use anonymous walk embeddings in conjunction with kernel methods to perform classification of new graphs. For this we define a kernel function on two graphs.

Definition 5 (Kernel function)**.** *Kernel function* is a symmetric, positive semidefinite function k: $X \times X \mapsto \mathbb{R}^n$ defined for a non-empty set X.

When $X \subseteq \mathbb{R}^n$, several popular choices of kernel exist (Schölkopf & Smola, 2002):

- **Inner product** $k(x, y) = \langle x, y \rangle, \forall x, y \in \mathbb{R}^n$,
- **Polynomial** $k(x, y) = (\langle x, y \rangle + c)^d, \forall x, y \in \mathbb{R}^n$,
- **RBF** $k(x, y) = \exp(-\frac{\|x - y\|_2^2}{2\sigma^2}), \forall x, y \in \mathbb{R}^n$.

With network embeddings, it is then easy to define a kernel function on two graphs:

$$K(G_1, G_2) = k(f(G_1), f(G_2)), \tag{5}$$

where $f(G_i)$ is an embedding of a graph G_i and k: $(x, y) \mapsto \mathbb{R}^n$ is a kernel function.

To train a graph classifier m one can then construct a square kernel matrix $\mathcal{K}$ for training data $G_1, G_2, \dots, G_N$ and feed this matrix to a kernelized algorithm such as SVM. Every element of kernel matrix equals to: $\mathcal{K}_{ij} = K(G_i, G_j)$. For classifying new test instance G_τ, one would first compute graph kernels with training instances $(K(G_1, G_\tau), K(G_2, G_\tau), \dots, K(G_N, G_\tau))$ and provide it to a trained classifier m.

In our experiments, we use anonymous walk embeddings to compute kernel matrices and show that kernelized SVM classifier achieves top performance comparing to more complex state-of-the-art models.

5. Experiments

We evaluate our embeddings on the task of graph classification for variety of widely-used datasets.

Datasets. We evaluate performance on two sets of graphs. One set contains *unlabeled* graph data and is related to social networks (Yanardag & Vishwanathan, 2015). Another set contains graphs with labels on node and/or edges and originates from bioinformatics (Shervashidze et al., 2011). Statistics of these ten graph datasets presented in Table 1.

Evaluation. We train a multiclass SVM classifier with one-vs-one scheme. We perform a 10-fold cross-validation and for each fold we estimate SVM parameter C from the range [0.001, 0.01, 0.1, 1, 10] using validation set. This process is repeated 10 times and an average accuracy is reported, i.e. the average number of correctly classified test graphs.

Table 1. Graph datasets used in classification experiments. The columns are: Name of dataset, Number of graphs, Number of classes (maximum number of graphs in a class), Average number of nodes/edges.

Dataset	Source	Graphs	Classes	Nodes	Edges
			(Max)	Avg.	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

Competitors. PSCN is a convolutional neural network algorithm (Niepert et al., 2016) with size of receptive field equals to 10. PSCN is the state-of-the-art instance of neural network algorithms, which has achieved strong classification accuracy in many datasets, and we use the best reported accuracy for these algorithms. GK is a graphlet kernel (Shervashidze et al., 2009) and DGK is a deep graphlet kernel (Yanardag & Vishwanathan, 2015) with graphlet size equals to 7. WL is Weisfeiler-Lehman graph kernel algorithm (Shervashidze et al., 2011) with height of subtree pattern equals to 7. WL proved consistenly strong results comparing to other graph kernels and supervised algorithms. ER is exponential random walk kernel (Gärtner et al., 2003) with exponent equals to 0.5 and kR is k-step random walk kernel with $k = 3$ (Sugiyama & Borgwardt, 2015).

Setup. For feature-based anonymous walk embeddings (Def. 1), we choose length l of walks from the range $[2, 3, \dots, 10]$ and approximate actual distribution of anonymous walks using sampling equation (2) with $\varepsilon = 0.1$ and $\delta = 0.05$.

For data-driven anonymous walk embeddings (Def. 4), we set length of walks $l = 10$ to generate a corpus of co-occurred anonymous walks. We run gradient descent with 100 iterations for 100 epochs with batch size that we vary from the range [100, 500, 1000, 5000, 10000]. Context walks are drawn from a window, which size varies in the range [2, 4, 8, 16]. The embedding size of walks and graphs d_a and d_g equals to 128. Finally, candidate sampling function for softmax equation (4) chooses uniform or loguniform distribution of sampled classes.

To perform classification, we compute a kernel matrix, where Inner product, Polynomial, and RBF kernels are tested. For RBF kernel function we choose parameter σ from the range $[10^{-5}, 10^{-4}, \dots, 1, 10]$; for Polynomial function we set $c = 0$ and $d = 2$. We run the experiments on a machine with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz and 32GB RAM[1]. We refer to our algorithms as AWE (DD) and AWE (FB) for data-driven and feature-based approaches correspondingly.

Classification results. Table 2 presents results on classification accuracy for Social unlabeled datasets. AWE approaches are consistently at the top, sharing top-2 results for all six social datasets, despite being unsupervised approach unlike PSCN. At the same time, Table 4 shows accuracy results for labeled bio datasets. Note that AWE are learned using only topology of the network and not node/edge labels. In this setting, embeddings obtained by AWE (FB) approach achieves competitive performance for the labeled datasets.

Overall observations.

- Tables 2 and 4 demonstrate that AWE is competitive to supervised state-of-the-art solutions in graph classification task. Importantly, even with simple classifiers such as SVM, AWE increases classification accuracy comparing to other more complex neural network models. Likewise, just comparing graph kernels, we can see that anonymous walks is at the top with tranditional graph objects such as graphlets (GK kernel) or subtree patterns (WL kernel).

- While feature-based and data-driven approaches are different in nature, the resulted classification accuracy is close across many datasets. As such, only on RE-B dataset data-driven approach has more than 5% increase in the accuracy. In practice, we found that using feature-based approach for small length l (e.g. ≤ 10) produces competitive results, while data-driven approach works best for large number of iterations and length l.

- Polynomial and RBF kernel functions bring non-linearity to the classification algorithm and are able to learn more complex classification boundaries. Table 3 shows that RBF and Polynomial kernels are well suited for feature-based and data-driven models respectively.

Scalability. To test for scalability, we learn network representations using AWE (DD) algorithm for Erdos-Renyi graphs with increasing sizes from [10, 10^1, 10^2, 10^3, 10^4, $3 \cdot 10^4$]. For each size we construct 10 Erdos-Renyi graphs with $\mu = np \in [2, 3, 4, 5]$, where n is the number of nodes and p is the probability of having an edge between two arbitrary nodes. In that case, a graph has $m \propto \mu n$ edges. We average time to train AWE (DD) embeddings across 10 graphs for every n and μ. Our setup: size of embeddings equals to 128, batch size equals to 100, window size equals to 100. We run AWE (DD) model for 100 iterations in one epoch. In Figure 4, we empirically observe that the model to learn AWE (DD) network representations scales to networks with tens of thousands of nodes and edges and requires no more than a few seconds to map a graph to a vector.

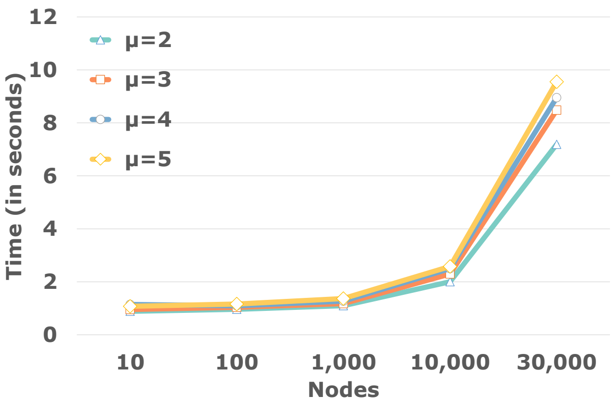


Figure 4. Average running time to generate anonymous walk embedding for Erdos-Renyi graphs, with $\mu = np \in [2, 3, 4, 5]$ where n is the number of nodes and p is probability parameter of Erdos-Renyi model. X-axis is in log scale.

Intuition behind performance. There is a couple of factors that leads anonymous walk embeddings to state-of-the-art performance in graph classification task. First, the use of anonymous walks is backed up by a recent discovery that, under certain condition, distribution of anonymous walks of a single node is sufficient to reconstruct a topology of the ball around a node. Hence, at least on a level of a single node, distribution of anonymous walk serves as a unique representation of subgraphs in a network. Second, data-driven approach reuses hitherto learned embeddings matrix W in previous iterations for learning embeddings of new graph instances. Therefore one can think of anonymous walks as words that have semantic meaning unified across all graphs. While learning graph embeddings, we simultaneously learn the meaning of different anonymous walks, which provides extra information for our model.

6. Related Work

Network representations were first studied in the context of graph kernels (Gärtner et al., 2003) and then have become a separate topic that found numerous applications beyond graph classification (Cai et al., 2017). Our feature-based embeddings originate from learning distribution on anonymous walks in a graph and is alike to the approach of graph kernels. Embeddings based on graph kernels include Ran-

[1] Code can be found at https://github.com/nd7141/AWE

Table 2. Comparison of classification accuracy (mean ± std., %) in Social datasets. Top-2 results are in **bold**. OOM is out-of-memory.

	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

Table 3. Kernel function comparison in classification task (%).

Algorithm	IMDB-M	COLLAB	RE-B
AWE (DD) *RBF*	50.73	**73.93**	**87.89**
AWE (DD) *Inner*	**51.54**	73.77	84.82
AWE (DD) *Poly*	45.32	70.45	79.35
AWE (FB) *RBF*	**51.58**	**70.99**	**82.97**
AWE (FB) *Inner*	46.45	69.60	76.83
AWE (FB) *Poly*	46.57	64.3	67.22

Table 4. Classification accuracy (%) in labeled 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

dom Walk (Gärtner et al., 2003), Graphlet (Shervashidze et al., 2009), Weisfeiler-Lehman (Shervashidze et al., 2011), Shortest-Path (Borgwardt & Kriegel, 2005) decompositions and all can be summarized as an instance of R-convolution framework (Haussler, 1999).

Distributed representations have become trendy after significant achievements in NLP applications (Mikolov et al., 2013a;b). Our data-driven network embeddings stem from paragraph-vector distributed-memory model (Le & Mikolov, 2014) that has become successful in learning document representations. Other related approaches include Deep Graph Kernel (Yanardag & Vishwanathan, 2015) that learns a matrix for graph kernel that encodes relationship between substructures; PSCN (Niepert et al., 2016) and 2D CNN (Tixier et al., 2017) algorithms that learn convolutional neural networks on graphs; graph2vec (Narayanan et al., 2017) learns network embeddings by extracting rooted subgraphs and training on skipgram negative sampling model (Mikolov et al., 2013b); FGSD (Verma & Zhang, 2017) that constructs feature vector from the histogram of the multiset of node pairwise distances. (Cai et al., 2017) provides a more comprehensive list of graph embeddings. Besides this, there is a list of aggregation techniques of node embeddings for the purpose of graph classification (Hamilton et al., 2017).

7. Conclusion

We described two unsupervised algorithms to compute network vector representations using anonymous walks. In the first approach, we use distribution of anonymous walks as a network embedding. As the exact calculation of network embeddings can be expensive we demonstrate how one can sample walks in a graph to approximate actual distribution with a given confidence. Next, we show how one can learn distributed graph representations in a data-driven manner, similar to learning paragraph vectors in NLP.

In our experiments, we show that our network embeddings even with simple SVM classifier achieve increase in classification accuracy comparing to state-of-the-art supervised neural network methods and graph kernels. This demonstrates that representation of your data can be more promising subject to study than the type and architecture of your predictive model.

Although the focus of this work was in representation of networks, AWE algorithm can be used to learn node, edge, or any subgraph representations by replacing graph vector with a corresponding subgraph vector. In all graph and subgraph representations, we expect data-driven approach to be a strong alternative to feature-based methods.

8. Acknowledgement

This work was supported by the Ministry of Education and Science of the Russian Federation (Grant no. 14.756.31.0001) and by the Skoltech NGP Program No. 1-NGP-1567 Simulation and Transfer Learning for Deep 3D Geometric Data Analysis (a Skoltech-MIT joint project).

References

Abraham, A. *Computational Social Networks: Security and Privacy*. Springer Publishing Company, Incorporated, 2012.

Babai, L. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pp. 684–697, 2016.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Borgwardt, K. M. and Kriegel, H. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pp. 74–81, 2005.

Cai, H., Zheng, V. W., and Chang, K. C. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017. URL http://arxiv.org/abs/1709.07604.

Cao, S., Lu, W., and Xu, Q. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 1145–1152, 2016.

Gärtner, T., Flach, P. A., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, pp. 129–143, 2003.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pp. 297–304, 2010.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

Haussler, D. Convolution kernels on discrete structures. Technical report, 1999.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. On using very large target vocabulary for neural machine translation. In *ACL 2015*, pp. 1–10, 2015.

Le, Q. V. and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1188–1196, 2014.

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

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *CoRR*, 2013a. URL http://arxiv.org/abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems NIPS*, pp. 3111–3119, 2013b.

Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5425–5434, 2017.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. graph2vec: Learning distributed representations of graphs. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2014–2023, 2016.

Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. Matching node embeddings for graph similarity. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 2429–2435, 2017.

Schölkopf, B. and Smola, A. J. *Learning with Kernels: support vector machines, regularization, optimization, and beyond.* Adaptive computation and machine learning series. MIT Press, 2002.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

Sugiyama, M. and Borgwardt, K. M. Halting in random walk kernels. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1639–1647, 2015.

Tixier, A. J., Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. Classifying graphs as images with convolutional neural networks. *CoRR*, abs/1708.02218, 2017. URL http://arxiv.org/abs/1708.02218.

Tubig, H. The number of walks and degree powers in directed graphs. Technical report, Computer Science Department, Rutgers University, TUM-I123, TU Munich, 2012.

Verma, S. and Zhang, Z. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 87–97, 2017.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010. ISSN 1532-4435.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pp. 1365–1374, 2015.