Multiple Kernel Representation Learning on Networks
Abdulkadir Çelikkanat®, Yanning Shen®, and Fragkiskos D. Malliaros®

Abstract—Learning representations of nodes in a low dimensional space is a crucial task with numerous interesting applications in network analysis, including link prediction, node classification, and visualization. Two popular approaches for this problem are matrix factorization and random walk-based models. In this paper, we aim to bring together the best of both worlds, towards learning node representations. In particular, we propose a weighted matrix factorization model that encodes random walk-based information about nodes of the network. The benefit of this novel formulation is that it enables us to utilize kernel functions without realizing the exact proximity matrix so that it enhances the expressiveness of existing matrix decomposition methods with kernels and alleviates their computational complexities. We extend the approach with a multiple kernel learning formulation that provides the flexibility of learning the kernel as the linear combination of a dictionary of kernels in data-driven fashion. We perform an empirical evaluation on real-world networks, showing that the proposed model outperforms baseline node embedding algorithms in downstream machine learning tasks.

Index Terms—Graph representation learning, node embeddings, kernel methods, node classification, link prediction

1 INTRODUCTION

With the development in data production, storage and consumption, graph data is becoming omnipresent; data from diverse disciplines can be represented as graph structures with prominent examples including various social, information, technological, and biological networks. Developing machine learning algorithms to analyze, predict, and make sense of graph data has become a crucial task with a plethora of cross-disciplinary applications [1], [2]. The major challenge in machine learning on graph data concerns the encoding of information about the graph structural properties into the learning model. To this direction, network representation learning (NRL), a recent paradigm in network analysis, has received substantial attention thanks to its outstanding performance in downstream tasks, including link prediction and classification. Representation learning techniques mainly target to embed the nodes of the graph into a lower-dimensional space in such a way that desired relational properties among graph nodes are captured by the similarity of the representations in the embedding space [3], [4], [5], [6], [7], [8].

The area of NRL has been highly impacted by traditional nonlinear dimensionality reduction approaches [5], [9]. Specifically, many proposed models had initially concentrated on learning node embeddings relying on matrix factorization techniques that encode structural node similarity[5], [6], [7]. Nevertheless, most of those approaches are not very efficient for large scale networks, mainly due to the high computational and memory cost required to perform matrix factorization. Besides, most such models require the exact realization of the target matrix [1], [2].

Inspired by the field of natural language processing [10], random-walk based embedding has gained considerable attention (e.g., [3], [4], [11], [12], [13], [14]). Typically, these methods first produce a set of node sequences by following a certain random walk strategy. Then, node representations are learned by optimizing the relative co-occurrence probabilities of nodes within the random walks. Although such random walk models follow similar approaches in modeling the relationships between nodes and optimizing the embedding vectors, their difference mainly stems from the way in which node sequences are sampled [3], [4].

On a separate topic, kernel functions have often been introduced along with popular learning algorithms, such as PCA [15], SVM [16], Spectral Clustering [17], and Collaborative Filtering [18], to name a few. Most of traditional learning models are insufficient to capture the underlying substructures of complex datasets, since they rely on linear techniques to model nonlinear patterns existing in data. Kernel functions [19], on the other hand, allow mapping non-linearly separable points into a (generally) higher dimensional feature space, so that the inner product in the new space can be computed without needing to compute the exact feature maps—bringing further computational benefits. Besides, to further reduce model bias, multiple kernel learning approaches have been proposed to learn optimal combinations of kernel functions [20]. Nevertheless, despite their wide applications in various fields [19], [21], kernel and
multiple kernel methods have not been thoroughly investigated for learning node embeddings.

In this paper, we aim at combining matrix factorization and random walks in a kernelized model for learning node embeddings. The potential advantage of such a modeling approach is that it allows for leveraging and combining the elegant mathematical formulation offered by matrix factorization with the expressive power of random walks to capture a notion of “stochastic” node similarity in an efficient way. More importantly, this formulation enables leveraging kernel functions in the node representation learning task. Because of the nature of matrix factorization-based models, node similarities can be viewed as inner products of vectors lying in a latent space—which allows to utilize kernels towards interpreting the embeddings in a higher dimensional feature space using non-linear maps. Besides, multiple kernels can be utilized to learn more discriminative node embeddings. Note that, although graph kernels is a well-studied topic in graph learning [22], it mainly focuses on graph classification—a task outside of the scope of this paper. To the best of our knowledge, random walk-based multiple kernel matrix factorization has not been studied before for learning node embeddings.

The main contributions of the present work can be summarized as follows:

- We propose **KernelNE** (Kernel Node Embeddings), a novel approach for learning node embeddings by incorporating kernel functions with models relying on weighted matrix factorization, encoding random walk-based structural information of the graph. We further examine the performance of the model with different types of kernel functions.
- To further improve expressiveness, we introduce **MKernelNE**, a multiple kernel learning formulation of the model. It extends the kernelized weighted matrix factorization framework by learning a linear combination of a predefined set of kernels.
- We demonstrate how both models (single and multiple kernel learning) leverage negative sampling to efficiently compute node embeddings.
- We extensively evaluate the proposed method’s performance in node classification and link prediction. We show that the proposed models generally outperform well-known baseline methods on various real-world graph datasets. Besides, due to the efficient model optimization mechanism, the running time is comparable to the one of random walk models.

**Notation.** We use the notation $M$ to denote a matrix, and the term $M_{v,u}$ represents the entry at the $v$th row and $u$th column of the matrix $M$. $M_{v,:}$ indicates the $v$th row of the matrix.

**Source Code.** The C++ implementation of the proposed methodology can be reached at: https://abdcelikka.nat.github.io/projects/kernelNE/.

## 2 RELATED WORK

**Node Embeddings.** Traditional dimension reduction methods such as Principal Component Analysis (PCA) [23], Locally Linear Embeddings [9], and ISOMAP [24] that rely on the matrix factorization techniques, have considerably influenced early approaches on learning node embeddings. For instance, HOPE [7] learns embedding vectors by factorizing a higher-order proximity matrix using SVD, while M-NMF [25] leverages non negative matrix factorization aiming to preserve the underlying community structure of the graph. Similarly, NetMF [26] and its scalable variant NetSMF [27], learn embeddings by properly factorizing a target matrix that has been designed based on the Positive Pointwise Mutual Information (PPMI) of random walk proximities. Similarly, SDAE [28] uses the PPMI matrix with stacked denoising autoencoders to learn the representations. SDNE [29] adapts a deep neural network architecture to capture the highly non-linear network patterns. Nevertheless, as we have mentioned in Section 1, most of these models suffer from high time complexity, while at the same time, they assume an exact realization of the target matrix. For a detailed description of matrix factorization models for node embeddings, the reader can refer to [1], [2], [30].

To further improve the expressiveness as well as to alleviate the computational burden of matrix factorization approaches, models that rely on random walk sampling have been developed (e.g., [3], [4], [11], [12], [13], [14], [31], [32], [33]). These models aim at modeling center-context node relationships in random walk sequences, leveraging advances on learning word embeddings in natural language processing [10]. Due to the flexible way that random walks can be defined, various formulations have been proposed, with the two most well-known being **DeepWalk** [3] and **Node2Vec** [4]. Most of these models though, are based on shallow architectures which do not allow to model complex non-linear data relationships, limiting the prediction capabilities on downstream tasks (e.g., node classification) [30]. There are also recent efforts towards developing scalable models relying on fast random projections for very large networks [34], [35], [36], [37].

As will be presented shortly, in this paper we aim to bring together the best of both worlds, by proposing an efficient random walk-based matrix factorization framework that allows to learn informative embeddings.

**Kernel Methods.** Although most algorithms in the broader field of machine learning have been developed for the linear case, real-world data often requires nonlinear models capable of unveiling the underlying complex relationships towards improving the performance of downstream tasks. To that end, kernel functions allow computing the inner product among data points in a typically high-dimensional feature space, in which linear models could still be applied, without explicitly computing the feature maps [19]. Because of the generality of the inner product, kernel methods have been widely used in a plethora of models, including Support Vector Machine [16], PCA [15], spectral clustering [17], [38], collaborative filtering [18], and non-negative matrix factorization for image processing tasks [39], to name a few. Kernel functions have also been utilized in the field of graph analysis. At the node level, diffusion kernels and their applications [40] constitute notable instances. At the graph level, **graph kernels** [41], such as the random walk and Weisfeiler-Lehman kernels [22], have mainly been utilized to measure the similarity between a pair of graphs for applications such as graph classification. Besides, they have also
used for capturing the temporal changes in time-evolving networks [42]. Recent approaches have also been proposed to leverage graph kernels for node classification, but in a supervised manner [43]. Other related applications of kernel methods on graphs include topology inference [44], [45], signal reconstruction [46], and anomaly detection [47].

The expressiveness of kernel methods can further be enhanced using multiple kernel functions in a way that the best possible combination of a set of predefined kernels can be learned [20], [48]. Besides improving prediction performance, multiple kernels have also been used to combine information from distinct heterogeneous sources (please see [20] and [49] for a detailed presentation of several multiple kernel learning algorithms and their applications). Examples include recent approaches that leverage multi-kernel strategies for graph and image datasets [50], [51].

Despite the widespread applications of kernel and multiple kernel learning methods, utilizing them for learning node embeddings via matrix factorization in an unsupervised way is a problem that has not been thoroughly investigated. Previous works that are close to our problem settings, which follow a methodologically different factorization approach (e.g., leveraging nonnegative matrix factorization [18]), targeting different application domains (e.g., collaborative filtering [52]). The multiple kernel framework for graph-based dimensionality reduction proposed by Lin et al. [53] is also close to our work. Nevertheless, their work focuses mainly on leveraging multiple kernel functions to fuse image descriptors in computer vision tasks properly.

In this paper we propose novel unsupervised models for node embeddings, that implicitly perform weighted matrix decomposition in a higher-dimensional space through kernel functions. The target pairwise node proximity matrix is properly designed to leverage information from random walk sequences, and thus, our models do not require the exact realization of this matrix. Both single and multiple kernel learning formulations are studied. Emphasis is also put on optimization aspects to ensure that node embeddings are computed efficiently.

### 3 Modeling and Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V} = \{1, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the vertex and edge sets, respectively. Our goal is to find node representations in a latent space, preserving properties of the network. More formally, we define the general objective function of our problem as a weighted matrix factorization [54], as follows:

$$\arg\min_{\mathbf{A}, \mathbf{B}} \left\| \mathbf{W} \odot (\mathbf{M} - \mathbf{AB}^\top) \right\|_F^2 + \frac{\lambda}{2} \left( \|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2 \right),$$

(1)

where $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the target matrix constructed based on the desired properties of a given network, which is used to learn node embeddings $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$, and $\| \cdot \|_F$ denotes the Frobenius norm. We will use $\mathcal{R}(\mathbf{A}, \mathbf{B})$ to denote the regularization term of Eq. (1). Each element $W_{uv}$ of the weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$ captures the importance of the approximation error between nodes $u$ and $v$, and $\odot$ indicates the Hadamard product. Depending on the desired graph properties that we are interested in encoding, there are many possible alternatives to choose matrix $\mathbf{M}$; such include the number of common neighbors between a pair of nodes, higher-order node proximity based on the Adamic-Adar or Katz indices [7], as well leveraging multi-hop information [6]. Here, we will design $\mathbf{M}$ as a sparse binary matrix utilizing information of random walks over the network.

As we have mentioned above, random walk-based node embedding models (e.g., [3], [4], [11], [14], [55], [56], [57]) have received great attention because of their good prediction performance and efficiency on large scale-networks. Typically, those models generate a set of node sequences by simulating random walks; node representations are then learned by optimizing a model which defines the relationships between nodes and their contexts within the walks. More formally, for a random walk $\mathbf{w} = (w_1, \ldots, w_L)$, the context set of center node $w_l \in \mathcal{V}$ appearing at the position $l$ of the walk $\mathbf{w}$ is defined by $\{w_{l-\gamma}, \ldots, w_{l-1}, w_{l+1}, \ldots, w_{l+\gamma}\}$, where $\gamma$ is the window size which is the furthest distance between the center and context nodes. The embedding vectors are then obtained by maximizing the likelihood of occurrences of node $v$ in the generated walks. Setting each term $W_{(v,a)}$ as the square root of $F_{(v,a)}$, the objective function in Eq. (1) can be expressed under a random walk-based formulation as follows:

$$\arg\min_{\mathbf{A}, \mathbf{B}} \left\| \mathbf{W} \odot (\mathbf{M} - \mathbf{AB}^\top) \right\|_F^2 + \mathcal{R}(\mathbf{A}, \mathbf{B})$$

$$= \arg\min_{\mathbf{A}, \mathbf{B}} \left\| \sqrt{\mathbf{F}} \odot (\mathbf{M} - \mathbf{AB}^\top) \right\|_F^2 + \mathcal{R}(\mathbf{A}, \mathbf{B})$$

$$= \arg\min_{\mathbf{A}, \mathbf{B}} \sum_{v \in \mathcal{V}} \sum_{u \in \mathcal{V}} F_{(v,u)} (M_{(v,u)} - \langle \mathbf{A}[u], \mathbf{B}[v] \rangle)^2 + \mathcal{R}(\mathbf{A}, \mathbf{B})$$

$$= \arg\min_{\mathbf{A}, \mathbf{B}} \sum_{v \in \mathcal{V}} \sum_{u \in \mathcal{V}} \sum_{s \in \mathcal{V}} \sum_{w \in \mathcal{V}} \mathbf{#}(v,s) (\|w = s\| - \langle \mathbf{A}[u], \mathbf{B}[v] \rangle)^2 + \mathcal{R}(\mathbf{A}, \mathbf{B})$$

$$= \arg\min_{\mathbf{A}, \mathbf{B}} \sum_{(u,s) \in \mathcal{E}} \sum_{w \in \mathcal{V}} \mathbf{#}(v,s) (\|w = s\| - \langle \mathbf{A}[u], \mathbf{B}[v] \rangle)^2 + \mathcal{R}(\mathbf{A}, \mathbf{B})$$

$$= \arg\min_{\mathbf{A}, \mathbf{B}} \sum_{w \in \mathcal{V}} \sum_{l = 1}^{L} \sum_{j \neq \emptyset} \sum_{u \in \mathcal{V}} (\|w_{l+j} = u\| - \langle \mathbf{A}[u], \mathbf{B}[w_i] \rangle)^2 + \mathcal{R}(\mathbf{A}, \mathbf{B}),$$

(2)

where each $\mathbf{w} \in \mathcal{V}^L$ indicates a random walk of length $L$ in the collection $\mathcal{W}$, $\#(v,s)$ denotes the number of appearances of the center and context pairs $(v, s)$ in the collection $\mathcal{W}$, $\mathbf{[\cdot]}$ is the the Iverson bracket, and $\mathcal{R}(\mathbf{A}, \mathbf{B})$ is the regularization term. Note that, in the equation above, the last line follows from the fact that $\#(v,s)$ is equal to zero for the pairs which
are not center-context, and the term \( \sum_{w \in W} \sum_{l=1}^{L} \sum_{j \geq y} \) traverses all center-context pairs in the collection. Matrix \( A \) in Eq. (2) indicates the embedding vectors of nodes when they are considered as centers; those will be the embeddings that are used in the experimental evaluation. The choice of matrix \( M \) and the reformulation of the objective function, offers a computational advantage during the optimization step. Moreover, such formulation also allows us to further explore kernelized version in order to exploit possible non-linearity of the model.

4 Kernel-Based Representation Learning

Most matrix factorization techniques that aim to find latent low-dimensional representations (e.g., [7], [26], [27]), adopt the Singular Value Decomposition (SVD) provides the best approximation of the objective function stated in Eq. (1), as long as the weight matrix is uniform [58]. Nevertheless, in our case the weight matrix is not uniform, therefore we need the exact realization of the target matrix in order to perform SVD. To overcome this limitation, we leverage kernel functions to learn node representations via matrix factorization.

Let \((X, d_X)\) be a metric space and \( \mathbb{H} \) be a Hilbert space of real-valued functions defined on \( X \). A Hilbert space is called reproducing kernel Hilbert space (RKHS) if the point evaluation map over \( \mathbb{H} \) is a continuous linear functional. Furthermore, a feature map is defined as a function \( \Phi : X \rightarrow \mathbb{H} \) from the input space \( X \) into feature space \( \mathbb{H} \). Every feature map defines a kernel \( K : X \times X \rightarrow \mathbb{R} \) as follows:

\[
K(x, y) := \langle \Phi(x), \Phi(y) \rangle \quad \forall (x, y) \in X^2.
\]

It can be seen that \( K(\cdot, \cdot) \) is symmetric and positive definite due to the properties of an inner product space.

A function \( g : X \rightarrow \mathbb{R} \) is induced by \( K \), if there exists \( h \in \mathbb{H} \) such that \( g = \langle h, \Phi(\cdot) \rangle \) for a feature vector \( \Phi \) of kernel \( K \). Note that, this is independent of the definition of the feature map \( \Phi \) and space \( \mathbb{H} \) [59]. Let \( \mathcal{I}_x := \{g : X \rightarrow \mathbb{R} \mid \exists h \in \mathbb{H} \text{ s.t. } g = \langle h, \Phi(\cdot) \rangle \} \) be the set of induced functions by kernel \( K \). Then, a continuous kernel \( K \) on a compact metric space \((X, d_X)\) is universal, if the set \( \mathcal{I}_x \) is dense in the space of all continuous real-valued functions \( \mathcal{C}(X) \). In other words, for any function \( f \in \mathcal{C}(X) \) and \( \epsilon > 0 \), there exists \( g_h \in \mathcal{I}_x \) satisfying

\[
\| f - g_h \|_\infty \leq \epsilon,
\]

where \( g_h \) is defined as \( \langle h, \Phi(\cdot) \rangle \) for some \( h \in \mathbb{H} \).

In this paper, we consider universal kernels, since we can always find \( h \in \mathbb{H} \) satisfying \( \| \langle h, \Phi(x) \rangle - a_i \| \leq \epsilon \) for given \( \{x_1, \ldots, x_N\} \subset X, \{a_1, \ldots, a_N\} \subset \mathbb{R} \) and \( \epsilon > 0 \) by Proposition 1. If we choose \( a_i \)'s as the entries of a row of our target matrix \( M \), then the elements \( h \) and \( \Phi(x) \) indicate the corresponding row vectors of \( A \) and \( B \), respectively. Then, we can obtain a decomposition of the target matrix by repeating the process for each row. However, the element \( h \) might not always be in the range of feature map \( \Phi \); in this case, we will approximate the correct values of \( M \).

Proposition 1 (Universal kernels [59]). Let \((X, d)\) be a compact metric space and \( K(\cdot, \cdot) \) be a universal kernel on \( X \). Then, for all compact and mutually disjoint subsets \( S_1, \ldots, S_n \subset X \), all \( \alpha_1, \ldots, \alpha_n \in \mathbb{R} \), and all \( \epsilon > 0 \), there exists a function \( g \) induced by \( K \) with \( \| g \|_\infty \leq \max_j |\alpha_j| + \epsilon \) such that

\[
\| g_S - \sum_{i=1}^{n} \alpha_i \mathbb{1}_{S_i} \|_\infty \leq \epsilon,
\]

where \( S := \bigcup_{i=1}^{n} S_i \) and \( g_S \) is the restriction of \( g \) to \( S \).

Universal kernels also provide a guarantee for the injectivity of the feature maps, as shown in Lemma 2; therefore, we can always find \( y \in X \), such that \( \Phi(y) = h \) if \( h \in \Phi(X) \). Otherwise, we can learn an approximate pre-image solution by using a gradient descent technique [60].

Lemma 2 ([59]). Every feature map of a universal kernel is injective.

Proof. Let \( \Phi(\cdot) : X \rightarrow \mathbb{H} \) be a feature vector of the kernel \( K(\cdot, \cdot) \). Assume that \( \Phi \) is not injective, so we can find a pair of distinct elements \( x, y \in X \) such that \( \Phi(x) \neq \Phi(y) \) and \( x \neq y \). By Proposition 1, for any given \( \epsilon > 0 \), there exists a function \( g = \langle h, \Phi(\cdot) \rangle \) induced by \( K \) for some \( h \in \mathbb{H} \), which satisfies

\[
\| g_S - (\mathbb{1}_{S_1} - \mathbb{1}_{S_2}) \|_\infty \leq \epsilon,
\]

where \( S := S_1 \cup S_2 \) for the compact sets \( S_1 = \{x\} \) and \( S_2 = \{y\} \). Then, we have that \( \| \langle \Phi(x), h \rangle - \langle \Phi(y), h \rangle \| \geq 2 - 2\epsilon \). In other words, we obtain \( \Phi(x) \neq \Phi(y) \), which contradicts our initial assumption. \( \square \)

4.1 Single Kernel Node Representation Learning

Following the kernel formulation described above, we can now perform matrix factorization in the feature space by leveraging kernel functions. In particular, we can move the inner product from the input space \( X \) to the feature space \( \mathbb{H} \), by reformulating Eq. (2) as follows:

\[
\begin{align*}
\arg\min_{A, B} & \sum_{x \in W} \sum_{l=1}^{L} \sum_{j \neq y \in \mathcal{V}} \sum_{w \in \mathcal{V}} \left( \| u_{l+j} - u \| - \langle \Phi(A_{w,u}), \Phi(B_{w,y}) \rangle \right)^2 \\
& + R(A, B) \\
= & \sum_{x \in W} \sum_{l=1}^{L} \sum_{j \neq y \in \mathcal{V}} \sum_{w \in \mathcal{V}} \left( \| u_{l+j} - u \| - K(A_{w,u}, B_{w,y}) \right)^2 \\
& + R(A, B).
\end{align*}
\]

In this way, we obtain a kernelized matrix factorization model for node embeddings based on random walks. For the numerical evaluation of our method, we use the following universal kernels [59], [61]:

\[
\begin{align*}
K_G(x, y) &= \exp \left( -\frac{\| x - y \|^2}{\sigma^2} \right) \quad \sigma \in \mathbb{R} \\
K_S(x, y) &= \frac{1}{1 + \| x - y \|^2} \quad \sigma \in \mathbb{R}_+,
\end{align*}
\]

where \( K_G \) and \( K_S \) correspond to the Gaussian and Schoenberg kernels respectively. We will refer to the proposed kernel-based node embeddings methodology as KERNELNE (the two different kernels will be denoted by GAUSS and SCH). A
KERNELNE model. Node sequences are first generated by following a random walk strategy. By using the co-occurrences of node pairs within a certain window size, node representations are learned by optimizing their maps in the feature space.

4.1.1 Model Optimization
The estimation problem for both parameters $A$ and $B$ is, unfortunately, non-convex. Nevertheless, when we consider each parameter separately by fixing the other one, it turns into a convex problem. By taking advantage of this property, we employ Stochastic Gradient Descent (SGD) [62] in the optimization step of each embedding matrix. Note that, for each context node $w_{i+j}$ in Eq. (3), we have to compute the gradient for each node $u \in V$, which is computationally intractable. However, Eq. (3) can be divided into two parts with respect to the values of $|w_{i+j} = u| \in \{0, 1\}$, as follows:

$$\sum_{u \in V} \left( \left( \sum_{u \in V} \right) - K(A(u, \cdot), B(w_{i+j})) \right)^2$$

$$= \left( \sum_{u \in V} \right) - K(A(u', \cdot), B(u'))^2 + \sum_{u \in V} \left( \sum_{u \in V} K(A(u', \cdot), B(u')) \right)^2$$

$$\approx \left( \sum_{u \in V} \right) - K(A(u', \cdot), B(u'))^2 + \left[ \sum_{u \in V} \right] \left( \sum_{u \in V} K(A(u', \cdot), B(u')) \right)^2$$

where $k := |V| - 1$ and $u^+ := w_{i+j}$. To this end, we apply negative sampling [10] which is a variant of noise-contrastive estimation [63], proposed as an alternative to solve the computational problem of hierarchical softmax. For each context node $u^+ \in V$, we sample $k$ negative instances $u^-$ from the noise distribution $p^-$. Then, we can rewrite the objective function Eq. (3) in the following way:

$$F_S := \arg\min_{A, B} \sum_{u \in V} \sum_{i=1}^{L} \sum_{j=1}^{L} \left( 1 - K(A(w_{i+j}), B(w_{i+j})) \right)^2$$

$$+ \sum_{i=1}^{L} \left( K(A(w_{i+j}), B(w_{i+j})) \right)^2 + R(A, B).$$

Equation (4) corresponds to the objective function of the proposed KERNELNE model. In the following subsection, we will study how this model could be further extended to leverage multiple kernels.

4.2 Multiple Kernel Node Representation Learning
Selecting a proper kernel function $K(\cdot, \cdot)$ and the corresponding parameters (e.g., the bandwidth of a Gaussian kernel) is a critical task during the learning phase. Nevertheless, choosing a single kernel function might impose potential bias, causing limitations on the performance of the model. Having the ability to properly utilize multiple kernels could increase the expressiveness of the model, capturing different notions of similarity among embeddings [20]. Besides, learning how to combine such kernels, might further improve the performance of the underlying model. In particular, given a set of base kernels $\{K_i\}_{i=1}^K$, we aim to find an optimal way to combine them, as follows:

$$K_c(x, y) = f_c(\{K_i(x, y)\}_{i=1}^K | c),$$

where the combination function $f_c$ is parameterized on $c \in \mathbb{R}^K$ that indicates kernel weights. Due to the generality of the multiple kernel learning framework, $f_c$ can be either a linear or nonlinear function.

In this paragraph, we examine how to further strengthen the proposed kernelized weighted matrix factorization, by linearly combining multiple kernels. Let $K_1, \ldots, K_K$ be a set of kernel functions satisfying the properties presented in the previous paragraph. Then, we can restate the objective function as follows:

$$F_M := \arg\min_{A, B, c} \sum_{u \in V} \sum_{i=1}^{L} \sum_{j=1}^{L} \left( 1 - \sum_{i=1}^{K} c_i K_i(A(w_{i+j}), B(w_{i+j})) \right)^2$$

$$+ \sum_{i=1}^{K} \left( \sum_{i=1}^{K} c_i K_i(A(w_{i+j}), B(w_{i+j})) \right)^2$$

$$+ \frac{\lambda}{2} \left( ||A||^2_F + ||B||^2_F \right) + \frac{\beta}{2} ||c||_2^2,$$

where $c := [c_1, \ldots, c_K]^T \in \mathbb{R}^K$. Here, we introduce an additional parameter $c_i$ representing the contribution of the corresponding kernel $K_i$. $\beta > 0$ is a trade-off parameter, and similarly, the coefficients $c_1, \ldots, c_K$ are optimized by fixing the remaining model parameters $A$ and $B$. Equation (5) corresponds to the objective function of the proposed multiple kernel learning model MKERNELNE. Unlike the common usage of multiple kernels methods [20], here we do not constrain the coefficients to be non-negative. We interpret each entry of the target matrix as a linear combination of inner products of different kernels’ feature maps. As discussed in
the previous sections, our main intuition relies on obtaining more expressive embeddings by projecting the factorization step into a higher dimensional space.

**Algorithm 1. MKERNELNE**

**Input:** Graph \( G = (\mathcal{V}, \mathcal{E}) \)
- Representation size \( d \)
- Set of walks \( \mathcal{W} \)
- Window size \( \gamma \)
- Kernel function \( K \)
- Kernel parameter(s) \( \sigma \)

**Output:** Embedding matrix \( A \)

1: Initialize matrices \( A, B \in \mathbb{R}^{n \times d} \)

/* Extract center-context node pairs */
2: for all \( w = (w_1, \ldots, w_L) \in \mathcal{W} \)
3: for \( l \leftarrow 1 \) to \( L \)
4: for \( j \neq 0 \leftarrow -\gamma \) to \( \gamma \)
5: \( A, B, c \leftarrow \text{UpdateEmb}(A, B, c, w_j, w_{l+j}, K, \sigma) \)
6: end for
7: end for
8: end for

Algorithm 1 summarizes the pseudocode of the proposed approach. For a given collection of random walks \( \mathcal{W} \), we first determine the center-context node pairs \( (w_l, w_{l+j}) \) in the node sequences. Recall that, for each center node in a walk, its surrounding nodes within a certain distance \( \gamma \), define its context. Furthermore, the corresponding embedding vectors \( B(u_c) \) of center node \( u_l \) and \( A(u_l, c) \) of context \( u_{l+j} \) are updated by following the rules which we describe in detail below. Note that, we obtain two different representations, \( A(u_c) \) and \( B(u_c) \), for each node \( v \in \mathcal{V} \) since the node can have either a center or context role in the walks.

The gradients in Algorithm 2 are given below. For notation simplicity, we denote each \( K_i(A(u_c), B(u_c)) \) by \( K_i(u, v) \).

\[
\nabla_{A(u_c)} F_M := [x = u] \left( -2 \sum_{i=1}^{K} c_i \nabla_{A(u_c)} K_i(x, v) \right) \left( 1 - \sum_{i=1}^{K} c_i K_i(x, v) \right) + 2 \sum_{i=1}^{K} \left[ x = u \right] c_i \nabla_{A(u_c)} K_i(x, v) K_i(u, v) + \lambda A(u_c)
\]

\[
\nabla_{B(u_c)} F_M := -2 \sum_{i=1}^{K} c_i \nabla_{B(u_c)} K_i(u, v) \left( 1 - \sum_{i=1}^{K} c_i K_i(u, v) \right) + 2 \sum_{i=1}^{K} \left[ x = u \right] c_i \nabla_{B(u_c)} K_i(u, v) K_i(u, v) + \lambda B(u_c)
\]

\[
\nabla_{c_i} F_M = -2 \left( 1 - \sum_{i=1}^{K} c_i K_i(u, v) \right) K_i(u, v) + 2 \sum_{i=1}^{K} \left[ x = u \right] c_i K_i(u, v) + \beta c_i.
\]

4.3 Complexity Analysis

For the generation of walks, the biased random walk strategy proposed in Node2Vec is used, which can be performed in \( O(|\mathcal{W}| \cdot L) \) steps [4] for the pre-computed transition probabilities, where \( L \) indicates the walk length and \( \mathcal{W} \) denotes the set of walks. For the algorithm’s learning procedure, we can carry out Line 5 of Algorithm 1 at most \( 2 \gamma \cdot |\mathcal{W}| \cdot L \) times for each center-context pair, where \( \gamma \) represents the window size. The dominant operation in Algorithm 2 is the multiplication operation of update rules in Lines 6, 7, and 9; the running time can be bounded by \( O(k \cdot K \cdot d) \), where \( k \) is the number of negative samples generated per center-context pair, \( K \) is the number of kernels, and \( d \) is the representation size. To sum up, the overall running time of the proposed approach can be bounded by \( O(\gamma \cdot |\mathcal{W}| \cdot L \cdot K \cdot d \cdot k) \) steps.

**Algorithm 2. UPDATEMB**

**Input:** Graph \( G = (\mathcal{V}, \mathcal{E}) \)
- Embedding matrices \( A \) and \( B \)
- Kernel coefficients \( c = (c_1, \ldots, c_K) \)
- Kernel function(s) \( K \)
- Center and context nodes \( v \) and \( u \)
- Learning rate \( \eta \)

**Output:** Embedding matrix \( A \)

1: node_list \( \leftarrow \{u\} \)

/* Extract negative samples */
2: for \( s \leftarrow 1 \) to \( k \)
3: node_list \( \leftarrow \text{SampleNode}(p^-) \)
4: end for

/* Update embedding vectors */
5: for each \( x \) in node_list
6: \( A(x) \leftarrow A(x) - \eta \nabla_{A(x)} F_M \)
7: \( B(x) \leftarrow B(x) - \eta \nabla_{B(x)} F_M \)
/* Update individual kernel weights */
8: if number of kernels \( > 1 \) then
9: \( c \leftarrow c - \eta \nabla_{c} F_M \)
10: end if
11: end if

5 EXPERIMENTS

This section presents the experimental set-up details, the datasets, and the baseline methods used in the evaluation. The performance of the proposed single and multiple kernel models is examined for node classification and link prediction tasks on various real-world datasets. The experiments have been performed on a computer with 16 Gb RAM.

5.1 Baseline Methods

We consider nine baseline models to compare the performance of our approach. (i) DeepWalk [3] performs uniform random walks to generate the contexts of nodes; then, the Skip-Gram model [10] is used to learn node embeddings. (ii) Node2Vec [4] combines Skip-Gram with biased random walks, using two extra parameters that control the walk in order to simulate a BFS or DFS exploration. In the experiments, we set those parameters to 1.0. In our approach we sample context nodes using this biased random walk strategy. (iii) LINE [64] learns nodes embeddings relying on first- and second-order proximity information of nodes. (iv) HOPE [7] is a matrix factorization approach aiming at capturing similarity patterns based on a higher-order node similarity measure. In the experiments, we consider the Katz index, since it demonstrates the best performance among other proximity indices. (v) NetMF [26] targets to factorize
the matrix approximated by pointwise mutual information of center and context pairs. The experiments have been conducted for large window sizes ($\gamma = 10$) due to its good performance. (vi) VERSE [33] learns the embedding vectors by optimizing similarities among nodes. As suggested by the authors, we set $\alpha = 0.85$ for the value of the hyper-parameter in the experiments. (vii) ProNE [34] learns the representations by relying on an efficient sparse matrix factorization and the extracted embeddings are improved with spectral propagation operations. (viii) GEMSEC [65] leverages the community structure of real-world graphs, learning node embeddings and the cluster assignments simultaneously. We have used the best performing number of cluster value from the set $\{5, 10, 15, 25, 50, 75, 100\}$. (ix) Lastly, M-NMF [66] extracts node embeddings under a modularity-based community detection framework based on non-negative matrix factorization. We have observed that the algorithm poses good performance by setting its parameters $\alpha = 0.1$ and $\beta = 5$. We performed parameter tuning for the number of communities using values from the following set: $\{5, 15, 20, 25, 50, 75, 100\}$.

Those baseline methods are compared against instances of KERNEL and MKERNEL using different kernel functions (GAUSS and SCH).

### 5.2 Datasets

In our experiments, we use eight networks of different types. To be consistent, we consider all network as undirected in all experiments, and the detailed statistics of the datasets are provided in Table 1. (i) CiteSeer [67] is a citation network obtained from the CiteSeer library. Each node of the graph corresponds to a paper, while the edges indicate reference relationships among papers. The node labels represent the subjects of the paper. (ii) Cora [68] is another citation network constructed from the publications in the machine learning area; the documents are classified into seven categories. (iii) DBLP [69] is a co-authorship graph, where an edge exists between nodes if two authors have co-authored at least one paper. The labels represent the research areas. (iv) PPI (Homo Sapiens) [4] is a protein-protein interaction network for Homo Sapiens, in which biological states are used as node labels. (v) AstroPh [70] is a collaboration network constructed from papers submitted to the ArXiv repository for the Astro Physics subject area, from January 1993 to April 2003. (vi) HepTh [70] network is constructed in a similar way from the papers submitted to ArXiv for the High Energy Physics - Theory category. (vii) Facebook [71] is a social network extracted from a survey conducted via a Facebook application. (viii) Gnutella [72] is the peer-to-peer file-sharing network constructed from the snapshot collected in August 2002 in which nodes and edges correspond to hosts and connections among them.

### 5.3 Parameter Settings

For random walk-based approaches, we set the window size ($\gamma$) to 10, the number of walks ($N$) to 80, and the walk length ($L$) to 10. We use the embedding size of $d = 128$ for each method. The instances of KERNEL and MKERNEL are fed with random walks generated by Node2Vec. For the training process, we adopt the negative sampling strategy [10] as described in Section 4.1. The negative samples are generated proportionally to its frequency raised to the power of 0.75. For our methods and the baseline models needing to generate negative node instances, we consistently sample 5 negative nodes for a fair comparison. In our experiments, the initial learning rate of stochastic gradient descent is set to 0.025; then it is decreased linearly according to the number of processed nodes until the minimum value, $10^{-4}$. For the kernel parameters, the value of $\sigma$ has been chosen as 2.0 for the single kernel version of the model (KERNEL). For MKERNEL, we considered three kernels and their parameters are set to $1.0, 2.0, 3.0$ and $1.0, 1.5, 2.0$ for MKERNEL-GAUSS and MKERNEL-SCH, respectively. As an exception for the PPI network, we employed $0.5, 1.0, 1.5$ for MKERNEL-GAUSS since it shows better performance due to the network’s structure. The regularization parameters are always set to $\lambda = 10^{-2}$ and $\beta = 0.1$.

### 5.4 Node Classification

**Experimental Setup.** In the node classification task, we have access to the labels of a certain fraction of nodes in the network (training set), and our goal is to predict the labels of the remaining nodes (test set). After learning the representation vectors for each node, we split them into varying sizes of training and test sets, ranging from 1% up to 90%. The experiments are carried out by applying an one-vs-rest logistic regression classifier with $L_2$ regularization [73]. We report the average performance of 50 runs for each representation learning method.

**Experimental Results.** Tables 2, 3, 4, and 5 report the Micro-$F_1$ and Macro-$F_1$ scores of the classification task. With boldface and underline we indicate the best and second-best performing model, respectively. As can be observed the single and multiple kernel versions of the proposed methodology outperform the baseline models, showing different characteristics depending on the graph dataset. While the Gaussian kernel comes into prominence on the Citeseer and DBlp networks, the Schoenberg kernel shows good performance for the PPI network. We further observe that leveraging multiple kernels with MKERNEL often has superior performance compared to the single kernel, especially for smaller training ratios, which corroborates the effectiveness of the data-driven multiple kernel approach. In only a few cases, Node2Vec shows comparable performances. The reason stems from the structure of the network and the used random walk sequences. Since the distribution of the node labels does not

---

**TABLE 1**

| Statistics of Networks | $|V|$ | $|E|$ | $|K|$ | $|C|$ | Avg. Degree |
|------------------------|------|------|------|------|-------------|
| CiteSeer               | 3,312| 4,660| 6    | 438  | 2.81        |
| Cora                   | 2,708| 5,278| 7    | 78   | 3.90        |
| DBLP                   | 27,199| 66,832| 115  | 2,115| 4.91        |
| PPI                    | 3,890| 38,739| 50   | 35   | 19.92       |
| AstroPh                | 17,903| 19,730| -    | 1    | 22.01       |
| HepTh                  | 8,638| 24,827| -    | 1    | 5.74        |
| Facebook               | 4,039| 88,234| -    | 1    | 43.69       |
| Gnutella               | 8,104| 26,008| -    | 1    | 6.42        |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
always follow the network structure, we observe such occasional cases due to the complexity of the datasets concerning the node classification task.

5.5 Link Prediction

For the link prediction task, we remove half of the network’s edges while retaining its connectivity, and we learn node embedding on the residual network. For networks consisting of disconnected components, we consider the giant component among them as our initial graph. The removed edges constitute the positive samples for the testing set; the same number of node pairs that do not exist in the original graph is sampled at random to form the negative samples. Then, the entries of the feature vector corresponding to each node pair \((u; v)\) in the test set are computed based on the element-wise operation \(\frac{|A(v;i)|}{C_0 A(u;i)}\), for each coordinate axis \(i\) of the embedding vectors \(A(u,:)\) and \(A(v,:)\). In the experiments, we use logistic regression with \(L_2\) regularization.

Experimental Results. Table 6 shows the Area Under Curve (AUC) scores for the link prediction task. As we can observe, both the KERNELNE and MKERNELNE models perform well across different datasets compared to the other...
## TABLE 4
Node Classification Task for Varying Training Sizes on Dblp

| Baselines          | 1%  | 2%  | 3%  | 5%  | 10% | 20% | 50% | 60% | 70% | 80% | 90% |
|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| DEEPWALK           | 0.518 | 0.500 | 0.537 | 0.526 | 0.607 | 0.607 | 0.671 | 0.661 | 0.614 | 0.627 | 0.630 |
| Node2Vec           | 0.464 | 0.496 | 0.515 | 0.527 | 0.535 | 0.540 | 0.543 | 0.547 | 0.550 | 0.551 | 0.560 |
| LINE               | 0.525 | 0.555 | 0.567 | 0.578 | 0.581 | 0.585 | 0.588 | 0.590 | 0.593 | 0.594 | 0.610 |
| HOPE               | 0.377 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.380 | 0.390 |
| VERSE              | 0.137 | 0.137 | 0.137 | 0.137 | 0.137 | 0.137 | 0.137 | 0.137 | 0.138 | 0.138 | 0.144 |
| ProNE              | 0.492 | 0.508 | 0.528 | 0.539 | 0.549 | 0.552 | 0.553 | 0.554 | 0.555 | 0.555 | 0.556 |
| NetMF              | 0.546 | 0.577 | 0.589 | 0.593 | 0.596 | 0.599 | 0.601 | 0.604 | 0.605 | 0.607 | 0.610 |
| GEMSEC             | 0.511 | 0.538 | 0.555 | 0.567 | 0.575 | 0.581 | 0.584 | 0.590 | 0.593 | 0.597 | 0.607 |
| M-NMF              | 0.435 | 0.477 | 0.492 | 0.501 | 0.508 | 0.513 | 0.516 | 0.519 | 0.521 | 0.523 | 0.531 |

For each method, the rows show the Micro-\(F^1\) and Macro-\(F^1\) scores, respectively.

## TABLE 5
Node Classification Task for Varying Training Sizes on PPI

| Baselines          | 1%  | 2%  | 3%  | 5%  | 10% | 20% | 50% | 60% | 70% | 80% | 90% |
|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| GAUS8              | 0.558 | 0.575 | 0.587 | 0.595 | 0.601 | 0.606 | 0.610 | 0.613 | 0.615 | 0.617 | 0.627 |
| RAUS8              | 0.498 | 0.517 | 0.528 | 0.536 | 0.542 | 0.545 | 0.548 | 0.551 | 0.552 | 0.554 | 0.556 |
| SCH                | 0.599 | 0.609 | 0.619 | 0.627 | 0.628 | 0.629 | 0.632 | 0.632 | 0.634 | 0.634 | 0.636 |

For each method, the rows show the Micro-\(F^1\) and Macro-\(F^1\) scores, respectively.

## TABLE 6
Area Under Curve (AUC) Scores for the Link Prediction Task

| Baselines          | KERNELNE | MKERNELNE |
|--------------------|-----------|------------|
| DEEPWALK           | 0.913 | 0.945 | 0.954 | 0.936 | 0.932 | 0.937 | 0.932 | 0.945 | 0.954 | 0.945 | 0.949 |
| Node2Vec           | 0.506 | 0.576 | 0.590 | 0.589 | 0.597 | 0.602 | 0.607 | 0.609 | 0.611 | 0.614 | 0.615 |
| LINE               | 0.097 | 0.115 | 0.136 | 0.133 | 0.138 | 0.141 | 0.147 | 0.151 | 0.153 | 0.156 | 0.158 |
| HOPE               | 0.065 | 0.078 | 0.092 | 0.091 | 0.096 | 0.100 | 0.103 | 0.106 | 0.108 | 0.110 | 0.114 |
| VERSE              | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 | 0.019 |
| ProNE              | 0.089 | 0.110 | 0.126 | 0.132 | 0.139 | 0.140 | 0.141 | 0.141 | 0.142 | 0.142 | 0.142 |
| NetMF              | 0.058 | 0.068 | 0.069 | 0.069 | 0.069 | 0.070 | 0.069 | 0.069 | 0.069 | 0.069 | 0.070 |
| GEMSEC             | 0.089 | 0.101 | 0.110 | 0.114 | 0.114 | 0.115 | 0.115 | 0.116 | 0.117 | 0.117 | 0.117 |
| M-NMF              | 0.067 | 0.068 | 0.069 | 0.069 | 0.070 | 0.071 | 0.070 | 0.070 | 0.070 | 0.070 | 0.070 |
| GAUS8              | 0.153 | 0.154 | 0.155 | 0.154 | 0.154 | 0.154 | 0.154 | 0.154 | 0.154 | 0.154 | 0.154 |
| SCH                | 0.050 | 0.059 | 0.062 | 0.064 | 0.064 | 0.065 | 0.067 | 0.067 | 0.067 | 0.068 | 0.068 |
| SCH                | 0.085 | 0.097 | 0.110 | 0.112 | 0.115 | 0.115 | 0.116 | 0.116 | 0.116 | 0.116 | 0.116 |

For each method, the rows show the Micro-\(F^1\) and Macro-\(F^1\) scores, respectively.

---

For each method, the rows show the Micro-\(F^1\) and Macro-\(F^1\) scores, respectively.
baseline models. Comparing now at the AUC score of KERNELNE and MKERNELNE, we observe that both models show quite similar behavior. In the case of the single kernel model KERNELNE, the Schoenberg kernel (SCH) performs significantly better than the Gaussian one. On the contrary, in the MKERNELNE model both kernels achieve similar performance. These results are consistent to the behavior observed in the node classification task.

5.6 Parameter Sensitivity

In this subsection, we examine how the performance of the proposed models is affected by the choice of parameters.

The Effect of Dimension Size. The dimension size $d$ is a critical parameter for node representation learning approaches since the desired properties of networks are aimed to be preserved in a lower-dimensional space. Fig. 2 depicts the Micro-$F_1$ score of the proposed models for varying embedding dimension sizes, ranging from $d = 32$ up to $d = 224$ over the Citeseer network. As it can be seen, all the different node embedding instances, both single and multiple kernel ones, have the same tendency with respect to $d$; the performance increases proportionally to the size of the embedding vectors. Focusing on 10% training ratio, we observe that the performance gain almost stabilizes for embedding sizes greater than 96.

The Effect of Kernel Parameters. We have further studied the behavior of the kernel parameter $\sigma$ of the Gaussian and Schoenberg kernel respectively (as described in Section 4.1). Fig. 3 shows how the Micro-$F_1$ node classification score is affected with respect to $\sigma$ for various training ratios on the Citeseer network. For the Gaussian kernel, we observe that the performance is almost stable for $\sigma$ values varying from 0.25 up to 2.0, showing a sudden decrease after 4.0. For the case of the Schoenberg kernel, we observe an almost opposite behavior. Recall that, parameter $\sigma$ has different impact on these kernels (Section 4.1). In particular, the Micro-$F_1$ score increases for $\sigma$ values ranging from 0.25 to 0.50, while the performance almost stabilizes in the range of 4.0 to 8.0.

5.7 Running Time Comparison

For running time comparison of the different models, we considered all the networks employed in the experiments. Additionally, we have generated an Erdős-Rényi $G_{n,p}$ graph model [74], by setting the number of nodes $n = 10^4$ and the edge probability $p = 10^{-4}$. Table 7 reports the running time (in seconds) for both the baseline and the proposed models. For this particular experiment, we have run all the models on a machine of 64 GB memory with a single thread when it is possible. The symbol "-" signifies that the corresponding method is unable to run due to excessive memory requirements or it takes more than one day to complete. As we can observe, both the proposed models have comparable running time—utilizing multiple kernels with MKERNELNE improves performance on downstream tasks without heavily affecting efficiency. Besides, KERNELNE runs

**TABLE 7**

Comparison of Running Time (in seconds) on an Erdős-Rényi Random Graph and the Real-World Networks Used in the Experiments

|          | DEEPWALK | NODE2VEC | LINE | HOPE | VERSE | PRONE | NETMF | GEMSEC | M-NMF | KERNELNE | MKERNELNE |
|----------|----------|----------|------|------|-------|-------|-------|--------|-------|-----------|-----------|
| Citeseer | 136      | 42       | 2,020| 26   | 294   | 5     | 34    | 2,012  | 530   | 93        | 186       |
| Cora     | 139      | 40       | 2,069| 27   | 256   | 5     | 22    | 2,064  | 392   | 87        | 170       |
| DBlp     | 1,761    | 335      | 2,717| 216  | 5,582 | 15    | 324   | 113,432| 35,988| 1,079     | 1,942     |
| PPI      | 316      | 248      | 3,532| 45   | 558   | 4     | 14    | 4,320  | 871   | 133       | 256       |
| AstroPh  | 1,274    | 318      | 3,152| 147  | 3,855 | 13    | 135   | 46,768 | 16,875| 555       | 1,055     |
| HepTh    | 804      | 128      | 3,866| 77   | 1,264 | 5     | 84    | 15,838 | 3,776 | 346       | 625       |
| Facebook | 217      | 75       | 2,411| 32   | 525   | 4     | 10    | 3,353  | 924   | 114       | 226       |
| Gnutella | 890      | 127      | 3,850| 82   | 1,130 | 5     | 69    | 11,907 | 3,523 | 330       | 575       |
| Erdős-Rényi | 9,525   | 1,684    | 3,449| 1,163| 25,341| 94    | 1,089 | -      | -     | 3,845     | 6,486     |

The symbol "-" denotes that the corresponding method is unable to run due to excessive memory requirements or it takes more than one day to complete.
faster than LINE in most cases, while MKERNELNE with two kernels shows also a comparable performance. It is also important to note that although matrix factorization-based models such as M-NMF are well-performing in some tasks, they are not very efficient because of the excessive memory demands (please see Section 2 for more details). On the contrary, our kernelized matrix factorization models are more efficient and scalable by leveraging negative sampling. The single kernel variant also runs faster than the random walk method DeepWalk.

In addition to the running time comparison of the proposed approach against the baseline models, we further examine the running time of KERNELNE and MKERNELNE (using three base kernels) over Erdős-Rényi graphs of varying sizes, ranging from $2^8$ to $2^{13}$ nodes. In the generation of random graphs, we set the edge probabilities so that the expected node degree of graphs is 10. Fig. 4 shows the running time of the proposed models. Since the Gaussian and Schoenberg kernels have similar performance, we report the running time for the Gaussian kernel only. As we observe, considering multiple kernels does not significantly affect the scalability properties of the MKERNELNE model.

5.8 Visualization and Clustering of Embedding Vectors

Modularity is a measure designed to assess the quality of the clustering structure of a graph [74]. High modularity implies good clustering structure—the network consists of substructures in which nodes densely connected with each other. Here, we perform a simple visualization and clustering experiment to examine the ability of the different node embedding models to capture the underlying community structure, preserving network’s modularity in the embedding space. Note that, to keep the settings of the experiment simple, we leverage the raw embedding vectors visualizing them in the two-dimensional space (instead of performing visualization with t-SNE [75] or similar algorithms).

We perform experiments on the Dolphins [76] toy network, which contains 62 nodes and 159 edges. We use the Louvain algorithm [77] to detect the communities in the network. Each of these detected five communities is represented with a different color in Fig. 5a. We also use the proposed and the baseline models to learn node embeddings in two-dimensional space.

As we can observe in Fig. 5, different instances of MKERNELNE learn embeddings in which nodes of the different communities are better distributed in the two-dimensional space. Besides, to further support this observation, we run the k-means clustering algorithm [73] on the embedding vectors, computing the corresponding normalized mutual information (NMI) scores [78], assuming as ground-truth communities the ones computed by the Louvain algorithm in the graph space. While the NMI scores for NODE2VEC and NETMF are 0.532 and 0.572 respectively, KERNELNE-SCH achieves 0.511 while for KERNELNE-GAUSS we have 0.607. The NMI scores significantly increase for proposed multiple kernel models MKERNELNE-SCH and MKERNELNE-GAUSS, which are 0.684 and 0.740 respectively.

6 CONCLUSION

In this paper, we have studied the problem of learning node embeddings with kernel functions. We have first introduced KERNELNE, a model that aims at interpreting random-walk based node proximity under a weighted matrix factorization framework, allowing to utilize kernel functions. To further boost performance, we have introduced MKERNELNE, extending the proposed methodology to the multiple kernel learning framework. Besides, we have discussed how parameters of both models can be optimized via negative sampling in an efficient manner. Extensive experimental evaluation showed that the proposed kernelized models substantially outperform baseline NRL methods in node classification and link prediction tasks.
The proposed kernelized matrix factorization opens further research directions in network representation learning that we aim to explore in future work. To incorporate the sparsity property prominent in real-world networks, a probabilistic interpretation [79] of the proposed matrix factorization mode would be suitable. Besides, it would be interesting to examine how the proposed models could be extended in the case of dynamic networks.

**References**

[1] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” IEEE Data Eng. Bull., vol. 40, no. 3, pp. 52–74, 2017.

[2] D. Zhou, J. Yin, X. Zhu, and C. Zhang, “Network representation learning: A survey,” IEEE Trans. Big Data, vol. 6, no. 1, pp. 3–28, Mar. 2020.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, “Node2Vec: Scalable feature learning for networks,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 855–864.

[5] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in Proc. 14th Int. Conf. Neural Inf. Process. Syst., 2001, pp. 585–591.

[6] S. Cao, W. Lu, and Q. Xu, “GraRep: Learning graph representations with global structural information,” in Proc. 24th ACM Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1549–1557.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 1105–1114.

[8] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, “Adaptive diffusions for scalable learning over graphs,” IEEE Trans. Signal Process., vol. 67, no. 5, pp. 1307–1321, Mar. 2019.

[9] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” Science, vol. 290, no. 5500, pp. 2323–2326, 2000.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J.Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. 26th Int. Conf. Neural Inf. Process. Syst., 2013, pp. 3111–3119.

[11] D. Nguyen and C. F. Malliaros, “BiasedWalk: Biased sampling for representation learning on graphs,” in Proc. IEEE Int. Conf. Big Data, 2018, pp. 4045–4053.

[12] A. Çelikkanat and F. D. Malliaros, “TNE: A latent model for representation learning on networks,” in Proc. NeurIPS Relational Representation Learn. Workshop, 2018.

[13] A. Çelikkanat and F. D. Malliaros, “Exponential family graph embeddings,” in Proc. 34th AAAI Conf. Artif. Intell., 2020, pp. 891–900.

[14] S. Chauanpuriya and C. Musco, “InfiniteWalk: Deep network embeddings as laplacian embeddings with a nonlinearity,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 1325–1333.

[15] B. Schölkopf, A. Smola, and K.-R. Müller, “Kernel principal component analysis,” in Proc. Int. Conf. Artif. Neural Nets., 1997, pp. 583–588.

[16] B. Schölkopf and A. J. Smola, Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2001.

[17] I. S. Dhillon, Y. Guan, and B. Kulis, “Kernel k-means: Spectral clustering and normalized cuts,” in Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2004, pp. 551–556.

[18] X. Liu, C. Aggarwal, Y.-F. Li, X. Kong, X. Sun, and S. Sathe, “Kernelized matrix factorization for collaborative filtering,” in Proc. SIAM Int. Conf. Data Mining, 2016, pp. 378–386.

[19] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” Ann. Stat., vol. 36, no. 3, pp. 1171–1200, 2008.

[20] M. Gönen and E. Alpaydın, “Multiple kernel learning methods,” J. Mach. Learn. Res., vol. 12, pp. 2211–2268, Jul. 2011.

[21] S. Wang, Q. Huang, S. Jiang, and Q. Tian, “SMKL: Scalable semi-supervised multiple kernel learning for real-world image applications,” IEEE Trans. Multimedia, vol. 14, pp. 1259–1274, 2012.

[22] N. M. Kriege, F. D. Johansson, and C. Morris, “A survey on graph kernels,” Appl. Netw. Sci., vol. 5, no. 1, 2020, Art. no. 6.
[58] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” Psychometrika, vol. 1, pp. 211–218, 1936.

[59] I. Steinwart, “On the influence of the kernel on the consistency of support vector machines,” J. Mach. Learn. Res., vol. 2, pp. 67–93, 2001.

[60] P. Honeine and C. Richard, “Preimage problem in kernel-based machine learning,” IEEE Signal Process. Mag., vol. 28, no. 2, pp. 77–88, Mar. 2011.

[61] C. A. Micchelli, Y. Xu, and H. Zhang, “Universal kernels,” J. Mach. Learn. Res., vol. 7, pp. 2651–2667, Dec. 2006.

[62] L. Bottou, “Stochastic gradient learning in neural networks,” in Proc. Neuro-Nimes. EC2, 1991.

[63] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, “The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations,” Behav. Ecol. Sociobiol., vol. 54, no. 4, pp. 396–405, 2003.

[64] J. Leskovec and J. Mcauley, “Learning to discover social circles in ego networks,” in Proc. 25th Int. Conf. Intell. Syst. Web Conf. (ISWC), 2016, pp. 77–88, Mar. 2011.

[65] J. Leskovec, J. Kleinberg, and C. Faloutsos, “Graph evolution: Densification and shrinking diameters,” ACM Trans. Knowl. Discov. Data, vol. 1, no. 1, pp. 2–es, 2007.

[66] J. Leskovec and J. J. Mcauley, “Learning to discover social circles in ego networks,” in Proc. 25th Int. Conf. Neural Inf. Process. Syst., 2012, pp. 539–547.

[67] M. Ripeanu, A. Iamnitchi, and I. Foster, “Mapping the gnutella network,” IEEE Internet Comput., vol. 6, no. 1, pp. 50–57, Jan./Feb. 2002.

[68] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Learning from social interactions in large graphs,” in Proc. 20th Int. Conf. Neural Inf. Process. Syst., 2013, pp. 1737–1745.

[69] Y. Lin, T. Liu, and C. Fuh, “Multiple kernel learning for dimensionality reduction,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 6, pp. 1147–1160, Jun. 2011.

[70] S. An, J. Yun, and S. Choi, “Multiple kernel nonnegative matrix factorization,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2011, pp. 1976–1979.

[71] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, “Don’t walk, skip!: Hierarchical learning of network community discovery,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2017, pp. 4129–4133.

[72] M. Ripeanu, A. Iamnitchi, and I. Foster, “Mapping the gnutella network: A survey,” Phys. Rep., vol. 533, 2013.

[73] A. Mnih and R. Salakhutdinov, “Probabilistic matrix factorization,” in Proc. 28th Int. Conf. Neural Inf. Process. Syst., 2006, pp. 1277–1284.

[74] M. Newman, “Modularity and community structure in networks,” Proc. Natl. Acad. Sci. USA, vol. 103, no. 23, pp. 8577–8582, 2006.

[75] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 86, pp. 2579–2605, 2008.

[76] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, “Dynamic network embeddings: From random walks to temporal random walks,” in Proc. IEEE Int. Conf. Big Data, 2018, pp. 1085–1092.

[77] J. Leskovec and J. Kleinberg, “Graphs over time: Densification laws, shrinking diameters and possible explanations,” in Proc. 19th Int. Conf. World Wide Web Conf., 2010, pp. 651–660.

[78] M. Newman, “The structure and function of complex networks,” SIAM Rev., vol. 45, no. 2, pp. 167–256, Jun. 2003.

[79] N. Srebro and T. Jaakkola, “Weighted low-rank approximations,” in Proc. 20th Int. Conf. Neural Inf. Process. Syst., 2007, pp. 720–727.

[80] A. Salim, S. Shiju, and S. Sumitra, “Design of multi-view graph embedding using multiple kernel learning,” Eng. Appl. Artif. Intell., vol. 90, 2020, Art. no. 103534.

[81] T. Wang, L. Zhang, and W. Hu, “Bridging deep and multiple kernel learning: A review,” Inf. Fusion, vol. 67, pp. 3–13, 2021.

[82] R. Wang, W. Li, and J. Kittler, “Graph embedding multiple-kernel metric learning for image set classification with grassmanian manifold-featured,” IEEE Trans. Multimedia, vol. 23, pp. 228–242, 2021.

[83] S. An, J. Yun, and S. Choi, “Multiple kernel nonnegative matrix factorization,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2013, pp. 2288–2291.

[84] T. Wang, L. Zhang, and W. Hu, “Bridging deep and multiple kernel learning: A review,” Inf. Fusion, vol. 67, pp. 3–13, 2021.

[85] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2831, 2011.

[86] M. Newman, Networks: An Introduction. Oxford, U.K.: Oxford Scholarship Online, 2010.

[87] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, no. 86, pp. 2579–2605, 2008.

[88] D. Lusseau, C. Schneider, O. J. Boisseau, K. Haase, P. Haarle, E. Slooten, and S. M. Dawson, “The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations,” Behav. Ecol. Sociobiol., vol. 54, no. 4, pp. 396–405, 2003.

[89] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” J. Statist. Mech., vol. 2008, no. 10, 2008, Art. no. P10008.

[90] F. D. Malliaros and M. Vazirgiannis, “Community and community detection in directed networks: A survey,” Phys. Rep., vol. 533, 2013.

[91] A. Mnih and R. Salakhutdinov, “Probabilistic matrix factorization,” in Proc. 28th Int. Conf. Neural Inf. Process. Syst., 2006, pp. 1277–1284.