Kernel Node Embeddings

Abdulkadir Çelikkanat, Fragkiskos Malliaros

Kernel Node Embeddings

Abdulkadir Çelikkanat
CentraleSupélec and Inria Saclay
University of Paris-Saclay
Gif-Sur-Yvette, France
Email: abdulkadir.celikkanat@centralesupelec.fr

Fragkiskos D. Malliaros
CentraleSupélec and Inria Saclay
University of Paris-Saclay
Gif-Sur-Yvette, France
Email: fragkiskos.malliaros@centralesupelec.fr

***Abstract*—Learning representations of nodes in a low dimensional space is a crucial task with many interesting applications in network analysis, including link prediction and node classification. Two popular approaches for this problem include *matrix factorization* and *random walk*-based models. In this paper, we aim to bring together the best of both worlds, towards learning latent node representations. In particular, we propose a weighted matrix factorization model which encodes random walk-based information about the nodes of the graph. The main benefit of this formulation is that it allows to utilize kernel functions on the computation of the embeddings. We perform an empirical evaluation on real-world networks, showing that the proposed model outperforms baseline node embedding algorithms in two downstream machine learning tasks.**

***Index Terms*—Network representation learning, node embedding, link prediction, node classification, kernel functions**

I. Introduction

With the advancements in data production, storage and consumption, networks are becoming omnipresent; data from diverse disciplines can be represented as graph structures with prominent examples here being various social, information, technological and biological networks. Developing machine learning algorithms to analyze, predict and make sense of the structure 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 its structural properties into the learning model. To this direction, a recent paradigm in network analysis, known as *network representation learning* (NRL), aims at embedding the nodes of the graph into a lower-dimensional space, in such a way that similarity among nodes in the graph is captured by the similarity of the embeddings in the latent space [3], [4], [5], [6], [7], [8]. Many of the proposed models in network representation learning have mostly concentrated on computing node embeddings relying on matrix factorization techniques that encode information about structural node similarity [5], [6], [7]. Nevertheless, the majority of those approaches are not efficient for large scale networks, mainly due to the high computational cost required to perform matrix factorization [1], [2].

Being inspired by the field of natural language processing [9], random-walk based models have gained considerable attention [3], [4], [10], [11]. Typically, these methods first generate a set of node sequences (i.e., *context* nodes) for every node (i.e., *center*) in the network, based on some random walk strategy; then, node representations are learned by predicting context-center node co-occurrences within the random walks.

In this paper, we aim at combining the previously proposed broad modeling approaches for NRL – namely matrix factorization and random walks. In particular, we focus on modeling the interactions between nodes based on random walks, under a weighted matrix factorization framework. The potential advantage of such a modeling approach is that it allows to take advantage of and combine the elegant mathematical formulation that matrix factorization can offer with the expressive power of random walks to capture a notion of "stochastic" node similarity in an efficient way. More importantly, this formulation allows us to utilize *kernel* functions in the node representation learning task.

Kernel functions have mostly been introduced along with popular learning algorithms, such as *PCA* [12], *SVMs* [13], *Spectral Clustering* [14] and *Collaborative Filtering* [15]. The idea is to map 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. Here, we aim at obtaining embeddings, given values that represent the relationships among nodes. Because of the nature of matrix factorization-based methods, these values are viewed as an inner product of vectors lying on a latent space, which allows us to utilize kernels interpreting the embeddings in a higher dimensional feature space using non-linear maps. The main contributions of the paper are the following:

- We propose 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 extensively evaluate the performance of the proposed method in the downstream tasks of node classification and link prediction and we show that the model generally outperforms the well-known baseline methods on various network datasets.

Notation. We use the notation $\mathbf{M}$ to denote a matrix, $\mathbf{M}_{i,j}$ points out the entry located at the i'th row and j'th column of the matrix, and $\mathbf{M}_{i,:}$ indicates the i'th row of the matrix.

Source code. The C++ implementation of the proposed methodology and the networks used in the study, can be reached at: https://abdcelikkanat.github.io/projects/kernelNE/.

II. Modeling and Problem Formulation

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph where $\mathcal{V} = \{1, ..., n\}$ 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 [16], as follows:

$$\arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \left\| \mathbf{W} \odot (\mathbf{M} - \mathbf{A}\mathbf{B}^\top) \right\|_F^2, \tag{1}$$

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 $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times d}$. $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the weight matrix in which each element $\mathbf{W}_{v,u}$ captures the importance of the approximation error between nodes v and u, and $\odot$ indicates the *Hadamard* product. Depending on the desired graph properties that we are interested to encode, 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 based on k-hop information [6]. Here, we will design $\mathbf{M}$ as a sparse binary matrix utilizing information of random walks over the network. Note that, matrices $\mathbf{M}$ and $\mathbf{W}$ do not need to be symmetric.

Random walk-based node embedding models [3], [4], [10], [17], [18], [19] 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 $\boldsymbol{w} = (w_1, ..., w_\ell)$, the context of the *center* node $w_l \in \mathcal{V}$ at position l in the walk $\boldsymbol{w}$ is defined as $\mathcal{C}_{\boldsymbol{w}}(w_l) := (w_{l-\gamma}, ..., w_{l-1}, w_{l+1}, ..., w_{l+\gamma})$, where γ is called the *window size* and it denotes the furthest distance between the *center* and *context* nodes $w_k \in \mathcal{V}$ for $l - \gamma \leq k \leq l + \gamma$ and $k \neq l$. The embedding vectors are then obtained by maximizing the likelihood of occurrences of nodes within the context of given center nodes. Here, we will also follow a similar random walk strategy, formulating the problem under a matrix factorization framework.

Let $\mathbf{M}_{v,u}$ be a binary value representing if node u appears in the context of v in any walk. Also, let $\mathbf{F}_{v,u}$ be the number of occurrences of node u in the contexts of v in the generated walks. Setting each term $\mathbf{W}_{v,u}$ as the square root of $\mathbf{F}_{v,u}$, the objective function in (1) can be expressed under a random walk-based formulation as follows:

$$\begin{aligned}
&\arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \left\| \sqrt{\mathbf{F}} \odot \left(\mathbf{M} - \mathbf{A}\mathbf{B}^\top\right) \right\|_F^2 \\
&= \arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{v \in \mathcal{V}} \sum_{u \in \mathcal{V}} \mathbf{F}_{v,u} \Big(\mathbf{M}_{v,u} - \langle \mathbf{A}_{v,:}, \mathbf{B}_{u,:} \rangle \Big)^2 \\
&= \arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{\boldsymbol{w} \in \mathcal{W}} \sum_{w_l \in \boldsymbol{w}} \sum_{u \in \mathcal{V}} \Big(\mathbf{M}^{\mathbf{w}}_{w_l,u} - \langle \mathbf{A}_{w_l,:}, \mathbf{B}_{u,:} \rangle \Big)^2,
\end{aligned} \tag{2}$$

where each $\boldsymbol{w} \in \mathcal{V}^\ell$ indicates a random walk of length ℓ in the collection $\mathcal{W}$ and $\mathbf{M}^{\mathbf{w}}_{w_l,u}$ represents the occurrence of u in the context $\mathcal{C}_{\mathbf{w}}(w_l)$. Matrix $\mathbf{A}$ in Eq. (2), contains 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 $\mathbf{M}$ and the reformulation of the objective function as stated above, offers a computational advantage during the optimization step. More importantly, as we will present in the next section, we can further benefit from a *kernelized* version of the objective function.

III. Kernel-based Representation Learning

Similar to other matrix factorization techniques that aim at finding latent representations in a lower dimensional space ($d \ll n$) (e.g., [20], [21], [22]), one can adopt *Singular Value Decomposition* (SVD) to provide the best approximation of the objective function in (1), as long as the weight matrix is uniform [23]. It is also implicitly assumed that every element of the target matrix $\mathbf{M}$ can be written as inner product of vectors in the latent space, and in that case, it becomes difficult to obtain an exact low-rank decomposition. To overcome this limitation, in our approach we utilize 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 $\Phi : \mathsf{X} \to \mathbb{H}$, $\mathbb{H}$ is referred to as *feature space* and every feature map defines a *kernel* $\kappa : \mathsf{X} \times \mathsf{X} \to \mathbb{R}$ as follows:

$$\kappa(x, y) := \langle \Phi(x), \Phi(y) \rangle \qquad \forall (x, y) \in \mathsf{X}^2.$$

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

A function $g : X \to \mathbb{R}$ is called *induced by* κ, if there exists $h \in \mathbb{H}$ such that $g = \langle h, \Phi(\cdot) \rangle$, for a feature vector Φ of kernel κ (note that, the definition is independent of the feature map Φ and space $\mathbb{H}$) [24]. Let $\mathcal{I}_\kappa := \{g : \mathsf{X} \to \mathbb{R} \mid \exists h \in \mathbb{H} \text{ s.t. } g = \langle h, \Phi(\cdot) \rangle\}$ be the set of induced functions by kernel κ. Then, a continuous kernel κ on a compact metric space $(\mathsf{X}, d_{\mathsf{X}})$ is called *universal*, if the set $\mathcal{I}_\kappa$ is dense in $\mathcal{C}(\mathsf{X})$. In other words, for any function $f \in \mathcal{C}(\mathsf{X})$ and $\epsilon > 0$, there exists $g_h \in \mathcal{I}_\kappa$ satisfying

$$\|f - g_h\|_\infty \leq \epsilon,$$

where g_h is defined as $\langle h, \Phi(\cdot) \rangle$ for some $h \in \mathbb{H}$. We use the next proposition as the basis for our approach.

Proposition 1 ([24]). *Let (X, d) be a compact metric space and $\kappa(\cdot, \cdot)$ be a universal kernel on X. Then, for all compact and mutually disjoint subsets $\mathsf{K}_1, ..., \mathsf{K}_n \subset \mathsf{X}$, all $\alpha_1, ..., \alpha_n \in \mathbb{R}$, and all $\epsilon > 0$, there exists a function g induced by κ with $\|g\|_\infty \leq \max_i |\alpha_i| + \epsilon$ such that*

$$\left\| g_{|K} - \sum_{i=1}^{n} \alpha_i \mathbf{1}_{K_i} \right\|_\infty \leq \epsilon,$$

where $K := \bigcup_{i=1}^{n} K_i$ *and* $g_{|K}$ *is the restriction of* g *to* K.

The universality property of a kernel helps us in finding the decomposition of matrix $\mathbf{M}$ in the feature space. Following Proposition (1), for each row of $\mathbf{M}$, we can always find $h \in \mathbb{H}$ to approximate the row values in a higher dimensional inner product space. We can choose node representations from the disjoint subsets, but note that, each element $h \in \mathbb{H}$ does not have to be in the image of the feature map.

Based on the above, we move the inner product from space X to the feature space $\mathbb{H}$, by reformulating Eq. (2) as follows:

$$\begin{aligned}&\arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{\boldsymbol{w}\in\mathcal{W}} \sum_{w_l\in\boldsymbol{w}} \sum_{u\in\mathcal{V}} \Big(\mathbf{M}^{\mathbf{w}}_{w_l,u} - \langle \Phi(\mathbf{A}_{w_l,:}), \Phi(\mathbf{B}_{u,:})\rangle\Big)^2 \\ =&\arg\min_{\mathbf{A},\mathbf{B}} \frac{1}{2} \sum_{\boldsymbol{w}\in\mathcal{W}} \sum_{w_l\in\boldsymbol{w}} \sum_{u\in\mathcal{V}} \Big(\mathbf{M}^{\mathbf{w}}_{w_l,u} - \kappa(\mathbf{A}_{w_l,:}, \mathbf{B}_{u,:})\Big)^2. \quad (3)\end{aligned}$$

That 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 [25], [24]:

$$\kappa_G(x,y) = \exp\left(\frac{-\|x-y\|^2}{\sigma^2}\right) \qquad \sigma \in \mathbb{R}$$

$$\kappa_S(x,y) = \frac{1}{\Big(1+\|x-y\|^2\Big)^{\alpha}} \qquad \alpha \in \mathbb{R}_+$$

where κ_G and κ_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).

Model Optimization. For the optimization step, we employ *Stochastic Gradient Descent* (SGD) [26]. Note that, Eq. (3) can be divided into two parts with respect to the values of $\mathbf{M}^{\mathbf{w}}_{v,u} \in \{0,1\}$. That way, we apply *negative sampling* [9] which is a variant of *noise-contrastive estimation* [27], proposed as an alternative to solve the computational problem of hierarchical softmax. For each context node $u^+ \in \mathcal{C}_{\mathbf{w}}(w_l)$, we sample k negative instances u^- from the noise distribution p^-:

$$\Big(1-\kappa(\mathbf{A}_{v,:}, \mathbf{B}_{u^+,:})\Big)^2 + \sum_{u^-\sim p^-} \Big(\kappa(\mathbf{A}_{v,:}, \mathbf{B}_{u^-,:})\Big)^2.$$

Each sample is generated proportionally to its frequency raised to the power of 0.75 and the number of negative instances is chosen as 5. In our experiments, we set the initial learning rate of SGD to 0.025; then it decreases linearly according to the number of processed nodes. The dimension of the embedding vectors is selected as $d = 128$ and the window size for the random walks as $\gamma = 10$.

IV. NUMERICAL TESTS

We evaluate the performance of our approach on the node classification and link prediction tasks. The experiments have been performed on a server with 60Gb RAM. Table I gives the statistics of the network datasets used in the experiments (all the networks are considered as undirected).

TABLE I
STATISTICS OF NETWORKS USED IN THE EXPERIMENTS. $|\mathcal{V}|$: NUMBER OF NODES, $|\mathcal{E}|$: NUMBER OF EDGES, $|\mathcal{K}|$: NUMBER OF LABELS AND $|\mathcal{C}|$: NUMBER OF CONNECTED COMPONENTS.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{C}\rvert$	**Avg. Degree**	**Type**
CiteSeer [28]	3,312	4,660	6	438	2.814	Citation
Cora [29]	2,708	5,278	7	78	3.898	Citation
DBLP [30]	27,199	66,832	4	2,115	4.914	Co-authorship
AstroPh [31]	17,903	19,7031	-	1	22.010	Collaboration
HepTh [31]	8,638	24,827	-	1	5.7483	Collaboration
Facebook [32]	4,039	88,234	-	1	43.6910	Social
Gnutella [31]	8,104	26,008	-	1	6.4186	Peer-to-peer

A. Baseline Methods

We consider five widely used baseline models to compare the performance of our approach. DEEPWALK [3] performs uniform random walks to generate the context of a node; then, the Skip-Gram model is used to learn node representations. NODE2VEC [4] combines Skip-Gram with biased random walks, using two extra parameters that control the walk to simulate a *BFS* or *DFS* exploration. In the experiments, we set those parameters to 1.0, the number of walks to 80 and walk length to 10. In our approach, we sample context nodes using NODE2VEC's random walk strategy. LINE [21] learns embeddings relying on first-order and second-order proximity information of nodes. HOPE [7] is a matrix factorization approach aiming at capturing higher-order node similarity patterns based on the *Katz* index. Lastly, NETMF [22] targets to factorize the matrix approximated by the pointwise mutual information of center and context pairs. Those methods are compared against the KERNELNE-GAUSS and KERNELNE-SCH models.

B. Node Classification

Experimental set-up. 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). In the experiments, we learn embeddings on varying sizes of training data, ranging from 1% up to 90%. The experiments have been carried out by applying an one-vs-rest logistic regression classifier with L_2 regularization; the average scores of 50 experiments are reported.

Experimental results. Table II shows the Micro-F_1 scores for each network. For the *CiteSeer* network, KERNELNE outperforms the baselines for all training sizes. The SCH kernel with $\alpha = 1$ gives gain of up to 7.0% against the best baseline model, while KERNELNE-GAUSS with $\sigma^2 = 2$ has the best performance for larger training sizes. For the *Cora* network, the GAUSS kernel with $\sigma^2 = 2$ also performs quite well especially for small training ratios (gain 0.91% up to 7.60%. Lastly, in the *DBLP* network, we choose $\sigma = 0.3$ for the GAUSS kernel, which is the best performing model. The SCH

TABLE II
MICRO-F_1 SCORES FOR THE NODE CLASSIFICATION TASK.

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DEEPWALK	0.416	0.460	0.489	0.505	0.517	0.566	0.584	0.595	0.592
NODE2VEC	0.450	0.491	0.517	0.530	0.541	0.585	0.597	0.601	0.599
LINE	0.323	0.387	0.423	0.451	0.466	0.532	0.551	0.560	0.564
HOPE	0.196	0.205	0.210	0.204	0.219	0.256	0.277	0.299	0.320
NETMF	0.451	0.496	0.526	0.540	0.552	0.590	0.603	0.604	0.608
GAUSS	0.479	0.514	0.535	0.548	0.560	**0.603**	**0.615**	**0.623**	**0.630**
SCH	**0.482**	**0.519**	**0.538**	**0.552**	**0.561**	0.599	0.613	0.620	0.627

(a) *CiteSeer*

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DEEPWALK	0.621	0.689	0.715	0.732	0.747	0.802	0.819	0.826	0.833
NODE2VEC	0.656	0.714	0.743	0.757	0.769	0.815	0.831	0.839	0.841
LINE	0.450	0.544	0.590	0.633	0.661	0.746	0.765	0.774	0.775
HOPE	0.277	0.302	0.299	0.302	0.302	0.301	0.302	0.303	0.302
NETMF	0.636	0.716	0.748	0.767	0.773	**0.821**	**0.834**	**0.841**	**0.844**
GAUSS	**0.706**	**0.746**	**0.761**	**0.774**	**0.782**	0.815	0.830	0.837	0.842
SCH	0.693	0.733	0.753	0.761	0.769	0.799	0.810	0.819	0.824

(b) *Cora*

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DEEPWALK	0.545	0.585	0.600	0.608	0.613	0.626	0.628	0.628	0.633
NODE2VEC	0.575	0.600	0.611	0.619	0.622	0.636	0.638	0.639	0.639
LINE	0.554	0.580	0.590	0.597	0.603	0.618	0.621	0.623	0.623
HOPE	0.379	0.378	0.379	0.379	0.379	0.379	0.379	0.378	0.380
NETMF	0.577	0.589	0.596	0.601	0.605	0.617	0.620	0.623	0.623
GAUSS	**0.611**	**0.621**	**0.626**	**0.628**	**0.630**	**0.637**	**0.641**	**0.642**	**0.644**
SCH	0.610	0.616	0.622	0.624	0.625	0.633	0.636	0.637	0.638

(c) *DBLP*

TABLE III
AREA UNDER CURVE (AUC) SCORES FOR THE LINK PREDICTION TASK.

	DEEPWALK	NODE2VEC	LINE	HOPE	NETMF	GAUSS	SCH
CiteSeecer	0.837	0.762	0.557	0.756	0.742	**0.886**	0.875
Cora	0.778	0.724	0.554	0.728	0.755	**0.819**	0.814
DBLP	0.944	0.905	0.590	0.930	0.930	**0.963**	0.958
AstroPh	0.960	0.935	0.679	0.967	0.897	**0.978**	0.970
HepTh	0.897	0.830	0.633	0.875	0.882	**0.920**	0.915
Facebook	0.983	**0.988**	0.696	0.980	0.987	0.987	0.987
Gnutella	0.680	0.498	0.702	0.599	0.651	**0.766**	0.677

kernel with $\alpha = 3.0$, also performs better than the baselines, especially for small training sizes.

C. Link Prediction

Experimental set-up. For the link prediction task, we remove half of the edges of the network in order to obtain positive samples for the test set; the same number of node pairs, not existing in the initial graph, are added to the test set. We then learn node embedding using the residual network; the feature vector of an edge (v, u) is formed with the operation $|x_i^v - y_i^u|^2$ for each coordinate i of the embedding vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ corresponding to nodes v and u. In the experiments, we use logistic regression with L_2 regularization.

Experimental results. Table III shows the *area under curve* (AUC) scores for the link prediction task. We choose kernel parameters as $\sigma = 0.3$ and $\alpha = 2$ except the *Gnutella* network in which $\sigma = 3.0$. In all cases, the largest connected component of the datasets is used. As we can observe, in almost all cases, the proposed kernel-based models outperform the baselines. The only exception is the *Facebook* dataset, where NODE2VEC is just slightly better than KERNELNE.

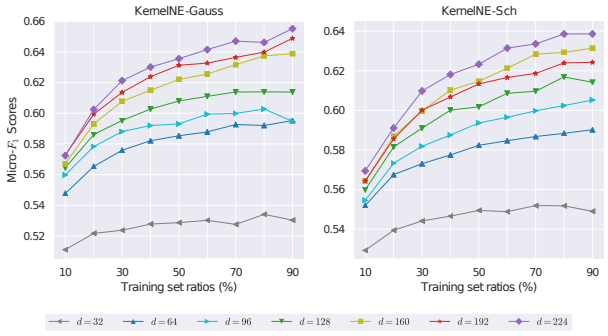

Fig. 1. Influence of the dimension size d on the *CiteSeer* network.

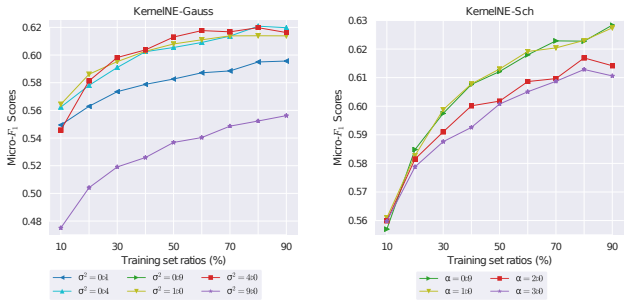

Fig. 2. Influence of kernel parameters on the *CiteSeer* network.

D. Parameter Sensitivity

The effect of dimension size. Figure 1 shows the Micro-F_1 scores of the proposed models for varying embedding dimension sizes, ranging from $d = 32$ up to $d = 224$. As it can be seen, both of the kernel instances have the same tendency, where the performance increases proportionally to the size of the embedding vectors.

The effect of kernel parameters. In Figure 2, we study the behaviour of kernel functions with respect to the chosen parameters. The GAUSS kernel shows comparable results for values of σ^2 between 0.4 and 4.0. In addition, we observed that its performance is limited for very big or very small values of this parameter. The SCH kernel also behaves similarly.We have reached the highest score with parameter values around $\alpha = 1.0$. Lastly, we observed poor performance for very small values of α, which are not included in the figure.

V. CONCLUSION

We have introduced the KERNELNE model for learning node embeddings. We interpret our random-walk based method under a weighted matrix factorization framework, which is then generalized to kernel functions. The numerical evaluation showed that the proposed kernel-based models substantially outperform baseline NRL methods in both node classification and link prediction tasks. An interesting future research direction concerns the extension of the proposed methodology to the multiple kernel learning framework [33].

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. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *CoRR*, 2018.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[4] A. Grover and J. Leskovec, "Node2Vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NIPS*, 2001, pp. 585–591.

[6] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *KDD*, 2016, pp. 1105–1114.

[8] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive diffusions for scalable learning over graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1307–1321, March 2019.

[9] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[10] D. Nguyen and F. D. Malliaros, "BiasedWalk: Biased sampling for representation learning on graphs," in *IEEE Big Data*, 2018, pp. 4045–4053.

[11] A. Çelikkanat and F. D. Malliaros, "TNE: A latent model for representation learning on networks," in *NeurIPS Relational Representation Learning Workshop*, 2018.

[12] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *ICANN*, 1997, pp. 583–588.

[13] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[14] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *KDD*. ACM, 2004, pp. 551–556.

[15] X. Liu, C. Aggarwal, Y.-F. Li, X. Kong, X. Sun, and S. Sathe, "Kernelized matrix factorization for collaborative filtering," in *SDM*, 2016, pp. 378–386.

[16] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *ICML*, 2003, pp. 720–727.

[17] A. Epasto and B. Perozzi, "Is a single embedding enough? learning node representations that capture multiple social contexts," in *WWW*, 2019, pp. 394–404.

[18] 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 *IEEE Big Data*, 2018, pp. 1085–1092.

[19] C. Lin, P. Ishwar, and W. Ding, "Node embedding for network community discovery," in *ICASSP*, 2017, pp. 4129–4133.

[20] J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang, "NetSMF: Large-scale network embedding as sparse matrix factorization," in *WWW*, 2019, pp. 1509–1520.

[21] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.

[22] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and Node2Vec," in *WSDM*, 2018, pp. 459–467.

[23] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep 1936.

[24] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 67–93, Mar. 2002.

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

[26] L. Bottou, "Stochastic gradient learning in neural networks," in *In Proceedings of Neuro-Nimes. EC2*, 1991.

[27] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *NIPS*, 2013, pp. 2265–2273.

[28] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *AAAI*, 2018.

[29] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, 2008.

[30] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip!: Online learning of multi-scale network embeddings," in *ASONAM*, 2017, pp. 258–265.

[31] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, 2007.

[32] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *NIPS*, 2012, pp. 539–547.

[33] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *ICML*, 2004, p. 6.