FAST GRAPH CLASSIFIER WITH OPTICAL RANDOM FEATURES

Hashem Ghanem *Nicolas Keriven* *Nicolas Tremblay*

CNRS, GIPSA-lab, FR-38402 Saint Martin d'Heres Cedex, France

ABSTRACT

Graph Classification is the problem of mapping graphs to the class they belong to. The graphlet kernel is a classical method in graph classification. However, it suffers from high computation cost due to the isomorphism test. We propose a generic algorithm that mainly replaces the isomorphism test with a user-defined mapping. This algorithm computes a feature vector for each graph. More importantly, we incorporate *optical processing units (OPUs)* technology by choosing the optical random features map. We show that OPUs perform such mapping in light-speed. We prove theoretically that using the OPU is effective in graph classification. We conduct the necessary experiments to empirically show the expected high performance.

Index Terms— Optical random features, Graph kernels

1. INTRODUCTION

Graph structures are used to model a set of objects and their interactions. In biology for instance, proteins can be modeled as graphs where amino acids are nodes and the chemical links between them are edges [1]. In social networks, each post thread is modeled with a graph whose nodes are users, where an edge between two nodes exists if one replied to the other's comment on that thread [2].

Having *a priori* known graph classes, the graph classification task is to predict the class of a given graph. In biology, proteins are to be classified to enzymes and to non-enzymes [1], as in the D&D dataset. In Social networks, one task is to discriminate between discussion-based and question/answer-based threads [3], as in Reddit-Binary dataset.

In some cases, nodes and edges may have extra features that can be used along with the graph structure to classify graphs. It has been shown that node features are important to have high performance [4]. Here, we focus on the sightly harder case where one has only access to the graph structure.

We place ourselves in the context of *supervised learning*, where we have access to a set of pre-labeled graphs ($\mathcal{X} = \{\mathcal{G}_1, \dots, \mathcal{G}_n\}, \mathcal{Y} = \{y_1, \dots, y_n\}$). Each graph $\mathcal{G}_i$ is a *a priori* known to belong to the class with label y_i. The graph classification problem is: given this prior information, design an algorithm that, given in input a new graph, outputs the label of the class to which it belongs.

Related work Structure-based graph classification has been tackled using many algorithms. Frequent sub-graph based algorithms, gSpan method [5] for instance, perform a prohibitive-cost analysis on the graph dataset $\mathcal{X}$ to catch the frequent and discriminative sub-graphs, then use them as features. Graph kernel-based algorithms [6] compute fixed representation for graphs by defining the kernel as a similarity function between graphs. One example is *the graphlet kernel*, which is an efficient method in the literature but is prone to high computation cost [7]. Moreover, graph convolutional networks (GCNs) are common in graph classification. Recently, a particular model called GIN (Graph Isomorphism Network) was developed and provided high performance classification [8].

Contribution: Random features are an computationally efficient method to approximate kernel functions, which satisfy some conditions [9]. Optical Processing Units (OPUs) is a recently developed hardware that computes optical random features in light speed. On the other hand, the graphlet kernel represents a graph by how many times graphs of a smaller size occur in it. This kernel includes the isomorphism test in the counting process which makes it computationally expensive. Can we leverage OPUs computations to reduce the computational complexity of a combinatorial problem like the graphlet kernel ? We show, empirically and theoretically, that yes we can do that to obtain fast and efficient graph representation.

2. BACKGROUND

2.1. The graphlet kernel

Let $\mathfrak{H} = \{\mathcal{H}_1, ..., \mathcal{H}_{N_k}\}$ be the set of all possible non-isomorphic graphs, also called graphlets, of size k, where two graphs are said to be isomorphic ($\mathcal{G} \cong \mathcal{G}'$) if they represent the same structure [10]. We define the matching function $\varphi_k^{match}(\mathcal{F}) = \left[1_{(\mathcal{F} \cong \mathcal{H}_i)}\right]_{i=1}^{N_k} \in \{0,1\}^{N_k}$, where 1_Ω is the indicator function. In words, φ_k^{match} is a Boolean vector of dimension N_k which has a 1 in the coordinate i if $\mathcal{F} \cong \mathcal{H}_i$, and 0 otherwise.

Denote $\mathfrak{F}_\mathcal{G} = \{\mathcal{F}_1, \mathcal{F}_2, \dots, \}$ the collection of all size-k sub-graphs existing in a graph $\mathcal{G}$. We assign each graph with the following representation, called k-spectrum, vector:

$$\mathbf{f}_\mathcal{G} = \frac{1}{|\mathfrak{F}_\mathcal{G}|} \sum_{\mathcal{F} \in \mathfrak{F}_\mathcal{G}} \varphi_k^{match}(\mathcal{F})$$

so that for two graphs $\mathcal{G}, \mathcal{G}'$, the graphlet kernel is defined by the inner product $\mathbf{f}_{\mathcal{G}}^T \mathbf{f}_{\mathcal{G}'}$. The graphlet kernel is used in kernel machine classifiers and performs well especially with sufficiently large value of k [7]. However, in each graph $\mathcal{G}$ of size v, there are $\binom{v}{k}$ size-k sub-graphs. Thus the computation cost to compute $\mathbf{f}_{\mathcal{G}}$ is $C_{gk} = \mathcal{O}\left(\binom{v}{k} N_k C_k^{\cong}\right)$, where $C_k^{\cong}$ is the cost of the isomorphism test between two graphs of size k. This cost is expensive due to: i/ $\binom{v}{k}$ explodes as v or k increase, ii/ N_k is exponential in k, iii/ yet there is no known method to test isomorphism in polynomial time of k [11].

Usually, uniform graph sampling is used to accelerate the graphlet kernel, where sampling a size-k sub-graph means: first we randomly at uniform choose k nodes from the graph, then we consider every edge that connects any pair of them to form the sub-graph.

Knowing this, the k-spectrum $\mathbf{f}_{\mathcal{G}}$ can be interpreted as follows: if one samples a subgraph from $\mathcal{G}$, then one has a probability $(\mathbf{f}_{\mathcal{G}})_i$ of obtaining $\mathcal{H}_i$, *i.e.*: $\mathbf{f}_{\mathcal{G}} = \mathbb{E}_{F \sim \text{unif}}\, \varphi_k^{match}(F)$. It is thus natural to approach $\mathbf{f}_{\mathcal{G}}$ with a sample average, where by sampling s subgraphs of size k to form the collection $\mathfrak{F}_{\mathcal{G}} = \{F_1, ..., F_s\}$, the estimator:

$$\hat{\mathbf{f}}_{\mathcal{G}} = \frac{1}{s} \sum_{F \in \hat{\mathfrak{F}}_{\mathcal{G}}} \varphi_k^{match}(F). \tag{1}$$

verifies by the law of large numbers that $\hat{\mathbf{f}}_{\mathcal{G}} \xrightarrow[s \to \infty]{} \mathbf{f}_{\mathcal{G}}$ with probability 1.

The computation cost per graph of the approximated graphlet kernel is $C_{gk+gs} = \mathcal{O}\left(s C_S N_k C_k^{\cong}\right)$, where C_s is the cost of sampling one subgraph. Although the term $\binom{v}{k}$ doesn't exist as a term in this cost, still for a specific certainty in estimating $\mathbf{f}_{\mathcal{G}}$, the required number of samples s must be proportional to N_k [7]. So this version is still expensive especially when k is large.

Although what follows is technically not related to the graphlet kernel as defined here, it could be considered related in some literature. There exist many different sampling techniques S_k, each follows a specific random process in sampling the k nodes from a graph $\mathcal{G}$. As a result, subgraphs sampled with a technique different from the uniform sampling will have a different histogram $\mathbf{f}_{\mathcal{G}, S_k}$ than the one defined by $\mathbf{f}_{\mathcal{G}}$ [12]. One such technique is the random walk (RW) sampler, that tends to sample a subgraph in which there is a path of edges between any two nodes. Therefore, RW subgraphs are more informative than the uniform ones, since the uniform sampler, with high probability, generates sparse subgraphs that don't have any information about the graph structure.

Algorithm 1: GSA-φ generic algorithm

Input: labelled graph dataset $\mathcal{X} = (\mathcal{G}_i, y_i)_{i=1,...,n}$
Tools *Graph random sampler S_k, a function φ, linear classifier (ex. SVM)*
Hyperparameters k*: graphlet size,* s*: number of graphlet samples per graph*, m: features number
Output: Trained model to classify graphs
Algorithm
Random initialization of the SVM weights
for $\mathcal{G}_i$ *in* $\mathcal{X}$ **do**
  $\mathbf{z}_i = \mathbf{0}$ (null vector of size m)
  for $j = 1 : s$ **do**
    $F_{i,j} \leftarrow S_k(\mathcal{G}_i)$
    $\mathbf{z}_i \leftarrow \mathbf{z}_i + \frac{1}{s}\varphi(F_{i,j})$
$\mathcal{D}_{\varphi} \leftarrow (\mathbf{z}_i, y_i)_{i=1,...,n}$
Train the linear classifier on the new vector-valued dataset $\mathcal{D}_{\varphi}$

3. METHOD

3.1. Proposed algorithm

We propose to replace φ_k^{match} with a user-defined map φ, and to replace the uniform sampler with a user-chosen one like RW sampler. The function φ here maps each subgraph to a new m-dimensional space $\mathbb{R}^m$. We refer to this framework as *Graph Sampling and Averaging* $GSA - \varphi$.

Note that choosing $\varphi = \varphi_k^{match}$ and S_k as the uniform sampler, GSA-φ_k^{match} turns out to be the approximated graphlet kernel. We see next that choosing φ as random maps is both fast and efficient in graph classification.

3.2. Efficiency of kernel random features with $GSA - \varphi$

A kernel κ associated to a random features (RF) decomposition is a positive definite function of two inputs in $\mathbb{R}^d$ that can be decomposed as follows [13]:

$$\kappa(\mathbf{x}, \mathbf{x}') = E_{\mathbf{w} \sim p}[\xi_{\mathbf{w}}(\mathbf{x}) \xi_{\mathbf{w}}(\mathbf{x}')] \tag{2}$$

where E stands for the expectation, p is a probability distribution, and ξ is a function parameterized by w. Most RF constructions in (2) are known as Random Fourier Features (RFF), which are based on Bochner's theorem. Specifically, if the kernel κ is continuous and shift invariant, its Fourier transform p is a correct probability distribution if it is well-scaled. Thus, scaling a kernels to obtain $\int p = 1$, we have:

$$\kappa(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} p(\mathbf{w}) cos(\mathbf{w}^T(\mathbf{x} - \mathbf{x}')) d\mathbf{w} \tag{3}$$

which leads to (2) by setting $\xi_{\mathbf{w}}(\mathbf{x}) = \sqrt{2} cos(\mathbf{w}^T \mathbf{x} + b)$ such that $\mathbf{w}$ is drawn from p and b is drawn uniformly from $[0, 2\pi]$.

Finally, the random maps we propose starting from (2) are:

$$\varphi_{RF}(\mathbf{x}) = \frac{1}{\sqrt{m}}(\xi_{\mathbf{w}_j}(\mathbf{x}))_{j=1}^{m} \in \mathbb{C}^m \tag{4}$$

with m is the number of features and the frequencies $\mathbf{w}_j$ are drawn identically and independently (iid) from p. For any two graphs $\mathcal{G}, \mathcal{G}'$, the next theorem states that using φ_{RF} in $GSA - \varphi$, the Euclidean distance between the representation vectors $\mathbf{z}_{\mathcal{G}}, \mathbf{z}_{\mathcal{G}'}$ converges to the MMD metric between their distributions $\mathbf{f}_{\mathcal{G},S_k}$, $\mathbf{f}_{\mathcal{G}',S_k}$. The MMD (Maximum Mean Discrepancy) metric between two distributions is defined by:

$$MMD^2(\mathbf{f}_{\mathcal{G},S_k}, \mathbf{f}_{\mathcal{G}',S_k}) = \sum_{(i,j)}^{N_k} (\mathbf{f}_{\mathcal{G},S_k})_i (\mathbf{f}_{\mathcal{G}',S_k})_j \kappa(\mathcal{H}_i, \mathcal{H}_j)$$

Theorem 1. *Let $\mathcal{G}$ and $\mathcal{G}'$ be two graphs, $\mathfrak{F}_{\mathcal{G}} = \{F_i\}_{i=1}^{s}$ (resp. $\mathfrak{F}_{\mathcal{G}'} = \{F_i'\}_{i=1}^{s}$) be iid size-k graphlet samples drawn from $S_k(\mathcal{G})$ (resp. $S_k(\mathcal{G}')$). Assume a random feature map 4. Assume that $|\xi_{\mathbf{w}}(F)| \leq 1$ for any $\mathbf{w}, F$. Proven in Appendix A, we have for all $\delta > 0$ and with probability at least $1 - \delta$:*

$$\left| \|\varphi(\mathfrak{F}_{\mathcal{G}}) - \varphi(\mathfrak{F}_{\mathcal{G}'})\|^2 - MMD(\mathbf{f}_{\mathcal{G},S_k}, \mathbf{f}_{\mathcal{G}',S_k})^2 \right| \leq \frac{4\sqrt{\log(6/\delta)}}{\sqrt{m}} + \frac{8\left(1 + \sqrt{2\log(3/\delta)}\right)}{\sqrt{s}}$$

Note that this theorem suggests choosing m of the same order of s. The main property of the MMD is that, for so-called *characteristic kernels*, it is a true metric on distributions, *i.e.* $MMD(\mathcal{P}, \mathcal{Q}) = 0 \Leftrightarrow \mathcal{P} = \mathcal{Q}$. In addition, most usual kernels, like the Gaussian kernel, are characteristic.

3.3. Considered choices of φ_{RF}

Gaussian maps φ_{Gs} applied on the adjacency matrix: φ_{Gs} is the RFF map of the Gaussian kernel. We for each subgraph $\mathcal{F}$ take its vectorized adjacency matrix $\mathbf{a}_{\mathcal{F}} = flatten(\mathbf{A}_{\mathcal{F}})$ as input. Then:

$$\varphi_{Gs}(\mathcal{F}) = \frac{1}{\sqrt{m}} \left(\sqrt{2}\cos(\mathbf{w}_j^T \mathbf{a}_{\mathcal{F}} + b_j) \right)_{j=1}^{m} \in \mathbb{R}^m \tag{5}$$

where the frequencies $w_j \in \mathbb{R}^{k^2}$ are drawn from a Gaussian distribution with the inverse variance of the original kernel.

Gaussian maps φ_{Gs+Eig} applied on the sorted Eigenvalues of the adjacency matrix: instead of passing the vectorized adjacency matrix as input, we pass the vector of its sorted eigenvalues $\boldsymbol{\lambda} \in \mathbb{R}^k$. The motive proposing φ_{Gs+Eig} is that it respects the isomorphism test since: i/ $\lambda(\mathbf{A}) = \lambda(\mathbf{PAP^T})$ for any permutation matrix $\mathbf{P}$, ii/ $F \cong F' \Rightarrow \exists \mathbf{P}, \mathbf{A}_F = P\mathbf{A}_{F'}P^T$. Thus, $F \cong F' \Rightarrow \varphi_{Gs+Eig}(F) = \varphi_{Gs+Eig}(F')$. φ_{Gs+Eig} maps isomorphic subgraphs to the same point in $\mathbb{R}^m$.

Graphlet kernel		$O(\binom{v}{k} N_k C_k^{\cong})$
GSA-φ with:	φ_k^{match}	$O(C_s s N_k C_k^{\cong})$
	φ_{Gs}	$O(C_s smk^2)$
	$\varphi_{Gs+Eigen}$	$O(C_s s(mk + k^3))$
	φ_{OPU}	$O(C_s s)$

Table 1. Per-graph complexities of GSA-φ.

Optical random feature maps φ_{OPU}: This corresponds to the fastest version of our algorithm. OPUs (Optical Processing Units) technology was developed to compute a specific random features mapping in *constant time* $\mathcal{O}(1)$ in both m and k using light scattering [14]. Having the random matrix $\mathbf{W}$, traditional random maps (ex. φ_{Gs}) need $\mathcal{O}(mk^2)$ cost to compute $\mathbf{Wx}$ as in (5). An OPU computes its associated map at the speed of light, this map is modeled as follows [14]:

$$\varphi_{OPU}(\mathbf{x}) = |\mathbf{Wx} + \mathbf{b}|^2;\ \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^d$$

Where $\mathbf{b}$ is a random bias vector, d is the input space dimension, the amplitude function $|\cdot|$ is taken element wise, and $\mathbf{W}$ is a random *iid* complex matrix with Gaussian real and imaginary parts. The complexities of the different mappings φ examined in this work are summarized in Table 1.

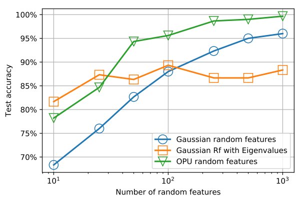
4. EXPERIMENTS

4.1. Setup

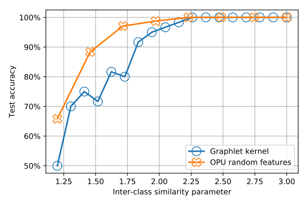
With respect to the performance and to the computation cost, we compare different choices of the map φ in $GSA - \varphi$. We benchmark the performance of $GSA - \varphi_{OPU}$ against GIN based graph convolutional network.

In all experiments except the last one, we use a synthetic dataset generated by *Stochastic Block Model (SBM)* [15]. We generate 300 graphs, 240 for training and 60 for testing. Each graph has $v = 60$ nodes divided equally between six communities. Moreover, graphs are divided into two classes $\{0, 1\}$ based on the edges distribution considered. For each class we fix two values (p_{in}, p_{out}) which are the probabilities of generating an edge between any two nodes when they are in the same community and when they are in different ones, respectively. Beside, to prevent the classes being easily discriminated by the average degree of nodes as a feature, the pairs $(p_{in,i}, p_{out,i})_{i=0,1}$ are chosen so all nodes have a fixed expected average degree equal to $\mu = 10$. Having one freedom degree left, we fix $p_{in,1} = 0.3$, and we vary $r = (p_{in,1}/p_{in,0})$ the inter-classes similarity parameter: the closer r is to 1, the more similar both classes are, and thus the harder it is to discriminate them.

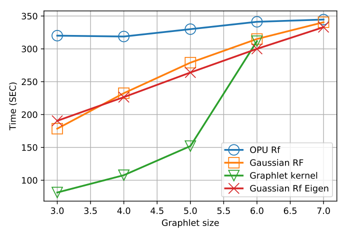
On the other hand, D&D is a labeled dataset of size $n = 1178$ protein graphs [16]. Also, nodes have 7 features each, which are not used by our algorithms, *i.e.* we will try to classify the graphs just based on their structure. In what follows, unless otherwise indicated, we use uniform sampling and the adjacency matrix of subgraphs as input.



(a) RF maps



(b) φ_{OPU} Vs. φ_k^{match}



(c) Computation cost

Fig. 1. Comparing different φ maps in $GSA - \varphi$. (a) test accuracy when using RF maps with $k = 6$ while varying m. (b) test accuracy of $GSA - \varphi_{OPU}$ and $GSA - \varphi_k^{match}$ fixing $k = 5$ while varying r. (c) Computation time as a function of k. If not specified: $r = 1.1$, $s = 2000$, $m = 5000$ and the Gaussian map variance $\sigma^2 = 0.01$.

4.2. Choice of feature map φ

Comparison of random features: Fig 1(a) shows that $GSA - \varphi_{OPU}$ applied on adjacency matrices gives better test accuracy with sufficiently large m than both $GSA - \varphi_{Gs}$ applied on adjacency matrices or $GSA - \varphi_{Gs+Eig}$ applied on its sorted eigenvalues. On the contrary, $GSA - \varphi_{Gs+Eig}$ performs best with a low number of random features, but increasing this number does not really improve the result and it is over-matched at high m. A possible justification is that the Eigenvalues of the adjacency matrix lose information about the subgraphs, even though respecting the isomorphism means that we are working with a smaller histogram and less random features are required.

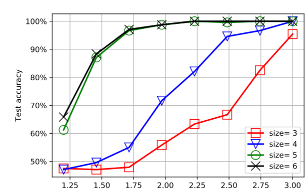
Comparing $GSA - \varphi_{OPU}$ to $GSA - \varphi_k^{match}$: from Fig 1(b) we observe that with the same limited number of samples s, $GSA - \varphi_{OPU}$ clearly outperforms the approximated graphlet kernel $GSA - \varphi_k^{match}$, so we conclude that $GSA - \varphi_{OPU}$ is more adapted in this case than the traditional graphlet kernel.

Computational time: Fig 1(c) shows the computation time of each of the previous methods with respect to the subgraphs size k. Other parameters are identically fixed for all methods. As expected, the execution time of the approximated graphlet kernel grows exponentially with k, and is polynomial for $GSA - \varphi_{Gs}$ and $GSA - \varphi_{Gs+Eig}$. On the contrary, it is almost constant for $GSA - \varphi_{OPU}$. We point out here that, with the current settings, there is a significant overhead between the point in time where we run our optimizer code on the OPUs server and the point when the OPU launches the computation. To be fair, this overhead time should be measured and subtracted from $GSA - \varphi_{OPU}$ computation time. As the technology comes into maturity, we can expect this additional time to be significantly reduced.

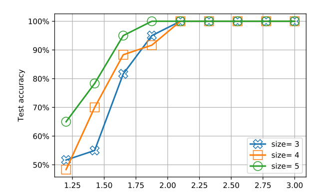
To summarize, $GSA - \varphi_{OPU}$ outperforms the traditional methods both in accuracy and computational cost.

4.3. Comparing $GSA - \varphi_{OPU}$ against GIN model

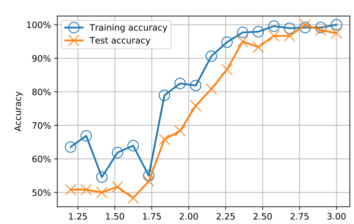
In Fig 2, we observe that for subgraphs sizes > 5, both RW and uniform sampling perform similarly well in $GSA - \varphi_{OPU}$, but RW sampling, as expected, provide more consistent results when the graphlet size k varies. Thus RW sampling is considered in this comparison. We see that $GSA - \varphi_{OPU}$ with RW performs better than the GIN model when the graphlet size is greater than 4. We note that we do not report the computational time for GIN, since it is highly dependent on high-speed graphical processing units (GPUs) to do the training process.



(a) φ_{OPU} & uniform sampling



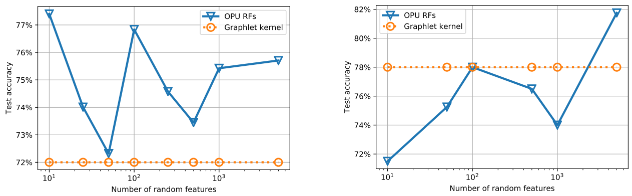
(b) φ_{OPU} & RW sampling



(c) GIN-based GCN model

Fig. 2. Comparing test accuracies of $GSA - \varphi_{OPU}$ and GIN-based GCN network when varying the problem difficulty r. We used $GSA - \varphi_{OPU}$ with uniform sampling in (a) and with random walk sampling in (b). In both cases: $s = 2000$ and $m = 5000$. (c) The model consists of 5 GIN layers then 2 fully connected layers, the dimensions of hidden layers: 4.

In Fig 3, we have the test accuracy with varying value of m and fixed $s = 4000$, $k = 7$. For each value of m we conduct the experiment 5 times and take the average accu-



(a) D&D (b) Reddit-Binary

Fig. 3. Bench-marking $GSA - \varphi$ against the graphlet kernel as a performance reference on real datasets with: s=4000,k=.

racy. Although we do not observe a clear, steady average improvement in accuracy when m grows, the results of the 5 corresponding experiments get more concentrated around the average value, giving a desirable reduced variance between experiments. On the other hand, accuracy results at low m show high variance between experiments, which might be accentuated by the fact that nodes features are ignored. However, using node features is without doubt necessary to reach state-of-the-art results, and it is an open question how to incorporate that in our algorithm, but our goal here is mainly to test our algorithm on real data as a proof of concept.

5. CONCLUSION

We proposed a family of algorithms that combines graph sampling with efficient mappings. Then, we proposed to choose this mapping as kernel random maps, and showed a concentration of the random embedding around the MMD metric. Finally, while classical random features still require expensive matrix-vector multiplication, we used optical random features projections, which can be computed in $\mathcal{O}(1)$ to get the algorithm's fastest version. Our Experiments showed that it is significantly faster than traditional graphlet kernel and generally performs better while concentrating around the MMD metric. In our settings, it even outperformed a particular graph convolutional network on graph classification.

A major point left open to be analyzed is how to use our algorithm to classify graphs with node features. One promising possibility is to use our algorithm to generate features embeddings on the graph level, and then feed these embeddings with the nodes' features to a deep neural network. Doing this, we take advantage of the speed of both our algorithm and GPUs. On the theoretical side, the properties of the MMD metric could be further analyzed on particular models of graphs to get a concentration with higher certainty.

6. REFERENCES

[1] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis, "Matching node embeddings for graph similarity," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Pinar Yanardag and SVN Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1365–1374.

[3] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[4] Chi Thang Duong, Thanh Dat Hoang, Ha The Hien Dang, Quoc Viet Hung Nguyen, and Karl Aberer, "On node features for graph neural networks," *arXiv preprint arXiv:1911.08795*, 2019.

[5] X Yan and J Han, "gspan: Graph-based substructure pattern mining, 2002," *Published by the IEEE Computer Society*, 2003.

[6] Nils M Kriege, Fredrik D Johansson, and Christopher Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.

[7] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[8] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[9] Aman Sinha and John C Duchi, "Learning kernels with random features," in *Advances in Neural Information Processing Systems*, 2016, pp. 1298–1306.

[10] Johannes Kobler, Uwe Schöning, and Jacobo Torán, *The graph isomorphism problem: its structural complexity*, Springer Science & Business Media, 2012.

[11] Anna Lubiw, "Some np-complete problems similar to graph isomorphism," *SIAM Journal on Computing*, vol. 10, no. 1, pp. 11–21, 1981.

[12] Jure Leskovec and Christos Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 631–636.

[13] Ali Rahimi and Benjamin Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems*, 2008, pp. 1177–1184.

[14] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan, and Florent Krzakala, "Random projections through multiple optical scattering: Approximating kernels at the speed of light," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6215–6219.

[15] Thorben Funke and Till Becker, "Stochastic block models: A comparison of variants and inference methods," *PloS one*, vol. 14, no. 4, 2019.

[16] Paul D Dobson and Andrew J Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.

Appendices

A. PROOF OF THEOREM 1

Proof. We decompose the proof in two steps.

Step 1: infinite s, finite m. First we define the random variables $x_j = |\mathbb{E}_{F\sim S_k(\mathcal{G})}\xi_{\mathbf{w}_j}(F) - \mathbb{E}_{F'\sim S_k(\mathcal{G}')}\xi_{\mathbf{w}_j}(F')|^2$, which are: i/independent, ii/have expectation $MMD(\mathcal{G},\mathcal{G}')^2$, /iii are bounded by the interval $[0,4]$ based on our assumption $|\xi_w| \le 1$. Thus, as a straight result of applying Hoeffding's inequality with easy manipulation: with probability $1-\delta$

$$\Big|\frac{1}{m}\sum_{j=1}^{m} x_j - MMD(\mathcal{G},\mathcal{G}')^2\Big| \le \frac{4\sqrt{\log(2/\delta)}}{\sqrt{m}} \tag{6}$$

Step 2: finite s and m. For any *fixed* set of random features $\{w_j\}_{1,\dots,m}$ and based on our previous assumptions we have: i/ φ_{RF} is in a ball of radius $M = \frac{\sqrt{m}}{\sqrt{m}} = 1$, ii/ $\mathbb{E}_{F\sim S_k(\mathcal{G})}\ \varphi(F) = \mathbb{E}\left(\frac{1}{s}\sum_i \varphi(F_i)\right)$. Therefore, we can directly apply the vector version of Hoeffding's inequality on the vectors $\frac{1}{s}\sum_i \varphi(F_i)$ to get that with probability $1-\delta$:

$$\left\|\mathbb{E}_{F\sim S_k(\mathcal{G})}\varphi(F) - \frac{1}{s}\sum_i \varphi(F_i)\right\| \le \frac{1+\sqrt{2\log\frac{1}{\delta}}}{\sqrt{s}} \tag{7}$$

Defining $J_{exp}(\mathcal{G},\mathcal{G}') = \|\mathbb{E}_{F\sim S_k(\mathcal{G})}\varphi(F) - \mathbb{E}_{F'\sim S_k(\mathcal{G}')}\varphi(F')\|$ and $J_{avg}(\mathcal{G},\mathcal{G}') = \|\frac{1}{s}\sum_i \varphi(F_i) - \frac{1}{s}\sum_i \varphi(F_i')\|$, then using triangular inequality followed by a union bound based on (7), we have the following with probability $1-2\delta$,

$$\Big|J_{exp}(\mathcal{G},\mathcal{G}') - J_{avg}(\mathcal{G},\mathcal{G}')\Big| \le \frac{2}{\sqrt{s}}\left(1+\sqrt{2\log\frac{1}{\delta}}\right)$$

On the other hand, $J_{exp}(\mathcal{G},\mathcal{G}') + J_{avg}(\mathcal{G},\mathcal{G}') \le 4$, so with same probability:

$$\Big|J_{exp}(\mathcal{G},\mathcal{G}')^2 - J_{avg}(\mathcal{G},\mathcal{G}')^2\Big| \le \frac{8}{\sqrt{s}}\left(1+\sqrt{2\log\frac{1}{\delta}}\right) \tag{8}$$

Since it is valid for any fixed set of random features, it is also valid with *joint* probability on random features and samples, by the law of total probability.

Finally, combining (6), (8) with a union bound and a triangular inequality, we have with probability $1-3\delta$,

$$\Big|\|\varphi(\mathfrak{F}_{\mathcal{G}}) - \varphi(\mathfrak{F}_{\mathcal{G}'})\|^2 - MMD(\mathcal{G},\mathcal{G}')^2\Big| \le \frac{4\sqrt{\log(2/\delta)}}{\sqrt{m}} + \frac{8}{\sqrt{s}}\left(1+\sqrt{2\log\frac{1}{\delta}}\right)$$

which concludes the proof by taking δ as $\delta/3$.

$\square$