FAST GRAPH CLASSIFIER BASED ON OPTICAL RANDOM FEATURES

Author(s) Name(s)

Author Affiliation(s)

ABSTRACT

The graphlet kernel is a classical method in graph classification. However, it suffers from high computation cost. This is due to its graph matching function which performs the isomorphism test. We propose a generic algorithm that mainly replaces the matching function with a user-defined map. The algorithm computes a feature vector for each graph which can be fed later to a linear classifier. Moreover, we prove that the algorithm is efficient when using random feature maps. In its finest version, we use *optical processing units (OPUs)*, which perform such random mapping in light-speed, and prove that the new computation cost is $\mathcal{O}(1)$ in both the input and the output dimensions. We conduct the necessary experiments to empirically show the high performance of this version.

***Index Terms*—** Optical random features, Graph kernels

1. INTRODUCTION

Graph structures are used to model a set of objects and their interactions, where an object is reduced to a node and an interaction between two objects is reduced to an edge between the two corresponding nodes. In biology for instance, proteins as graphs are composed of amino acids as nodes and the chemical links as edges.

Graph classification has been addressed in many fields. In biology for example, proteins are to be classified to enzymes and to non-enzymes, which is the case in [illegible]D dataset [1]. However, nodes and/or edges may have extra information that can be used along with the graph structure to classify graphs. In fact, the existence of such extra-information is very application-dependent, so we focus here on the case where such information is absent and the only information one has access to is the graph structure (topology).

Furthermore, We place ourselves in the context of *supervised learning*, where we have access to a set of pre-labeled graphs ($\mathcal{X} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}, \mathcal{Y} = \{y_1, \ldots, y_n\}$). Each graph $\mathcal{G}_i$ is a *a priori* known to belong to the class with label y_i. Simply, 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. In general, a pre-chosen metric is used to compare the performance of different algorithms as the test accuracy.

Thanks to XYZ agency for funding.

Related work Structure-based graph classification has been tackled in many previous works, where proposed algorithms can be divided in three main categories. The first is frequent sub-graph based algorithms, gSpan method [2] for instance, which perform a prohibitive-cost analysis on the graph dataset $\mathcal{X}$ to catch the frequent and discriminative sub-graphs, then use them as features. The second is graph convlutional networks (GCNs), which have a poor performance when all we have is the graphs structure. Recently, a particular model called GIN (Graph Isomorphism Network) was developed to have high performance in this case [3]. The third is Graph kernel-based algorithms **??**,which compute fixed representation for graphs by defining the kernel as a similarity function between graphs. In this work, we focus on one such kernel called *the graphlet kernel*, which is an efficient method in the literature but is prone to high computation cost [4].

Contribution: inspired by the graphlet kernel with uniform graph sampling, we propose a new algorithm $GSA - \varphi$ that:

- is a general framework in graph classification, which maps graphs to a new feature space using a user-defined pair (map function, graph sampling technique).
- proves, both theoretically and empirically, to be efficient with using random feature maps.
- can make use of the light-speed optical random feature maps. This is in $\mathcal{O}(1)$ in both the size of sampled subgraphs and in the number of features.

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 graphlets, of size k, where two graphs are said to be isomorphic ($\mathcal{G} \cong \mathcal{G}'$) if they represent the same structure [5]. 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 $\mathfrak{F}_\mathcal{G} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \}$ 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 inner product $f_{\mathcal{G}}^T f_{\mathcal{G}'}$ is graphlet kernel between them. The graphlet kernel is used in kernel machine classifiers [6] and performs well especially with sufficiently large value of k [4]. 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 sub-graphs of size k. This cost is expensive due to: i/ $\binom{v}{k}$ explodes as v creases, ii/ N_k is exponential in k, iii/ yet there is no known method to test isomorphism in polynomial time [7].

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 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 accelerated graphlet kernel is $C_{gk+gs} = \mathcal{O}\left(sC_S N_k C_k^{\cong}\right)$, 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, b ll for a specific certainty in estimating $\mathbf{f}_{\mathcal{G}}$, the required number of samples s must be proportional to N_k [4]. So this version is still expensive especially when k is large.

Although what follows is not related to the graphlet kernel, but here is the place to point out that there exist other graph sampling techniques. Each technique S_k follows a specific random process in sampling the k nodes from a graph $\mathcal{G}$ to form a size-k subgraph. As a result, subgraphs sampled with a technique different from the uniform sampling will have a different histogram than the one defined by $\mathbf{f}_{\mathcal{G}}$ [8]. One such technique is the random walk (RW) sampler, which 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.

3. METHOD

3.1. Proposed algorithm

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*
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 $\mathcal{D}_\varphi$

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 $\mathbb{R}^m$. Fin we refer to this framework as *Graph Sampling and Averaging* $GSA - \varphi$.

Note that choosing $\varphi = \varphi_k^{match}$ and S_k a uniform sampler, GSA-φ_k^{match} turns out to be the accele 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 [9]:

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

where E stands for the ctation, p is a cor probability distribution, and ξ is a randomized function. 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:

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

with m is the number of features and the random frequencies $\mathbf{w}_j$ are drawn identically and independently (iid) from p. The next theorem states that using φ_{RF} in $GSA{-}\varphi$, the Euclidean distance between their representation vectors $\mathbf{z}$ converges to the MMD metric between their corresponding distributions.

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 $\mathbf{w}, F$. We have that, for all $\delta > 0$, 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:

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

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 $\mathbb{R}^m$.

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 [10]. 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 [10]:

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

Where $\mathbf{b}$ is a random bias vector, m is the number of random features, d is the input space dimension, the amplitude function $|\cdot|$ is taken element wise, and the matrix $\mathbf{W}$ is a random *iid* complex matrix with Gaussian real and imaginary parts.

In the limit where the number of random features $m \to \infty$, it can be proven that the inner product between two projected data points tends to a kernel between them [10]. The complexities of the different mappings φ examined in this work are summarized in Table 1.

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}	$O(C_s s m k^2)$
	$\varphi_{Gs+Eigen}$	$O(C_s s(mk + k^3))$
	φ_{OPU}	$O(C_s s)$

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

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 then we test D&D dataset.

In all experiments except the last one, we use a synthetic dataset generated by *Stochastic Block Model (SBM)* [11]. We have 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 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 $(p_{in,i}, p_{out,i})_{i=1,2}$ 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 [12]. 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.

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 accelerated 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 accelerated 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 $GSA - \varphi_{OPU}$, where the slight variation are due to overhead computation and not the feature map itself. We should 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 computation time should be measured and subtracted from $GSA - \varphi_{OPU}$ execution time. As the technology comes into maturity, we can expect this additional time to be significantly reduced.

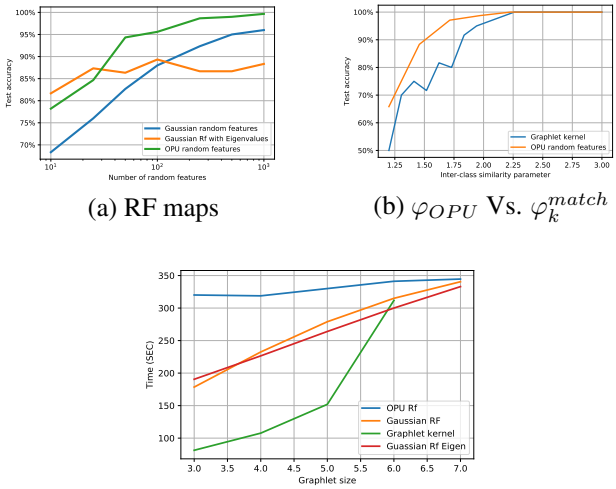
To summarize, $GSA - \varphi_{OPU}$ outperforms the traditional methods both in accuracy and computational cost.

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.

4.4. $GSA - \varphi_{OPU}$ on D&D dataset

In Fig 3, we have the 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 accuracy. 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, at low m accuracy results 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.



(a) RF maps

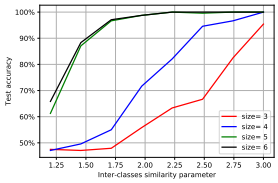
(b) φ_{OPU} Vs. φ_k^{match}

(c) Computation cost

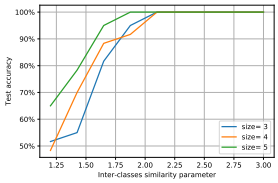
Fig. 1. Comparing different φ maps in $GSA - \varphi$. (a) test accuracy when using RF maps with $k = 6$ while varying m. (b) test accuracy of $GSA - \varphi_{OPU}$ and $GSA - \varphi_k^{match}$ fixing $k = 5$ while varying r. (c) Computation time as a function of k. If not specified: $r = 1.1$, $s = 2000$, $m = 5000$ and $\sigma = 0.1$.

5. CONCLUSION

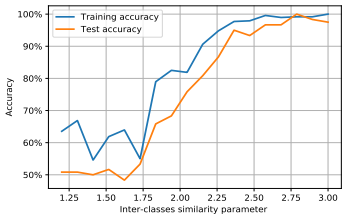
In this work, we proposed a family of algorithms that combines graph sampling with efficient mappings. Then, we proposed to choose this mapping as kernel random maps, and showed a concentration of the random embedding around the MMD metric. Finally, while classical random features still require expensive matrix-vector multiplication, we used optical random features projections, which can be computed in constant time 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 a well-defined MMD metric. Furthermore, in our settings it even performed better than a particular graph convolutional network on graph classification.



(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.

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 our new MMD metric could be further analyzed on particular models of graphs to get a concentration with higher certainty.

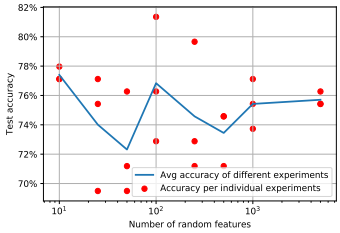


Fig. 3. $GSA-\varphi_{OPU}$ test accuracy on D&D. With $k=7$, $s=4000$. Varying the value of m, the experiment is done five times, then the 5 resulted accuracies are averaged.

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

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

[4] 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.

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

[6] Vikramaditya Jakkula, "Tutorial on support vector machine (svm)," *School of EECS, Washington State University*, vol. 37, 2006.

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

[8] 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.

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

[10] 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.

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

[12] 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 ??

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 $|\xi_w| \leq 1$. Thus, as a straight result of applying Hoeffding's inequality with easy manipulation that with probability $1-\delta$:

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

Step 2: finite s and m. For any *fixed* set of random features $\{w_j\}_{1,\ldots,m}$ 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 $1-\delta$:

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

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 $1-2\delta$,

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

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

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

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), a union bound and a triangular inequality, we have: with probability $1-3\delta$,

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

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

□