TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

Theoretical analysis

Before proceeding to the analysis of used methods, we first introduce the necessary notations related to Graph definition and Graph kernels. A graph by definition is a pair $G = (V, E)$, where V is the set of the graph nodes (vertices) $V = v_1, ..., v_n$, and $E \in V \times V$ is the set of edges between these nodes, i.e. $(v_i, v_j) \in E$ means that the graph has an edge from node v_i to node v_j (or vice versa since we consider undirected graphs in this work).

$H = (V_H, E_H)$ is said to be a subgraph (graphlet) of G ($H \sqsubseteq$) if and only if exist an injective function $\mathcal{M} : V_H \to V$ such that $(v, w) \in E_H \Leftrightarrow (\mathcal{M}(v), \mathcal{M}(w)) \in E$.

Any edge (v_i, v_i) is called a self loop. In a general graph two vertices v_i and v_j may be connected by more than one edge. A simple graph is a graph with no self loops or multiple edges. Here we always consider simple graphs.

A simple graph can equivalently be represented by an adjacency matrix A of size $n \times n$. The $(i, j) - th$ entry of A is 1 if an edge (v_i, v_j) exists and zero otherwise.

Two graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic ($G' \cong G$) if there exists a bijective function $\mathcal{M} : V \to V'$ such that $(v_i, v_j) \in E$ iff $(\mathcal{M}(v_i), \mathcal{M}(v_j)) \in E'$

1.1 Kernels and Random Features

The kernel trick is a way to generate features for algorithms that depend only on the inner product between pairs of input points. The mathematical basis behind it is that any positive definite function $k(x, y)$ with $x, y \in \mathcal{R}^d$ defines an inner product and a lifting function ϕ so that the inner product between lifted data points equals its kernel $< \phi(x), \phi(y) >= k(x, y)$. The convenience one gets is that there is no need to access the lifting function ϕ, instead, the used algorithm accesses the data only through evaluations of $k(x, y)$. To better illustrate this benefit, we take the Gaussian Radial Basis function (RBF) kernel as an example, where:

$$K_{RBF}(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2}) \tag{1.1}$$

where σ is called the bandwidth parameter of the kernel. The Hilbert space associated with the Gaussian RBF kernel has infinite dimension, but the kernel may be easily evaluated for any pair (x,y).

1.1.1 Graph kernels design

Traditional kernel machines approach problems with vector-valued input data, where it compare different objects ($x, y \in \mathcal{R}^d$) using difference between vector components. Based on that, these kernels are imperfect to be used with graphs, since the structure of a graph is invariant to permutations of its representation (e.g. Adjacency matrix), i.e. the ordering by which nodes and edges are enumerated, which leads to isomorphic graphs, does not change the structure. So in this case distance-kernels between graph representation vectors are uninformative.As a result it is necessary to measure distance between graphs in ways that are themselves permutation invariant. While the concept of isomorphism is important in learning algorithms on graphs, not only there is no known polynomial-time algorithm for testing graph isomorphism (except for graphs with specific structures), but isomorphism is also too strict for learning in a similar way to learning with equality operator (Kriege, Johansson,

and Morris, 2020).

The majority of kernels developed to solve graph learning problems are convolution kernels, where given two discrete structures (e.g., two graphs), the idea of convolution framework is to subdivide these to structures into sub-structures (nodes or subgraphs) and evaluate a kernel between each pair of such sub-structures.

Definition 1 (Convolution Kernel) *let $\mathcal{R} = \mathcal{R}_1 \times ... \times \mathcal{R}_1$ denote a space of components such that a composite object $X \in \mathcal{X}$ decomposes into elements of $\mathcal{R}$. Let $R : \mathcal{R} \to \mathcal{X}$ denote the mapping from components to objects, such that $R(x) = X$ iff the components $x \in \mathcal{R}$ make up the object $X \in \mathcal{X}$, and let $R^{-1}(X) = \{x \in \mathcal{R} : R(x) = X\}$. then, the R-convolution kernel is:*

$$K_{CV}(X, Y) = \sum_{x \in R^{-1}(X)} \sum_{y \in R^{-1}(Y)} \underbrace{\prod_{i=1}^{d} k_i(x_i, y_i)}_{k(x,y)} \tag{1.2}$$

with k_i is a kernel on $\mathcal{R}$ for $i \in \{1, ..., d\}$.

Projecting this definition on Graphs, the inverse map $R^{-1}(G)$ of the convolution kernel can be seen as the set of all components of a graph G that we want to compare. An example of these kernels is the node label kernel whose mapping R takes the attributes $x_u \in \mathcal{R}$ of each node $u \in G \times H$ and maps them to the graph that u is a member of. The attractive advantage using convolution kernel framework when working with graphs is that while the kernels on substructures are invariant to orderings of nodes and edges, so is the resulting graph kernel. On the other hand, the sum in Eq. 1.2 iterate over all pairs of components. When the considered components become more and more specific, each object becomes increasingly similar to itself, but no longer to any other objects, this problem is called the diagonal dominance problem, since the entries on the main diagonal of the kernel matrix (Gram matrix) are much higher than other entries,Thus, weights between the components usually are added to balance Gram matrix entries so this drawback is resolved.

1.1.2 Graphlet Kernel

Referring by $\mathcal{G} = \{graphlet(1), ..., graphlet(N_k)\}$ to the set of k-nodes graphlets, and considering a graph G of size n we can define a vector f_G of length N_k whose i-th component equals the normalized-number of occurrences of $graphlet(i)$ in G ($\#(graphlet(i) \sqsubseteq G$. What should be noticed based on this definition is that no two different graphlets in $\mathcal{G}$ are isomorphic. f_G is referred to by k-spectrum of G, and this vector is the key idea behind graphlet kernel.

Definition 2 (Graphlet Kernel) *Given two graphs G and G' of size $n, n' \geq k$, the graphlet kernel k_g is defined as (Shervashidze et al., 2009):*

$$k_g(G, G') = f_G^T f'_G. \tag{1.3}$$

The drawback of this kernel is that computing the k-spectrum vector costs a lot of computational time, since there are $\binom{n}{k}$ size-k subgraphs in a graph G of size n ($O(n^k)$ processing steps required), thus there is a trade off between a more accurate representation of the graph (large value of k) and the computational cost. However, some techniques are used in order to resolve this limitation as Sampling From Graph technique (section 1.1.3).

1.1.3 Sampling From Graph

The problem of Graph Sampling arises when we deal with a large-scale graph and the task is the pick a small-size sample subgraph/s that would be similar to the original graph with respect to some important properties.

Sampling from graph techniques are used to resolve the processing cost limitation of graphlet kernel, and it can deployed in two different manners:

1. directly sample m sugraphs $\{H_1, ..., H_m\}$ of size k, and then estimate the k-spectrum vector empirically: $f_G(i) = \frac{1}{m}\Sigma_{j=1}^{m} \mathbb{1}[H_j = graphlet(i)]$
2. sample m subgraphs $\{H_1, ..., H_m\}$ of size n' such that $n \gg n' > k$, then estimate f_G as follows: $f_G = \frac{1}{m}\Sigma_{j=1}^{m} f_{H_j}$

The important thing here is whether a sufficiently large number of random samples will lead to an empirical k-spectrum vector close to the actual vector. The number of samples needed to achieve a given confidence with a small probability of error is called the sample complexity.

Theorem 1 *Let f be a probability distribution on the finite set $\mathcal{G} = g_1, ..., g_a$, $X = X_j{}_{j=1}^{m}$ be a set of independent identically distributed (iid) random variables X_j drawn from f, and $\hat{f}(g_i) = \frac{1}{m}\Sigma_{j=1}^{m}\mathbb{1}(X_j = g_i)$. Then for a given $\epsilon > 0$ and $\delta > 0$ we have (Shervashidze et al., 2009):*

$$m = \left\lceil \frac{2(log(2)a + log(\frac{1}{\delta}))}{\epsilon^2} \right\rceil \tag{1.4}$$

samples suffice to ensure that $P\{\left\| f - \hat{f} \right\|_1 \geq \epsilon\} \leq \delta$

Therefore this theorem gives a lower bound on the number of samples considered respecting the first manner using Graph Sampling to approximate the Graphlet Kernel in order to ensure some certaintyϵ with prabability $1 - \delta$.

1.1.4 Random Features

Kernel machines are of interest as they approximate any function arbitrarily well with sufficiently large training data set. On the other hand, the methods that operate on the kernel matrix (Gram matrix) require a lot of time in order to compute this matrix and to train the machine; for example, a dataset with half a million training examples might take days to train on modern workstations (Rahimi and Recht, 2008). Unlike kernel machines, linear support vector machines and regularized regression run much faster especially with low-dimensionality training data. One way to combine the advantages of the linear and nonlinear vector machines is to convert the training and evaluation of any kernel machine into the corresponding operations of a linear machine by mapping data into a relatively low-dimensional randomized feature space. Instead of considering the implicit lifting function which corresponds to the kernel, it was proposed to explicitly map the data to a low-dimensional

Euclidean inner product space using a randomized feature map $z : \mathcal{R}^d \rightarrow \mathcal{R}^D$ so that the inner product between a pair of transformed points approximates their kernel:

$$k(x, y) =< \phi(x), \phi(y) >\approx z(x)^T z(y) \tag{1.5}$$

Considering this approximation, we can simply transform the input with z and then apply a fast linear learning method to approximate the answer of the real kernel.
In what follows, two methods to construct the random feature map function z are to be presented.

Random Fourier Features

The following Theorem represents the key idea behind this mapping technique

Theorem 2 (Bochner's theorem) *A continuous and shift-invariant kernel $k(x, y) = k(x-y$ on $\mathcal{R}^d$ is positive definite if and only if $k(\delta)$ is the Fourier transform of a non-negative measure.*

What that means is that when a shift-invariant kernel k is properly scaled, its Fourier transform $p(w)$ is a proper probability distribution, we can write:

$$k(x - y) = \int_{\mathcal{R}^d} p(w) e^{jw'(x-y)} dw = E_w[e^{jw'x} e^{jw'y^*}] \tag{1.6}$$

But both $p(w)$ and $k(\delta)$ are real-valued functions, thus from Eq. 1.6 we can prove that:

$$k(x - y) = \int_{\mathcal{R}^d} p(w) cos(w'(x - y)) dw = E_w[z_w(x) z_w(y)] \tag{1.7}$$

where $z_w(x) = \sqrt{2} cos(w'x + b)$ such that w is drawn from $p(w)$ and b is drawn uniformly from $[0, 2\pi]$.

As a straight result, $z_w(x)z_w(y)$ is an unbiased estimate of k(x,y). We can achieve lower variance estimation to the expectation (1.7) by averaging D instances of the estimator with different random frequencies w. i.e. the low-variance estimator can be written as:

$z(x)'z(y) = \frac{1}{D}\Sigma_{j=1}^{D} z_w(x)z_w(y)$. this estimator and based on Hoeffding's inequality guarantees exponentially fast convergence in D between $z(x)'z(y)$ and the kernel true value:

$$Pr(|z(x)'z(y) - k(x,y)| \geq \epsilon) \leq 2e^{\frac{-D\epsilon^2}{4}} \tag{1.8}$$

1.1.5 Random Projections with Optical Processing Units (OPU's)

Random projections is one of the important techniques in machine learning and signal processing, but it requires either to store a very large random matrix, or to use a different, structured matrix to reduce the computational and memory costs. Optical processing units (OPU's) are the tools used to overcome this difficulty, an OPU performs random projections at the speed of light without having to store any matrix in memory. In general, Random projections are the result of two procedures where the first one is the linear-random projections and the second one is non-linear mapping. Mathematically speaking, OPU's perform the following operation (Saade et al., 2016):

$$X = \phi(WU + b);\ W \in \mathcal{R}^{r\times d}, b \in \mathcal{R}^r, U \in \mathcal{R}^d \tag{1.9}$$

where W is the random projection matrix, b is a bias, U is an input point, r is the number of random features and d is the input space dimension.With, W is a random i.i.d complex matrix with Gaussian real and imaginary parts and ϕ is the non-linear mapping function.
In the limit where the number of random features $r \to \infty$, it can be proven by the concentration of the measure that the inner product between the projected data points ($X_i \in \mathcal{R}^r$) in the new feature space tends to a kernel function that depends only on the input points in the original feature space ($U_i \in \mathcal{R}^d$)

1.1.6 OPU structure and functionality

Eq. 1.9 still imply that an OPU need to store and multiply by the random projection matrix. In such units a heterogeneous material, such as paper or any white translucid material, is

used to scatter incident light in a very complex way, so the behavior of light scattering is considered random because of the extremely high complexity. one can argue that light scattering is a linear, deterministic, and reproducible phenomenon, but what can be said is that the unpredictable nature of the process makes it effectively a random process, that is why these materials are called opaque since all information carried within the incident light is seemingly lost during the propagation through the material (Saade et al., 2016). An example used to demonstrate and justify the resulted randomness is a cube of edge length $100\mu m$, such cube can comprise $\approx 10^7$ paint nanoparticles, all the positions and shape of these particles must be known in order to predict its effect on light. Propagation through such a layer can be seen as a random walk because of frequent scattering with the nanoparticles, where light would explore the whole volume and endure on average tens of thousands of such scattering steps before exiting on the other side in a few picoseconds.

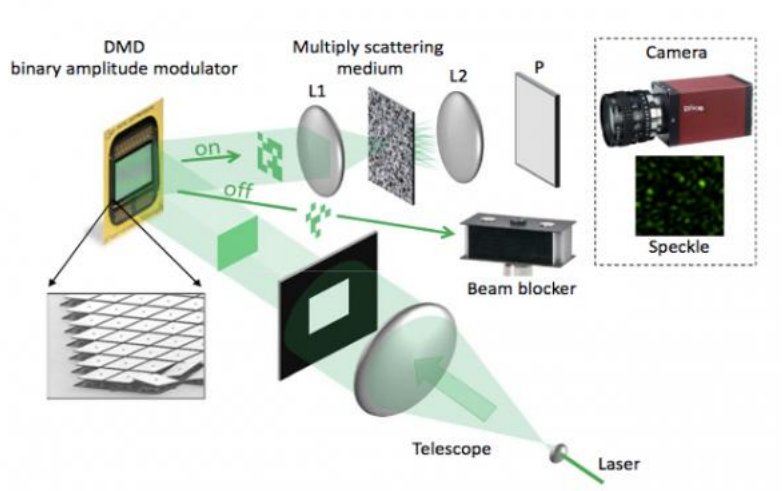

Figure 1.1 OPU's Experimental setup (Saade et al., 2016): A monochromatic laser is expanded by a telescope, then illuminates a digital micromirror device (DMD), able to spatially encode digital information on the light beam by amplitude modulation. The light beam carrying the signal is then focused on a random medium by means of a lens. The transmitted light is collected on the far side by a second lens, passes through a polarizer, and is measured by a standard monochrome CCD camera for example .

When the incident light is coherent, it gives rise to interferences, and the complex interference pattern arising from the multiple scattering process is called speckle. These patterns are not characteristic only of the propagation medium but also of the shape of the incident light, and this can be modeled by $y = Wx$, where y and x are the vector amplitudes between a set of spatial modes at the output and at the input of the medium respectively and W is the transmission matrix of the medium and it was shown to be very close to Gaussian i.i.d matrices (Saade et al., 2016). What's more convenient to use the aforementioned configuration as a platform for random projections, even without W being determined, is that for a stable medium, such as a paint layer, the transmission matrix W is stable as well. So lightening this layer with an appropriate set of incident illuminations, which is possible using a spatial light modulator and a laser, and measuring the output speckle using CCD or CMOS camera, we can record the resulting intensity $|y|^2$, which is the principle concept behind OPU's functionality as seen in Fig 1.1.

The DMD (digital micromirror device) used in OPU's is a Binary Amplitude Modulator consisting of an array of micro-mirrors, Each encodes a binary value(lit or not). In order to represent grey values, each value is encoded on a square sub-array (4×4 for example) in the DMD,where the number of lit mirrors reflects the desired level of grey. DMD reflects the data through the disordered medium, and a snapshot of the resulting random projection is acquired using a standard camera, all that in a very high speed compared to the traditional random features techniques.

REFERENCES

Kriege, Nils M, Fredrik D Johansson, and Christopher Morris (2020). "A survey on graph kernels". In: *Applied Network Science* 5.1, pp. 1–42.

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

Saade, Alaa et al. (2016). "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, pp. 6215–6219.

Shervashidze, Nino et al. (2009). "Efficient graphlet kernels for large graph comparison". In: *Artificial Intelligence and Statistics*, pp. 488–495.