Unsupervised Neighborhood Propagation Kernel Layers for Semi-supervised Node Classification

Sonny Achten, Francesco Tonin, Panagiotis Patrinos, Johan A.K. Suykens

KU Leuven, Department of Electrical Engineering (ESAT),
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
{sonny.achten, francesco.tonin, panos.patrinos, johan.suykens}@esat.kuleuven.be

Abstract

We present a deep Graph Convolutional Kernel Machine (GCKM) for semi-supervised node classification in graphs. The method is built of two main types of blocks: (i) We introduce unsupervised kernel machine layers propagating the node features in a one-hop neighborhood, using implicit node feature mappings. (ii) We specify a semi-supervised classification kernel machine through the lens of the Fenchel-Young inequality. We derive an effective initialization scheme and efficient end-to-end training algorithm in the dual variables for the full architecture. The main idea underlying GCKM is that, because of the unsupervised core, the final model can achieve higher performance in semi-supervised node classification when few labels are available for training. Experimental results demonstrate the effectiveness of the proposed framework.

Introduction

Semi-supervised node classification has been an important research area for several years. In many real-life applications, one has structured data for which the entire graph can be observed (e.g., a social network where users are represented as nodes and the relationships between users as edges). However, the node labels can only be observed for a small subset of nodes. The learning task is then to predict the label of unsupervised nodes, given the node attributes of all nodes and the network structure of the graph. In many cases, exploiting the information in a local neighborhood can boost performance (e.g., friends in the social network are likely to share the same preferences). In recent years, graph neural networks (GNNs) have rapidly transformed the field of learning on graphs. Their performance follows from their ability to effectively propagate the node information through the network iteratively and from end-to-end training (Hamilton 2020; Bacciu et al. 2020; Wu et al. 2021).

More traditionally, kernel-based methods such as support vector machines were the standard in graph learning tasks because of the possibility to use a kernel function that represents pairwise similarities between two graphs as the dot product of their embeddings, without the need to explicitly know these potentially high-dimensional embeddings (i.e., the "kernel-trick") (Ghosh et al. 2018; Kriege, Johansson, and Morris 2020). An additional advantage of kernel machines is that they have strong foundations in learning theory and have clear and interpretable optimization (Vapnik 1998; Schölkopf and Smola 2002; Suykens et al. 2002). A drawback however, is that they do not benefit from hierarchical representation learning as deep learning methods do.

In recent years, a new branch of research emerged that relates the training of infinitely wide neural networks to kernel methods (Nikolentzos et al. 2018; Belkin, Ma, and Mandal 2018), gaining understanding in the generalization capabilities of highly parameterized neural networks. While extensions exist for graph learning (Du et al. 2019), these are studied for graph level tasks (e.g. graph classification), and the effectiveness on using kernels in a message passing scheme for node level tasks remains an open question. This work therefore uses multiple kernel functions (i.e., one for each layer) where the node representations are aggregated and implicitly transformed into an infinite-dimensional feature space, and aims to study the effectiveness of this approach for semi-supervised node classification.

While GNNs typically employ a regression-based core model, we utilize an unsupervised core model. Specifically, our model incorporates unsupervised message-passing levels based on Kernel Principal Component Analysis (kernel PCA), and a semi-supervised layer based on a weighted version of kernel PCA, i.e. Kernel Spectral Clustering (KSC). This approach is motivated to address real-world problems where few labels are available, leading to improved performance compared to existing methods. Kernel methods are used in their dual form, where they directly learn node representations. In fact, unlike typical parametric GNNs that learn parameters, our method directly learns $\boldsymbol{H}$, the matrix of node embeddings. This allows greater modelling flexibility, where node representations can be used for a variety of tasks, including classification with varying levels of supervision and clustering. This adaptability makes our method a versatile tool for node tasks in applications.

Contributions We introduce a deep Graph Convolutional Kernel Machine (GCKM) for node classification made of multiple shallow layers with non-parametric aggregation functions. Our main contributions are the following.

- We propose an architecture consisting of multiple unsu-

Appears in the Proceedings of the AAAI 2024 Conference on Artificial Intelligence.

pervised kernel machine layers for one-hop node aggregation and a final semi-supervised kernel machine layer. The main idea underlying our method is to combine multiple unsupervised kernel machines s.t. the final model can achieve higher performance in semi-supervised node classification where most nodes in the graph are unlabeled.

- We show how to train the proposed deep kernel machine in its dual form by directly learning the hidden node representations. Because of the appropriate regularization mechanisms, the neighborhood aggregation of each layer is implicitly embedded in the final representation, which is a key difference with GNNs.
- We propose a two-step optimization algorithm with an initialization and fine-tuning phase. Because the model is built on unsupervised core models, augmented with a supervised loss term, we illustrate the possibility to use an unsupervised validation metric.
- In experiments, we show that our model outperforms the state-of-the-art in a transductive node classification setting when few labels are available, which is of particular interest in many real-world applications where labels are difficult or expensive to collect.

The reported results can be reproduced using our code on GitHub[1] and the Appendix is available below.

Preliminaries and related work

An undirected and unweighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is defined by a set of nodes or vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$ between these nodes. The node degree is simply the number of adjacent nodes: $d_v = |\mathcal{N}_v|$, where $\mathcal{N}_v$ is the one-hop neighborhood of node v. As the task of the proposed method will be node classification, we will consider attributed graphs $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ where each node v has a d-dimensional node features vector $\boldsymbol{x}_v$ and a class label y_v. By concatenating the feature vectors, we obtain the node feature matrix $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

We will use lowercase symbols (e.g., x) for scalars, lowercase bold (e.g., $\boldsymbol{x}$) for vectors and uppercase bold (e.g., $\boldsymbol{X}$) for matrices. A single entry of a matrix is represented by X_{ij} where i and j indicate the row and column respectively. Superscripts in brackets indicate the layer in deep architectures whereas subscripts indicate datapoints (e.g., $\boldsymbol{h}_v^{(l)}$). Subscript c indicates a centering, as will be explained. We represent sets with curly brackets $\{\cdot\}$ and use double curly brackets $\{\{\cdot\}\}$ for multisets (i.e., sets that allow multiple instances of a same element). At any point, the reader can consult the list of symbols in Appendix for clarification.

Graph neural networks Many convolutional GNN layers can be decomposed into a nonparametric aggregation step $\psi(\cdot, \cdot)$, followed by a nonlinear transformation $\phi(\cdot)$. In this case, the hidden representation of node v in layer l is of the form:

$$\boldsymbol{h}_v^{(l)} = \phi\left(\psi\left(\boldsymbol{h}_v^{(l-1)}, \left\{\left\{\boldsymbol{h}_u^{(l-1)} | u \in \mathcal{N}_v\right\}\right\}\right)\right). \tag{1}$$

Well-known examples are GCN (Kipf and Welling 2017) and GIN (Xu et al. 2019), which is maximally powerful in the class of message passing neural networks and as expressive as the one-dimensional Weisfeiler-Lehman graph isomorphism test (Weisfeiler and Lehman 1968). Xu et al. (2019) have demonstrated that GIN's expressiveness follows from the sum aggregator and the injectiveness of the transformation function, for which they proposed a multilayer perceptron with at least one hidden layer, motivated by the universal approximator theorem (Hornik, Stinchcombe, and White 1989; Hornik 1991).

Restricted kernel machines In deep kernel learning, the recently proposed restricted kernel machine (RKM) framework (Suykens 2017) connects least squares support vector machines (LS-SVMs) and kernel PCA with restricted Boltzmann machines (Suykens and Vandewalle 1999; Suykens et al. 2003; Salakhutdinov 2015). They possess primal and dual model representations based on the concept of conjugate feature duality, which introduces dual variables as hidden features based on an inequality of quadratic forms. The feature map can be defined explicitly (e.g., with a deep neural network) or implicitly by means of a kernel function when using the dual representation. The RKM interpretation of kernel PCA leads to an eigendecomposition of the kernel matrix. Asides kernel PCA, Suykens (2017) also formulated different types of kernel machines in the RKM framework. Deep RKMs are then obtained by combining multiple RKM layers, where the dual variables are the input for the next layer. RKMs have been successfully applied to unsupervised problems, including generative modelling (Pandey et al. 2022), disentangled deep feature learning (Tonin, Patrinos, and Suykens 2021), and multi-view clustering (Tao et al. 2022).

Kernels in GNNs Recent works have established explicit connections between GNNs and kernel machines. Nikolentzos et al. (2018) and Feng et al. (2022) used graph kernels as convolutional filters in a GNN setting. Therefore, they are in essence not kernel machines. In Lei et al. (2017), a deep neural network is modularly built with recurrent kernel modules. On this modular scale, the kernels are used as convolutional filters, rather than in a kernel machine setting. On the model scale, they show that the feature mappings of the graph neural networks lie in the same Hilbert spaces as some common graph kernels. Their proposed model has no dual representation, typical for kernel machines.

GNN inspired shallow kernel learning Conversely, Du et al. (2019) designed the graph neural tangent kernel, based on infinitely wide GNN architectures that are trained by gradient descent. Chen, Jacob, and Mairal (2020) use path and walk kernels to embed the local graph topology in an iteratively constructed feature map. Although one can refer to the feature maps of these methods as deep feature maps, their method remains a shallow kernel machine; whereas in this paper, we consider several feature maps over multiple layers, where each layer is associated with a kernel based objective function and dual variables. Also, Du et al. (2019) and Chen, Jacob, and Mairal (2020) designed and implemented their model for a graph learning setting, whereas the focus of our work is node representation learning.

[1] https://github.com/sonnyachten/GCKM

Method

This section introduces the deep Graph Convolutional Kernel Machine (GCKM): a semi-supervised kernel machine propagating information through the network for node classification in graphs. First, the GCKM layer (GCKMℓ) for single-hop propagation is proposed. Also, the semi-supervised kernel machine (Semi-SupRKM) is described. We then explain how to combine these shallow kernel machines in a deep model to increase the receptive field of the model to multiple hops and to perform semi-supervised node classification, after which we conclude with some key properties of the model. All proofs and derivations for both the GCKMℓ as the Semi-SupRKM can be found in Appendix.

The graph convolutional and semi-supervised kernel machine layers as building blocks

Graph convolutional kernel machine layer The derivation of GCKMℓ starts from the primal minimization problem:

$$\min_{\boldsymbol{W},\boldsymbol{e}_v} J = \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}) - \frac{1}{2}\sum_{i=1}^{n}\boldsymbol{e}_v^T\boldsymbol{\Lambda}^{-1}\boldsymbol{e}_v$$
$$\text{s.t.}\begin{cases}\boldsymbol{e}_v = \boldsymbol{W}^T\phi_c(\boldsymbol{a}_v), & i=1,\dots,n\\ \boldsymbol{a}_v = \psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u\in\mathcal{N}_v\}\})\end{cases}, \quad (2)$$

where $\boldsymbol{W}\in\mathbb{R}^{d_f\times s}$ is an unknown interconnection matrix, $\boldsymbol{e}_v\in\mathbb{R}^s$ the error variables, $n=|\mathcal{V}_{\text{tr}}|$ the number of training nodes, and symmetric hyperparameter matrix $\boldsymbol{\Lambda}\succ 0$. Given a feature map $\phi(\cdot)$, the centered feature map is defined as $\phi_c(\cdot)\triangleq\phi(\cdot)-\Sigma_i\phi(\boldsymbol{x}_i)/n$. Because of the minus sign in the objective function, one can interpret this minimization problem conceptually as maximizing the variance of the error variables $\boldsymbol{e}_i$ around zero target, while keeping the weights $\boldsymbol{W}$ small (Suykens et al. 2003). Note that the formulation of the error variables has the same form as GNN layers such as GCN and GIN (1). In this regard, the GCKM layer relates in the same way to a WL-iteration (i.e., an iteration of the Weisfeiler-Lehman graph isomorphism test) as these GNN layers.

We now introduce dual variables $\boldsymbol{h}_i$ using a case of Fenchel-Young inequality (Rockafellar 1974):

$$\frac{1}{2}\boldsymbol{e}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{e}+\frac{1}{2}\boldsymbol{h}^T\boldsymbol{\Lambda}\boldsymbol{h}\geq\boldsymbol{e}^T\boldsymbol{h},\quad\forall\boldsymbol{e},\boldsymbol{h}\in\mathbb{R}^s,\forall\boldsymbol{\Lambda}\in\mathbb{R}^{s\times s}_{\succ 0}.$$

When substituting the above in (2) and eliminating the error variables, one obtains a primal-dual minimization problem as an upper bound on the primal objective function:

$$\min_{\boldsymbol{W},\boldsymbol{h}_v}\bar{J}\triangleq-\sum_{v=1}^{n}\phi_c(\boldsymbol{a}_v)^T\boldsymbol{W}\boldsymbol{h}_v + \frac{1}{2}\sum_{v=1}^{n}\boldsymbol{h}_v^T\boldsymbol{\Lambda}\boldsymbol{h}_v+\frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}). \quad (3)$$

Note that problem (3) is generally nonconvex. Whether or not it has a solution depends on hyperparameters $\boldsymbol{\Lambda}$. In the next lemma we show how to determine $\boldsymbol{\Lambda}$ automatically by the optimization. We next define the Gram matrix $\boldsymbol{K}$ with $K_{uv}=\phi(\boldsymbol{a}_u)^T\phi(\boldsymbol{a}_v)$, which depends on the aggregated node features; $\boldsymbol{K}_c=\boldsymbol{M}_c\boldsymbol{K}\boldsymbol{M}_c$ with $\boldsymbol{M}_c=(\boldsymbol{I}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^T)$ the centering matrix; and $\boldsymbol{H}=[\boldsymbol{h}_1,\dots,\boldsymbol{h}_n]^T$. We now arrive at formulating the minimization problem w.r.t. the dual variables:

Lemma 1. *The solution to the dual minimization problem:*

$$\min_{\boldsymbol{H}}-\frac{1}{2\eta}Tr(\boldsymbol{H}^T\boldsymbol{K}_c(\boldsymbol{X},\mathcal{E})\boldsymbol{H})\quad s.t.\ \boldsymbol{H}^T\boldsymbol{H}=\boldsymbol{I}_s, \quad (4)$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ *as* (3) *when the hyperparameters* $\boldsymbol{\Lambda}$ *in* (3) *are chosen to equal the symmetric part of the Lagrange multipliers* $\boldsymbol{Z}$ *of the equality constraints in* (4)*; i.e.,* $\boldsymbol{\Lambda}=(\boldsymbol{Z}+\boldsymbol{Z}^T)/2$ *.*

Now, (4) is bounded and is guaranteed to have a minimizer. Indeed, it is a minimization of a concave objective over a compact set. Note that (4) can be solved by a gradient-based algorithm. The solution satisfies the following property:

Proposition 2. *Given a symmetric matrix* $\boldsymbol{K}_c$ *with eigenvalues* $\lambda_1\geq\cdots\geq\lambda_s>\lambda_{s+1}\geq\cdots\geq\lambda_n\geq 0$*, and* $\eta>0$ *a hyperparameter; and let* $\boldsymbol{g}_1,\dots,\boldsymbol{g}_s$ *be the columns of* $\boldsymbol{H}$*; then* $\boldsymbol{H}$ *is a minimizer of* (4) *if and only if* $\boldsymbol{H}^T\boldsymbol{H}=\boldsymbol{I}_s$ *and span*$(\boldsymbol{g}_1,\dots,\boldsymbol{g}_s)=$ *span*$(\boldsymbol{v}_1,\dots,\boldsymbol{v}_s)$*, where* $\boldsymbol{v}_1,\dots,\boldsymbol{v}_s$ *are the eigenvectors of* $\boldsymbol{K}_c$ *corresponding to the* s *largest eigenvalues.*

Remark 3. One can obtain a solution of (4) by solving the eigendecomposition problem:

$$\frac{1}{\eta}\boldsymbol{K}_c(\boldsymbol{X},\mathcal{E})\boldsymbol{H}=\boldsymbol{H}\boldsymbol{\Lambda}, \quad (5)$$

and selecting the eigenvectors corresponding to the s largest eigenvalues as the columns of $\boldsymbol{H}$.

Notice that (5) is the kernel PCA formulation, a nonlinear generalization of PCA (Schölkopf and Smola 2002; Suykens et al. 2003), with the aggregated node features as the input, and where the first s components represent the data. The solution of (4) generally yields any orthonormal basis for the same subspace as spanned by the first s components, and therefore embeds the same information in the dual representations. Further, instead of explicitly defining a feature map, one can apply the kernel trick using Mercer's theorem, stating that for any positive definite kernel $k(\cdot,\cdot)$ there exists a, possibly infinite dimensional, feature map $\phi(\cdot)$ such that $\phi(\boldsymbol{a}_u)^T\phi(\boldsymbol{a}_v)=k(\boldsymbol{a}_u,\boldsymbol{a}_v)$ (Mercer 1909). In this case, the transformation function $\boldsymbol{W}^T\phi(\cdot)$ is only implicitly defined. As the kernel function, one could choose for example a linear kernel, a polynomial kernel, a radial basis function (RBF), or a kernel that is particularly suited for the inherent characteristics of the data (e.g., for categorical node features (Couto 2005)). We can now define the GCKM layer:

Definition 4 (Graph Convolutional Kernel Machine layer)**.** GCKMℓ is a kernel machine for unsupervised node representation learning that propagates information through the network in a one-hop neighborhood in a convolutional flavour. More formally, it can be interpreted as a principal component analysis on the aggregated node features in a kernel induced feature space, where the latent representations are obtained by solving either (4) or (5), and are used as the input for the subsequent layer in a deep GCKM.

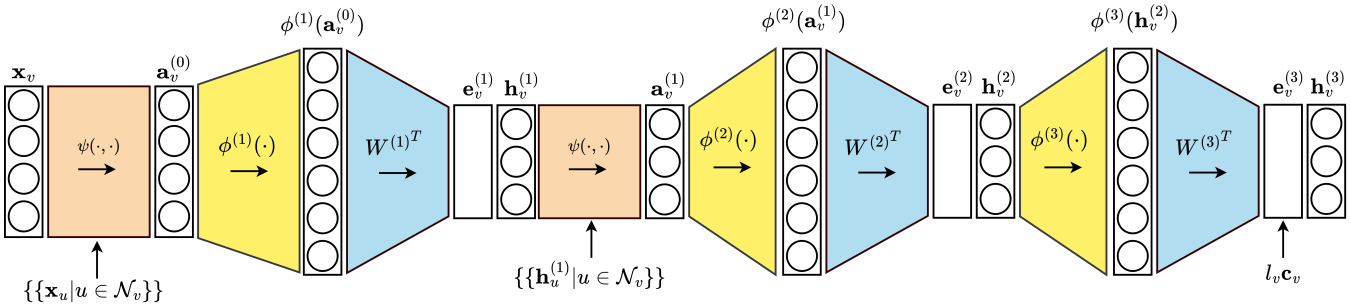

Figure 1: A deep GCKM architecture for semi-supervised node classification, consisting of two GCKM layers (GCKMℓ_1, GCKMℓ_2) and a Semi-SupRKM layer. In each GCKMℓ, the node features are aggregated and then (implicitly) transformed to obtain the error variables. The dual variables are coupled with these error variables by means of conjugate feature duality and serve as input for the next layer. In the final Semi-SupRKM layer, the dual variables directly represent the class labels of the unsupervised nodes.

The key difference between a GNN layer and GCKMℓ is thus that the former learns a parametric mapping for the feature transformation, whereas GCKMℓ learns the node representations themselves (the transformation is only implicitly defined by a kernel induced feature map). For the aggregation step, we can choose any function that can handle multisets of different sizes and that is invariant to permutations on this multiset. In our experiments, we use GCN aggregation or sum aggregation:

$$\psi_{\text{GCN}}(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u|u\in\mathcal{N}_v\}\}) = \sum_{u\in\mathcal{N}_v\cup\{v\}} \frac{\boldsymbol{x}_u}{\sqrt{\tilde{d}_u\tilde{d}_v}},$$
$$\psi_{\text{sum}}(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u|u\in\mathcal{N}_v\}\}) = \boldsymbol{x}_v + \sum_{u\in\mathcal{N}_v} \boldsymbol{x}_u,$$

where $\tilde{d}_v$ is the node degree of node v after self-loops were added to the graph.

We proceed with some properties of GCKMℓ. The model-based approach gives us the possibility to use one set of nodes for training $\mathcal{V}_{\text{tr}}$ and use an out-of-sample extension for another (super)set of nodes $\mathcal{V}$, possibly from another graph. The following result follows from the stationarity conditions of (3):

Lemma 5. *Let $n = |\mathcal{V}_{tr}|$, $m = |\mathcal{V}|$, and $\boldsymbol{K}^{\mathcal{V}_1,\mathcal{V}_2} \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$ a kernel matrix containing kernel evaluations of all elements of set $\mathcal{V}_1$ w.r.t. all elements of set $\mathcal{V}_2$ (i.e., $K^{\mathcal{V}_1,\mathcal{V}_2}_{uv} = k(\boldsymbol{a}_u, \boldsymbol{a}_v)\ \forall u\in\mathcal{V}_1, \forall v\in\mathcal{V}_2$). The dual representations can then be obtained using:*

$$\hat{\boldsymbol{H}}_{\mathcal{V}} = \frac{1}{\eta}\boldsymbol{K}^{\mathcal{V},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}}\boldsymbol{\Lambda}^{-1} - \frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}^{\mathcal{V}_{tr},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}}}{n\eta}\boldsymbol{\Lambda}^{-1}, \quad (6)$$

Equation (6) is useful for large-scale problems, when subsets are used for training, and for inductive tasks. It also satisfies the permutation equivariance condition.

Proposition 6. *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, the aggregated node features $\{a_v : v\in\mathcal{V}_{tr}\}$ and latent representations $\boldsymbol{H}_{\mathcal{V}_{tr}}$ of the training nodes $\mathcal{V}_{tr}$, and a local aggregation function $\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u|u\in\mathcal{N}_v\}\})$ that is permutation invariant; the mapping f from $\mathcal{G}$ to $\mathcal{G}' = (\mathcal{V}, \mathcal{E}, \hat{\boldsymbol{E}}_{\mathcal{V}})$ using (6) is equivariant w.r.t. any permutation $\pi(\mathcal{G})$, i.e., $\mathcal{G}' = f(\mathcal{G}) \iff \pi(\mathcal{G}') = f(\pi(\mathcal{G}))$.*

A theoretical analysis of the expressiveness of GCKMℓ can be put in context of the theoretical results by Xu et al. (2019). Any associated feature map of the RBF-kernel is injective (Christmann and Steinwart 2008). Furthermore, it has been established that SVMs using the RBF-kernel are universal approximators (Burges 1998; Hammer and Gersmann 2003).

Lemma 7. *A GCKMℓ that uses sum aggregation and a RBF-kernel is as expressive as an iteration of the Weisfeiler-Lehman graph isomorphism test (Weisfeiler and Lehman 1968).*

Semi-supervised restricted kernel machine layer Next, we introduce a multi-class semi-supervised kernel machine for classification (Semi-SupRKM). Like Mehrkanoon et al. (2015), we start from kernel spectral clustering as an unsupervised core model, and augment it with a supervised loss term. Here however, to be able to use it in a deep kernel machine, we introduce duality with conjugated features, rather than by means of Lagrange multipliers.

The primal minimization problem is given by:

$$\min_{\boldsymbol{W},\boldsymbol{e}_i,\boldsymbol{b}} \quad J = \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}) - \frac{1}{2\lambda_1}\sum_{i=1}^n v_i\boldsymbol{e}_i^T\boldsymbol{e}_i + \frac{1}{2\lambda_2}\sum_{i=1}^n l_i(\boldsymbol{e}_i-\boldsymbol{c}_i)^T(\boldsymbol{e}_i-\boldsymbol{c}_i) \quad (7)$$

$$\text{s.t.} \quad \boldsymbol{e}_i = \boldsymbol{W}^T\phi(\boldsymbol{x}_i) + \boldsymbol{b}, \quad i = 1,\ldots,n, \quad (8)$$

with hyperparameters η, λ_1, and λ_2, where $l_i \in \{0,1\}$ indicates whether the label of datapoint i is used in training, $\boldsymbol{c}_i \in \{-1,1\}^p$ encodes its class label (e.g., in a one-vs-all encoding), and v_i is a weighting scalar obtained as the inverse degree of the datapoint in the similarity graph defined by $\boldsymbol{K} = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$, i.e., $v_i = 1/\Sigma_j K_{ij}$.

By introducing Fenchel-Young inequalities:

$$\frac{1}{2}\boldsymbol{h}_i^T\boldsymbol{h}_i - \boldsymbol{e}_i^T\boldsymbol{h}_i \geq -\frac{1}{2}\boldsymbol{e}_i^T\boldsymbol{e}_i,$$
$$-\frac{1}{2}\boldsymbol{h}_i^T\boldsymbol{h}_i + (\boldsymbol{e}_i-\boldsymbol{c}_i)^T\boldsymbol{h}_i \leq \frac{1}{2}(\boldsymbol{e}_i-\boldsymbol{c}_i)^T(\boldsymbol{e}_i-\boldsymbol{c}_i),$$

and defining $r_i = \frac{v_i}{\lambda_1} - \frac{l_i}{\lambda_2}$, one obtains the primal-dual minimization problem, which corresponds to minimizing the upper bound on the negative variance while minimizing the lower bound on the supervision error:

$$\min_{\boldsymbol{W},\boldsymbol{h}_i,\boldsymbol{b}} \tilde{J} \triangleq \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}) + \frac{1}{2}\sum_{i=1}^{n} r_i \boldsymbol{h}_i^T \boldsymbol{h}_i - \sum_{i=1}^{n} r_i(\boldsymbol{W}^T\phi(\boldsymbol{x}_i) + \boldsymbol{b})^T\boldsymbol{h}_i - \sum_{i=1}^{n}\frac{l_i}{\lambda_2}\boldsymbol{c}_i^T\boldsymbol{h}_i. \quad (9)$$

We next define matrices $\boldsymbol{R} = \text{diag}(r_1,\ldots,r_n)$; $\boldsymbol{L} = \text{diag}(l_1,\ldots,l_n)$; $\boldsymbol{S} = \boldsymbol{I}_n - \frac{\mathbf{1}_n\mathbf{1}_n^T\boldsymbol{R}}{\mathbf{1}_n^T\boldsymbol{R}\mathbf{1}_n}$; $\boldsymbol{H} = [\boldsymbol{h}_1,\ldots,\boldsymbol{h}_n]^T$; and $\boldsymbol{C} = [\boldsymbol{c}_1,\ldots,\boldsymbol{c}_n]^T$.

Lemma 8. *The solution to the dual minimization problem:*

$$\min_{\boldsymbol{H}} -\frac{1}{2\eta}Tr(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{K}(\boldsymbol{X})\boldsymbol{R}\boldsymbol{H}) + \frac{1}{2}Tr(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{H}) - \frac{1}{\lambda_2}Tr(\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{C}) \qquad s.t.\ \boldsymbol{H}^T\boldsymbol{R}\mathbf{1}_n = \mathbf{0}_p \quad (10)$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ as (9) *where the Lagrange multipliers equal the bias $\boldsymbol{b}$.*

Remark 9. Alternatively, one can find the dual variables by solving a linear system in the dual variables:

$$(\boldsymbol{I}_n - \frac{1}{\eta}\boldsymbol{R}\boldsymbol{S}\boldsymbol{K}(\boldsymbol{X}))\boldsymbol{R}\boldsymbol{H} = \frac{1}{\lambda_2}\boldsymbol{S}^T\boldsymbol{L}\boldsymbol{C}. \quad (11)$$

From the stationarity conditions of (9), one obtains $\boldsymbol{e}_i = \boldsymbol{h}_i - \frac{l_i\boldsymbol{c}_i}{r_i\lambda_2}$, which simplifies to $\boldsymbol{e}_i = \boldsymbol{h}_i$ for the unsupervised training points. One can thus directly infer the class label $\hat{y}_i$ from the learned representation by comparing the class codes and select the one with closest Hamming distance to the error variable $\boldsymbol{e}_i$. For one-vs-all encoding, this is simply the index with highest value: $\hat{y}_i = \text{argmax}_j(\boldsymbol{h}_i)_j$. When using a subsample for training, one can use the out-of-sample extension described in Appendix.

Deep graph convolutional kernel machine

Next, we construct a deep graph convolutional kernel machine for semi-supervised node classification by combining multiple GCKMℓ's with a Semi-SupRKM read-out layer (Figure 1). Similar to GNNs, the dual variables of the GCKMℓ's ($\boldsymbol{H}^{(1)}$ and $\boldsymbol{H}^{(2)}$) serve as input for the subsequent layer and can thus be viewed as hidden representations. The dual variables of the Semi-SupRKM layer ($\boldsymbol{H}^{(3)}$), can directly be used to infer the class label of the unlabeled nodes. The optimization problem for end-to-end learning is given by combining the dual minimization problems of the different layers (i.e., (4) and (10), with $\boldsymbol{K}^{(l)}$ the kernel matrix of layer l). For two GCKM layers and a Semi-SupRKM layer, this yields:

$$\min_{\boldsymbol{H}^{(1)},\boldsymbol{H}^{(2)},\boldsymbol{H}^{(3)}} J_{\text{GCKM}} \triangleq -\frac{1}{2\eta^{(1)}}\text{Tr}({\boldsymbol{H}^{(1)}}^T\boldsymbol{K}_c^{(1)}\boldsymbol{H}^{(1)}) - \frac{1}{2\eta^{(2)}}\text{Tr}({\boldsymbol{H}^{(2)}}^T\boldsymbol{K}_c^{(2)}\boldsymbol{H}^{(2)}) - \frac{1}{2\eta^{(3)}}\text{Tr}({\boldsymbol{H}^{(3)}}^T\boldsymbol{R}\boldsymbol{K}^{(3)}\boldsymbol{R}\boldsymbol{H}^{(3)}) + \frac{1}{2}\text{Tr}({\boldsymbol{H}^{(3)}}^T\boldsymbol{R}\boldsymbol{H}^{(3)}) + \frac{1}{\lambda_2^{(3)}}\text{Tr}({\boldsymbol{H}^{(3)}}^T\boldsymbol{L}\boldsymbol{C}) \quad \text{s.t. } {\boldsymbol{H}^{(1,2)}}^T\boldsymbol{H}^{(1,2)} = \boldsymbol{I}_{s_{1,2}},\ {\boldsymbol{H}^{(3)}}^T\boldsymbol{R}\mathbf{1}_n = \mathbf{0}_p. \quad (12)$$

Algorithm 1 Optimization algorithm of GCKM.

1: Initialize $\{\boldsymbol{H}_0^{(1)}, \boldsymbol{H}_0^{(2)}, \boldsymbol{H}_0^{(3)}\}$
2: **for** $k \leftarrow 0, 1, \ldots, T$ **do**
3: Compute $\boldsymbol{K}_c^{(1)}$ from aggregated $\boldsymbol{X}$
4: Compute $\boldsymbol{K}_c^{(2)}$ from aggregated $\boldsymbol{H}_k^{(1)}$
5: Update $\{\boldsymbol{H}_{k+1}^{(1)}, \boldsymbol{H}_{k+1}^{(2)}\} \leftarrow$ CayleyAdam(J_{GCKM})
6: Compute $\boldsymbol{K}^{(3)}$ from $\boldsymbol{H}_{k+1}^{(2)}$
7: Update $\{\boldsymbol{H}_{k+1}^{(3)}\} \leftarrow$ Solve (11) with $\boldsymbol{K}^{(3)}$
8: **end for**

Like in GNNs, the number of GCKMℓ's used in the deep GCKM determines the receptive field of the model (i.e., the number of hops that the information propagates through the network). However, the key difference is that in GCKM, this message passing is implicitly embedded in the final representation. In our experiments, we will also train a multiview variant, called GCKM-MV, which is the same as GCKM but with the kernel of the last layer defined as $k^{(3)}(\{\boldsymbol{x}_u, \boldsymbol{h}_u\}, \{\boldsymbol{x}_v, \boldsymbol{h}_v\}) = k_1(\boldsymbol{h}_u, \boldsymbol{h}_v)k_2(\boldsymbol{x}_u, \boldsymbol{x}_v)$, where k_1 and k_2 are positive definite kernel functions that use the representations of the last GCKMℓ and the initial node representations respectively. By construction, $k^{(3)}$ is also positive definite.

Training deep graph convolutional kernel machines Similar to stacked auto-encoders (Bengio 2009), the dual variables $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ and $\boldsymbol{H}^{(3)}$ are initialized by sequentially solving the layers as individual kernel machines before finetuning end-to-end. Then, the constrained optimization problem (12) is addressed with an alternating minimization scheme, as shown in Algorithm 1. First, note that the constraint set for the two GCKM layers is the Stiefel manifold $\text{St}(s_j, n) = \{\boldsymbol{H}^{(j)} \in \mathbb{R}^{n\times s_j} \,|\, {\boldsymbol{H}^{(j)}}^T\boldsymbol{H}^{(j)} = \boldsymbol{I}_{s_j}\}$, $j = 1, 2$. We therefore employ the Cayley Adam optimizer (Li, Li, and Todorovic 2019) to update $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ with $\boldsymbol{H}^{(3)}$ fixed. Then, $\boldsymbol{H}^{(3)}$ is updated by solving the linear system (11) associated with the semi-supervised layer.

As a validation metric, one can use the accuracy of the validation set or a different supervised metric. Alternatively, because the core model of the Semi-SupRKM is based on kernel spectral clustering, one can use an unsupervised metric that quantifies the quality of obtained clustering (Langone, Mall, and Suykens 2013). For node v, the centered cosine distance w.r.t. class s is $d_{v,s}^{\cos} = 1 - \frac{(\boldsymbol{c}_s - \boldsymbol{\mu})^T(\boldsymbol{e}_v - \boldsymbol{\mu})}{||\boldsymbol{c}_s - \boldsymbol{\mu}||\,||\boldsymbol{e}_v - \boldsymbol{\mu}||}$, where $\boldsymbol{c}_s$ is the coding of class s and $\boldsymbol{\mu}$ is the center of all codings. The unsupervised performance metric for nodes $\mathcal{V}_{\text{unsup}}$ is then obtained by assigning each node to its closest class encoding and averaging the cosine distances: $\mathcal{L}_{\text{unsup}} = {}^1/_{|\mathcal{V}_{\text{unsup}}|}\sum_{v=1}^{|\mathcal{V}_{\text{unsup}}|}\min_s d_{v,s}^{\cos}$.

Experiments

Datasets and main setting As datasets, we use four homophilious graphs: Cora, CiteSeer, PubMed (Sen et al. 2008; Yang, Cohen, and Salakhudinov 2016), and OGB-Arxiv (Hu et al. 2020), which are citation graphs, as

METHOD	NODE ATTR.	GRAPH STRUCTURE	DIRECTED GRAPHS	WEIGHTED GRAPHS	PROPAGATION RULE
SVM-RBF	YES	NO	NO	NO	NONE
SVM-DIFF	NO	YES	YES	YES	LAPLACIAN
SVM-WWL	YES	YES	YES	NO	1-HOP/ITERATION
GCKN	YES	YES	YES	YES	PATH/WALK KERNELS
MLP	YES	NO	NO	NO	NONE
GCN	YES	YES	NO	YES	1-HOP/LAYER
APPNP	YES	YES	NO	YES	PAGERANK
GPR-GNN	YES	YES	NO	YES	PAGERANK
BERNNET	YES	YES	NO	YES	POLYNOMIAL FILTERS
CHEBNETII	YES	YES	NO	YES	POLYNOMIAL FILTERS
GCKM (OURS)	YES	YES	YES	YES	1-HOP/LAYER

Table 1: Qualitative comparison.

well as two heterophilious graphs: Chameleon and Squirrel (Wikipedia graphs (Rozemberczki, Allen, and Sarkar 2021)). All graphs are undirected and unweighted and a summary Table is provided in Appendix. We trained a deep GCKM with two GCKM layers and a Semi-SupRKM readout layer, as depicted in Figure 1, which we will simply refer to as GCKM. We used GCN aggregation for the citation networks and sum aggregation for the heterophilious graphs. We also trained GCKM-MV, where the second view is obtained by a RBF-kernel operating on the input node features. We compare our method to a multilayer perceptron (MLP) and to GCN (Kipf and Welling 2017) as it is the most comparable GNN counterpart to our method. Furthermore, we add APPNP (Klicpera, Bojchevski, and Günnemann 2019), BernNet (He et al. 2021), GPR-GNN (Chien et al. 2021), and ChebNetII (He, Wei, and Wen 2022) to the comparison because all these methods achieve state-of-the-art performance on at least one of the datasets. For these GNN baselines, we used the code of Lim et al. (2021) and He, Wei, and Wen (2022). For the SVM-based models, we compare with a standard RBF-kernel (SVM-RBF), with a diffusion kernel (SVM-DIFF (Smola and Kondor 2003)), and to Wasserstein Weisfeiler-Lehman kernel (SVM-WWL (Togninalli et al. 2019)). A qualitative comparison between the different models is given in Table 1, showing whether the models exploit the node attributes and/or graph structure, whether they are applicable for directed and/or weighted graphs, and giving an indication of the complexity of the propagation rules. The reader can consult Appendix for more details about the hyperparameter search.

Semi-supervised node classification with few labels In the first experiment, we assess the models in a true semi-supervised setting where only few labels are given. For Cora, CiteSeer, and PubMed, there are 4 training labels per class, 100 labels for validation, and 1000 labels for testing. For Chameleon and Squirrel, a 0.5%/0.5%/99% train/validation/test-split is used. For the aforementioned datasets, we merge the training and validation labels into one set and use the following training and validation strategies: (i) for GCKM, all labels are used for training and the model is selected based on the cosine similarity metric after a random search; and (ii) for the baseline models, we use a 5-fold crossvalidation scheme and take the best model based on the average validation accuracy over the 5 folds from a grid search.[2] For OGB-Arxiv, we use a 2.5%/2.5%/95% random split and select the model that had highest combined score: $\mathcal{L}_{\text{comb}} = (|\mathcal{V}_{\text{val}}|\mathcal{L}_{\text{val}} + |\mathcal{V}_{\text{test}}|\mathcal{L}_{\text{unsup}})/(|\mathcal{V}_{\text{val}}| + |\mathcal{V}_{\text{test}}|)$ for GCKM, or highest validation accuracy for the baselines. All results are reported in Table 2. We observe that GCKM outperforms the GNN baselines in 5 out of 6 cases. However, the usage of a multiview kernel in GCKM-MV further improves the model, especially for the heterophilious datasets, such that it additionally outperforms the basic kernel methods for all datasets.

Sparsity property In our experiments, we observe that the linear system (11), is very sparse ($\sim 98\%$). This can be explained by the fact that the r_i values are small and appear quadratically in the equation. However, to be able to fully exploit this sparsity and scale up the method to very large graphs, we have to find a good implementation of a sparse solver as proposed by Benzi, Golub, and Liesen (2005) or similar.

Semi-supervised node classification with standard split and few validation labels In this experiment, we use the standard fixed splits for Cora, CiteSeer and PubMed, with 20 training labels per class and 1000 test labels, but the experiment is repeated for different validation sets. We perform a random search for the hyperparameters and report in each run the unsupervised metric $\mathcal{L}_{\text{unsup}}$ on the test set, as well as the validation accuracies $\mathcal{L}_{\text{val}}$ for each validation set. For each validation set, we selected the model that had the highest combined score: $\mathcal{L}_{\text{comb}} = (|\mathcal{V}_{\text{val}}|\mathcal{L}_{\text{val}} + |\mathcal{V}_{\text{test}}|\mathcal{L}_{\text{unsup}})/(|\mathcal{V}_{\text{val}}| + |\mathcal{V}_{\text{test}}|)$. Table 3 shows the resulting performances of the experiment. When few validation labels are given, we see that GCKM achieves best performance on all but two cases. Even with no validation labels, when only the unsupervised metric is used, GCKM is able to obtain a

[2]We use 5-fold crossvalidation because the validation sets might be too small in a 10-fold.

METHOD	CHAM.	SQUI.	CORA	CITE.	PUBM.	OGB-ARXIV
SVM-RBF	29.37	29.09	38.84	33.94	62.23	36.41
SVM-DIFF	27.95	19.89	75.50	47.96	39.19	43.17
SVM-WWL	27.73	28.37	44.34	39.12	71.87	42.23
MLP	22.20±1.22	23.50±2.33	40.95±1.47	50.05±1.38	68.65±0.59	31.17±5.67
GCN	25.09±3.05	23.22±2.37	76.70±3.41	64.29±0.96	76.68±0.36	38.06±4.22
APPNP	25.07±2.50	22.59±2.29	80.06±0.45	66.46±0.92	77.18±0.46	37.80±4.14
GPR-GNN	29.58±3.06	24.93±3.57	80.16±0.66	64.58±1.27	76.95±2.96	36.28±6.60
BERNNET	29.17±3.07	24.59±2.59	80.68±0.62	64.88±1.03	**77.51±0.33**	40.49±0.24
CHEBNETII	30.07±0.83	24.58±2.50	78.86±0.55	67.26±0.68	74.84±0.76	54.98±0.09
GCKM (OURS)	30.17	25.38	80.74	67.53	75.99	**58.03**
GCKM-MV(OURS)	**38.15**	**29.11**	**81.05**	**68.44**	75.32	57.63

Table 2: Mean test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification with fixed splits where fewer labels are given. The best model is highlighted in bold for each dataset. Since GCKM and the SVM baselines have a deterministic training procedure, no confidence intervals are reported.

	METHOD	VALIDATION LABELS PER CLASS 0	1	5	10
CORA	GCN	-	81.18±0.66	79.50±0.56	80.51±0.25
	CHEBNETII	-	60.15±2.13	77.67±1.03	79.79±1.07
	GCKM	**82.40**	**82.40**	**82.40**	**81.90**
CITE.	GCN	-	63.91±0.95	67.71±1.30	**69.22±1.07**
	CHEBNETII	-	51.52±4.40	67.72±1.06	68.13±1.14
	GCKM	**68.10**	**68.10**	**68.10**	68.10
PUBM.	GCN	-	75.43±0.41	76.42±0.79	74.52±0.53
	CHEBNETII	-	63.31±2.30	**76.79±0.95**	71.65±0.70
	GCKM	**76.80**	**76.60**	76.60	**76.80**

Table 3: Test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification on Cora, CiteSeer and PubMed with fixed split for smaller validation set sizes. The best performing model is highlighted in bold.

decent performing model, whereas it is not possible to do early stopping or even select a model with the other methods. We conclude that GCKM is less sensitive to a decrease in validation set size than these baseline methods. A Table containing the performances on all datasets using the standard splits with full validation set is given in Appendix.

Ablation studies We empirically assess the effect of the number of GCKM layers in our model, and compare it to a GCN with the same amount of message passing layers. Results for the Cora dataset are given in Table 4. We observe that the performance decrease for GCKM is less than GCN and that GCKM scales linearly with model depth. Next, we perform an ablation study to evaluate the unsupervised setting separately and compare to DMoN (Tsitsulin et al. 2023). We use deep GCKM with only two unsupervised layers, with RBF bandwidth $\sigma^2_{\text{RBF}} = m\sigma^2$ with m the input dimension and σ^2 the variance of the inputs, and clustering obtained by k-means on $\boldsymbol{H}^{(2)}$. Table 5 shows that our method is also effective for the node clustering task. In Appendix, we also compare our model with two simplified versions to study the effect of the aggregation step and GCKM-layers, and discuss the computational complexity.

LAYERS	1	2	4	8	10
GCN	63.60	76.70	74.86	70.37	72.69
GCKM	72.09	80.74	80.27	80.31	79.84
TIME (S)	9.1	12.9	19.8	22.6	27.3

Table 4: Test accuracy (%) and computation time for different numbers of layers on Cora in semi-supervised setting with few labels.

Method	Cora	Cite.	Pubm.	Cham.	Squi.
DMoN	48.8	33.7	**29.8**	14.0	3.7
GCKM	**49.0**	**37.5**	25.2	**14.9**	**5.4**

Table 5: NMI performance in the unsupervised setting.

Discussion

We now recapitulate some key properties of our model and contrast this with related methods: (i) Since the depth of our model is obtained by combining multiple kernel machines, we are able to use simple kernel functions such as the RBF-kernel. (ii) We directly train the dual variables, which are the node representations themselves. Inversely, Chen, Jacob, and Mairal (2020) uses graph kernels and approximates them using the Nyström-method to work in a primal representation. Nevertheless, our modular framework easily allows to augment or replace the kernel function by a graph kernel, at any layer. (iii) The layerwise structure of our deep kernel machine yields an effective initialization scheme and levelwise objective function during finetuning, which can yield good conditioning of the training process. Other methods construct a deep feature map but train it in a shallow kernel machine setting (Du et al. 2019; Chen, Jacob, and

Mairal 2020). (iv) Our method uses a 1-hop message passing scheme to learn the graph topology, whereas most state-of-the-art convolutional GNNs use higher order polynomial filters (He et al. 2021; He, Wei, and Wen 2022). (v) Our framework, is highly modular. One can for example easily extend it to directed graphs or categorical data by choosing appropriate aggregation functions and kernel functions (e.g., (Couto 2005)) respectively. We refer the reader to Table 1 for a qualitative comparison with some different models.

On one hand, the end-to-end objective (12) maximizes the variance in each hidden node representation. On the other hand, the last three terms yield a spectral clustering of a learned similarity graph, where each cluster is pushed towards the few labeled nodes. Because of this regularization on the hidden representations, and the fact that the readout layer is based on an unsupervised core model (augmented with a supervised term for the few labels), our model achieves good generalization capabilities. Furthermore, we include an unsupervised validation metric in our model selection task. We believe that these characteristics of our model explain the performances in Table 2, Table 3, and Table 5. Further, Table 2 shows a significant performance gain of GCKM-MV over GCKM for the heterophilious graphs. This can be explained by the fact that the second view in the last layer is constructed of the initial node attributes, which yields a similar effect as skip connections in a conventional GNN.

Conclusion

We introduce GCKM, a new approach for message passing in graphs based on a deep kernel machine with convolutional aggregation functions. In GCKM, intermediate node representations are embedded in an infinite-dimensional feature space through the use of duality. We derive optimization problems for the unsupervised and semi-supervised building blocks and we show optimization algorithms for end-to-end training for the semi-supervised node classification task. Experiments on several benchmark datasets verify the effectiveness of the proposed method in its elementary form. Thanks to the unsupervised core, our model outperforms current state-of-the-art GNNs and kernel methods when few labels are available for training, which can be a considerable advantage in real-world applications. Furthermore, it does so consistently for both heterophilious and homophilious graphs. Many directions for future work exist, such as extending the method to inductive tasks or attentional message passing and investigating methods for scaling up the kernel machines to very large graphs.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. This work was supported in part by the KU Leuven Research Council (Optimization frameworks for deep kernel machines C14/18/068); the Flemish Government FWO projects GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant; Flemish Government (AI Research Program). Sonny Achten, Francesco Tonin, Panagiotis Patrinos, and Johan Suykens are also affiliated with Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. The following references appear in Appendix only: Lehoucq, Sorensen, and Yang (1998); Boyd and Vandenberghe (2004).

References

Bacciu, D.; Errica, F.; Micheli, A.; and Podda, M. 2020. A Gentle Introduction to Deep Learning for Graphs. *Neural Networks*, 129: 203–221.

Belkin, M.; Ma, S.; and Mandal, S. 2018. To understand deep learning we need to understand kernel learning. In *ICML*.

Bengio, Y. 2009. Learning Deep Architectures for AI. *Foundations*, 2: 1–55.

Benzi, M.; Golub, G. H.; and Liesen, J. 2005. Numerical solution of saddle point problems. *Acta Numerica*, 14: 1–137.

Boyd, S.; and Vandenberghe, L. 2004. *Convex Optimization*. New York: Cambridge University Press.

Burges, C. J. 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2: 121–167.

Chen, D.; Jacob, L.; and Mairal, J. 2020. Convolutional Kernel Networks for Graph-Structured Data. In *ICML*.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

Christmann, A.; and Steinwart, I. 2008. *Support Vector Machines*. NY: Springer New York.

Couto, J. 2005. Kernel K-Means for Categorical Data. In *Advances in Intelligent Data Analysis VI*, 46–56.

Du, S. S.; Hou, K.; Póczos, B.; Salakhutdinov, R.; Wang, R.; and Xu, K. 2019. Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. In *NIPS*.

Feng, A.; You, C.; Wang, S.; and Tassiulas, L. 2022. KerGNNs: Interpretable Graph Neural Networks with Graph Kernels. In *AAAI*.

Ghosh, S.; Das, N.; Gonçalves, T.; Quaresma, P.; and Kundu, M. 2018. The journey of graph kernels through two decades. *Computer Science Review*, 27: 88–111.

Hamilton, W. L. 2020. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3): 1–159.

Hammer, B.; and Gersmann, K. 2003. A Note on the Universal Approximation Capability of Support Vector Machines. *Neural Processing Letters*, 17: 43–53.

He, M.; Wei, Z.; Huang, Z.; and Xu, H. 2021. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. In *NIPS*.

He, M.; Wei, Z.; and Wen, J.-R. 2022. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. In *NIPS*.

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2): 251–257.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5): 359–366.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NIPS*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Kriege, N. M.; Johansson, F. D.; and Morris, C. 2020. A survey on graph kernels. *Applied Network Science*, 5(1): 1–42.

Langone, R.; Mall, R.; and Suykens, J. A. K. 2013. Soft Kernel Spectral Clustering. In *IJCNN*.

Lehoucq, R. B.; Sorensen, D. C.; and Yang, C. 1998. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. Philadelphia, PA: SIAM.

Lei, T.; Jin, W.; Barzilay, R.; and Jaakkola, T. 2017. Deriving Neural Architectures from Sequence and Graph Kernels. In *ICML*.

Li, J.; Li, F.; and Todorovic, S. 2019. Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform. In *ICLR*.

Lim, D.; Hohne, F.; Li, X.; Huang, S. L.; Gupta, V.; Bhalerao, O.; and Lim, S.-N. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *NIPS*.

Mehrkanoon, S.; Alzate, C.; Mall, R.; Langone, R.; and Suykens, J. A. K. 2015. Multiclass Semisupervised Learning Based Upon Kernel Spectral Clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 26(4): 20–33.

Mercer, J. 1909. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, Series A*, 209(441-458): 415–446.

Nikolentzos, G.; Meladianos, P.; Tixier, A. J.-P.; Skianis, K.; and Vazirgiannis, M. 2018. Kernel Graph Convolutional Neural Networks. In *ICANN*.

Pandey, A.; Fanuel, M.; Schreurs, J.; and Suykens, J. A. K. 2022. Disentangled Representation Learning and Generation With Manifold Optimization. *Neural Computation*, 34(10): 2009–2036.

Rockafellar, R. T. 1974. *Conjugate Duality and Optimization*. SIAM.

Rozemberczki, B.; Allen, C.; and Sarkar, R. 2021. Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(1): 1–22.

Salakhutdinov, R. 2015. Learning Deep Generative Models. *Annual Review of Statistics and Its Application*, 2(1): 361–385.

Schölkopf, B.; and Smola, A. 2002. *Learning with Kernels*. Cambridge, MA: MIT Press.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *AI magazine*, 29(3): 93–93.

Smola, A. J.; and Kondor, R. 2003. Kernels and Regularization on Graphs. In *Learning Theory and Kernel Machines*, 144–158.

Suykens, J. A. K. 2017. Deep Restricted Kernel Machines Using Conjugate Feature Duality. *Neural Computation*, 29(8): 2123–2163.

Suykens, J. A. K.; Van Gestel, T.; De Brabanter, J.; De Moor, B.; and Vandewalle, J. 2002. *Least Squares Support Vector Machines*. Singapore: World Scientific.

Suykens, J. A. K.; Van Gestel, T.; Vandewalle, J.; and De Moor, B. 2003. A support vector machine formulation to PCA analysis and its Kernel version. *IEEE Transactions on Neural Networks*, 14(2): 447–450.

Suykens, J. A. K.; and Vandewalle, J. 1999. Least Squares Support Vector Machine Classifiers. *Neural Processing Letters*, 9(3): 293–300.

Tao, Q.; Tonin, F.; Patrinos, P.; and Suykens, J. A. K. 2022. Tensor-based Multi-view Spectral Clustering via Shared Latent Space.

Togninalli, M.; Ghisu, E.; Llinares-López, F.; Rieck, B.; and Borgwardt, K. 2019. Wasserstein Weisfeiler-Lehman Graph Kernels. In *NIPS*.

Tonin, F.; Patrinos, P.; and Suykens, J. A. K. 2021. Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints. *Neural Networks*, 142: 661–679.

Tsitsulin, A.; Palowitch, J.; Perozzi, B.; and Müller, E. 2023. Graph Clustering with Graph Neural Networks. *Journal of Machine Learning Research*, 24(127): 1–21.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

Weisfeiler, B.; and Lehman, A. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9): 12–16.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*.

List of symbols

We provide a list of symbols with short descriptions in Table 6.

Proofs and derivation of the Graph Convolutional Kernel Machine layer

Proof 1

The minimization problem in primal and dual variables is given by:

$$\min_{\boldsymbol{W},\boldsymbol{h}_v} \bar{J} \triangleq -\sum_{v=1}^{n} \phi_c(\boldsymbol{a}_v)^T \boldsymbol{W}\boldsymbol{h}_v + \frac{1}{2}\sum_{v=1}^{n} \boldsymbol{h}_v^T \boldsymbol{\Lambda}\boldsymbol{h}_v + \frac{\eta}{2}\text{Tr}(\boldsymbol{W}^T\boldsymbol{W}). \quad (13)$$

Lemma. *The solution to the dual minimization problem:*

$$\min_{\boldsymbol{H}} -\frac{1}{2\eta} Tr(\boldsymbol{H}^T \boldsymbol{K}_c(\boldsymbol{X},\mathcal{E})\boldsymbol{H}) \quad \textit{s.t. } \boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I}_s, \quad (14)$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ *as* (13) *when the hyperparameters* $\boldsymbol{\Lambda}$ *in* (13) *are chosen to equal the symmetric part of the Lagrange multipliers* $\boldsymbol{Z}$ *of the equality constraints in* (14)*; i.e.,* $\boldsymbol{\Lambda} = (\boldsymbol{Z} + \boldsymbol{Z}^T)/2$.

Proof. The stationarity conditions of (13) are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial \boldsymbol{W}} = 0 \iff & \boldsymbol{W} = \frac{1}{\eta}\sum_{v=1}^{n} \phi_c(\boldsymbol{a}_v)\boldsymbol{h}_v^T \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_v} = 0 \iff & \boldsymbol{h}_v\boldsymbol{\Lambda} = \boldsymbol{W}^T\phi_c(\boldsymbol{a}_v), \end{cases}$$

or equivalently,

$$\begin{cases} \boldsymbol{W} = \frac{1}{\eta}\boldsymbol{\Phi}_c^T\boldsymbol{H} \\ \boldsymbol{h}_v\boldsymbol{\Lambda} = \boldsymbol{W}^T\phi_c(\boldsymbol{a}_v) \iff \boldsymbol{H}\boldsymbol{\Lambda} = \boldsymbol{\Phi}_c\boldsymbol{W}, \end{cases} \quad (15)$$

where $\boldsymbol{\Phi}_c = [\phi_c(\boldsymbol{a}_1), \ldots, \phi_c(\boldsymbol{a}_n)]^T$. Given that $\boldsymbol{K}_c = \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c^T$, and by substituting the first condition into the second, one can eliminate the primal variable $\boldsymbol{W}$:

$$\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{H} = \boldsymbol{H}\boldsymbol{\Lambda}. \quad (16)$$

Next, the Lagrangian of (14) is:

$$\mathcal{L}(\boldsymbol{H},\boldsymbol{Z}) = -\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{K}_c\boldsymbol{H}) + \frac{1}{2}\text{Tr}(\boldsymbol{Z}^T(\boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s)),$$

and the Karush-Kuhn-Tucker conditions are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{H}} = -\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{H} + \boldsymbol{H}(\boldsymbol{Z} + \boldsymbol{Z}^T)/2 = 0 \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{Z}} = \boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s = \boldsymbol{0}_s. \end{cases} \quad (17)$$

By choosing $\boldsymbol{\Lambda} = \tilde{\boldsymbol{Z}} = (\boldsymbol{Z} + \boldsymbol{Z}^T)/2$ we obtain (16) from the first condition, which proves the Lemma. □

Proof 2

Proposition. *Given a symmetric matrix* $\boldsymbol{K}_c$ *with eigenvalues* $\lambda_1 \geq \cdots \geq \lambda_s > \lambda_{s+1} \geq \cdots \geq \lambda_n \geq 0$*, and* $\eta > 0$ *a hyperparameter; and let* $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s$ *be the columns of* $\boldsymbol{H}$*; then* $\boldsymbol{H}$ *is a minimizer of* (14) *if and only if* $\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I}_s$ *and span*$(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s) =$ *span*$(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$*, where* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ *are the eigenvectors of* $\boldsymbol{K}_c$ *corresponding to the* s *largest eigenvalues.*

Proof. Let us define the columns of $\boldsymbol{H}$ as $\boldsymbol{g}_i = [(\boldsymbol{h}_1)_i, \ldots, (\boldsymbol{h}_n)_i]^T$, and $\boldsymbol{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_s] = \boldsymbol{H}$. We can then rewrite (17) in vector notation:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{g}_i} = -\frac{1}{\eta}\boldsymbol{K}_c\boldsymbol{g}_i + \sum_{j=1}^{s} \tilde{Z}_{ij}\boldsymbol{g}_j = \boldsymbol{0}_n \quad \forall i = 1 \ldots s \\ \frac{\partial \mathcal{L}}{\partial \tilde{Z}_{ij}} = \boldsymbol{g}_i^T\boldsymbol{g}_j - \delta_{ij} = 0 \quad \forall ij = 1 \ldots s \end{cases} \quad (18)$$

where δ_{ij} is the Kronecker delta, and derive the second order derivatives for the optimization parameters:

$$\nabla^2_{\boldsymbol{g}_i\boldsymbol{g}_j}\mathcal{L} = -\frac{\delta_{ij}}{\eta}\boldsymbol{K}_c + \tilde{Z}_{ij}\boldsymbol{I}_n. \quad (19)$$

By defining $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_s]$, the second order necessary conditions can be formulated as:

$$\sum_{i=1}^{s}\sum_{j=1}^{s} \boldsymbol{d}_i^T(-\frac{\delta_{ij}}{\eta}\boldsymbol{K}_c + \tilde{Z}_{ij}\boldsymbol{I}_n)\boldsymbol{d}_j = -\frac{1}{\eta}\sum_{i=1}^{s} \boldsymbol{d}_i^T\boldsymbol{K}_c\boldsymbol{d}_i + \sum_{i=1}^{s}\sum_{j=1}^{s} \tilde{Z}_{ij}\boldsymbol{d}_i^T\boldsymbol{d}_j \geq 0 \quad \forall \boldsymbol{D} \in C(\boldsymbol{G}^*), \quad (20)$$

where $C(\boldsymbol{G}^*) = \{\boldsymbol{D} \in \mathbb{R}^{n\times s} \mid \boldsymbol{D}^T\boldsymbol{G}^* = \boldsymbol{0}_s\}$ is the critical cone at $\boldsymbol{G}^*$. For the critical cone, it can be deduced that the following properties hold:

$$\begin{aligned} \boldsymbol{d}_i^T\boldsymbol{g}_j^* + \boldsymbol{d}_j^T\boldsymbol{g}_i^* = 0 \quad & \forall ij = 1, \ldots, s \\ \boldsymbol{d}_i^T\boldsymbol{g}_i^* = 0 \quad & \forall i = 1, \ldots, s \\ \boldsymbol{d}_i^T\boldsymbol{d}_j = 0 \quad & \forall ij = 1, \ldots, s \quad i \neq j. \end{aligned}$$

Without loss of generality, we further assume $\|\boldsymbol{d}_i\| = 1$. From (18), one can derive $\tilde{Z}_{ij} = \frac{1}{\eta}\boldsymbol{g}_j^{*T}\boldsymbol{K}_c\boldsymbol{g}_i^*$. By substituting this and $\boldsymbol{d}_i^T\boldsymbol{d}_j = 0$ in (20), the second order necessary condition becomes:

$$\sum_{i=1}^{s} \frac{1}{\eta}\boldsymbol{g}_i^{*T}\boldsymbol{K}_c\boldsymbol{g}_i^* \geq \sum_{i=1}^{s} \frac{1}{\eta}\boldsymbol{d}_i^T\boldsymbol{K}_c\boldsymbol{d}_i,$$

or after reworking this algebraically:

$$\text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) \geq \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}), \quad (21)$$

with $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ the eigenvectors of $\boldsymbol{K}_c$ with corresponding eigenvalues $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$. For the case where span$(\boldsymbol{g}_1^*, \ldots, \boldsymbol{g}_s^*) =$ span$(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s)$, the left hand side is maximal:

$$\sum_{i=1}^{s} \lambda_i = \text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) \geq \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}).$$

The right hand side becomes maximal when span$(\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n) =$ span$(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$. In this case,

Symbol	Space	Description
$\mathbf{0}_n$	$\{0\}^n$	n-dimensional vector of all zeros
$\mathbf{1}_n$	$\{1\}^n$	n-dimensional vector of all ones
δ_{ij}	$\{0,1\}$	Kronecker delta
η	$\mathbb{R}_+$	hyperparameter
λ	$\mathbb{R}_+$	hyperparameter
$\boldsymbol{\Lambda}$	$\mathbb{R}^{s\times s}_{\succ 0}$	symmetric positive definite hyperparameter matrix
$\phi(\cdot)$	$\mathbb{R}^d \to \mathbb{R}^{d_f}$	feature map, transforming an input from a d-dimensional space to a d_f-dimensional space
$\phi_c(\cdot)$	$\mathbb{R}^d \to \mathbb{R}^{d_f}$	centered feature map $\phi_c(\cdot)\phi(\cdot) - \Sigma_i\phi(x_i)/n$
Φ	$\mathbb{R}^{n\times d_f}$	matrix containing all feature maps as row vectors
Φ_c	$\mathbb{R}^{n\times d_f}$	matrix containing all centered feature maps as row vectors
$\psi(\cdot,\cdot)$	$\mathbb{R}^s \times \{\{\mathbb{R}^s\}\} \to \mathbb{R}^s$	aggregation function
$\boldsymbol{a}_v$	$\mathbb{R}^d$	vector containing the aggregated node features
$\boldsymbol{b}$	$\mathbb{R}^p$	bias vector
$\boldsymbol{c}_i$	$\{-1,1\}^p$	the class encoding of point i
$\boldsymbol{C}$	$\{-1,1\}^{n\times p}$	matrix containing all class encodings as row vectors
d_v	$\mathbb{N}$	node degree
$\boldsymbol{e}_i$	$\mathbb{R}^s$ or $\mathbb{R}^p$	error variable
$\boldsymbol{h}_i$	$\mathbb{R}^s$ or $\mathbb{R}^p$	dual variable/hidden representation (GCKMℓ) or dual variable (Semi-SupRKM)
$\boldsymbol{H}$	$\mathbb{R}^{n\times s}$ or $\mathbb{R}^{n\times p}$	matrix containing all dual vectors as row vectors
$\boldsymbol{I}_n$	$\{0,1\}^{n\times n}$	n by n identity matrix
$k(\cdot,\cdot)$	$\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$	a positive definite kernel function
$\boldsymbol{K}$	$\mathbb{R}^{n\times n}$	kernel matrix $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, or Gram matrix $K_{ij} = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$
l_i	$\{0,1\}$	indicator that is equal to one when the label of i is used for training
$\boldsymbol{L}$	$\{0,1\}^{n\times n}$	diagonal matrix containing all label indicators on the diagonal
p	$\mathbb{R}$	dimensionality of the final representation (equals the number of classes in one-vs-all encoding)
r_i	$\mathbb{R}$	a weighted combination of v_i and l_i
$\boldsymbol{R}$	$\mathbb{R}^{n\times n}$	diagonal matrix containing all r variables on the diagonal
s	$\mathbb{R}$	dimensionality of a hidden representation
$\boldsymbol{S}$	$\mathbb{R}^{n\times n}$	see Section of SemiSupRKM
v_i	$\mathbb{R}$	weighting scalar for the datapoints in Semi-SupRKM (equals the inverse degree of the kernel matrix)
$\boldsymbol{W}$	$\mathbb{R}^{d_f\times s}$	linear transformation matrix
$\boldsymbol{x}_i$	$\mathbb{R}^d$	the feature vector of datapoint i
y_i	$\mathbb{N}$	class label of datapoint i
$\boldsymbol{Z}$	$\mathbb{R}^{s\times s}$	a matrix containing Lagrange multipliers

Table 6: List of symbols

there exists an orthonormal transformation matrix $\boldsymbol{O}$ such that $\boldsymbol{G}^* = \boldsymbol{D}\boldsymbol{O}$:

$$\sum_{i=1}^{s}\lambda_i = \text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*)$$
$$= \text{Tr}(\boldsymbol{O}^T\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}\boldsymbol{O}) = \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}).$$

We verified that the second order necessary conditions are satisfied for in the case $\text{span}(\boldsymbol{g}_1^*,\dots,\boldsymbol{g}_s^*) = \text{span}(\boldsymbol{v}_1,\dots,\boldsymbol{v}_s)$. Let us now proceed by assuming that $\text{span}(\boldsymbol{g}_1^*,\dots,\boldsymbol{g}_s^*) \neq \text{span}(\boldsymbol{v}_1,\dots,\boldsymbol{v}_s)$. In this case there exists a matrix D such that (21) becomes:

$$\text{Tr}(\boldsymbol{G}^{*T}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{G}^*) < \text{Tr}(\boldsymbol{D}^T\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\boldsymbol{D}) \leq \sum_{i=1}^{s}\lambda_i.$$

We thus established that the second order necessary conditions are satisfied if and only if $\text{span}(\boldsymbol{g}_1^*,\dots,\boldsymbol{g}_s^*) = \text{span}(\boldsymbol{v}_1,\dots,\boldsymbol{v}_s)$. Generally, the second order condition is not sufficient. However, as the feasible set $\{\boldsymbol{H} \in \mathbb{R}^{N\times s} \mid \boldsymbol{H}^T\boldsymbol{H} - \boldsymbol{I}_s = \boldsymbol{0}_s\}$ is compact, and the objective function $-\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{K}_c\boldsymbol{H})$ is concave, (14) is guaranteed to have a global minimizer (Boyd and Vandenberghe 2004). Therefore, any $\boldsymbol{H}^*$ such that $\text{range}(\boldsymbol{H}^*) = span(\boldsymbol{v}_1 \dots \boldsymbol{v}_s)$ is a global minimizer. □

Proof 3

Lemma. *Let $n = |\mathcal{V}_{tr}|$, $m = |\mathcal{V}|$, and $\boldsymbol{K}^{\mathcal{V}_1,\mathcal{V}_2} \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$ a kernel matrix containing kernel evaluations of all elements of set $\mathcal{V}_1$ w.r.t. all elements of set $\mathcal{V}_2$ (i.e., $K_{uv}^{\mathcal{V}_1,\mathcal{V}_2} = k(\boldsymbol{a}_u, \boldsymbol{a}_v)\ \forall u \in \mathcal{V}_1, \forall v \in \mathcal{V}_2$). The dual representations can then be obtained using:*

$$\hat{\boldsymbol{H}}_{\mathcal{V}} = \frac{1}{\eta}\boldsymbol{K}^{\mathcal{V},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}}\boldsymbol{\Lambda}^{-1} - \frac{\boldsymbol{1}_m\boldsymbol{1}_n^T\boldsymbol{K}^{\mathcal{V}_{tr},\mathcal{V}_{tr}}\boldsymbol{H}_{\mathcal{V}_{tr}}}{n\eta}\boldsymbol{\Lambda}^{-1}, \quad (22)$$

Proof. We start from the vector formulation of the stationarity conditions (15). From the second condition, we get $\hat{\boldsymbol{e}}_v = \hat{\boldsymbol{h}}_v \boldsymbol{\Lambda} = \boldsymbol{W}^T \phi_c(\hat{\boldsymbol{a}}_v)$. By substituting the first condition into this, we obtain:

$$\hat{\boldsymbol{e}}_v = \hat{\boldsymbol{h}}_v \boldsymbol{\Lambda} = \frac{1}{\eta} \sum_{u \in \mathcal{V}_{\text{tr}}} \boldsymbol{h}_u \phi_c(\boldsymbol{a}_u)^T \phi_c(\hat{\boldsymbol{a}}_v),$$

where $\mathcal{V}_{\text{tr}}$ is the set of nodes that is used for training. By doing this for every node in set $\mathcal{V}$, and writing it in matrix notation, we obtain:

$$\hat{\boldsymbol{E}}_{\mathcal{V}} = \hat{\boldsymbol{H}}_{\mathcal{V}} \boldsymbol{\Lambda} = \frac{1}{\eta} \boldsymbol{K}_c^{\mathcal{V}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}.$$

Or in terms of the uncentered kernel matrix:

$$\hat{\boldsymbol{E}}_{\mathcal{V}} = \hat{\boldsymbol{H}}_{\mathcal{V}} \boldsymbol{\Lambda} = \frac{1}{\eta} \boldsymbol{K}^{\mathcal{V}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}} - \frac{\boldsymbol{1}_m \boldsymbol{1}_n^T \boldsymbol{K}^{\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta},$$

and thus:

$$\hat{\boldsymbol{H}}_{\mathcal{V}} = \frac{1}{\eta} \boldsymbol{K}^{\mathcal{V}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}} \boldsymbol{\Lambda}^{-1} - \frac{\boldsymbol{1}_m \boldsymbol{1}_n^T \boldsymbol{K}^{\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta} \boldsymbol{\Lambda}^{-1}.$$

□

Proof 4

Proposition. *Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, the aggregated node features $\{a_v : v \in \mathcal{V}_{tr}\}$ and latent representations $\boldsymbol{H}_{\mathcal{V}_{tr}}$ of the training nodes $\mathcal{V}_{tr}$, and a local aggregation function $\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\})$ that is permutation invariant; the mapping f from $\mathcal{G}$ to $\mathcal{G}' = (\mathcal{V}, \mathcal{E}, \hat{\boldsymbol{E}}_{\mathcal{V}})$ using* (22) *is equivariant w.r.t. any permutation $\pi(\mathcal{G})$, i.e., $\mathcal{G}' = f(\mathcal{G}) \iff \pi(\mathcal{G}') = f(\pi(\mathcal{G}))$.*

Proof. Since the aggregation function is permutation invariant, the kernel evaluations $k(\boldsymbol{a}_u, \boldsymbol{a}_v)$ are as well. Now, let the permutation $\pi(\cdot)$ be defined by a permutation matrix $\boldsymbol{P}$. When $\mathcal{G}$ gets permuted, the rows of the corresponding kernel matrix $\boldsymbol{K}_{\mathcal{V}, \mathcal{V}_{\text{tr}}}$ get permuted accordingly by construction. The first term in (22) thus becomes $\frac{1}{\eta} \boldsymbol{P} \boldsymbol{K}_{\mathcal{V}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}$. The second term is a matrix with constant rows, and is therefore permutation invariant: $\frac{\boldsymbol{1}_m \boldsymbol{1}_n^T \boldsymbol{K}_{\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta} = \boldsymbol{P} \frac{\boldsymbol{1}_m \boldsymbol{1}_n^T \boldsymbol{K}_{\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta}$. We thus get $\frac{1}{\eta} \boldsymbol{P} \boldsymbol{K}_{\mathcal{V}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}} - \boldsymbol{P} \frac{\boldsymbol{1}_m \boldsymbol{1}_n^T \boldsymbol{K}_{\mathcal{V}_{\text{tr}}, \mathcal{V}_{\text{tr}}} \boldsymbol{H}_{\mathcal{V}_{\text{tr}}}}{n\eta} = \boldsymbol{P} \hat{\boldsymbol{E}}_{\mathcal{V}}$, which proves the permutation equivariance of the mapping. □

Proof 5

Lemma. *A GCKMℓ that uses sum aggregation and a RBF-kernel is as expressive as an iteration of the Weisfeiler-Lehman graph isomorphism test (Weisfeiler and Lehman 1968).*

Proof. The theoretical results of Xu et al. (2019) showed that a message passing iteration of the form:

$$\boldsymbol{h}_v^{(l)} = f^{(l)} \left((1 + \epsilon^{(l)}) \cdot \boldsymbol{h}_v^{(l-1)} + \sum_{u \in \mathcal{N}_v} \boldsymbol{h}_u^{(l-1)} \right),$$

where $\epsilon^{(l)}$ can be a fixed or a learnable parameter, is maximally powerful in the class of message passing neural networks and as expressive as the one dimensional Weisfeiler-Lehman graph isomorphism test (Weisfeiler and Lehman 1968); and that this follows from the sum aggregator and the injectiveness of the transformation function $f^{(l)}$. Therefore, when GCKMℓ uses the sum aggregator (i.e., $\psi_{\text{sum}}(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) = \boldsymbol{x}_v + \sum_{u \in \mathcal{N}_v} \boldsymbol{x}_u$), it satisfies the first condition. When the feature map is chosen to be an RBF-kernel, the second condition is also satisfied because any feature map of the RBF-kernel is injective (we refer to Proposition 4.54, Lemma 4.55, and Corollary 4.58 in Christmann and Steinwart (2008)). □

Remark 10. from an empirical perspective, Xu et al. (2019) proposed a multilayer perceptron with at least one hidden layer for the function $f^{(l)}$, motivated by the universal approximator theorem (Hornik, Stinchcombe, and White 1989; Hornik 1991). Anagolously, it has been established that SVMs using the RBF-kernel are universal approximators (Burges 1998; Hammer and Gersmann 2003).

Proofs and derivation of the Semi-Supervised Restricted Kernel Machine

Proof 1

The minimization problem in primal and dual variables is given by:

$$\min_{\boldsymbol{W}, \boldsymbol{h}_i, \boldsymbol{b}} \bar{J} \triangleq \frac{\eta}{2} \text{Tr}(\boldsymbol{W}^T \boldsymbol{W}) + \frac{1}{2} \sum_{i=1}^{n} r_i \boldsymbol{h}_i^T \boldsymbol{h}_i - \sum_{i=1}^{n} r_i (\boldsymbol{W}^T \phi(\boldsymbol{x}_i) + \boldsymbol{b})^T \boldsymbol{h}_i - \sum_{i=1}^{n} \frac{l_i}{\lambda_2} \boldsymbol{c}_i^T \boldsymbol{h}_i. \quad (23)$$

Lemma. *The solution to the dual minimization problem:*

$$\min_{\boldsymbol{H}} -\frac{1}{2\eta} Tr(\boldsymbol{H}^T \boldsymbol{R} \boldsymbol{K}(\boldsymbol{X}) \boldsymbol{R} \boldsymbol{H}) + \frac{1}{2} Tr(\boldsymbol{H}^T \boldsymbol{R} \boldsymbol{H}) - \frac{1}{\lambda_2} Tr(\boldsymbol{H}^T \boldsymbol{L} \boldsymbol{C}) \quad \text{s.t. } \boldsymbol{H}^T \boldsymbol{R} \boldsymbol{1}_n = \boldsymbol{0}_p \quad (24)$$

satisfies the same first order conditions for optimality w.r.t. $\boldsymbol{H}$ as (23) *where the Lagrange multipliers equal the bias $\boldsymbol{b}$.*

Proof. The stationarity conditions of (23) are:

$$\begin{cases} \frac{\partial \bar{J}}{\partial \boldsymbol{W}} = 0 \iff & \boldsymbol{W} = \frac{1}{\eta} \sum_i r_i \phi(\boldsymbol{x}_i) \boldsymbol{h}_i^T \\ \frac{\partial \bar{J}}{\partial \boldsymbol{h}_i} = 0 \iff & \boldsymbol{h}_i = (\boldsymbol{W}^T \phi(\boldsymbol{x}_i) + \boldsymbol{b}) + r_i^{-1} \frac{l_i \boldsymbol{c}_i}{\lambda_2} \\ \frac{\partial \bar{J}}{\partial \boldsymbol{b}} = 0 \iff & \sum_i r_i \boldsymbol{h}_i = 0, \end{cases}$$

or equivalently,

$$\begin{cases} \boldsymbol{W} = \frac{1}{\eta} \boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{H} \\ \boldsymbol{H} = \boldsymbol{\Phi} \boldsymbol{W} + \boldsymbol{1}_n \boldsymbol{b}^T + \frac{\boldsymbol{R}^{-1} \boldsymbol{L} \boldsymbol{C}}{\lambda_2} \\ \boldsymbol{H}^T \boldsymbol{R} \boldsymbol{1}_n = \boldsymbol{0}_p, \end{cases} \quad (25)$$

where $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_n)]^T$. Given that $\boldsymbol{K} = \boldsymbol{\Phi} \boldsymbol{\Phi}^T$, and by substituting the first condition into the second, one can eliminate the primal variable $\boldsymbol{W}$:

$$\begin{cases} \boldsymbol{H} = \frac{1}{\eta} \boldsymbol{K} \boldsymbol{R} \boldsymbol{H} + \boldsymbol{1}_n \boldsymbol{b}^T + \frac{\boldsymbol{R}^{-1} \boldsymbol{L} \boldsymbol{C}}{\lambda_2} \\ \boldsymbol{H}^T \boldsymbol{R} \boldsymbol{1}_n = \boldsymbol{0}_p. \end{cases} \quad (26)$$

Next, the Lagrangian of (24) is:

$$\mathcal{L}(\boldsymbol{H}, \boldsymbol{z}) = -\frac{1}{2\eta}\text{Tr}(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H}) + \frac{1}{2}\text{Tr}(\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{H}) - \frac{1}{\lambda_2}\text{Tr}(\boldsymbol{H}^T\boldsymbol{L}\boldsymbol{C}) - \boldsymbol{z}^T\boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n,$$

and the KKT conditions are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \boldsymbol{H}} = -\frac{1}{\eta}\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H} + \boldsymbol{R}\boldsymbol{H} - \frac{\boldsymbol{L}\boldsymbol{C}}{\lambda_2} - \boldsymbol{R}\boldsymbol{1}_n\boldsymbol{z}^T = \boldsymbol{0}_{n\times s} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}} = \boldsymbol{H}^T\boldsymbol{R}\boldsymbol{1}_n = \boldsymbol{0}_p. \end{cases}$$

By left multiplying the first condition with $\boldsymbol{R}^{-1}$, and moving all negative terms to the right, one obtains (26) with $\boldsymbol{z} = \boldsymbol{b}$. □

Derivations of Semi-SupRKM

We next proceed with some derivations of which some results are given in the main text.

Eliminating $\boldsymbol{W}$ from the stationarity conditions (25) yields the following linear system:

$$\left[\begin{array}{c|c} \frac{1}{\eta}\boldsymbol{K} - \boldsymbol{R}^{-1} & \boldsymbol{1}_n \\ \hline \boldsymbol{1}_n^T & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{R}\boldsymbol{H} \\ \hline \boldsymbol{b}^T \end{array}\right] = \left[\begin{array}{c} -\frac{1}{\lambda_2}\boldsymbol{R}^{-1}\boldsymbol{L}\boldsymbol{C} \\ \hline \boldsymbol{0}_p^T \end{array}\right].$$

Alternatively, using all stationarity conditions, the expression for the bias term becomes:

$$\boldsymbol{b}^T = -\frac{1}{\boldsymbol{1}_n^T\boldsymbol{R}\boldsymbol{1}_n}(\frac{1}{\eta}\boldsymbol{1}_n^T\boldsymbol{R}\boldsymbol{K}\boldsymbol{R}\boldsymbol{H} + \frac{1}{\lambda_2}\boldsymbol{1}_n^T\boldsymbol{L}\boldsymbol{C}), \quad (27)$$

and eliminating both $\boldsymbol{W}$ and $\boldsymbol{b}$ from the stationarity conditions then yields:

$$(\boldsymbol{I}_n - \frac{1}{\eta}\boldsymbol{R}\boldsymbol{S}\boldsymbol{K})\boldsymbol{R}\boldsymbol{H} = \frac{1}{\lambda_2}\boldsymbol{S}^T\boldsymbol{L}\boldsymbol{C}.$$

From the first stationarity condition, we can derive an expression that can be used for out-of-sample extensions:

$$\hat{\boldsymbol{e}} = \boldsymbol{W}^T\phi(\boldsymbol{x}) + \boldsymbol{b} = \frac{1}{\eta}\sum_i r_i\boldsymbol{h}_i\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) + \boldsymbol{b} = \frac{1}{\eta}\sum_i r_i\boldsymbol{h}_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \boldsymbol{b} \quad (28)$$

Note that the second condition gives $\boldsymbol{e}_i = \boldsymbol{h}_i - \frac{l_i\boldsymbol{c}_i}{r_i\lambda_2}$, which simplifies to $\boldsymbol{e}_i = \boldsymbol{h}_i$ for the unsupervised training points. One can thus directly infer the class label $\hat{\boldsymbol{y}}_v$ from the learned representation by comparing the class codes and select the one with closest Hamming distance to the error variable $\boldsymbol{e}_i$. For one-vs-all encoding, this simply corresponds to selecting the index with the highest value.

For $\eta = 1$, these results are the same as those of Mehrkanoon et al. (2015), where their dual variables $\boldsymbol{\alpha}^{(l)}$ correspond to the columns of $\boldsymbol{R}\boldsymbol{H}$.

Hyperparameter Search

Regarding hyperparameter selection, we tune the hyperparameters of GCKM by random search. The σ^2 of RBF kernels is tuned between e^{-3} and e^5, the employed degree of the polynomial kernel is $p \in \{1, 2\}$, and the t parameter is chosen between e^{-5} and e^5. Note that for $p = 1$, this is in fact the linear kernel. The number of components is tuned in $s \in \{16, 32, 64\}$. For OGB-Arxiv, this was extended to $s \in \{16, 32, 64, 128, 256\}$. The $\lambda^{(l)}$, $\eta^{(l)}$ are tuned between e^{-4} and e^4. Regarding the GNN baselines, we tune the hyperparameters of each tested method by grid search in the ranges suggested by the authors in their papers. Finally, for the SVM baselines, we tuned the models with a gridsearch where c and γ are between 10^{-5} and 10^5 with unit steps on a log-scale, and with h ranging from 0 to 5 for the WWL kernel.

Dataset statistics Table 7 summarizes the dataset statistics for chameleon (Rozemberczki, Allen, and Sarkar 2021), cora (Sen et al. 2008; Yang, Cohen, and Salakhudinov 2016), citeseer (Sen et al. 2008; Yang, Cohen, and Salakhudinov 2016), pubmed (Sen et al. 2008; Yang, Cohen, and Salakhudinov 2016), and ogb-arxiv (Hu et al. 2020), in which the class insensitive edge homophily ratio $\mathcal{H}(\mathcal{G})$ (Lim et al. 2021) is a measure for the level of homophily in the graph.

DATASET	NODES	EDGES	FEATURES
CHAMELEON	2,277	31,371	2,325
SQUIRREL	5,201	198,353	2,089
CORA	2,708	5,278	1,433
CITESEER	3,327	4,552	3,703
PUBMED	19,717	44,324	500
OGB-ARXIV	169,343	1,157,799	128

DATASET	CLASSES	$\mathcal{H}(\mathcal{G})$
CHAMELEON	5	0.041
SQUIRREL	5	0.031
CORA	7	0.766
CITESEER	6	0.627
PUBMED	3	0.664
OGB-ARXIV	40	0.421

Table 7: Dataset statistics.

Experimental results for semi-supervised node classification with standard fixed splits

The next experiment uses the standard fixed splits: for Chameleon and Squirrel, this means a 2.5%/2.5%/95% train/validation/test-split; For Cora, Citeseer and PubMed, there are 20 training labels per class, 500 validation labels and 1000 test labels; and for OGB-Arxiv, the standard split is a 53.7%/17.6%/28.7% train/validation/test-split, where the splits are temporal, based on publication date. For each dataset, we performed a random search to determine the hyperparameters and selected the model with highest validation accuracy. For the baseline models, we use the mean test accuracies and 95% confidence intervals as reported in He, Wei, and Wen (2022). Table 8 summarizes the results.

Comparing to GCN, we observe that for Squirrel and CiteSeer the performance of GCKM is similar, whereas for all other datasets, GCKM outperforms its GNN counterpart. Comparing our model to the models with more advanced

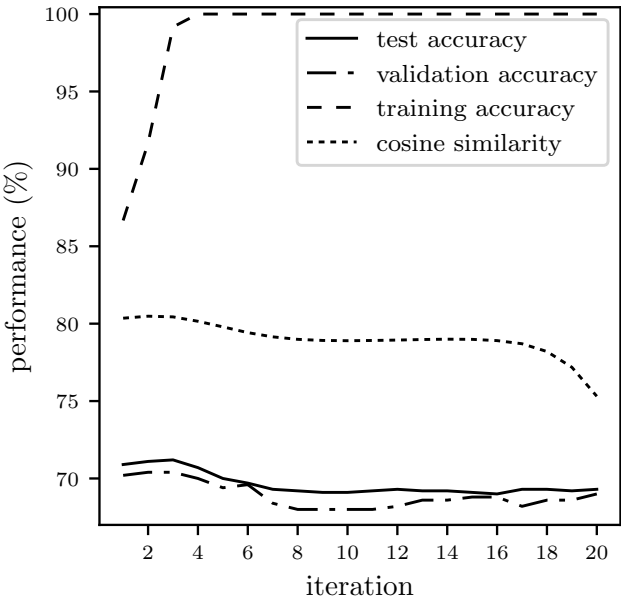

Figure 2: Train, validation and test accuracies, and cosine similarity score during training on CiteSeer dataset.

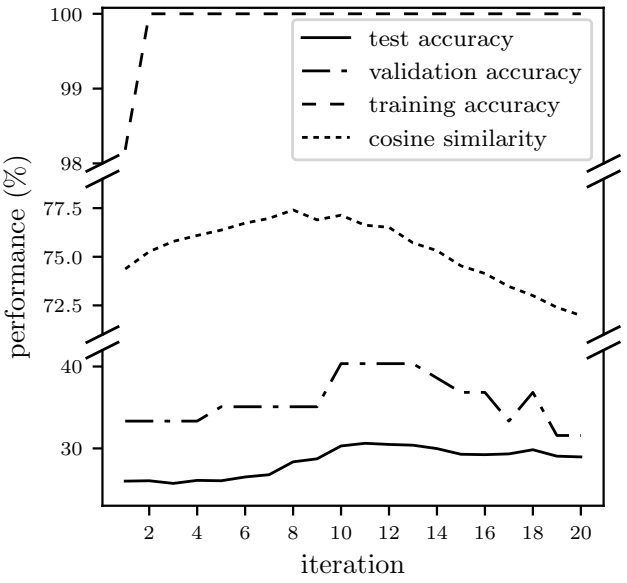

Figure 3: Train, validation and test accuracies, and cosine similarity score during training on Chameleon dataset.

aggregation techniques, we see that GCKM achieves second best performance on Chameleon and Squirrel.

Ablation studies

Analysis of initialization significance and validation metrics Figure 2 and 3 show the training progress on CiteSeer and Chameleon respectively of models where the dual variables were initialized by sequentially solving the shallow kernel machines. For Figure 2, the same hyperparameters were used as in Figure 4. By comparing Figure 2 with Figure 4, we observe that the test performance of the model after initialization is already better than that of the model without initialization after training. Further, we see that for CiteSeer, the finetuning increases the validation and test accuracy, as well as the cosine similarity score for a few iterations, after which these metrics decrease again. For Chameleon, we see a similar trend, though the performance increase is more clear. Generally, the finetuning phase after the sequential initialization only takes few iterations as observed. Next we see that the cosine similarity is also a good indicator for the early stopping of the finetuning phase. For CiteSeer, a homophilious dataset, the trend is indeed well aligned with that of the test accuracy, and although this is less the case for Chameleon, a heterophilious dataset, the general trend is aligned as well. We conclude that the initialization phase is crucial to obtain a good performance, and that the finetuning indeed helps further improving the model at a low cost. We also conclude that both validation accuracy as well as cosine similarity, which is unsupervised, are good indicators for the test performance.

Analysis of training progress and initialization

Analysis of training objective under random initialization Figure 4 shows the training progress on CiteSeer of a model where the dual variables were initialized randomly. Several observations can be made. First, It should be noted that the Cayley Adam algorithm (Li, Li, and Todorovic 2019) does not guarantee the exact feasibility of the orthogonality constraints. We thus observe a burn-in phase for a few iterations untill the orthogonality loss $\sum_{l=1}^{2} ||\boldsymbol{H}^{(l)^T}\boldsymbol{H}^{(l)} - \boldsymbol{I}||_F$ is small. After this burn-in, we see an almost monotonically decreasing training objective. Second, there is a fast increase in the training accuracy to 100%, whereas the validation and training performance keep increasing more gently, until they reach a certain level and start decreasing again. A third observation is that the cosine similarity increases only slightly for a few iterations until it keeps decreasing. Finally, we observe a sudden change in behavior around iteration 150, caused by another jump in the orthogonality loss. Overall we conclude that minimizing the objective under the constraints indeed yields improving generalization, up to some point, similarly as in training deep neural networks.

Computational complexity The proposed algorithm consists of two main computational parts: eigendecomposition for initialization of the unsupervised blocks and solving a linear system for the semi-supervised block. Initialization of $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}$ requires computing the first s eigenvectors of the kernel matrices $\boldsymbol{K}_c^{(1)}, \boldsymbol{K}_c^{(2)}$, respectively, with time complexity $\mathcal{O}(sn^2)$. This computation needs to be run only once and we exploit the symmetric structure of the kernel matrices by using the method of Lehoucq, Sorensen, and Yang (1998). Finding $\boldsymbol{H}^{(3)}$ in the employed alternating minimization algorithm requires solving the linear system (27) with worst-case complexity $\mathcal{O}(n^3)$. When using out-of-

METHOD	CHAM.	SQUI.	CORA	CITE.	PUBM.	OGB-ARXIV
MLP	21.91±2.11	23.42±0.94	58.88±0.62	56.97±0.54	73.15±0.28	47.27±0.64
GCN	39.14±0.60	30.06±0.75	81.32±0.18	71.77±0.21	79.15±0.18	71.74±0.29
APPNP	30.06±0.96	25.18±0.35	83.52±0.24	72.09±0.25	80.23±0.15	65.47±0.34
GPR-GNN	30.56±0.94	25.11±0.51	83.95±0.22	70.92±0.57	78.97±0.27	71.78±0.18
BERNNET	26.35±1.04	24.57±0.72	83.15±0.32	72.24±0.25	79.65±0.25	71.96±0.27
CHEBNETII	**46.45±0.53**	**36.18±0.46**	83.67±0.33	**72.75±0.16**	**80.48±0.23**	**72.32±0.23**
GCKM (OURS)	41.16	30.10	**84.20**	71.80	80.10	70.95

Table 8: Mean test accuracy (%) and 95% confidence interval (%) for semi-supervised node classification with standard fixed splits. The best model is highlighted in bold and the second best is underlined for each dataset. Since GCKM has a deterministic training procedure, no confidence intervals are reported.

sample extensions, one could use a subset of size $m \ll n$, such that the complexity scales w.r.t. m instead of n. First however, a thorough analysis is needed to study the effect of the out-of-sample extension on the hyperparameter choices as well as on the overall performance. Also, since the linear system is sparse, an efficient implementation of a sparse solver like proposed by (Benzi, Golub, and Liesen 2005) is expected to speed up computations significantly. Experiments are implemented in Python on a PC with a 3.7GHz Intel i7-8700K processor and 64GB RAM, and on a PC with 512GB RAM for ogbn-arxiv experiments.

Ablation studies We compare the performance of GCKM with that of two simplified alternatives. The first base model is also a GCKM with the same architecture, but the aggregation function is $\psi(\boldsymbol{x}_v, \{\{\boldsymbol{x}_u | u \in \mathcal{N}_v\}\}) = \boldsymbol{x}_v$ for both GCKMℓ's. This means that actually no aggregation happens and we will refer to this model as GCKM-NoAggr. The second base model is a shallow Semi-SupRKM, where we combine the information of the node features with that of the network structure in the kernel matrix, using a trade-off parameter α: $K_{uv} = \alpha\, k(\boldsymbol{x}_u, \boldsymbol{x}_v) + (1-\alpha)\, e_{u,v}$, where $e_{u,v}$ is a binary variable that indicates whether or not an edge is present between nodes u and v. By comparing GCKM with deepRKM and Semi-SupRKM in Table 9, we see the significance of the aggregation function and iterative message passing. We further refer the interested reader to Appendix for an empirical analysis of the initialization significance, validation metrics, training progress, and number of layers.

	SEMI-SUPRKM	GCKM-NOAGGR	GCKM
CHAM.	29.37	33.67	**41.16**
SQUI.	29.09	24.09	**30.10**
CORA	59.90	46.10	**84.20**
CITE.	67.60	58.50	**71.80**
PUBM.	75.00	67.80	**80.10**

Table 9: Test accuracy (%) of base models for semi-supervised setting with standard splits

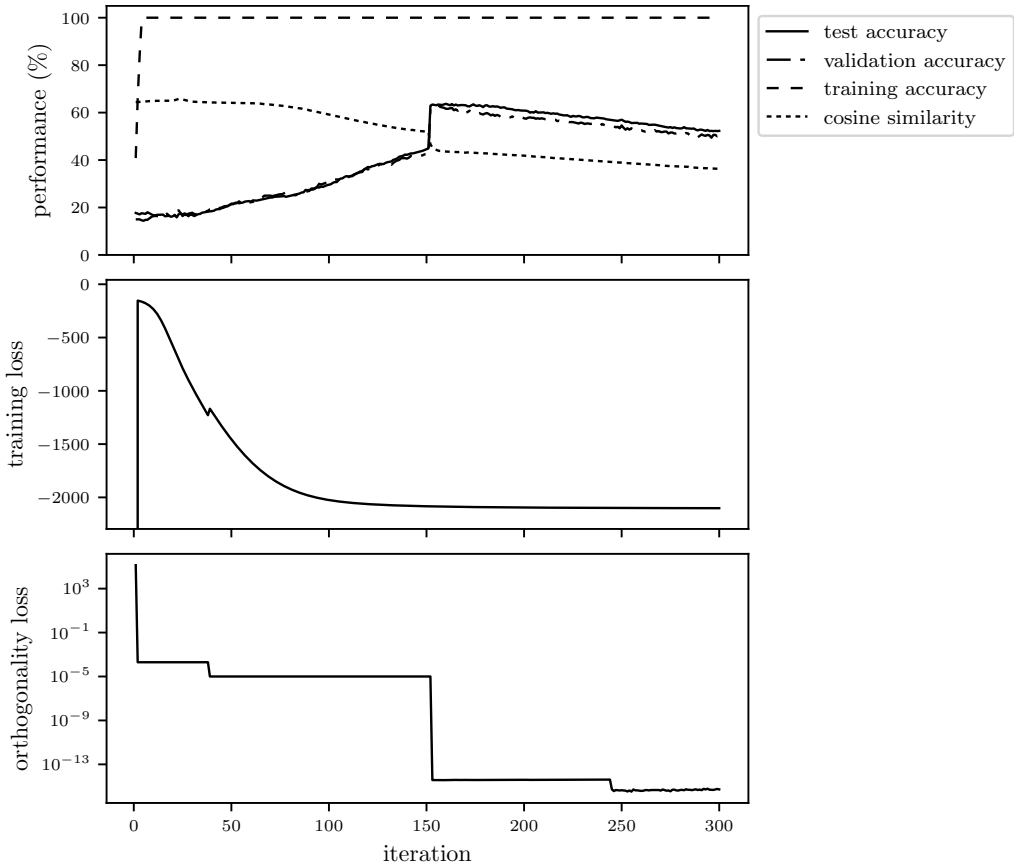

Figure 4: Top: train, validation and test accuracies, and cosine similarity score; middle: training loss; and bottom: orthogonality loss, during training on CiteSeer dataset after random initialization.