Global Weisfeiler-Lehman Graph Kernels

Christopher Morris [1] **Kristian Kersting** [1] **Petra Mutzel** [1]

Abstract

Most state-of-the-art graph kernels only take local graph properties into account, i.e., the kernel is computed with regard to properties of the neighborhood of vertices or other small substructures only. On the other hand, kernels that do take global graph properties into account may not scale well to large graph databases. Here we propose to start exploring the space between local and global graph kernels, striking the balance between both worlds. Specifically, we introduce a novel graph kernel based on the k-dimensional Weisfeiler-Lehman algorithm, and show that it takes local as well as global properties into account. Unfortunately, the k-dimensional Weisfeiler-Lehman scales exponentially in k. Consequently, we devise a stochastic version of the kernel with provable approximation guarantees using conditional Rademacher averages. On bounded-degree graphs, it can even be computed in constant time. We support our theoretical results with experiments on several graph classification benchmarks, showing that our kernels often outperform the state-of-the-art in terms of classification accuracies.

1. Introduction

In several domains like chemo- and bioinformatics, or social network analysis large amounts of structured data, i.e., *graphs*, are prevalent. Unfortunately, most state-of-the-art *graph kernels* only take *local* graph properties into account, i.e., they compute the kernel based on properties of the neighborhood of vertices or other small substructures, e.g., see (Shervashidze et al., 2009; Costa & De Grave, 2010; Shervashidze et al., 2011; Orsini et al., 2015). Moreover, kernels that do take global graph properties into account may not scale to large graph databases, e.g., see (Johansson et al., 2014).

Recently, a graph kernel based on the 1-*dimensional Weisfeiler-Lehman* algorithm (Babai & Kucera, 1979) has been proposed (Shervashidze et al., 2011). The 1-dimensional Weisfeiler-Lehman or *color refinement* algorithm is a well known heuristic for deciding whether two graphs are isomorphic: Given an initial *coloring* or *labeling* of the vertices, e.g., their degree, of both graphs, in each iteration two vertices are assigned a different label if the number of neighbors labeled with a certain label is not equal. In this case the algorithm terminates. It is easy to see that the algorithm is not able to distinguish all non-isomorphic graphs, e.g., see (Arvind et al., 2015). On the other hand, this simple algorithm is already quite powerful since it can distinguish almost all graphs with high probability (Babai & Kucera, 1979; Babai et al., 1980), and has been applied in other areas (Grohe et al., 2014; Kersting et al., 2014; Li et al., 2016). Moreover, it can be generalized to k-tuples leading to a more powerful graph isomorphism heuristic (Cai et al., 1992). Now the so called *Weisfeiler-Lehman subtree kernel* is defined by computing the above heuristic for a predefined number of steps and the kernel finally counts the number of common labels arising in all refinement steps.

Clearly, the Weisfeiler-Lehman subtree kernel takes only local graph properties into account when performed for a fixed number of iterations. On the other hand, the k-dimensional variant does take global properties but does not consider local properties.

1.1. Contributions

Our contributions can be summed up as follows.

A Kernel Based on the k-dimensional Weisfeiler-Lehman Algorithm We propose a graph kernel based on the k-dimensional Weisfeiler-Lehman algorithm. In order to take local properties into account we propose a *local* variant of the above algorithm. Moreover, we show that our local variant converges to the global kernel, which implies that it also takes global properties into account.

An Approximation Algorithm for the Local k-dimensional Weisfeiler-Lehman Kernel Since the k-dimensional Weisfeiler-Lehman algorithm has a worst

[1]TU Dortmund University, Dortmund, Germany. Correspondence to: Christopher Morris <christopher.morris@tu-dortmund.de>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML).

case running time of $\mathcal{O}(n^k \log n)$ for fixed k (Immerman & Lander, 1990), our local kernel does not scale to large graph databases. Hence, we propose algorithms to approximate it. For bounded-degree graphs we show that the running time of the approximation algorithm is *constant*, i.e., it does not depend on the number of vertices or edges of a graph. Moreover, for general graphs, we propose an *adaptive* sampling algorithm which uses results from statistical learning theory, in particular, *conditional Rademacher averages*.

Experimental Evaluation Finally, we implemented our algorithms and evaluated them on graph databases stemming from chemoinformatics and social networks.

1.2. Related Work

In recent years, various graph kernels have been proposed, e.g., see (Vishwanathan et al., 2010; Neumann et al., 2016). Gärtner et al. and Kashima et al. simultaneously proposed graph kernels based on random walks, which count the number of walks two graphs have in common. Since then, random walk kernels have been studied intensively (Kang et al., 2012; Kriege et al., 2014; Mahé et al., 2004; Sugiyama & Borgwardt, 2015; Vishwanathan et al., 2010). Kernels based on tree patterns were initially proposed by Ramon & Gärtner and later refined by Mahé & Vert. Kernels based on shortest paths were first proposed by Borgwardt & Kriegel and are computed by performing one-step walks on transformed input graphs, where edges are annotated with shortest-path distances. A drawback of the approaches mentioned above is their *high computational cost*. They all employ the kernel trick, leading to a quadratic overhead in the size of the data set, and the running time for a single kernel function evaluation can be only bounded by a polynomial of relatively high degree, e.g., the random-walk and the shortest-path graph kernel have a running time in $\mathcal{O}(n^{2\omega})$ and $\mathcal{O}(n^4)$, respectively, where n denotes the maximum number of vertices of two graphs, and ω denotes the exponent for the running time of matrix multiplication. Moreover, recently graph kernels using matchings (Kriege et al., 2016) and geometric embeddings (Johansson & Dubhashi, 2015) have been proposed.

A different line in the development of graph kernels focused particularly on scalable graph kernels. These kernels are typically computed efficiently by explicit feature maps, which allow to bypass the computation of a gram matrix, and allow applying scalable linear classification algorithms (Chiang et al., 2016; Fan et al., 2008; Joachims, 2006). Prominent examples are kernels based on subgraphs up to a fixed size, so called *graphlets* (Shervashidze et al., 2009; Wale et al., 2008), or specific subgraphs like cycles and trees (Horváth et al., 2004). Other approaches of this category encode the neighborhood of every node by different techniques, e.g., see (Bai et al., 2015; Hido & Kashima, 2009; Neumann et al., 2016), and most notably the Weisfeiler-Lehman subtree kernel (Shervashidze et al., 2011). Subgraph and Weisfeiler-Lehman kernels have been successfully employed within frameworks for smoothed and deep graph kernels (Yanardag & Vishwanathan, 2015a;b).

In the past few works considered graph kernels that use global graph properties. In (Johansson et al., 2014) a kernel based on the Lovàsz number is proposed. Moreover, Kondor & Pan proposed a graph kernel based on the graph Laplacian.

Moreover, few works considered sampling as way to speed up graph kernel computation. In (Shervashidze et al., 2009) a sampling algorithm for the graphlet kernel is introduced, which relies on approximating the relative counts of graphlets of a certain size. The authors provide a bound on the number of samples needed to approximate this quantity for a specific graphlet. However, they do not show an approximation result for the kernel. One drawback here is that the algorithm has to solve the graph isomorphism problem as a subproblem. More refined approaches can be found, e.g., in (Ahmed et al., 2016; Chen et al., 2016).

In (Johansson et al., 2014) sampling techniques for approximating the kernel based on the Lovàsz number were used. However, the running time of their algorithm is at least quadratic in the number of vertices for a single evaluation of the kernel function.

1.3. Outline

In section 2, we fix some notation and describe the 1-dimensional Weisfeiler-Lehman algorithm and the corresponding kernel. Moreover, we describe its k-dimensional sibling. In the subsequent chapter we propose our local kernel based on the k-dimensional Weisfeiler-Lehman algorithm and show that our kernel converges to the global kernel. In the next section, we present our approximation algorithm for bounded-degree graphs and the approximation algorithm for general graphs based on conditional Rademacher averages. In Section 4, we report on the results of our experimental evaluation.

2. Preliminaries

An *(undirected) graph* G is a pair (V, E) with a *finite* set of *nodes* V and a set of *edges* $E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$. We denote the set of nodes and the set of edges of G by $V(G)$ and $E(G)$, respectively. For ease of notation we denote the edge $\{u, v\}$ in $E(G)$ by (u, v) or (v, u). A *labeled graph* is a graph G endowed with a *label function* $l\colon V(G) \to \Sigma$, where Σ is some finite alphabet. We say that $l(v)$ is a *label* of v for v in $V(G)$.

Moreover, $N(v)$ denotes the *neighborhood* of v in $V(G)$, i.e., $N(v) = \{u \in V(G) \mid (v, u) \in E(G)\}$. A graph is of

d-*bounded degree* if its maximum degree is at most d, where d is always independent of the number of vertices, i.e., d is in $\mathcal{O}(1)$. We say that two graphs G and H are *isomorphic* if there exists an edge preserving bijection $\varphi : V(G) \to V(H)$, i.e., (u, v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$. Let $S \subseteq V(G)$ then $G[S] = (S, E_S)$ is the *subgraph induced* by S with $E_S = \{(u, v) \in E(G) \mid u, v \in S\}$.

Let χ be a non-empty set and let $k\colon \chi \times \chi \to \mathbb{R}$ be a function. Then k is a *kernel* on χ if there is a real Hilbert space $\mathcal{H}_k$ and a mapping $\phi\colon \chi \to \mathcal{H}_k$ such that $k(x, y) = \langle \phi(x), \phi(y) \rangle$ for x and y in χ, where $\langle \cdot, \cdot \rangle$ denotes the inner product of $\mathcal{H}_k$. We call ϕ a *feature map*, and $\mathcal{H}_k$ a *feature space*. Moreover, $\mathbf{K}$ in $\mathbb{R}^{n \times n}$ denotes the *gram matrix* for a kernel k for some finite subset S of χ of cardinality n, i.e., $\mathbf{K}_{ij} = k(x_i, x_j)$ for x_i and x_j in S. Let $\mathbb{G}$ be a non-empty set of graphs, then a kernel $k\colon \mathbb{G} \times \mathbb{G} \to \mathbb{R}$ is called *graph kernel*. Moreover, let $[1\,{:}\,n] = \{1, \ldots, n\} \subset \mathbb{N}$ for $n > 1$.

2.1. Weisfeiler-Lehman Subtree Kernel

We now describe the 1-dimensional Weisfeiler-Lehman algorithm (1-WL) and the corresponding kernel. Let G and H be graphs, and let l be a label function $V(G) \cup V(H) \to \Sigma$, e.g., $l(v) = |N(v)|$ for v in $V(G) \cup V(H)$. In each iteration $i \geq 0$, the 1-WL algorithm computes a new label function $l^i\colon V(G) \cup V(H) \to \Sigma$. In iteration 0 we set $l^0 = l$. Now in iteration $i > 0$, we set

$$l^i(v) = \mathsf{relabel}((l^{i-1}(v), \mathsf{sort}(\{\!\{l^{i-1}(u) \mid u \in N(v)\}\!\}))),$$

for v in $V(G) \cup V(H)$, where $\mathsf{sort}(S)$ returns a (ascendantly) sorted tuple of the multiset S and the bijection $\mathsf{relabel}(p)$ maps the pair p to an unique value in Σ, which has not been used in previous iterations. Now if G and H have an unequal number of vertices labeled σ in Σ, we conclude that the graphs are not isomorphic. Moreover, if $l^{i-1} = l^i$ the algorithm terminates. After at most $|V(G)| + |V(H)|$ iterations the algorithm terminates. It is easy to see that the algorithm is not able to distinguish all non-isomorphic graphs, e.g., see (Arvind et al., 2015). On the other hand, this simple algorithm is already quite powerful, since it can distinguish almost all graphs with high probability (Babai & Kucera, 1979; Babai et al., 1980). The running time of the algorithm is in $\mathcal{O}(m \log n)$ where $n = \max\{|V(G)|, |V(H)|\}$ and $m = \max\{|E(G)|, |E(H)|\}$, e.g., see (Berkholz et al., 2013).

The idea of the Weisfeiler-Lehman subtree graph kernel is to compute the above algorithm for $h \geq 0$ iterations and after each iteration i compute a feature map $\phi^i(G)$ in $\mathbb{R}^{|\Sigma_i|}$ for each graph G, where $\Sigma_i \subseteq \Sigma$ denotes the codomain of l^i. Each component $\phi^i(G)_{\sigma_j}$ counts the number of occurrences of vertices labeled with σ_j in Σ_i. The overall feature map $\phi(G)$ is defined as the concatenation of the feature maps of all h iterations, i.e.,

$$\Big(\phi^0(G)_{\sigma_1^1}, \ldots, \phi^0(G)_{\sigma_{|\Sigma_0|}^1}, \ldots, \phi^h(G)_{\sigma_1^h}, \ldots \phi^h(G)_{\sigma_{|\Sigma_h|}^h}\Big).$$

Then the Weisfeiler-Lehman subtree kernel for h iterations is $k_{\mathrm{WL}}^h(G, H) = \langle \phi(G), \phi(H) \rangle$. The running time for a single feature map computation is in $\mathcal{O}(hm)$ and $\mathcal{O}(Nhm + N^2hn)$ for a set of N graphs (Shervashidze et al., 2011), where n and m denote the maximum number of vertices and edges over all N graphs, respectively.

2.2. The k-dimensional Weisfeiler-Lehman Algorithm

Based on the 1-WL, the k-dimensional Weisfeiler-Lehman algorithm (k-WL) can be described as follows for $k \geq 2$. We follow the description of (Cai et al., 1992). Let G and H be graphs. Instead of iteratively labeling vertices, the k-WL computes a labeling function defined on the set of k-tuples $V(G)^k \cup V(H)^k$. In order to describe the algorithm we define the neighborhood of a tuple t in $V(G)^k$ (analogously for $V(H)^k$). For i in $[1, k]$ the j-th neighbor of t is the set

$$N_j(t) = \{(t_1, \ldots, t_{j-1}, r, t_{j+1}, \ldots, t_k) \,|\, r \in V(G)\}. \quad (1)$$

That is, the j-th neighbor is obtained by replacing the j-th component of t by a vertex from $V(G)$. In iteration 0, the algorithm labels each k-tuple with its isomorphism type, i.e., two tuples s and t in V^k get the same label if the corresponding induced subgraphs are isomorphic. Now in iteration $i > 0$ we set

$$l^i(t) = \mathsf{relabel}((l^{i-1}(t), \mathsf{sort}(\{\!\{l^{i-1}(s) \mid s \in N(t)\}\!\}))),$$

where $\mathsf{sort}(S)$ returns a (ascendantly) sorted tuple of the multiset S of labels, and $N(t) = \bigcup_{i=1}^k N_i(t)$. The algorithm then proceeds analogously to the 1-WL.

3. A Local Kernel Based on the k-dimensional Weisfeiler-Lehman Algorithm

The k-WL, see the Introduction and Section 2.2, is inherently *global*, i.e., it labels a k-tuple by considering all other k-tuples. In order to capture the local properties of a graph, we propose a *local* variant. The ideal of our *local* k-WL (k-LWL) is the following: The algorithm again labels all k-tuples. But in order to extract local features we define the *local* j*-th neighbors* of a k-tuple t in $V(G)^k$,

$$N_j^{\mathrm{L}}(t) = \{(t_1, \ldots, t_{j-1}, r, t_{j+1}, \ldots, t_k) \mid (t_l, r) \in E(G) \text{ for } l \in [1, k] \setminus \{j\} \text{ and } r \neq t_j\},$$

i.e., we consider a tuple s a local j-th neighbor of a tuple t if s is in $N_j(t)$, and there is at least one edge between vertices of s and t, not considering t_j. Now the local algorithm works the same way as the k-WL but in each iteration $i \geq 0$

considers the local neighbors of a tuple and computes a labeling function $l^i_{\mathrm{L}}(t)\colon V(G)^k \to \Sigma$.

The following theorem shows that the above algorithm is able to capture the same global properties of a graph as the ordinary k-WL, i.e., after a finite number of iterations two tuples have the same label with regard to the k-LWL if and only if they have the same label with regard to the k-WL.

Theorem 3.1. *Let G be a connected graph, and let s and t be k-tuples from $V(G)^k$. Then for every iteration $h^* \geq 0$ of the k-WL there exists $h \geq h^*$ such that*

$$l^h_L(s) = l^h_L(t) \iff l^{h^*}(s) = l^{h^*}(t)\,.$$

In order to prove the above theorem, we need the following definitions.

Definition 3.2. Let G be a connected graph, and let s and t be k-tuples from $V(G)^k$, then the *k-tuple graph* $T(G) = (V_T, E_T)$, where $V_T = \{v_r \mid r \in V(G)^k\}$, and

$$(v_s, t_t) \in E_T \iff s \in N^{\mathrm{L}}_j(t) \text{ for } j \in [1,k]\,.$$

Definition 3.3. Let G be a connected graph, and let t be a k-tuple from $V(G)^k$, then the *c-neighborhood* for $c \geq 0$,

$$\mathbf{N}(t,c) = \{s \in V(G)^k \mid d_T(v_t, v_s) \leq c\},$$

where $d_T\colon T(G) \times T(G) \to \mathbb{N}$ denotes the shortest-path distance in the k-tuple graph $T(G)$ of G.

Moreover, we need the following result which states that two tuples s and t get a different label if and only if there exists a different number of tuples in the c-neighborhood of s and t labeled σ in Σ at some iteration of the k-LWL.

Lemma 3.4. *Let G be a connected graph, and let s and t be k-tuples from $V(G)^k$, then $l^h_L(s) \neq l^h_L(t)$ for some iteration $h \geq 0$ if and only if there exists some iteration $0 \leq i \leq h$, and $c \geq 0$ such that there exists σ in Σ_i such that*

$$\left|\left\{r \in \mathbf{N}(s,c) \mid l^i_L(r) = \sigma\right\}\right| \neq \left|\left\{r \in \mathbf{N}(t,c) \mid l^i_L(r) = \sigma\right\}\right|.$$

Proof. Assume that $l^h_{\mathrm{L}}(s) \neq l^h_{\mathrm{L}}(t)$. After some iteration $j \leq h$, the labels of s and t first differ. Hence, we set $i = j$ and $c = 0$.

Now assume that the above inequality holds. After iteration i, without loss of generality, there will be a tuple r^* in $\mathbf{N}(s,c)$ that is labeled $\sigma^* = l^i_{\mathrm{L}}(r^*)$. Moreover, since relabel is bijective there is no tuple in $\mathbf{N}(t,c)$ that is labeled σ^* after iteration i. Now by applying Corollary 3.5 iteratively, the result follows. □

Corollary 3.5. *Let G be a connected graph, and let s and t be k-tuples from $V(G)^k$, then $l^i_L(s) \neq l^i_L(t)$ for some iteration $i \geq 1$ if and only if there exists σ in Σ_{i-1} such that*

$$\left|\left\{r \in \mathbf{N}(s,1) \mid l^{i-1}_L(r) = \sigma\right\}\right| \neq \left|\left\{r \in \mathbf{N}(t,1) \mid l^{i-1}_L(r) = \sigma\right\}\right|.$$

We can now prove Theorem 3.1.

Proof of Theorem 3.1. Assume that $l^{h^*}(s) = l^{h^*}(t)$ holds. Since $N^{\mathrm{L}}_j(t) \subseteq N_j(t)$ for j in $[1,k]$ for t in $V(G)^k$ holds, it follows that $l^h_{\mathrm{L}}(s) = l^h_{\mathrm{L}}(t)$ by definition of the algorithm.

Now assume that $l^{h^*}(s) \neq l^{h^*}(t)$ and $l^h_{\mathrm{L}}(s) = l^h_{\mathrm{L}}(t)$ holds. Since $l^{h^*}(s) \neq l^{h^*}(t)$, observe that, without loss of generality, after some iteration $i < h$ there is a tuple s^* in the c-neighborhood of s for some $c > 0$ that is labeled σ in Σ but there is no such labeled tuple in the c-neighborhood of t. Moreover, observe that the same is true for the label of s^* after some iteration $j \geq i$ with regard to the k-LWL. To see this, assume that the label σ is caused by labels of tuples in $N_j(t) \setminus N^{\mathrm{L}}_j(t)$. Since G is connected, the labels will eventually be propagated to s^*. Hence, by applying Lemma 3.4, we arrive at a contradiction. □

3.1. A Kernel Based on the k-LWL

The idea for a kernel based on the k-LWL is the same as for the 1-WL. We compute the k-LWL for $h \geq 0$ iterations, and after each iteration i compute a feature map $\phi^i_{k\text{-LWL}}(G)$ in $\mathbb{R}^{|\Sigma_i|}$ for each graph G, where $\Sigma_i \subseteq \Sigma$ denotes the codomain of l^i_{L}. Each component $\phi^i_{k\text{-LWL}}(G)_{\sigma_j}$ counts the number of occurrences of tuples labeled with σ_j in Σ_i. The overall feature map $\phi_{k\text{-LWL}}(G)$ is defined as the concatenation of the feature maps of all h iterations, i.e.,

$$\left(\phi^0(G)_{\sigma^1_1}, \ldots, \phi^0(G)_{\sigma^1_{|\Sigma_0|}}, \ldots, \phi^h(G)_{\sigma^h_1}, \ldots \phi^h(G)_{\sigma^h_{|\Sigma_h|}}\right).$$

Let G and H be two graphs then the kernel $k^h_{k\text{-LWL}} = \langle \phi(G)_{k\text{-LWL}}, \phi(H)_{k\text{-LWL}} \rangle$. Moreover, we will need the *normalized feature vector*

$$\widehat{\phi}_{k\text{-LWL}}(G) = \phi_{k\text{-LWL}}(G) / \|\phi_{k\text{-LWL}}(G)\|_1\,. \tag{2}$$

Observe that the components of the normalized feature vector are in $[0,1]$. We denote the corresponding kernel by $\widehat{k}^h_{k\text{-LWL}}$.

3.2. A Sampling Algorithm for Bounded-degree Graphs

Since the worst-case running time of the computation of $\phi_{k\text{-LWL}}(G)$ is in $\mathcal{O}(h \cdot n^k)$ for fixed k, the corresponding kernel is not feasible for large graphs. Thereto, we describe approximation algorithms for the k-LWL kernel for a fixed number of iterations. First, we introduce an algorithm restricted to bounded-degree graphs, and show that it can be computed in constant time. Subsequently, in Section 3.3, we describe an algorithm for general graphs which employs an adaptive sampling strategy.

The idea of the first algorithm is the following: Let G be a graph and let $\widetilde{\phi}_{k\text{-LWL}}(G)$ be a zero vector with the same

Algorithm 1 Approximation Algorithm for the k-LWL for Bounded-degree Graphs

Require: A d-bounded degree graph G, number of iterations $h \geq 0$, $k \geq 2$, failure parameter δ in $(0, 1)$, and an additive error term ε in $(0, 1]$.

Ensure: A feature map $\widetilde{\phi}_{k\text{-LWL}}(G)$ according to Inequality (3).

1: Let $\widetilde{\phi}_{k\text{-LWL}}(G)$ be a feature vector
2: Draw a multiset S of tuples independently and uniformly from $V(G)^k$ with $|S|$ according to Inequality (4).
3: **parallel for** $s \in S$ **do**
4: Compute the h-neighborhood $\mathbf{N}(s, h)$ around s
5: Compute $\sigma = l^h_{k\text{-LWL}}(s)$ on $G[\mathfrak{N}(s, h)]$
6: $\widetilde{\phi}_{k\text{-LWL}}(G)_\sigma = \widetilde{\phi}(G)_\sigma + 1/|S|$
7: **end**
8: **return** $\widetilde{\phi}_{k\text{-LWL}}(G)$

number of components as $\widehat{\phi}_{k\text{-LWL}}(G)$. First, we sample a multiset S of tuples from $V(G)^k$. The exact cardinality of S will be determined later. Secondly, for each such tuple s in S, we compute the h-neighborhood $\mathbf{N}(s, h)$ and compute the k-LWL on the subgraph induced by all vertices in $\mathfrak{N}(s, h) = \{u, v, \ldots, w \mid (u, v, \ldots, w) \in \mathbf{N}(s, h)\}$ resulting in a label σ for the tuple s. The following result shows that the label of a tuple after h iterations can be computed *locally* by only considering its h-neighborhood.

Lemma 3.6. *Let G be a graph and let s be a tuple in $V(G)^k$. Moreover, let $l^h_{L,N}(s)$ be the label of s after the h-th iteration of the k-LWL on $G[\mathfrak{N}(s, h)]$, then $l^h_{L,N}(s) = l^h_L(s)$.*

Proof (Sketch). Induction on the number of iterations. □

Now the algorithm proceeds by adding $1/|S|$ to $\widetilde{\phi}_{k\text{-LWL}}(G)_\sigma$. An easy implication of the above lemma is that for tuples that are in $\mathbf{N}(t, i)$ for $i \leq h$ it holds that $l^j_{\text{L,N}}(t) = l^j_{\text{L}}(t)$ for $j \leq h$ such that $i + j \leq h$. See Algorithm 1 for pseudo code. Note that lines 4 to 6 can be computed in parallel for all samples. Moreover, note that the algorithm can be easily adapted so that it computes the feature vector of Equation (2). We get the following result for bounded-degree graphs.

Theorem 3.7. *Let G be a d-bounded degree graph, then Algorithm 1 approximates the normalized feature vector $\widehat{\phi}_{k\text{-LWL}}(G)$ of the k-LWL for h-iterations such that with probability $(1 - \delta)$ for δ in $(0, 1)$,*

$$\left\| \widehat{\phi}_{k\text{-LWL}}(G) - \widetilde{\phi}_{k\text{-LWL}}(G) \right\|_1 \leq \varepsilon \tag{3}$$

for any ε in $(0, 1]$. Morever, the running time of the algorithm is only dependent of d, k, and h, i.e., it does not dependent on $|V(G)|$.

Proof. First, observe that for bounded-degree graphs the maximum number of different labels induced by h-neighborhoods is only dependent on the number of iterations h, and the maximum degree d. Let $\Gamma(d, h)$ be an upper bound on this quantity.

Let $X_{i,\sigma}$ denote the random variable that is 1 if we sample a triple s such that $l^i_{\text{L,N}}(s) = \sigma$ in iteration i of Algorithm 1, otherwise 0. Now observe that $\mathbb{E}(X_{i,\sigma}) = \widehat{\phi}_{k\text{-LWL}}(G)_\sigma$. Moreover, let $\bar{X}_\sigma = 1/|S| \cdot \sum_{i=1}^{S} X_{i,\sigma} = \widetilde{\phi}_{k\text{-LWL}}(G)_\sigma$, then, by the linearity of expectation, $\mathbb{E}\big(\bar{X}_\sigma\big) = \widehat{\phi}_{k\text{-LWL}}(G)_\sigma$. Hence, by the Hoeffding bound (Hoeffding, 1963), we get

$$\mathbb{P}\Big(\Big|\bar{X}_\sigma - \widehat{\phi}_{k\text{-LWL}}(G)_\sigma\Big| \geq \varepsilon\Big) \leq 2e^{-2|S|\cdot\varepsilon^2}.$$

By setting the sample size

$$|S| \geq \left\lceil \frac{\log(2 \cdot \Gamma(d, h) \cdot 1/\delta)}{2\lambda^2} \right\rceil, \tag{4}$$

it follows that

$$\mathbb{P}\Big(\Big|\bar{X}_\sigma - \widehat{\phi}_{k\text{-LWL}}(G)_\sigma\Big| \geq \lambda\Big) \leq \frac{\delta}{\Gamma(d, h)}.$$

The result then follows by setting $\lambda = \varepsilon/\Gamma(d,h)$, and the Union bound. Finally the bound on the running time follows from the observation that Inequality (4), and the running time of lines 4 to 6 in Algorithm 1 are independent of the size of G, i.e., the number of vertices and edges. The correctness follows from Lemma 3.6. □

Now let $\widetilde{k}^h_{k\text{-LWL}}$ denote the corresponding kernel, i.e., $\widetilde{k}^h_{k\text{-LWL}} = \langle \widetilde{\phi}_{k\text{-LWL}}(G), \widetilde{\phi}_{k\text{-LWL}}(H) \rangle$ for two graphs G and H. The following proposition shows that the above kernel approximates the normalized k-LWL kernel arbitrarily close.

Proposition 3.8. *Let $\mathbb{G}$ be (non-empty, finite) set of d-bounded degree graphs and let $\widehat{k}^h_{k\text{-LWL}}$ be the normalized k-LWL kernel. Then with probability $(1 - \delta)$ for δ in $(0, 1)$,*

$$\sup_{G,H \in \mathbb{G}} \left| \widehat{k}^h_{k\text{-LWL}}(G, H) - \widetilde{k}^h_{k\text{-LWL}}(G, H) \right| \leq 3\lambda$$

for any λ in (0,1]. The running time for computing $\mathbf{K}^h_{k\text{-LWL}}$ for $\mathbb{G}$ does only depend on the cardinality of $\mathbb{G}$, the number of iterations, k, and the maximum degree d.

Proof. First observe that by setting the sample size $|S|$ to

$$|S| \geq \left\lceil \frac{\log(2 \cdot \Gamma(d, h) \cdot 1/\delta \cdot |\mathbb{G}|)}{2\varepsilon^2} \right\rceil,$$

we get that with probability $(1 - \delta)$ for all G in $\mathbb{G}$

$$\left| \widehat{\phi}^h_{k\text{-LWL}}(G)_i - \widetilde{\phi}^h_{k\text{-LWL}}(G)_i \right| \leq \varepsilon$$

for any $1 \leq i \leq \Gamma(d,h)$ holds. Now

$$\begin{aligned}
&\widetilde{k}^h_{k\text{-LWL}}(G,H) = \left\langle \widetilde{\phi}^h_{k\text{-LWL}}(G), \widetilde{\phi}_{k\text{-LWL}}(H) \right\rangle \\
&= \sum_{i=1}^{\Gamma(d,h)} \widetilde{\phi}^h_{k\text{-LWL}}(G)_i \cdot \widetilde{\phi}^h_{k\text{-LWL}}(H)_i \\
&\leq \sum_{i=1}^{\Gamma(d,h)} \left(\widehat{\phi}^h_{k\text{-LWL}}(G)_i + \varepsilon\right) \cdot \left(\widehat{\phi}^h_{k\text{-LWL}}(H)_i + \varepsilon\right) \\
&\leq \sum_{i=1}^{\Gamma(d,h)} \left(\widehat{\phi}(G)_i \cdot \widehat{\phi}(H)_i\right) + \varepsilon \cdot \sum_{i=1}^{\Gamma(d,h)} \left(\widehat{\phi}(G)_i + \widehat{\phi}(H)_i\right) \\
&+ \sum_{i=1}^{\Gamma(d,h)} \varepsilon^2 \leq \hat{k}^h_{k\text{-LWL}}(G,H) + 2\varepsilon + \Gamma(d,h) \cdot \varepsilon\,.
\end{aligned}$$

Where the last inequality follows from the fact that the components of $\widehat{\phi}(\cdot)$ are in $[0,1]$. The result then follows by setting $\varepsilon = \lambda / \Gamma(d,h)$. □

Note that the above technique also leads to an approximation result for the (normalized) Weisfeiler-Lehman subtree kernel.

Corollary 3.9. *Let $\mathbb{G}$ be (non-empty, finite) set of d-bounded degree graphs and let $\widehat{k}^h_{WL}$ be the normalized Weisfeiler-Lehman subtree kernel. Then with probability $(1-\delta)$ for δ in $(0,1)$,*

$$\sup_{G,H \in \mathbb{G}} \left| \widehat{k}^h_{WL}(G,H) - \widetilde{k}^h_{WL}(G,H) \right| \leq 3\lambda$$

for any λ in $(0,1]$. The running time for computing $\mathbf{K}^h_{WL}$ for $\mathbb{G}$ does only depend on the size of $\mathbb{G}$, the number of iterations, and the maximum degree d.

Observe that instead of considering bounded-degree graphs Proposition 3.8 and Corollary 3.9 also hold for graphs with a constant label alphabet, i.e., the cardinality of the label alphabet is independent of the size of the graphs.

3.3. Adaptive Sampling Algorithm for General Graphs

In order to calculate the sample size of algorithm Algorithm 1, we assumed to know the number of different labels in advance. Hence, in order to circumvent this problem, we propose a variant of Algorithm 1 that relies on an adaptive sampling scheme based on *conditional Rademacher averages*.

Let $\mathcal{D}$ be a finite set, and let $\mathcal{F} = \{f \mid \mathcal{D} \to [0,1]\}$ be a family of functions. Now let $S = \{s_1, \ldots, s_m\}$ be sample of points sampled independently and uniformly from $\mathcal{D}$. Next we define the *true average* and the *sample average*, respectively,

$$L_{\mathcal{D}}(f) = \mathbb{E}_{s \sim \mathcal{U}(\mathcal{D})}[f(s)], \quad \text{and} \quad L_S(f) = \frac{1}{m}\sum_{i=1}^{m} f(s_i)\,.$$

Given S we are interested in bounding

$$\sup_{f \in \mathcal{F}} |L_{\mathcal{D}}(f) - L_S(f)|, \tag{5}$$

i.e., determining the maximum deviation of the sample average to the true average using only the sample S. For i in $[1,m]$ let σ_i be a *Rademacher variable*, i.e., it is i.i.d. to $\mathbb{P}[\sigma_i = 1] = \mathbb{P}[\sigma_i = -1] = 1/2$. Now the conditional Rademacher average (Mohri et al., 2012) is defined as

$$\mathcal{R}_{\mathcal{F}}(S) = \frac{1}{m}\mathbb{E}_{\sigma \sim \{\pm 1\}^m}\left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i f(s_i)\right]. \tag{6}$$

We can now state a classical result from learning theory which employs Equation (6) to bound Equation (5) depending on the sample S.

Theorem 3.10 (Mohri et al. (2012))**.** *Let δ be in $(0,1)$, then with probability of at least $(1-\delta)$,*

$$\sup_{f \in \mathcal{F}} |L_{\mathcal{D}}(f) - L_S(f)| \leq 2 \cdot \mathcal{R}_{\mathcal{F}}(S) + 3\sqrt{\frac{\log(2/\delta)}{2m}}\,. \tag{7}$$

This bound can be further improved (Oneto et al., 2013). In order to bound the conditional Rademacher average we can use *Massart's Lemma*. Let $\mathbf{v}_{f,S} = (f(s_1), \ldots, f(s_m))$ for f in $\mathcal{F}$, and let the set $\mathcal{V}_S = \{v_S \mid f \in \mathcal{F}\}$. We get the following result.

Lemma 3.11 (Mohri et al. (2012))**.**

$$\mathcal{R}_{\mathcal{F}}(S) \leq \max_{f \in \mathcal{F}} \|v_{S,f}\| \frac{\sqrt{2\ln|\mathcal{V}_S|}}{m}\,.$$

Moreover, based on the above lemma Riondato & Upfal proposed a convex optimization problem to achieve tighter (empirical) bounds on the conditional Rademacher average.

Now in order to use the above theorem to approximate the feature vector of the k-LWL, we define the following family of functions $\mathcal{F}_{G,k\text{-LWL}} = \{\mathbf{1}_{\sigma,h} \mid t \in V(G)^k, \sigma \in \Sigma, \text{ and } h \geq 0\}$ for a graph G, which equals 1 for a triple t in $V(G)^k$ if it has label σ after h iterations of the k-LWL, and otherwise 0, i.e.,

$$\mathbf{1}_{\sigma,h}(v) = \begin{cases} 1 & \text{if } l^h_{\mathrm{L}}(v) = \sigma, \\ 0 & \text{otherwise}\,. \end{cases}$$

Observe that

$$L_{\mathcal{D}}(\mathbf{1}_{\sigma,h}) = \widehat{\phi}(G)^h_\sigma\,.$$

The idea of the sampling algorithm is the following: In each iteration $i \geq 0$ we determine a sample size $|S_i|$. Draw $|S_i|$ samples independently and uniformly at random from $V(G)^k$, and proceed as in Algorithm 1, i.e., we compute the h-neighborhood of each tuple t in S_i, and compute its label

$l^h_{\text{L,D}}(t)$ after h iterations on the induced subgraph of the set $\mathfrak{N}(t,h)$. The algorithms terminates if Inequality (7) holds, otherwise we proceed with iteration $i+1$ and a sample multiset S_{i+1} such that $|S_i| \leq |S_{i+1}|$. See Algorithm 2 for pseudo code. We get the following result.

Algorithm 2 Adaptive Approximation Algorithm for the k-LWL

Require: A graph G, number of iterations $h \geq 0$, $k \geq 2$, failure parameter δ in $(0,1)$, and an additive error term ε in $(0,1]$.

Ensure: A feature map $\widetilde{\phi}_{k\text{-LWL}}(G)$ according to Theorem 3.12.

Let $\widetilde{\phi}_{k\text{-LWL}}(G)$ be a feature vector
$i \leftarrow 0$, $s \leftarrow 0$

while Inequality (7) not satisfied **do**
  Compute S_i
  $s \leftarrow s + |S_i|$
  parallel for $t \in S_i$ **do**
    Compute the t-neighborhood $\mathbf{N}(t,h)$ around s
    Compute $l^h_{k\text{-LWL}}(t) = \sigma$ on $G[\mathfrak{N}(t,h)]$
    $\widetilde{\phi}_{k\text{-LWL}}(G)_\sigma = \widetilde{\phi}_{k\text{-LWL}}(G)_\sigma + 1$
  end
  $i \leftarrow i + 1$
end while

$\widetilde{\phi}_{k\text{-LWL}}(G) = {}^1\!/\!_s \cdot \widetilde{\phi}_{k\text{-LWL}}(G)$
return $\widetilde{\phi}_{k\text{-LWL}}(G)$

Theorem 3.12. *Let G be a graph, then Algorithm 2 approximates the normalized feature vector $\widehat{\phi}_{k\text{-LWL}}(G)$ of the k-LWL for h iterations such that with probability $(1-\delta)$ for δ in $(0,1)$,*

$$\sup_{\sigma \in \Sigma} \left| \widehat{\phi}_{k\text{-LWL}}(G)_\sigma - \widetilde{\phi}_{k\text{-LWL}}(G)_\sigma \right| \leq \varepsilon, \tag{8}$$

for any ε in $(0,1]$.

Proof. The finiteness follows from the fact that we increase the sample size in every iteration. Hence, eventually Inequality (7) will hold. The approximation guarantee then follows from Algorithm 2, Inequality (7), and the correctness of Algorithm 1. □

The above algorithm and the above result can be adapted straightforwardly to a set of graphs instead of a single graph.

4. Experiments

Our intention here is to investigate the benefits of the k-LWL and the adaptive sampling algorithm k-LWL-APPROX compared to the state-of-the-art. More precisely, we address the following questions:

Q1 How does the k-LWL compare to state-of-the-art graph kernels in terms of classification accuracy?
Q2 How does the k-LWL compare to the global k-WL in terms of classification accuracy and running time?
Q3 Does our adaptive sampling algorithm to approximate the k-LWL speed up the computation time of the kernel computation?
Q4 Does the adaptive sampling algorithm lead to worse classification accuracies compared to the exact algorithm?

4.1. Data Sets and Graph Kernels

We used the following data sets to evaluate the k-LWL kernels: ENZYMES (Borgwardt et al., 2005; Feragen et al., 2013), IMDB-BINARY (Yanardag & Vishwanathan, 2015b), MUTAG (Debnath et al., 1991; Kriege & Mutzel, 2012), NCI1 and NCI109 (Wale et al., 2008; Shervashidze et al., 2011), PROTEINS (Borgwardt et al., 2005; Feragen et al., 2013), PTC_FM (Kriege & Mutzel, 2012), and REDDIT-BINARY (Yanardag & Vishwanathan, 2015b).[1]

We implemented the 3-LWL and the adaptive sampling algorithm for the 3-LWL (3-LWL-APPROX). We compare our kernels to the Weisfeiler-Lehman subtree kernel (Shervashidze et al., 2011), the graphlet kernel (Shervashidze et al., 2009), and the shortest-path kernel (Borgwardt & Kriegel, 2005). Moreover, we implemented a kernel (k-GWL) based on the (global) k-WL that uses the definition of neighborhood of Equation (1). All kernels were implemented in C++11.[2]

4.2. Experimental Protocol

For each kernel, we computed the normalized gram matrix. We computed the classification accuracies using the C-SVM implementation of LIBSVM (Chang & Lin, 2011), using 10-fold cross validation. The C-parameter was selected from $\{10^{-3}, 10^{-2}, \ldots, 10^2, 10^3\}$ by 10-fold cross validation on the training folds. For the 3-LWL-APPROX we set the ε of Inequality (8) to 0.01 (0.05, 0.1) and the initial sample size was set to $10\,000$ (100, 100). We doubled the sampled size when Inequality (7) was not fulfilled.

We repeated each 10-fold cross validation ten times with different random folds, and report average accuracies and standard deviations. Since the adaptive approximation algorithm is a randomized algorithm, we computed each gram matrix ten times and report average classification accuracies and running times. We report running times for the 1-WL,

[1] Due to space constraints we refer to `http://graphkernels.cs.tu-dortmund.de` for further descriptions, references, and statistics.

[2] The source code can be obtained from `https://github.com/chrsmrrs/globalwl`.

Table 1. Classification accuracies in percent and standard deviations, OOT— Computation did not finish within one day, OOM— Out of memory, *— Initial sample size 10 000, $\varepsilon = 0.01$, †— Initial sample size 100, $\varepsilon = 0.05$, ‡— Initial sample size 100, $\varepsilon = 0.1$.

Graph Kernel	Data Set							
	ENZYMES	IMDB-BINARY	MUTAG	NCI1	NCI109	PTC_FM	PROTEINS	REDDIT-BINARY
1-WL	53.4 ±1.4	72.4 ±0.5	78.3 ±1.9	83.1 ±0.2	**85.2** ±0.2	**62.9** ±1.6	73.7 ±0.5	75.3 ±0.3
3-LWL	**60.5** ±1.2	**74.6** ±0.6	87.8 ±1.4	**84.7** ±0.2	83.6 ±0.3	61.0 ±1.4	OOT	OOT
3-LWL-APPROX*	54.9 ±1.4	72.9 ±0.8	87.5 ±1.4	79.3 ±0.4	78.2 ±0.3	61.8 ±1.6	**76.4** ±0.4	83.1 ±0.6
3-LWL-APPROX†	54.3 ±0.9	72.8 ±0.7	87.7 ±1.4	79.4 ±0.2	78.0 ±0.3	**62.9** ±0.9	75.2 ±0.7	82.8 ±0.6
3-LWL-APPROX‡	52.7 ±1.1	72.7 ±1.0	86.3 ±1.5	78.4 ±0.4	77.3 ±0.4	62.7 ±0.9	75.4 ±0.6	82.7 ±0.7
3-GWL	44.4 ±1.3	72.3 ±0.7	**89.1** ±1.2	77.0 ±0.3	76.0 ±0.2	61.8 ±1.4	OOT	OOM
GRAPHLET	41.0 ±1.2	59.4 ±0.4	87.7 ±1.4	72.1 ±0.3	72.3 ±0.2	58.3 ±1.6	72.9 ±0.3	60.1 ±0.2
SHORTEST-PATH	42.3 ±1.3	59.2 ±0.3	81.7 ±1.3	74.5 ±0.3	73.4 ±0.1	62.1 ±0.9	**76.4** ±0.4	**84.7** ±0.2

Table 2. Average running times in seconds (Number of iterations for 1-WL, 3-LWL, 3-LWL-APPROX, and 3-GWL: 5, OOT— Computation did not finish within one day, OOM— Out of memory, *— Initial sample size 10 000, $\varepsilon = 0.01$, †— Initial sample size 100, $\varepsilon = 0.05$, ‡— Initial sample size 100, $\varepsilon = 0.1$.

Graph Kernel	Data Set							
	ENZYMES	IMDB-BINARY	MUTAG	NCI1	NCI109	PTC_FM	PROTEINS	REDDIT-BINARY
1-WL	<1	<1	<1	1.6	1.6	<1	<1	3.1
3-LWL	364.9	582.0	9.1	1 303.8	1 292.4	13.0	OOT	OOT
3-LWL-APPROX*	3 428.9	19 425.1	469.3	10 761.5	11 132.3	738.1	6 136.9	36 978.5
3-LWL-APPROX†	216.3	1 183.8	14.7	688.6	712.6	40.7	383.4	2 440.9
3-LWL-APPROX‡	54.0	280.8	7.1	186.2	185.9	9.9	96.0	553.4
3-GWL	2 716.1	845.6	17.1	9 887.6	9 575.7	56.2	OOT	OOM
GRAPHLET	<1	<1	<1	<1	<1	<1	<1	3.3
SHORTEST-PATH	<1	<1	<1	2.3	2.1	<1	<1	1 115.6

3-GWL, the 3-LWL, and the 3-LWL-APPROX with 5 refinement steps. For the graphlet kernel we counted (labeled) connected subgraphs of size three. The number of iterations for the 1-WL, 3-GWL, 3-LWL, and 3-LWL-APPROX were selected from $\{0, \ldots, 5\}$ using 10-fold cross validation on the training folds only.

Moreover, for the 3-{GWL, LWL, LWL-APPROX} we selected by 10-fold cross validation on the training folds only if the initial labels were set to the isomorphism type or the number of edges of the 3-tuples and the node labels. All experiments were conducted on a workstation with an Intel Core i7-3770 with 3.40GHz and 16GB of RAM running Ubuntu 14.04.5 LTS. Moreover, we used GNU C++ Compiler 4.8.4 with the flag –O2.

4.3. Results and Discussion

In the following we answer questions **Q1** to **Q4**. See also Tables 1 and 2.

A1 On five out of eight data sets the 3-LWL or the 3-LWL-APPROX performs better than the state-of-the-art, e.g., on the challenging ENZYMES data set the 3-LWL performs more than 7% better than the 1-WL. Moreover, on the other data sets the 3-LWL is at most 2% worse than the best performing kernel. Notice that the 3-LWL performs well over all data sets. This is not the case for the other kernels.

A2 On all data sets besides MUTAG the 3-LWL achieves better classification accuracies than the 3-GWL. On the ENZYMES data set the 3-LWL is more than 15% better. On all data sets the 3-LWL is faster, e.g., on the ENZYMES data set, the 3-LWL is more than seven times faster. On the PROTEINS and the REDDIT-BINARY data set the 3-GWL did not finish within one day, or was out of memory, respectively. On these data sets the 3-LWL-APPROX was able to finish within one day.

A3 On all data sets the adaptive sampling algorithm is several times faster (excluding the small data sets MUTAG and PTC_FM) than the exact algorithm (with inital sample size 100 and $\varepsilon = 0.1$), e.g., on the ENZYMES and the NCI1 data set the 3-LWL-APPROX is more than 6 and 7 times faster, respectively, than the 3-LWL. Moreover, on the PROTEINS and the REDDIT-BINARY data sets the exact algorithm did not finish within one day.

A4 On all data sets excluding ENZYMES, NCI1, and NCI109, the 3-LWL-APPROX is at most 2% (with inital sample size 10 000 and $\varepsilon = 0.01$) worse than the

exact algorithm in terms of classification accuracies. On ENZYMES the 3-LWL-APPROX is about 6% worse than the exact algorithm. But it still achieves better classification accuracies than the other kernels.

5. Conclusion and Future Work

We proposed a graph kernel based on a local variant of the (global) k-dimensional Weisfeiler algorithm, which explores the space between local and global graph properties. We demonstrated that it can be computed approximately in constant time for bounded-degree graphs. For general graphs we proposed an adaptive sampling algorithm. Our experimental study showed that our kernel advances the state-of-the-art. In particular, the predictive accuracy of our kernel is favorable on all data sets which is not the case for the other kernels. We can draw the following conclusion:

> *The local k-*LWL *is able to extract local as well as global graph properties, and behaves favorable for all data sets. Moreover, it can be approximated efficiently. For bounded-degree graphs the approximation can be computed in constant time.*

Our work provides several interesting directions for future work. First, it should be combined with the recent progress in deep learning approaches for graph classification (Duvenaud et al., 2015; Niepert et al., 2016) as our sampling is a form of dropout. Moreover, the running time should be reduced further, e.g., by using parallelization or random walks (Kersting et al., 2014).

Acknowledgement

This work has been supported by the German Science Foundation (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", project A6 "Resource-efficient Graph Mining".

References

Ahmed, N. K., Willke, T., and Rossi, R. A. Estimation of local subgraph counts. In *IEEE International Conference on Big Data*, pp. 1–10, 2016.

Arvind, V., Köbler, J., Rattan, G., and Verbitsky, O. On the power of color refinement. In *20th International Symposium on Fundamentals of Computation Theory*, pp. 339–350, 2015.

Babai, L. and Kucera, L. Canonical labelling of graphs in linear average time. In *20th Annual Symposium on Foundations of Computer Science*, pp. 39–46, 1979.

Babai, L., Erdős, P., and Selkow, S. M. Random graph isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980.

Bai, L., Rossi, L., Zhang, Z., and Hancock, E. R. An aligned subtree kernel for weighted graphs. In *32nd International Conference on Machine Learning*, pp. 30–39, 2015.

Berkholz, C., Bonsma, P. S., and Grohe, M. Tight lower and upper bounds for the complexity of canonical colour refinement. In *21st Annual European Symposium on Algorithms*, pp. 145–156, 2013.

Borgwardt, K. M. and Kriegel, H.-P. Shortest-path kernels on graphs. In *5th IEEE International Conference on Data Mining*, pp. 74–81, 2005.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21 (Supplement 1):i47–i56, 2005.

Cai, J., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identifications. *Combinatorica*, 12(4):389–410, 1992.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, X., Li, Y., Wang, P., and Lui, J. C. S. A general framework for estimating graphlet statistics via random walk. *VLDB Endowment*, 10(3):253–264, 11 2016.

Chiang, W.-L., Lee, M.-C., and Lin, C.-J. Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1485–1494, 2016.

Costa, F. and De Grave, K. Fast neighborhood subgraph pairwise distance kernel. In *26th International Conference on Machine Learning*, pp. 255–262, 2010.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34 (2):786–797, 1991.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pp. 2224–2232, 2015.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Feragen, A., Kasenburg, N., Petersen, J., Bruijne, M. D., and M., Borgwardt K. Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systems*, pp. 216–224, 2013. Erratum available at `http://image.diku.dk/aasa/papers/graphkernels_nips_erratum.pdf`.

Gärtner, T., Flach, P., and Wrobel, S. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pp. 129–143. 2003.

Grohe, M., Kersting, K., Mladenov, M., and Selman, E. Dimension reduction via colour refinement. In *22th European Symposium on Algorithms*, pp. 505–516, 2014.

Hido, S. and Kashima, H. A linear-time graph kernel. In *Ninth IEEE International Conference on Data Mining*, pp. 179–188, 2009.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Horváth, T., Gärtner, T., and Wrobel, S. Cyclic pattern kernels for predictive graph mining. In *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 158–167, 2004.

Immerman, N. and Lander, E. *Describing Graphs: A First-Order Approach to Graph Canonization*, pp. 59–81. 1990.

Joachims, T. Training linear SVMs in linear time. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226, 2006.

Johansson, F. D. and Dubhashi, D. Learning with similarity functions on graphs using matchings of geometric embeddings. In *21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 467–476, 2015.

Johansson, F. D., Jethava, V., Dubhashi, D. P., and Bhattacharyya, C. Global graph kernels using geometric embeddings. In *31st International Conference on Machine Learning*, pp. 694–702, 2014.

Kang, U., Tong, H., and Sun, J. Fast random walk graph kernel. In *SIAM International Conference on Data Mining*, pp. 828–838, 2012.

Kashima, H., Tsuda, K., and Inokuchi, A. Marginalized kernels between labeled graphs. In *20th International Conference on Machine Learning*, pp. 321–328, 2003.

Kersting, K., Mladenov, M., Garnett, R., and Grohe, M. Power iterated color refinement. In *28th AAAI Conference on Artificial Intelligence*, pp. 1904–1910, 2014.

Kondor, R. and Pan, H. The multiscale laplacian graph kernel. In *Advances in Neural Information Processing Systems*, pp. 2982–2990, 2016.

Kriege, N. and Mutzel, P. Subgraph matching kernels for attributed graphs. In *20th International Conference on Machine Learning*, 2012.

Kriege, N., Neumann, M., Kersting, K., and Mutzel, M. Explicit versus implicit graph feature maps: A computational phase transition for walk kernels. In *14th IEEE International Conference on Data Mining*, pp. 881–886, 2014.

Kriege, N. M., P.-L., Giscard., and Wilson, R. C. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pp. 1615–1623, 2016.

Li, W., Saidi, H., Sanchez, H., Schäf, M., and Schweitzer, P. Detecting similar programs via the weisfeiler-leman graph kernel. In *15th International Conference on Software Reuse: Bridging with Social-Awarenes*, pp. 315–330, 2016.

Mahé, P. and Vert, J.-P. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, 2009.

Mahé, P., Ueda, N., Akutsu, T., Perret, J.-L., and Vert, J.-P. Extensions of marginalized graph kernels. In *21st International Conference on Machine Learning*, pp. 552–559, 2004.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. MIT Press, 2012.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: Efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *33rd International Conference on Machine Learning*, pp. 2014–2023, 2016.

Oneto, L., Ghio, A., Anguita, D., and Ridella, S. An improved analysis of the rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44: 107–111, 2013.

Orsini, F., Frasconi, P., and De Raedt, L. Graph invariant kernels. In *24th International Joint Conference on Artificial Intelligence*, pp. 3756–3762, 2015.

Ramon, J. and Gärtner, T. Expressivity versus efficiency of graph kernels. In *1st International Workshop on Mining Graphs, Trees and Sequences*, 2003.

Riondato, M. and Upfal, E. ABRA: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1145–1154, 2016.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T. H., Mehlhorn, K., and Borgwardt, K. M. Efficient graphlet kernels for large graph comparison. In *12th International Conference on Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

Sugiyama, M. and Borgwardt, K. M. Halting in random walk kernels. In *Advances in Neural Information Processing Systems*, pp. 1639–1647, 2015.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

Wale, N., Watson, I. A., and Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14 (3):347–375, 2008.

Yanardag, P. and Vishwanathan, S. V. N. A structural smoothing framework for robust graph comparison. In *Advances in Neural Information Processing Systems*, pp. 2125–2133, 2015a.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015b.