Isolation Graph Kernel

Bi-Cun Xu, Kai Ming Ting, Yuan Jiang
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
{xubc,tingkm,jiangy}@lamda.nju.edu.cn

Abstract

A recent Wasserstein Weisfeiler-Lehman (WWL) Graph Kernel has a distinctive feature: Representing the distribution of Weisfeiler-Lehman (WL)-embedded node vectors of a graph in a histogram that enables a dissimilarity measurement of two graphs using Wasserstein distance. It has been shown to produce better classification accuracy than other graph kernels which do not employ such distribution and Wasserstein distance. This paper introduces an alternative called Isolation Graph Kernel (IGK) that measures the similarity between two attributed graphs. IGK is unique in two aspects among existing graph kernels. First, it is the first graph kernel which employs a distributional kernel in the framework of kernel mean embedding. This avoids the need to use the computationally expensive Wasserstein distance. Second, it is the first graph kernel that incorporates the distribution of attributed nodes (ignoring the edges) in a dataset of graphs. We reveal that this distributional information, extracted in the form of a feature map of Isolation Kernel, is crucial in building an efficient and effective graph kernel. We show that IGK is better than WWL in terms of classification accuracy, and it runs orders of magnitude faster in large datasets when used in the context of SVM classification.

Introduction

Measuring the similarity between graphs is a fundamental problem in learning on graphs. Graph kernels have been highly successful and have shown good predictive performance on a variety of classification problems (Morris et al. 2016; Yanardag and Vishwanathan 2015; Togninalli et al. 2019).

A recent work has shown that it is important to use distributional information in a graph to construct a graph kernel called Wasserstein Weisfeiler-Lehman or WWL kernel (Togninalli et al. 2019). Because most existing graph kernels do not take distributional information into consideration (Kriege, Johansson, and Morris 2020), the WWL kernel has better classification accuracy than these existing kernels. However, the price WWL paid is high computational cost because of the use of the Wasserstein distance. As a result, it could not be used for large datasets.

This paper proposes an alternative approach called Isolation Graph Kernel (IGK). It takes more than one type of distribution of the given dataset into consideration, and it can deal with large scale datasets.

IGK is distinguished from WWL in four aspects. First, IGK employs two types of distributions: the distribution of node vectors in a dataset of graphs (ignoring the edges); and the distribution of the Weisfeiler-Lehmah (WL)-embedded node vectors. Second, although both IGK and WWL employ the same distribution of WL-embedded node vectors, IGK invokes the first level of kernel mean embedding and WWL does not. As a result, IGK operates in Hilbert space and WWL in input space. Third, WWL uses the Wasserstein distance to measure the distance between two distributions; whereas IGK uses the second level of kernel mean embedding to measure the similarity of two distributions. Fourth, IGK has an explicit feature map and WWL does not.

Our contributions are summarized as follows:

1. Introducing Isolation Graph Kernel (IGK) which has the following unique characteristics: [i] It is the first graph kernel which employs a distributional kernel in the framework of kernel mean embedding (KME) (Smola et al. 2007; Muandet et al. 2017). IGK is probably the first non-trivial application of two levels of KME. [ii] It is the first graph kernel which incorporates the distribution of attributed nodes in an entire dataset of graphs. We show that this distributional information extracted in terms of a feature map of Isolation Kernel (Ting, Zhu, and Zhou 2018) is crucial in building an efficient and effective graph kernel.
2. Verifying that the power of IGK is derived from the use of Isolation Kernel (IK) in two aspects. [I] The data dependent property of IK makes full use of the data distribution of the given dataset; while existing data independent kernel such as Gaussian Kernel could not. [II] A current WL scheme (Togninalli et al. 2019) has weak discriminative power (i.e., weak in distinguishing two non-isomorphic graphs). The use of IK in the KME framework is a key to improve the discriminative power of the WL scheme.
3. Showing that IGK has higher predictive accuracy than the state-of-the-art WWL (Togninalli et al. 2019); and it runs orders of magnitude faster in large datasets. This is because it does not employ the computationally expensive Wasserstein distance; and the feature map of IK enables the use of fast linear SVM.

Related Work

Most existing graph kernels are an instance of $\mathcal{R}$-Convolution kernel (Haussler 1999). The core idea is to extract substructures from graphs and then concatenate the substructures in a vectorial representation. The kernel between two graphs is computed in terms of the similarity between these substructures. This conversion is often dubbed embedding. An influential embedding is Weisfeiler-Lehman (WL) scheme (Weisfeiler and Lehman 1968; Shervashidze et al. 2011a) which captures dependencies in a graph in the form of subtrees. This WL scheme applies to labelled graphs only. There are variants/extensions of this scheme since, e.g., Shervashidze et al. (2011b); Morris, Kersting, and Mutzel (2017).

Recently, a different version of the WL scheme has been suggested for attributed graphs which have nodes with continuous attributes and weighted edges (Togninalli et al. 2019). It then represents each graph as a histogram, i.e., the distribution of all WL-embedded node vectors of an attributed graph. The distance between two graphs is measured by the Wasserstein distance between two histograms; and the measured distance is converted to similarity using a Laplacian kernel. This graph kernel is called Wasserstein Weisfeiler–Lehman (WWL) kernel (Togninalli et al. 2019). We use the term WL to refer to this WL scheme in this paper.

Hash Graph Kernel (HGK) (Morris et al. 2016) is another competitive method. It iteratively converts continuous attributes into discrete labels using a randomized hash function. HGK-WL employs the Weisfeiler-Lehman scheme as a hash function; while HGK-SP uses the shortest-path kernel. Both have been shown to be the closest contender to WWL (Togninalli et al. 2019) among several other contenders.

A comprehensive survey of other graph kernels is provided by Kriege, Johansson, and Morris (2020).

One of the criticisms of existing graph kernels (that advocates for the use of graph neural networks for graph representation learning) is that the feature construction scheme of graph kernels is fixed, which does not adapt to the given data distribution (Morris et al. 2019). Despite considering the distribution of WL-embedded node vectors of a graph, WWL (Togninalli et al. 2019) does not address this criticism because it does not consider the distribution of the given dataset either. The proposed graph kernel is the only graph kernel that addresses this criticism.

Kernel mean embedding (Smola et al. 2007; Muandet et al. 2017) on distributions is an effective way to build a distributional kernel from a point kernel, enabling similarity between distributions to be measured. The current approach has focused on point kernels which have a feature map with intractable dimensionality such as Gaussian Kernel. In this paper, we combine kernel mean embedding with Isolation Kernel (Ting, Zhu, and Zhou 2018) which has a finite-dimensional feature map.

Although kernel mean embedding has been applied to anomaly detection (Muandet and Schölkopf 2013; Ting et al. 2020), we are probably the first in applying kernel mean embedding in graph representation.

Isolation Graph Kernel

The idea of Isolation Graph Kernel is to incorporate the distribution of node vectors in a dataset of graphs (ignoring all edges) and the distribution of WL node vectors in a graph. These are considered separately in different layers of embedding. None of the existing graph kernels have taken the first distribution or both these distributions into consideration.

Given a dataset $D = \{G_1, \dots, G_n\}$ of attributed graphs. We denote a graph $G_i = (V_i, E_i)$, where V_i and E_i are the sets of nodes and edges of the graph, respectively. Each node $v \in V$ is associated a vector with m attributes $\mathbf{v} \in \mathbb{R}^m$. A node $v \in V$ has a neighborhood $\mathcal{N}(v) = \mathcal{N}^1(v) = \{u \in V | (v,u) \in E\}$ and $|\mathcal{N}(v)| = deg(v)$. At WL iteration $h > 1$, the neighborhood of v is defined as $\mathcal{N}^h(v) = \{u \in V | (w,u) \in E, w \in \mathcal{N}^{h-1}(v)\}$.

Let a set of node vectors $\mathbf{D} = \cup_{v \in V_i, i \in [1,n]} \mathbf{v}$, extracted from D by ignoring all edges in all graphs. The proposed embedding method has three layers of embeddings:

1 The first layer applies the dataset-wide node embedding $\varphi(\mathbf{v}|\mathbf{D})$ via Isolation Kernel (IK), derived from $\mathbf{D}$. In other words, IK considers the distribution of all nodes in the dataset, ignoring the edges in the graphs.

2 The second layer applies the graph-specific node embedding $\Phi(\mathbf{v}|G)$ on each node in a graph G. The WL embedding is used here. Applying the first two layers gives a feature map $\Phi(\varphi(\mathbf{v}|\mathbf{D})|G)$ for each node in G.

3 The third layer applies a mean embedding over all nodes in each graph $G \in D$. This produces a feature mean map for each graph in the dataset: $\widehat{\Phi}(G \mid \Phi(\varphi(\mathbf{v}|\mathbf{D})|G))$.

The details of each embedding are provided in the next three subsections.

Layer #1: Dataset-Wide Node Embedding Using Isolation Kernel

Isolation Kernel maps each node vector $\mathbf{v} \in \mathbb{R}^m$ into a vector $\varphi(\mathbf{v}|\mathbf{D}) \in \{0,1\}^{t \times \psi}$.

The definition of Isolation Kernel (Ting, Zhu, and Zhou 2018) is given as follows.

Let $\mathbf{D} \subset \mathcal{X} \subseteq \mathbb{R}^m$ be a dataset sampled from an unknown $\mathcal{P}_{\mathbf{D}}$; and $\mathcal{D} \subset \mathbf{D}$, where $|\mathcal{D}| = \psi \geq 2$. Each point $\mathbf{z} \in \mathcal{D}$ has the equal probability of being selected from $\mathbf{D}$.

Given $\mathcal{D}$, we assume that there is a space partitioning mechanism which produces a partitioning H with ψ partitions $\theta[\mathbf{z}] \in H$, where each partition isolates a point $\mathbf{z} \in \mathcal{D}$ from the rest of the points in $\mathcal{D}$. Let $\mathbb{H}_\psi(\mathbf{D})$ denote the set of all partitionings H that are admissible from $\mathcal{D} \subset \mathbf{D}$.

Definition 1. *(Ting, Zhu, and Zhou 2018; Qin et al. 2019) For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, Isolation Kernel of $\mathbf{x}$ and $\mathbf{y}$ is defined to be the expectation taken over the probability distribution on all partitionings $H \in \mathbb{H}_\psi(\mathbf{D})$ that both $\mathbf{x}$ and $\mathbf{y}$ fall into the same isolating partition $\theta[\mathbf{z}] \in H$, where $\mathbf{z} \in \mathcal{D} \subset \mathbf{D}$, $\psi = |\mathcal{D}|$:*

$$\kappa_\psi(\mathbf{x}, \mathbf{y} \mid \mathbf{D}) \quad = \quad \mathbb{E}_{\mathbb{H}_\psi(\mathbf{D})}[\mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta[\mathbf{z}] \mid \theta[\mathbf{z}] \in H)] \tag{1}$$

where $\mathbb{1}(\cdot)$ is an indicator function.

In practice, κ_ψ is constructed using a finite number of partitionings $H_i, i = 1, \ldots, t$, where each H_i is created using randomly subsampled $\mathcal{D}_i \subset \mathbf{D}$; and θ is a shorthand for $\theta[\mathbf{z}]$:

$$\begin{aligned}\kappa_\psi(\mathbf{x}, \mathbf{y} \mid \mathbf{D}) &= \frac{1}{t}\sum_{i=1}^{t} \mathbb{1}(\mathbf{x}, \mathbf{y} \in \theta \mid \theta \in H_i) \\ &= \frac{1}{t}\sum_{i=1}^{t}\sum_{\theta \in H_i} \mathbb{1}(\mathbf{x} \in \theta)\mathbb{1}(\mathbf{y} \in \theta) \quad (2)\end{aligned}$$

Isolation Kernel has been shown to be positive definite (Ting et al. 2020). Thus, Isolation Kernel defines an RKHS $\mathscr{H}$. Its feature map in RKHS is defined below.

Given a partitioning H_i, let $\varphi_i(\mathbf{x})$ be a ψ-dimensional binary column vector representing all partitions $\theta_j \in H_i$, $j = 1, \ldots, \psi$; where $\mathbf{x}$ falls into only one of the ψ partitions. The j-component of the vector is: $\varphi_{ij}(\mathbf{x}) = \mathbb{1}(\mathbf{x} \in \theta_j \mid \theta_j \in H_i)$. Given t partitionings, $\varphi(\mathbf{x})$ is the concatenation of $\varphi_1(\mathbf{x}), \ldots, \varphi_t(\mathbf{x})$.

Definition 2. ***Feature map of Isolation Kernel.*** *For point $\mathbf{x} \in \mathbb{R}^m$, the feature mapping $\varphi : \mathbf{x} \to \{0,1\}^{t\times\psi}$ of κ_ψ is a vector that represents the partitions in every partitioning $H_i \in \mathbb{H}_\psi(\mathbf{D})$, $i = 1, \ldots, t$; where $\mathbf{x}$ falls into only one of the ψ partitions in each partitioning H_i.*

Therefore, Equation 2 can be re-expressed as:

$$\kappa_\psi(\mathbf{x}, \mathbf{y} \mid \mathbf{D}) = \frac{1}{t}\left\langle \varphi(\mathbf{x}|\mathbf{D}), \varphi(\mathbf{y}|\mathbf{D}) \right\rangle$$

Hereafter, $\varphi(\cdot)$ is used as a shorthand for $\varphi(\cdot|\mathbf{D})$.

Layer #2: Graph-Specific Node Embedding Using a Weisfeiler–Lehman Scheme

The Weisfeiler–Lehman (WL) embedding used here is the same as that proposed by Togninalli et al. (2019). The only exception is that we use the Isolation Kernel (IK) mapped vector $\varphi(\mathbf{v})$ instead of node vector $\mathbf{v}$ as the input representation of the WL scheme.

Let $\varphi^0(\mathbf{v}) = \varphi(\mathbf{v})$ be the IK mapped vector of node vector $\mathbf{v}$ for node $v \in V$ in a graph $G = (V, E)$. The WL embedding at iteration $h > 0$ is recursively defined as:

$$\varphi^h(\mathbf{v}) = \frac{1}{2}\left(\varphi^{h-1}(\mathbf{v}) + \frac{1}{deg(v)}\sum_{u\in\mathcal{N}(v)} \mathsf{w}(v,u)\cdot\varphi^{h-1}(\mathbf{u})\right) \quad (3)$$

The weight $\mathsf{w}(v,u)$ is set to the given edge weight of a graph. For unweighted edges, it is set to 1.

Definition 3. *IK-induced WL-features: Let $G = (V, E)$ and h be the number of WL iterations. The IK-induced WL-features for node $v \in V$ is defined as $\Phi(\mathbf{v}) \in \mathbb{R}^{(h+1)\times t\times\psi}$:*

$$\Phi(\mathbf{v}) = [\varphi^0(\mathbf{v}), \ldots, \varphi^h(\mathbf{v})]^\top$$

Layer #3: Mean Embedding of a Graph

Here we view each graph $G = (V, E)$ as having a representative sample of WL-embedded node vectors $\Phi(\mathbf{v})\ \forall v \in V$ of an unknown distribution, where each node vector is represented using the IK-induced WL embedding. Having this view, a mean embedding can be used to represent each graph.

For a graph $G = (V, E)$, its mean embedding or its feature mean map is given as

$$\widehat{\Phi}(G) = \frac{1}{|V|}\sum_{v\in V}\Phi(\mathbf{v}) = \frac{1}{|V|}\sum_{v\in V}[\varphi^0(\mathbf{v}), \ldots, \varphi^h(\mathbf{v})]^\top$$

For any pair of two graphs, Isolation Graph Kernel (IGK), using the above three layers of embedding, is calculated by a dot product between their two mean embeddings or feature mean maps, i.e.,

$$K^{(1,2,3)}(G, G') = \left\langle \widehat{\Phi}(G), \widehat{\Phi}(G') \right\rangle$$

$K^{(1,2,3)}$, expressed in a typical formulation of kernel mean embedding (Smola et al. 2007; Muandet et al. 2017) in terms of a base kernel κ, is given as follows:

$$K^{(1,2,3)}(G, G') = \frac{1}{|V||V')|}\sum_{v\in V}\sum_{v'\in V'}\kappa(\mathbf{v}, \mathbf{v}')$$

where $\kappa(\mathbf{v}, \mathbf{v}') = \langle\Phi(\mathbf{v}), \Phi(\mathbf{v}')\rangle$ is the base kernel as a result of the combined layers 1 & 2 embeddings.

The algorithm of generating the feature mean map of IGK and its mapped dataset is given in Algorithm 1.

Algorithm 1: Generate IGK mapped dataset

Input : Dataset $\{G_i = (V_i, E_i), i = 1, \ldots, n\}$, IK parameter ψ, WL parameter h.
Output: IGK mapped dataset $\{\widehat{\Phi}(G_i), i = 1, \ldots, n\}$.

$\mathbf{D} = \cup_{v\in V_i, i\in[1,n]}\ \mathbf{v}$ /* Set of node vectors */
Produce $\varphi(\cdot|\mathbf{D})$ from $\mathbf{D}$ /* Embedding #1 */
for *($i = 1$; $i < n$; i++)* **do**
  for *(each $v \in V_i$)* **do**
    /* Embedding layer #2 */
    Map $\varphi^{j-1}(\mathbf{v})$ into $\varphi^j(\mathbf{v})$ for $j = 1, \ldots, h$;
    $\Phi(\mathbf{v}) = [\varphi^0(\mathbf{v}), \ldots, \varphi^h(\mathbf{v})]^\top$;
  end
  /* Embedding layer #3 */
  $\widehat{\Phi}(G_i) = \frac{1}{|V_i|}\sum_{v\in V_i}\Phi(\mathbf{v})$;
end
return $\{\widehat{\Phi}(G_i), i = 1, \ldots, n\}$

The IGK mapped dataset is used as input to a linear SVM to produce a classifier.

We describe three advantages of IGK over WWL in the following three sections.

Advantage 1: Harnessing Information in The Dataset-Wide Node Distribution

IGK has a dataset-wide node embedding in the first layer using Isolation Kernel; but WWL has no such embedding.

The distribution of node vectors in a dataset of graphs is an important part of the information in the dataset, in addition to the connectivity information due to edges. This dataset-wide distribution is often ignored in existing graph kernels, just like the ordinary data independent kernels such as Gaussian kernel and Laplacian kernel which ignore the dataset-wide distribution of points in $\mathbb{R}^m$.

The explanation of why IGK has better accuracy than that using the Gaussian kernel (which replaces Isolation Kernel) is the same as that provided in the first paper on Isolation Kernel (see Section 3 in Ting, Zhu, and Zhou (2018)): In a dataset consists of sparse and dense regions, Gaussian kernel makes more errors in the sparse region bordering the dense region; and Isolation Kernel makes fewer errors—a direct result of the data dependent property of Isolation Kernel.

Advantage 2: Improve the Discriminative Power of WL

The WL embedding is the only mechanism in both WWL and IGK (as well as many existing graph kernels) in extracting information from substructures in a graph.

We explain that the WL scheme (Togninalli et al. 2019) has weak discriminative power in distinguishing non-isomorphic graphs and how this power can be improved in the next two subsections. The third subsection explains the phenomenon of low WL iterations in existing WL applications.

The Current WL Has Weak Discriminative Power

Figure 1(a) shows an example of applying the WL scheme in the input space of node vectors. Two non-isomorphic graphs G_1 and G_2 have increasing similarity as h increases because each graph is shrunk into increasingly smaller size. At $h = \infty$, each of the two graphs is shrunk into one point which is the center of the node representations in the space.

In the extreme case that both graphs have the same graph center, then both graphs are shrunk into the same point! We formalize this notion as follows.

The WL scheme for $h > 0$ in the input space (as proposed by Togninalli et al. (2019) in Eq 4) can be re-expressed in terms of node vectors $\mathbf{u}^0 = \mathbf{u}$ in Eq 5 as follows:

$$\mathbf{v}^h = \frac{1}{2}[(\mathbf{v}^{h-1} + \frac{1}{deg(v)} \sum_{u \in \mathcal{N}(v)} \mathsf{w}(v,u) \cdot \mathbf{u}^{h-1}] \quad (4)$$

$$= \sum_{u \in \mathsf{N}^h(v)} \varpi_u^h \mathbf{u} \quad (5)$$

where $\sum_{u \in \mathsf{N}^h(v)} \varpi_u^h = 1$; and ϖ_u^h is the weight of $\mathbf{u}$ after h iterations; and $\mathsf{N}^h(v) = \cup_{k \in [1,h]} \mathcal{N}^k(v) \cup v$ is the h-hop neighborhood of v after h iterations. Note that all superscripts are indexes, not power.

Definition 4. *The graph center $\mathbf{c}_G$ of graph $G = (V, E)$ in the input space of node vectors is defined to be the point such that for all $v \in V$, $\mathbf{v}^\infty = \mathbf{c}_G$.*

Proposition 1. *Given two graphs of arbitrary distributions of node vectors in the input space, the only requirement for them to shrink into the same point at $h = \infty$ is that they both have the same graph center.*

This proposition is a direct consequence of Definition 4: Given two non-isomorphic graphs G and G' such that all $v \in V$,$v' \in V'$, $\mathbf{v}^\infty = \mathbf{v'}^\infty \Rightarrow \mathbf{c}_G = \mathbf{c}_{G'}$.

In other words, applying the WL scheme in the input space of node vectors reduces the discriminative power between graphs as h increases. This means that the WL scheme fails to differentiate two completely different graphs when they have the same graph centers. Proposition 1 delineates the condition under which the WL scheme is not suitable for graph isomorphism test.

Mapping Node Vectors to IK's Feature Space Increases the Discriminative Power of WL

There is a simple way to improve the power of the WL scheme to discriminate two different graphs. The idea is to treat each graph as a sample of nodes generated from an unknown distribution. Then, we can use the kernel mean embedding (Smola et al. 2007) to measure the similarity between two graphs as two distributions.

The key criterion in kernel mean embedding is to employ a point kernel κ which is positive definite and a characteristic kernel. This ensures that the kernel mean map is injective, *i.e.*, $\widehat{\phi} : \mathbb{P} \to \mathscr{H}$ is injective or $\|\widehat{\phi}(\mathcal{P}) - \widehat{\phi}(\mathcal{P}')\|_{\mathscr{H}} = 0$ if and only if $\mathcal{P} = \mathcal{P}'$ (Smola et al. 2007). If the kernel κ is non-characteristic, two different distributions $\mathcal{P} \neq \mathcal{P}'$ may be mapped to the same $\widehat{\phi}(\mathcal{P}) = \widehat{\phi}(\mathcal{P}')$.

Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ and a dataset $D \sim \mathcal{P}_D$. Given a base kernel $\kappa(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$, the kernel mean map of $\mathcal{P}_D$ is defined as an estimation from D (Smola et al. 2007):

$$\widehat{\phi}(\mathcal{P}_D) = \frac{1}{|D|} \sum_{\mathbf{x} \in D} \phi(\mathbf{x}). \quad (6)$$

Like Eq 5, the IK-induced WL for $h > 0$ in Eq 3 can be similarly re-expressed in terms of $\varphi^0(\mathbf{u}) = \varphi(\mathbf{u})$ as

$$\varphi^h(\mathbf{v}) = \sum_{u \in \mathsf{N}^h(v)} \omega_u^h \varphi(\mathbf{u}) \quad (7)$$

Comparing Eq 7 with Eq 6, we have $\omega_u^h \varphi(\mathbf{u}) \equiv \phi(\mathbf{u})$ as the weighted feature map of κ, and $\mathsf{N}^h(v)$ the h-hop neighborhood of node v is the representative sample of an unknown distribution associated with node vector $\mathbf{v}$ after h iterations of WL. Then, $\varphi^h(\mathbf{v}) \equiv \widehat{\phi}(\mathcal{P}_\mathbf{v})$ is the corresponding feature mean map of the distribution.

Proposition 2. *The IK-induced WL expressed in Eq 7 is a kernel mean map using Isolation Kernel as the base kernel, where $\omega_u^h \varphi$ is the (weighted) feature map of Isolation Kernel and φ^h is the kernel mean map.*

Recall that Isolation Kernel is positive definite and a characteristic kernel (Ting et al. 2020). Thus, its kernel mean

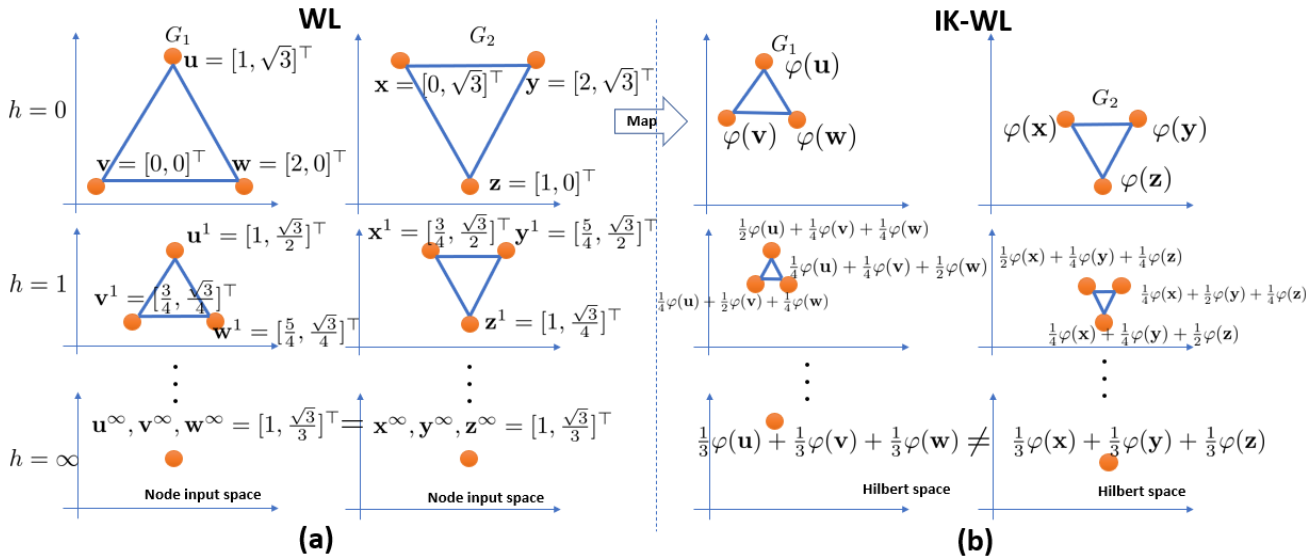

Figure 1: The effect of h in the WL scheme on two non-isomorphic graphs G_1 and G_2 when applied in (a) node input space (WL) where both graphs have the same graph center as defined in Definition 4; and (b) Hilbert space (IK-WL). $\mathbf{u}, \mathbf{v}$ and $\mathbf{w}$ are node vectors of G_1; $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are node vectors of G_2. φ is the feature map of IK. For $h > 0$, the vectors in input space are computed using Eq 5 (WL); and the weights of the components of a vector in Hilbert space are derived from Eq 7 (IK-WL).

map is injective. Note that the WL scheme (Togninalli et al. 2019) is an approximate isomorphism test. Being injective improves its discriminative power; though this does not guarantee a foolproof graph isomorphism test (which is an NP-hard problem (Garey and Johnson 1979).)

As a direct consequence of mapping node vectors to Hilbert space using Isolation Kernel and applying WL in Hilbert space, we have effectively improved the discriminative power of WL because φ^h is equivalent to the (weighted) mean map of Isolation Kernel.

Figure 1(b) illustrates an example of mapping node vectors in Figure 1(a) to Hilbert space. This enables different distributions (representing G_1 and G_2) to be mapped to different regions in Hilbert space. As h increases to ∞, each sample of a distribution will shrink from a group of individual points to a single point in Hilbert space (as in the case in input space). But two different distributions are shrunk into two different points, enabling them to be discriminated; while their counterparts in input space fail to do so when they have the same graph center (as shown in Figure 1(a).)

The Issue of Low Iterations of WL Explained

The WL scheme has been recognized as a successful method for approximate graph isomorphism test. Yet, some have noticed that only low WL iterations are most effective (even the WL incarnations in Graph Neural Networks or GNNs have the same phenomenon[1] see e.g., Liu, Gao, and Ji (2020).

We explain this phenomenon as follows: It is a result of applying WL in the node input space, without considering the distribution of the node vectors. As described previously, this leads to weak discriminative power as h increases.

[1] An explanation of this phenomenon is that GNNs produces the same effect as a low pass filter (Liu, Gao, and Ji 2020). However, this does not explain the effect due to the direct application of WL on the node input space. Also note that the WL scheme for labelled graphs has no such issue.

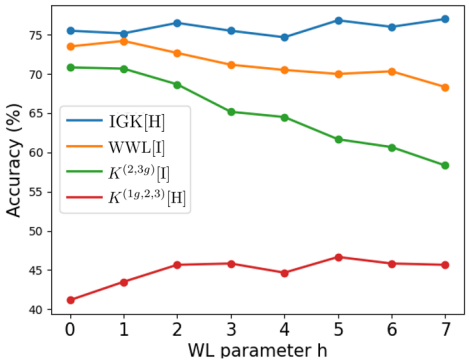

Figure 2: Different outcomes of h parameter when employing WL in Hilbert space [H] and input space [I] in four methods. The experiment was done on the ENZYMES dataset.

Figure 2 shows the effect of h on methods that employ (a) WL in Hilbert space [H]: IGK and $K^{(1g,2,3)}$; and (b) WL in input space [I]: WWL and $K^{(2,3g)}$. $K^{(1g,2,3)}$ and $K^{(2,3g)}$ are variants of IGK which use Gaussian kernel in layers 1 and 3, respectively (see the experiments section for details).

Because IGK and $K^{(1g,2,3)}$ employ WL in Hilbert space [H], the substructure information of nodes in a graph is preserved with high h iterations. This is manifested as non-decreasing accuracy for IGK and $K^{(1g,2,3)}$ as h increases. In contrast, the accuracy of the methods which employ WL in input space [I] decreases significantly as h increases. In a nutshell, the discriminative power of WL for large subgraphs, which is subdued in input space, is released in Hilbert space.

Advantage 3: IGK Has an Explicit Feature Map

IGK has an explicit feature map which enables the use of fast linear SVM. In contrast, WWL has no explicit feature map and it must use a SVM which solves the dual optimization problem, in addition to the use of computationally expensive Wasserstein distance. As a result, SVM using WWL runs much slower than that using IGK.

Section Summary: The source of the three advantages of IGK over WWL is the use of Isolation Kernel which takes the distribution of node vectors into consideration and has an explicit feature map.

IGK Uses Two Levels of Kernel Mean Embedding

As described above, $\varphi^j(\mathbf{v})$ for every $j \in [1, h]$ can be viewed as the first level of kernel mean embedding with Isolation Kernel as the base kernel. This mean embedding is applied to all node vectors of the j-hop neighbourhood of $\mathsf{N}^h(v)$ associated with $\mathbf{v}$ of graph G—this set of node vectors is treated as the representative sample of an unknown distribution of $\mathbf{v}$ of G created by the WL scheme with j iterations. Note that this kernel mean embedding is 'concealed' because the averaging is part of the WL scheme.

The second level of kernel mean embedding is explicit, and it employs a base kernel which has a feature map $\Phi(\mathbf{v})$ consisting of a concatenation of $\varphi^j(\mathbf{v})$ for all $j \in [0, h]$, as given in Definition 3. This mean embedding is applied to a graph, averaged over all WL-embedded node vectors of the graph. The set of WL-embedded node vectors having feature map $\Phi(\mathbf{v})$ is a representative sample of an (unknown) distribution created from all $\mathbf{v}$ in G.

This is interesting because IGK is probably the first non-trivial application of two levels of kernel mean embedding.

Note that Isolation Kernel is crucial in the framework of kernel mean embedding to get high accuracy. A head-to-head comparison between IGK and $K^{(1g,2,3)}$ in Figure 2 reveals its importance: Isolation Kernel which considers the distribution of node vectors produces significantly higher accuracy than Gaussian kernel which is data independent; though both IGK and $K^{(1g,2,3)}$ use two levels of kernel mean embedding. Further evidence is provided in the next section.

Experiments

The experiments have two aims: (i) Examine the role of Isolation Kernel in IGK; and (ii) Compare the relative performance of IGK and existing graph kernels in terms of classification accuracy and runtime.

To achieve the first aim, the following variants of IGK ($K^{(1,2,3)}$) are used in an ablation study:

(a) $K^{(2,3)}$: No dataset-wide node embedding but has the next two layers of embedding. This variant is a head-to-head contender to the proposed 3-layer embedding $K^{(1,2,3)}$ (to test the utility of IK in the first layer).

(b) $K^{(2,3g)}$: This variant applies the existing kernel mean embedding (Muandet et al. 2017) which employs Gaussian kernel in layer 3. It is expressed as follows:

$$K^{(2,3g)}(G, G') = \sum_{v \in V} \sum_{v' \in V'} \kappa_g(X(\mathbf{v}), X(\mathbf{v}'))$$

where $X(\mathbf{v}) = [(\mathbf{v}^0, \ldots, \mathbf{v}^h]^\top$ & κ_g is Gaussian Kernel. This is denoted as RBF-WL in Togninalli et al. (2019).

(c) $K^{(1g,2,3)}$ employs the approximation feature map of Gaussian kernel derived from the Nystrom method (Williams and Seeger 2001; Musco and Musco 2017) in layer 1. This variant examines whether Gaussian kernel could be as effective as Isolation Kernel.

The key difference between IGK and these variants is that none of them utilize the distribution of all nodes in a dataset.

To achieve the second aim, three existing graph kernels (which performed the best in Togninalli et al. (2019)) are used in the comparison:

- The WWL kernel (Togninalli et al. 2019) has high accuracy; but it has slow runtime because it needs to calculate the Wasserstein distance and the kernel matrix.
- Hash graph kernels HGK-WL and HGK-SP (Morris et al. 2016) employs WL and the shortest path kernel as the hash function, respectively.

We use the codes provided by the authors of WWL and HGK; and IGK[2] is coded based on IK implemented using Isolation Forest (Ting, Zhu, and Zhou 2018; Liu, Ting, and Zhou 2008).

We use SVM classifiers in scikit-learn (Pedregosa et al. 2011), i.e., LinearSVC and SVC which are based on Liblinear (Fan et al. 2008) and Libsvm (Chang and Lin 2011), respectively. Because Liblinear usually produces lower accuracy than Libsvm, both our result and the previous result (e.g., Togninalli et al. (2019)) have used the latter for all three existing graph kernels. We have used Liblinear for IGK as the accuracy degradation is small.

See the Appendix for the full experimental settings and some additional results.

Classification Accuracy

The accuracy results of the comparison are shown in Table 1. We have the following observations:

- IGK outperforms its three variants in all datasets. The only exception is AIDS wrt $K^{(2,3g)}$ which is IGK's closest contender variant. The comparisons with $K^{(1g,2,3)}$ and $K^{(2,3)}$ are head-to-head comparisons in the same framework which shows the value of using data dependent IK versus data independent Gaussian kernel or no kernel in layer 1. The differences in accuracy are large on e.g., ENZYMES, IMDB-B, COIL-DEL and COX2_MD.
- IGK outperforms WWL, HGK-WL and HGK-SP on 11 out of 13 data sets. IGK has the highest average rank of all these contenders. A rank-based Nemenyi test (Demšar 2006) shows that IGK is significantly better than WWL at $p = 0.02$ level.

[2]The IGK code is available at github.com/IsolationKernel.

Dataset	#graphs	#attr	#class	IGK	$K^{(2,3)}$	$K^{(2,3g)}$	$K^{(1g,2,3)}$	WWL	HGK-WL	HGK-SP
BZR	405	3	2	**86.8**	78.8	84.0	83.8	84.4	85.4	84.1
ENZYMES	600	18	6	**77.0**	46.7	70.8	47.8	74.2	66.2	68.4
IMDB-B	1000	1	2	**75.0**	53.4	73.2	65.6	74.3	73.8	73.3
COIL-DEL	3900	2	100	**94.1**	12.5	91.0	14.6	91.3	93.4	92.3
AIDS	2000	4	2	98.5	90.7	98.8	95.4	**99.5**	99.4	99.2
DHFR	467	3	2	76.6	61.3	75.7	73.7	78.9	80.3	**80.5**
COX2	467	3	2	**79.5**	78.1	75.6	79.2	78.2	78.4	78.3
COIL-RAG	3900	64	100	**98.9**	95.9	96.9	15.8	96.2	94.4	94.2
PROTEINS_full	1113	29	2	**75.1**	74.3	74.0	65.6	74.7	73.0	74.7
BZR_MD	306	1	2	**74.5**	55.5	71.9	55.7	74.2	72.9	72.6
COX2_MD	303	1	2	**69.0**	49.8	66.4	53.7	68.8	67.7	67.4
DHFR_MD	393	1	2	**79.2**	67.9	73.5	59.1	74.8	73.8	71.8
TWITTER	144033	1	2	**58.1**	52.8	>24hr	51.9	>24hr	>24hr	52.3
Rank				1.53	6.00	4.61	5.76	2.76	3.38	3.61

Table 1: 10-fold CV SVM classifier results in accuracy(%). The last row shows the rank of each algorithm in each dataset, averaged over all datasets. The algorithms having the smallest and largest errors are ranked 1 and 7, respectively.

In addition, we show that adding IK as the first layer embedding to WWL, denoted as IK-WWL, improves WWW's accuracy on ENZYMES from 74.2% to 75.5%. We are unable to run more experiments on IK-WWL because its runtime increased significantly due to the increased dimensionality of IK feature map.

Time Complexities And Scaleup Test

Algorithm	Time complexity
IGK	$O(t\psi(n+he)N)$
WWL	$O(heN + n^3\log(n)N^2)$
HGK-WL	$O(hn^2 4^d N^2)$
HGK-SP	$O(n^3N^2)$

Table 2: Time complexity. n and e are the maximum numbers of nodes and edges, respectively, in each graph. N is the number of graphs. d is the maximum degree of nodes.

The time complexities of IGK and existing graph kernels are given in Table 2.

Figure 3 shows the result of the scaleup test of these four methods on TWITTER which has more than 144 thousand graphs. IGK is significantly faster than the other three methods when dataset sizes are large. This is because the time complexity of IGK is linear in the number of graphs. The other methods have quadratic time complexity. For the two methods which could complete the run on 100% data size, IGK and HGK-SP increase their runtimes by a factor 8 and 80, respectively, as the data size increases by a factor of 10. This difference is expected to enlarge for larger datasets.

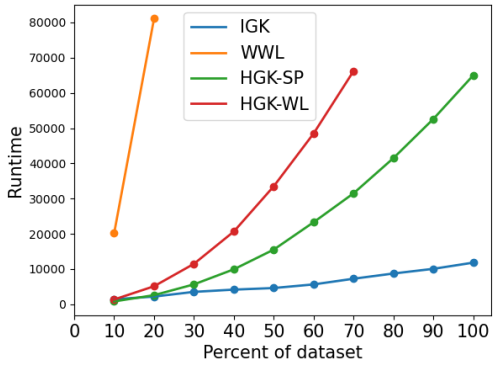

Figure 3: Runtimes (in CPU seconds) of IGK, WWL and HGK on increasing data sizes of the TWITTER dataset (consists of 144033 graphs of two classes). Note that both WWL and HGK-WL could not complete the runs in 24 hours on data size larger than 20% and 70%, respectively.

Discussion

Aggregate via averaging and WL. It is interesting to note that WWL was motivated to avoid the use of sum or average to aggregate the final set of substructures to prevent losing valuable information about the distribution of individual components (Togninalli et al. 2019). Yet, it uses the average in the WL scheme to aggregate neighborhood information. This averaging has been shown to be not injective (Xu et al. 2019). We show that using the same average aggregation, but operated in the Hilbert space of a characteristic kernel in the framework of KME, the WL scheme becomes injective.

Using Gaussian kernel in the kernel mean embedding (KME) framework is injective in terms of discriminating two distributions (Sriperumbudur et al. 2010; Muandet et al. 2017). However, like in the case of applying KME in anomaly detection (Ting et al. 2020), Gaussian kernel has produced weaker accuracy than Isolation Kernel when applied to build graph kernels for classification. This is shown in the comparison between IGK and $K^{(1g,2,3)}$ in the experiments section. This shows that IK which incorporates the dataset-wide node distribution is critical for KME to work well in practice.

Relation to Graph Neural Networks (GNNs). For labelled graphs, it is shown theoretically that Graph Isomorphism Network (GIN), a maximally powerful GNN, is at most as powerful as the WL test (Xu et al. 2019) in distinguishing two non-isomorphic graphs.

For attributed graphs, IGK offers to be a proxy to the WL test which is as powerful as the original WL test. In contrast to the approach in GIN (Xu et al. 2019) which advocates the use of sum (instead of average) as the aggregation function, IGK shows that the averaged version of WL can be as powerful if it is applied in the IK's feature space instead of the input space. This is because the average in IK's feature space denotes the distributional information that incorporates both the dataset-wide node distribution and the neighboring node distribution specific to a graph, rather than just the proportion between distinct neighbouring nodes in input space as described in Xu et al. (2019).

We compare IGK with GIN. IGK ran orders of magnitude faster than GIN. For example on the 10 percent of the TWITTER dataset that GIN could afford to run in one day, GIN took 87,542 seconds (outside the scale shown in Figure 3); whereas IGK took 1,431 seconds only. Because of high runtime and having many tuning parameters (hidden units, batch size, dropout ratio and epochs), we ran a few small datasets for comparison: In terms of accuracy, GIN is competitive with IGK on IMDB-B (75.1%), AIDS (99.5%), COX2 (77.7%) and COIL-RAG (95.9%).

The WL scheme we used is the first order WL. Higher-order WL schemes as used in GNNs (Morris et al. 2019) can be similarly applied in IGK.

Conclusions

The proposed IGK has higher predictive accuracy than the state-of-the-art graph kernel WWL in most datasets; and it runs orders of magnitude faster in large datasets.

IGK achieves high accuracy for two reasons. First, it makes use of two types of distributions via two levels of kernel mean embedding. Second, it applies the Weisfeiler-Lehman (WL) scheme in IK's feature space instead of node input space to perform node embedding. We show that the use of IK's feature space is critical in preserving the structure of subgraphs, making the WL scheme injective—leading to more discriminative power. WWL employs only one type of distribution and applies WL in node input space.

Because IGK has linear time feature mapping and an explicit feature map, it enables the use of a linear SVM classifier. In contrast, WWL employs the computationally expensive Wasserstein distance and has no explicit feature map.

The use of Isolation Kernel as the base kernel is instrumental in IGK achieving high accuracy and fast runtime.

Acknowledgements

Kai Ming Ting is supported by Natural Science Foundation of China (62076120).

Appendix A: Experimental Settings

The parameter search ranges of all these methods in the experiments are given in Table 3.

Algorithm	Parameter search ranges
IGK	$h = \{0, \dots, 7\}; \psi = \{2^4, \dots, 2^{11}\}; t = 100$
$K^{(2,3)}$	$h = \{0, \dots, 7\}$
$K^{(2,3g)}$	$h = \{0, \dots, 7\}; \gamma = \{2^{-3}, \dots, 2^4\}$
$K^{(1g,2,3)}$	$h = \{0, \dots, 7\}; \gamma = \{2^{-3}, \dots, 2^4\}$
WWL	$h = \{0, \dots, 7\}; \lambda = \{10^{-4}, \dots, 10^1\}$
HGK-WL	$h = \{0, \dots, 7\}; \sigma = \{2^{-3}, \dots, 2^4\}$
HGK-SP	$\sigma = \{2^{-3}, \dots, 2^4\}$

Table 3: Parameter search ranges.

We use 13 benchmark datasets commonly used to evaluate graph kernels (Kersting et al. 2016). In each dataset, every graph $G = (V, E)$ has a set of attributed nodes V and a set of edges E, where each node v is associated with a vector $\mathbf{v} \in \mathbb{R}^m$. For datasets that don't have node attributes but have edge attributes, we convert graphs in these datasets into dual graphs (where each edge is represented as a node, and connectivity is established if two edges in the primal graph share the same node). These dual graph datasets are: BZR_MD, COX2_MD, DHFR_MD, TWITTER.

The machine used in the experiments has 755G memory and Intel E5 CPU.

IGK and its variants are normalised to $[0, 1]$ as follows:

$$\hat{K}(G, G') = \frac{K(G, G')}{\sqrt{K(G, G)}\sqrt{K(G', G')}}$$

Appendix B: Additional Experimental Results

Figure 4 shows that IGK becomes more stable and produces potentially higher accuracy as parameter t increases.

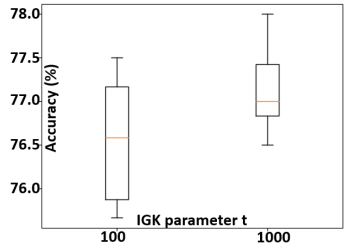

Figure 4: Stability analysis of IGK on ENZYMES dataset.

As shown in Table 4, HGK-WL and HGK-SP using Libsvm have better accuracy than those using Liblinear.

Dataset	HGK-WL		HGK-SP	
	liblinear	libsvm	liblinear	libsvm
BZR	85.1	85.7	81.9	84.1
ENZYMES	64.8	66.2	64.8	68.4
IMDB-B	68.7	73.8	66.4	73.3
COIL-DEL	89.8	93.4	89.6	92.3

Table 4: Accuracy comparison between Liblinear and Libsvm for HGK-WL and HGK-SP.

References

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2: 27:1–27:27.

Demšar, J. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 1–30.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 1871–1874.

Garey, M.; and Johnson, D. S. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. New York, Freeman.

Haussler, D. 1999. *Convolution kernels on discrete structures*. Technical report, Department of Computer Science, University of California.

Kersting, K.; Kriege, N. M.; Morris, C.; Mutzel, P.; and Neumann, M. 2016. Benchmark Data Sets for Graph Kernels. Http://graphkernels.cs.tu-dortmund.de[accessed 09/09/2020].

Kriege, N. M.; Johansson, F. D.; and Morris, C. 2020. A survey on graph kernel. *Applied Network Science* 5(6).

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining*, 413–422.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards Deeper Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348. ACM.

Morris, C.; Kersting, K.; and Mutzel, P. 2017. Glocalized Weisfeiler-Lehman Graph Kernels: Global-Local Feature Maps of Graphs. In *IEEE International Conference on Data Mining*, 327–336.

Morris, C.; Kriege, N. M.; Kersting, K.; and Mutzel, P. 2016. Faster Kernels for Graphs with Continuous Attributes via Hashing. In *IEEE 16th International Conference on Data Mining*, 1095–1100.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence*.

Muandet, K.; Fukumizu, K.; Sriperumbudur, B.; and Schölkopf, B. 2017. Kernel Mean Embedding of Distributions: A Review and Beyond. *Foundations and Trends in Machine Learning* 10 (1–2): 1–141.

Muandet, K.; and Schölkopf, B. 2013. One-class Support Measure Machines for Group Anomaly Detection. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, 449–458.

Musco, C.; and Musco, C. 2017. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, 3833–3845.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

Qin, X.; Ting, K. M.; Zhu, Y.; and Lee, V. C. S. 2019. Nearest-Neighbour-Induced Isolation Similarity and Its Impact on Density-Based Clustering. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence*, 4755–4762.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011a. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12(9).

Shervashidze, N.; Schweitzer, P.; van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011b. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research* 12(77): 2539–2561.

Smola, A.; Gretton, A.; Song, L.; and Schölkopf, B. 2007. A Hilbert Space Embedding for Distributions. In Hutter, M.; Servedio, R. A.; and Takimoto, E., eds., *Algorithmic Learning Theory*, 13–31. Springer.

Sriperumbudur, B. K.; Gretton, A.; Fukumizu, K.; Schölkopf, B.; and Lanckriet, G. R. 2010. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research* 11: 1517–1561.

Ting, K. M.; Xu, B.-C.; Takashi, W.; and Zhou, Z.-H. 2020. Isolation Distributional Kernel: A new tool for kernel based anomaly detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 198–206. ACM.

Ting, K. M.; Zhu, Y.; and Zhou, Z.-H. 2018. Isolation Kernel and its effect on SVM. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2329–2337. ACM.

Togninalli, M.; Ghisu, E.; Llinares-López, F.; Rieck, B.; and Borgwardt, K. 2019. Wasserstein Weisfeiler-Lehman Graph Kernels. In *Advances in neural information processing systems*.

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

Williams, C. K.; and Seeger, M. 2001. Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems*, 682–688.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *International Conference on Learning Representation*.

Yanardag, P.; and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374.