Graph Invariant Kernels

Francesco Orsini[1,2] and **Paolo Frasconi**[2] and **Luc De Raedt**[1]
[1]Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Heverlee, Belgium
{francesco.orsini,luc.deraedt}@cs.kuleuven.be
[2]Department of Information Engineering
Università degli Studi di Firenze
Via di Santa Marta 3
I-50139 Firenze, Italy
paolo.frasconi@unifi.it

Abstract

We introduce a novel kernel that upgrades the Weisfeiler-Lehman and other graph kernels to effectively exploit high-dimensional and continuous vertex attributes. Graphs are first decomposed into subgraphs. Vertices of the subgraphs are then compared by a kernel that combines the similarity of their labels and the similarity of their structural role, using a suitable vertex invariant. By changing this invariant we obtain a family of graph kernels which includes generalizations of Weisfeiler-Lehman, NSPDK, and propagation kernels. We demonstrate empirically that these kernels obtain state-of-the-art results on relational data sets.

1 Introduction

Renewed interest has been manifested in recent years on graph kernels which can handle continuous (possibly high dimensional) attributes. A vast body of literature on Graph kernels is devoted to symbolic only structures. Which means that vertices (and possibly edges) are labeled by a number of discrete attributes. A popular approach to define graph kernels in this case is to count the number of common substructures (which we call *patterns* in this paper). Examples include tree-structured patterns [Ramon and Gärtner, 2003; Mahé and Vert, 2009], paths [Kashima *et al.*, 2003] or shortest paths [Borgwardt and Kriegel, 2005], cyclic and tree patterns [Horváth *et al.*, 2004], neighborhoods [Costa and De Grave, 2010], or arbitrary frequent patterns [Deshpande *et al.*, 2005].

Graphs with continuous attributes have been much less investigated, but one graph kernel that handles continuous attributes, is based on random walks (RW) labels [Kashima *et al.*, 2003]. The RW kernel can be computed in $O(V^2)$. Borgwardt and Kriegel (2005) proposed a variant based on shortest-paths (SP), which avoids the tottering problem of RW kernels. The SP kernel has a running time of $O(V^4)$. This has been recently improved by Feragen *et al.* (2013) who introduced the GRAPHHOPPER kernel, also based on shortest-paths, reporting a running time of $O(V^2(E + \log V + \Delta^2))$ (where Δ is the graph diameter). Kriege and Mutzel (2012) propose graph kernels in which subgraphs are matched with a score which computed as the product between a weight function and the kernels on vertex and edge attributes.

We upgraded existing graph kernels to continuous attributes by using graph and vertex invariants. Vertex invariants are functions that color vertices of a graph in a way that is not affected by isomorphism. They form the basis for several practical isomorphism checking algorithms [McKay and Piperno, 2014]. We consider the commonalities between graph kernels like the Weisfeiler-Lehman graph kernel (WLGK) [Shervashidze *et al.*, 2011], the neighborhood subgraph pairwise distance kernel (NSPDK) [Costa and De Grave, 2010], the propagation kernels [Neumann *et al.*, 2012] and GRAPHHOPPER [Feragen *et al.*, 2013] and summarize them in a general formulation which we call Graph Invariant Kernels (GIK, pronounce "Geek"). GIKs decompose graphs into sets of vertices which are compared by a kernel that measures both their attribute and their structural similarity. The structural similarity indicates to which extent vertices play the same role in the graph they belong to. Our formulation allows arbitrary patterns (e.g. other than the shortest paths used by GRAPHHOPPER) and arbitrary graph and vertex invariants that can be obtained with color propagation schemas (e.g. Weisfeiler-Lehman, Propagation kernel). We also propose spectral coloring which exploits eigen-decompositions.

We show that the upgrade to continuous values provided by GIKs perform very well on a number of new and existing benchmarks. We experiment with different types of vertex invariants, including Weisfeiler-Lehman and spectral colors and compare the shortest paths used by GRAPHHOPPER with neighborhood subgraphs.

2 Notation and definitions

We consider undirected graphs $G = (V, E)$ where V is the vertex set and E the edge set. Both vertices and edges may be labeled. $\ell : V \mapsto \mathcal{X}$ is the vertex labeling function where $\mathcal{X}$ may incorporate both discrete and continuous attributes. When necessary, we distinguish continuous and discrete labels as $\ell_c(v)$ and $\ell_d(v)$, respectively. Similarly, we denote by $l : E \mapsto \mathcal{Z}$ the edge labeling function. When needed we represent the connectivity of the graph G with the adjacency matrix A. Two graphs G and G' are *isomorphic*, written $G \approx G'$, if there exist a bijection $f : V \mapsto V'$ (called an *isomorphism*) such that $\{u, v\} \in E$ iff $\{f(u), f(v)\} \in E'$. In the case of labeled graphs, we also require $\ell(v) = \ell(f(v))$ and $l(u, v) = l(f(u), f(v))$. An *invariant* $\mathcal{I}$ is a function on graphs such that $G \approx G'$ implies $\mathcal{I}(G) = \mathcal{I}(G')$. If the reverse implication is also true, then $\mathcal{I}$ is called a *complete* invariant. A *vertex invariant* is a function $\mathcal{L} : V \mapsto \mathcal{C}$ that assigns each vertex v of G a value $\mathcal{L}(v)$, called the *color* of v, that is preserved under isomorphism f, i.e. $\mathcal{L}(v) = \mathcal{L}(f(v))$. The symbol $\preceq$ denotes the lexicographical order on discrete colors.

3 Graph Invariant Kernels

GIKs measure the similarity between two attributed graphs G and G' by comparing their vertices with a suitable kernel function $k_{\text{ATTR}}(v, v')$ between the continuous attributes ℓ_c and reweighing their similarity with a function $w(v, v')$:

$$k(G, G') = \sum_{v \in V(G)} \sum_{v' \in V(G')} w(v, v') k_{\text{ATTR}}(v, v'). \tag{1}$$

We define $w(v, v')$ as a count on common graph invariants:

$$w(v, v') = \sum_{\substack{g \in \mathcal{R}^{-1}(G) \\ g' \in \mathcal{R}^{-1}(G')}} k_{\text{INV}}(v, v') \frac{\delta_m(g, g')}{|V_g||V_{g'}|} \mathbb{1}\{v \in V_g \wedge v' \in V_{g'}\}. \tag{2}$$

The weight $w(v, v')$ measures the structural similarity between vertices and can be designed combining an $\mathcal{R}$-decomposition relation [Haussler, 1999], a function $\delta_m(g, g')$ and a kernel on vertex invariants k_{INV}.

An $\mathcal{R}$-decomposition relation is a binary relation $\mathcal{R}(G, g)$ which encodes that "g is part of G" and specifies a decomposition of G into its *parts* (*patterns*). We denote with $\mathcal{R}^{-1}(G)$ the multiset of all patterns in G.

According to Equation 2 the weight between a pair of vertices increases whenever the two vertices appear in the same pattern with the same structural role.

The function $\delta_m(g, g')$ is used to determine whether the two patterns match, while the indicator function $\mathbb{1}\{v \in V_g \wedge v' \in V_{g'}\}$ is introduced to select only the subgraphs g and g' in which the vertices v and v' are involved respectively.

The kernel function k_{INV} is used to measure the similarity between the colors produced by a vertex invariant $\mathcal{L}$ and encodes the extent to which the vertices play the same structural role in the pattern. A complete vertex invariant gives the most fine-grained matches and has the same effect as using an isomorphism map f, while weaker invariants induce spurious matches. Weaker invariants can be desirable because they allow to compare non isomorphic graphs. Vertex invariants can be weakened by exploiting the $\mathcal{R}$-decomposition relation and computing the vertex invariants with respect to the patterns. We specialize the notation for $k_{\text{INV}}(v, v')$ into *local* $k^g_{\text{INV}}(v, v')$ and *global* $k^G_{\text{INV}}(v, v')$ as to distinguish whether the vertex invariants in question were computed with respect to the patterns $g \in \mathcal{R}^{-1}(G)$ or the whole graph G respectively. We now explain how a suitable $\delta_m(g, g')$ function can define a hard-match between subgraphs generated by some $\mathcal{R}$-decomposition relation, while we refer the reader to Section 4 for the vertex invariants that can be used to define structural similarity $k_{\text{INV}}(v, v')$ between vertices.

3.1 Hard-match kernels for discrete labels

If graphs are labeled by discrete symbols, a simple choice for $\delta_m(g, g')$ is the hard match function

$$\delta(g, g') \doteq \begin{cases} 1 & \text{if } g \equiv g' \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

for a suitable definition of the equivalence relation $\equiv$. By instantiating the definition of $\mathcal{R}$ and $\equiv$, we can obtain a fairly large family of hard pattern-match graph kernels. For example, in the case of SP kernels [Borgwardt and Kriegel, 2005] $\mathcal{R}(G, g)$ iff g is a shortest-path between some pair of vertices u and v in G. If edges are unlabeled, then a special case of the kernel presented in [Borgwardt and Kriegel, 2005] can be interpreted as a hard pattern-match kernel by defining $g \equiv g'$ iff the two serializations (obtained by concatenating the vertex labels) of g and g' are identical. A second example is the kernel described in [Horváth *et al.*, 2004], where $\mathcal{R}(G, g)$ iff g is either a cycle in G, or a tree in the forest obtained by deleting all cycles from G. For this kernel, the equivalence $g \equiv g'$ is established by taking all possible serializations (for example all cyclic permutations in the case of cycles) and checking that the lexicographical minima are identical. As a last example, in the NSPDK [Costa and De Grave, 2010] the decomposition relation is parameterized by two natural numbers r and d and $\mathcal{R}_{r,d}(G, g)$ iff g is a pair of neighborhood subgraphs of G, g_1 and g_2, rooted at vertices v_1 and v_2, respectively, such that the two following conditions hold: (1) the shortest-path distance between v_1 and v_2 is d, and (2) for $i = 1, 2$, g_i is the neighborhood subgraph rooted in v_i. A neighborhood subgraph rooted in v_i consists of the subgraph induced by all vertices of G whose shortest-path distance from v_i is at most r. In [Costa and De Grave, 2010], $\equiv$ is defined as the graph isomorphism relation, approximated by hashing a canonical representation of the patterns.

A weaker hard-match kernel can be obtained using a graph invariant $\mathcal{I}$, i.e. $g \equiv g'$ if $\mathcal{I}(g) = \mathcal{I}(g')$. A simple graph invariant $\mathcal{I}_{\text{V}}$ is the number of vertices of a graph. Vertex colors can be used to construct stronger graph invariants because they are preserved under graph isomorphism. A graph invariant can be obtained as the lexicographically sorted set of vertex colors or edge colors, where the color of an edge $\{u, v\} \in E$ can be represented as $(\mathcal{L}(u), \mathcal{L}(v), l(u, v))$. The latter method was introduced by [Costa and De Grave, 2010] combined with distance coloring as in Section 4.3. Graph invariants are hashed to integers so that we can compute the subgraph match function $\delta_m(g, g')$ in constant time. Collisions due to hashing can only weaken the graph invariant.

On the other hand graph invariants used by δ_m are always computed on patterns (i.e. at local level). Computing δ_m at the global level would not allow to capture meaningful correlations between a pair of graphs G and G' leading to poor generalization performance.

GIKs can also be expressed as $\mathcal{R}$-convolutional kernels on graphs [Haussler, 1999] so they are positive semidefinite (PSD).

4 Vertex invariants

In this section we review some possible vertex invariants to construct specific instances of the vertex coloring kernel.

4.1 Weisfeiler-Lehman coloring

The 1-dimensional Weisfeiler-Lehman (WL) refinement algorithm [Weisfeiler, 1976] finds a partition of the vertices of a graph in an iterative fashion:

- Initialization: All colors are initialized using vertex labels, i.e. $\forall v \in V\ \mathcal{L}^{(0)}(v) \doteq \ell_d(v)$
- Recoloring step: $\forall v \in V$

$$\mathcal{L}^{(t+1)}(v) = id(\{\mathcal{L}^{(t)}(w) | w \in \mathcal{N}_G(v)\}) \tag{4}$$

  where $\mathcal{N}(v)$ is the set of vertices adjacent to v and the function id returns the string of the vertex colors of the neighbors sorted in lexicographic order. Often a hash function is used to represent this string with an integer.
- Termination criterion: recoloring converges when the number of distinct colors stops increasing, which means that the vertices in the graph cannot be further partitioned. In practice the recoloring process is stopped after a predefined number h of iterations.

  The 1-dimensional WL color of a vertex is finally obtained as the concatenation of the colors at all time steps:

$$\mathcal{L}(v) = \left[\mathcal{L}^{(0)}(v) | \dots | \mathcal{L}^{(h)}(v)\right] \tag{5}$$

For edge-labeled graphs, we propagate the vertex colors of the neighbors together with the edge labels:

$$\mathcal{L}^{(t+1)}(v) = id(\{(\mathcal{L}^{(t)}(w), l(v,w)) | w \in \mathcal{N}_G(v)\}) \tag{6}$$

The positive definite kernel on this kind of vertex invariant can be defined as:

$$\langle \mathcal{L}(v), \mathcal{L}(v') \rangle = \sum_{i=0}^{h} \mathbb{1}\left\{\mathcal{L}^{(i)}(v) = \mathcal{L}^{(i)}(v')\right\} \tag{7}$$

4.2 Label updates in propagation kernels

When introducing propagation kernels (PROP), Neumann *et al.* (2012) proposed two WL-like label update approaches:

- Diffusion updates:

$$\mathcal{L}^{(t+1)}(v_i) = \sum_{v_j \in \mathcal{N}_G(v_i)} T_{ij} \mathcal{L}^{(t)}(v_j) \tag{8}$$

  where $T = D^{-1}A$ is the transition matrix derived from the normalization of the rows of the adjacency matrix, and D is the diagonal degree matrix $D_{ii} = \sum_{k=0}^{V} a_{ik}$
- Label propagation: as above except that before each iteration $t+1$ the labels $\ell_d(v_i)$ of the originally labeled vertices overwrite the value assigned to $\mathcal{L}^{(t)}(v_i)$ by the propagation process.

4.3 Distance coloring

This method was initially suggested in [Costa and De Grave, 2010] to compute the NSPDK and assumes connected and rooted patterns, i.e. g has a distinguished vertex r.

- Starting from r, a breadth-first visit is used to compute all-pairs distances $d(v,w)$ between the vertices.
- Each vertex is colored as follows:

$$\mathcal{L}(v) = id(\{(\ell_x(w), d(v,w)) | w \in V(G)\})\ \forall v \in V(G) \tag{9}$$

Information about the root of the subgraph can be included by adding to each vertex color information about its distance from the root vertex.

4.4 Spectral coloring

Eigenvalues and eigenvectors of a graph adjacency matrix are insensitive to vertex permutation. For this reason, starting from the seminal work of Umeyama (1988), spectral methods have been popular in pattern recognition as graph matching tool. Spectra can be also used for isomorphism testing if eigenvalues have bounded multiplicity [Babai *et al.*, 1982].

A weighted adjacency matrix W can be defined by using vertex labels $\ell_c(v)$, assuming that $\mathcal{X}$ is a metric space endowed with a distance function d:

$$w_{ij} = a_{ij} e^{-\gamma d^2(\ell_c(v_i), \ell_c(v_j))} \tag{10}$$

where a_{ij} are the entries in the adjacency matrix A of the graph and γ is a non-negative scalar hyperparameter of the heat kernel. In spectral coloring, a vertex invariant is obtained from the the Laplacian embedding of the graph as follows. Let $L \doteq D - W$ denote the graph Laplacian matrix, with D the diagonal degree matrix. The embedding $X \in \mathbb{R}^{V \times V}$ is the solution of

$$\min_{X : X^t X = I} \sum_{i=1}^{V} \sum_{j=1}^{V} \|\mathbf{x}(v_i) - \mathbf{x}(v_j)\|^2 w_{ij}. \tag{11}$$

This equation finds vertex coordinates $\mathbf{x}(v)$ (rows of X) whose euclidean distance preserves the graph connectivity and is equivalent to:

$$\min_{X : X^t X = I} tr(X^t L X) \tag{12}$$

the solution can be obtained by solving an eigenvalue problem:

$$L\mathbf{x}_i = \lambda_i \mathbf{x}_i \tag{13}$$

where the eigenvectors $\mathbf{x}_i$ (columns of X) are sorted according to the corresponding eigenvalues, i.e. $0 = \lambda_1 \leq \lambda_2 \cdots \leq \lambda_V$. The row $\mathbf{x}(v)$ of X is the embedding coordinate of the vertex v in the graph. The Spectral vertex invariant is obtained by switching to zero the λ-eigenvectors whose eigenvalue λ has multiplicity $\mu > 1$ and taking the absolute value of the components of the eigenvalues otherwise.

$$\mathcal{L}^{(i)}(v) = \begin{cases} |x_i(v)| & \text{if } \mu_i = 1 \\ 0 & \text{if } \mu_i > 1 \end{cases} \tag{14}$$

Some lifted inference approaches [Mladenov *et al.*, 2012] perform color passing on factor graphs to cluster symmetric variables and speed up intractable inference problems. We solve a complementary problem, and propose the canonized Laplacian embedding to reveal the symmetries between the vertices of a graph. The graph Laplacian is preferred over the one of the adjacency matrix because its spectrum gives more insight on the connectivity of the graph. The smallest eigenvalue λ_1 of the Laplacian spectrum of a connected graph is 0, has multiplicity 1 and its λ_1-eigenvector is constant. All the other eigenvectors are orthogonal to the constant eigenvector and thus balanced. It is this appealing property that allows spectral coloring features to encode the symmetries of a graph. From an implementation point of view it is possible to limit the dimensionality of the embedding to the first h components. In case there are less than h vertices the embedding can be padded with zeros.

5 Algorithmic issues and running time

We limit our analysis to kernels that use an $\mathcal{R}$-decomposition relation that generates connected subgraphs whose enumeration scales linearly with the number of vertices in the graph G. The time complexity of our method is $V^2(C_1 + C_2\tau V_g^2)$ where V is the number of vertices, V_g is the number of vertices in a subgraph, C_1 and C_2 are two costs and τ is the subgraph matching count:

$$\tau = \sum_{g \in \mathcal{R}^{-1}(G)} \sum_{g' \in \mathcal{R}^{-1}(G')} \delta_m(g, g'). \tag{15}$$

We assume that the cost of computing k_{INV} or k_{ATTR} scales with the dimension d_{INV} and d_{ATTR} of the corresponding features. According to Equation 2 we have $C_1 = d_{\text{ATTR}}$ and $C_2 = d^g_{\text{INV}}$. If we use global instead of local graph invariants we can rewrite Equation 2 as:

$$w(v, v') = k^G_{\text{INV}}(v, v') \sum_{\substack{g \in \mathcal{R}^{-1}(G) \\ g' \in \mathcal{R}^{-1}(G')}} \frac{\delta_m(g, g')}{|V_g||V_{g'}|} \mathbb{1}\{v \in V_g \wedge v' \in V_{g'}\} \tag{16}$$

and thus we obtain $C_1 = d_{\text{ATTR}} + d^G_{\text{INV}}$ and $C_2 = 1$.

The worst case scenario for τ occurs when the subgraph matching function $\delta_m(g, g')$ is always 1, in this case we have $\tau = V^2$. This can happen in two cases: 1) when the graph invariant $\mathcal{I}$ is constant and every pair of patterns matches and 2) when the patterns are all identical. In the former case we should resort to a more selective invariant, while in the latter case we should avoid computing $w(v, v')$ because its value is constant. This case in which all the patterns are identical is verified for 0-neighborhood subgraphs (vertices) or r-neighborhood subgraphs in which $r \geq \frac{\Delta}{2}$, where Δ is the diameter of the G. When the radius r is at least twice the diameter Δ of the graph G, all the corresponding r-neighborhood subgraphs are the graph G itself. In both cases there is no structural contribution, these cases are easy to detect and being aware of this fact we can compute the kernel in $O(V^2 C_1)$. Another option is to add structural information in order to avoid the extreme cases. We can incorporate discrete labels if present when $r = 0$ and root information when $r \geq \frac{\Delta}{2}$. Two rooted patterns must have both same graph invariant and same root vertex invariant in order to match . We obtain an upper bound on the number of vertices V_g of a r-neighborhood subgraph with maximum vertex degree d and diameter $\Delta = 2r$ using the Moore bound [Miller and Širáň, 2005]:

$$V_g \leq 1 + d \sum_{i=0}^{\Delta - 1} (d-1)^i = d^\Delta + O(d^{\Delta - 1}) \tag{17}$$

So $V_g \in O(d^{2r})$ and the complexity of our method is $O(V^2(C_1 + C_2\tau d^{4r}))$.

6 Experimental evaluation

We answer the following experimental questions:

Q1 How do GIKs compare to the state of the art?

Q2 Do our kernels produce more accurate classifiers when continuous attributes are introduced?

Q3 Can the graph representation of a sentence benefit from word vector attributes instead of discrete word attributes?

Q4 What are the best graph invariants?

6.1 Datasets

PROTEINS and **ENZYMES$_{\text{SYMM}}$** are sets of proteins from Dobson and Doig (2003) and from the BRENDA database [Schomburg *et al.*, 2004], respectively. Vertices are secondary structure elements as in [Borgwardt *et al.*, 2005].

SYNTHETIC$_{\text{NEW}}$ is a dataset of 300 examples with two classes, it is based on a random graph G with 100 nodes and 196 edges, whose vertices were annotated with scalar attributes sampled from $\mathcal{N}(0, 1)$. It was introduced in [Feragen *et al.*, 2013].

FRANKENSTEIN is a dataset created by the fusion of the BURSI and MNIST datasets. BURSI is made by 4337 molecules with mutagenicity (AMES) classification, with 2401 mutagens and 1936 nonmutagens [Kazius *et al.*, 2005]. Each molecule is represented as a graph whose vertices are labeled by the chemical atom symbol and edges by the bond type.

FRANKENSTEIN is a modified version of the BURSI dataset: we discarded bond type information and remapped the most frequent atom symbols (vertex labels) to MNIST digit images. The original atom symbols can only be recovered through the high dimensional MNIST vectors of pixel intensities, in this sense this is a challenging problem for a graph kernel that can handle continuous attributes.

QC and **WEASEL** are natural language processing datasets: QC is a dataset for question classification with fixed split (5452 train / 500 test) originally proposed in [Li and Roth, 2002]. We consider the classification task with six coarse labels. WEASEL is part of the CoNNL-2010 shared task dedicated to the detection of hedge cues, it is a binary classification task with fixed split (11111 train / 9634 test).

For both QC and WEASEL we represent a sentence as a graph whose vertices are tokens annotated with 300-dimensional word-vector attributes, and whose edges represent the word adjacency and the typed dependency relations extracted with the Stanford dependency parser [De Marneffe *et al.*, 2006]. Word vectors were obtained using the WORD2VEC software [Mikolov *et al.*, 2013] and the whole Wikipedia corpus for training.

	ENZYMES$_{\text{SYM}}$		PROTEIN		SYNTHETIC$_{\text{NEW}}$		FRANKENSTEIN		QC	
	WITHOUT CONT.	WITH CONT.	WITHOUT CONT.	WITH CONT.	WITHOUT CONT.	WITH CONT.	WITHOUT CONT.	WITH CONT.	WITHOUT CONT.	WITH CONT.
NSK$_{\text{V}}$	25.9 ± 1.1	71.8 ± 1.0	72.1 ± 0.4	74.2 ± 0.7	78.4 ± 1.9	81.9 ± 1.1	67.9 ± 0.2	72.9 ± 0.3	37.8	92.6
NSK$_{\text{WL}}$	56.5 ± 1.1	72.2 ± 0.8	71.7 ± 0.4	76.2 ± 0.4	50.0 ± 0.0	50.0 ± 0.0	74.2 ± 0.3	77.3 ± 0.1	47.8	91.0
GWL$_{\text{V}}$	55.7 ± 1.0	72.6 ± 0.8	**74.9 ± 0.6**	76.1 ± 0.8	**80.8 ± 1.2**	82.8 ± 1.0	73.5 ± 0.3	77.3 ± 0.2	49.8	93.6
GWL$_{\text{WL}}$	**58.6 ± 1.4**	71.3 ± 1.1	73.6 ± 0.5	75.8 ± 0.6	50.0 ± 0.0	50.0 ± 0.0	**75.1 ± 0.2**	**78.9 ± 0.3**	48.6	89.6
LWL$_{\text{V}}$	54.5 ± 1.1	**73.3 ± 0.9**	74.4 ± 0.4	**76.6 ± 0.6**	80.6 ± 1.5	**83.0 ± 1.0**	73.0 ± 0.2	77.6 ± 0.2	47.2	**94.6**
LWL$_{\text{WL}}$	57.0 ± 1.1	72.0 ± 0.9	71.9 ± 0.6	76.5 ± 0.5	50.0 ± 0.0	50.0 ± 0.0	74.1 ± 0.2	78.3 ± 0.3	47.4	91.8
GSGK$_{\text{V}}$	29.8 ± 0.6	71.8 ± 1.0	73.2 ± 0.3	74.7 ± 0.5	78.2 ± 2.1	82.4 ± 0.9	70.1 ± 0.3	74.0 ± 0.3	44.4	92.6
GSGK$_{\text{WL}}$	56.7 ± 1.2	72.2 ± 0.7	72.9 ± 0.5	76.4 ± 0.4	50.0 ± 0.0	50.0 ± 0.0	75.0 ± 0.3	77.6 ± 0.2	47.8	91.0
LSGK$_{\text{V}}$	31.9 ± 1.0	71.9 ± 1.0	72.3 ± 0.4	74.4 ± 0.6	78.7 ± 2.0	82.2 ± 1.1	72.1 ± 0.2	74.9 ± 0.2	42.4	92.2
LSGK$_{\text{WL}}$	56.6 ± 1.3	72.1 ± 0.8	71.7 ± 0.3	76.1 ± 0.5	50.0 ± 0.0	50.0 ± 0.0	74.2 ± 0.2	77.4 ± 0.2	**51.4**	91.0
GRAPHHOPPER		69.5 ± 0.7		72.7 ± 0.3		73.9 ± 1.7		68.7 ± 0.4		91.4

Table 1: Comparison between different GIKs and GRAPHHOPPER

6.2 Kernels

We combined vertex invariants to construct some GIKs that can handle continuous attributes (see also Table 2).

NSK [Costa and De Grave, 2010] was instantiated using an $\mathcal{R}$-decomposition relation that generates neighborhood subgraphs (NS) of increasing radius $r = 0, \ldots, R$ and δ_m as an indicator function that matches two r-neighborhood subgraphs only if they have the same radius r and $\mathcal{I}_{\text{WL}}$ subgraph invariant. Costa and De Grave (2010) originally used distance coloring $\mathcal{I}_{\text{DC}}$.

LWL and **GWL** are the local and global variant of the WL subtree kernel respectively. Strictly speaking, GWL is not equivalent to the WL subtree kernel. In fact instead of using an $\mathcal{R}$-decomposition relation which generates vertex patterns we used r-neighborhood subgraphs.

LSGK and **GSGK** are local and global version of Spectral Graph Kernel in which we used our spectral coloring method.

kernel	$k_{\text{INV}}(v, v')$	
NSK	1	global
GWL	$\langle \mathcal{L}^{G}_{\text{WL}}(v), \mathcal{L}^{G}_{\text{WL}}(v') \rangle$	global
LWL	$\langle \mathcal{L}^{g}_{\text{WL}}(v), \mathcal{L}^{g}_{\text{WL}}(v') \rangle$	local
GSGK	$e^{-\gamma \Vert \mathcal{L}^{G}_{\text{SC}}(v) - \mathcal{L}^{G}_{\text{SC}}(v') \Vert^2}$	global
LSGK	$e^{-\gamma \Vert \mathcal{L}^{g}_{\text{SC}}(v) - \mathcal{L}^{g}_{\text{SC}}(v') \Vert^2}$	local

Table 2: Some instances of GIKs

Each GIK was instantiated in combination with two different kernels on subgraph invariants: $\mathcal{I}_{\text{V}}$ and $\mathcal{I}_{\text{WL}}$. $\mathcal{I}_{\text{WL}}$ is more selective and reduces the subgraph matching count τ. When necessary we use the subscript to specify which subgraph invariant was employed. We used an $\mathcal{R}$-decomposition relation which generates neighborhood subgraphs of increasing radius $r = 0, \ldots, R$. Based on preliminary experiments, we fixed $R = 3$. The Gram matrices were normalized as proposed in [Costa and De Grave, 2010]: for each radius a different Gram matrix is extracted then normalized, we compute their sum and apply normalization again. For ENZYMES$_{\text{SYMM}}$, PROTEINS, SYNTHETIC$_{\text{NEW}}$ we chose the parameter γ of the RBF kernel as in [Feragen *et al.*, 2013], to be $1/d_{\text{ATTR}}$ which is the inverse of the number of dimensions of the attributes. We did the same for the γ_{INV} of the RBF kernel on the continuous vertex invariants derived from spectral graph coloring $\mathcal{L}_{\text{SC}}$. The number of dimensions d_{INV} of $\mathcal{L}_{\text{SC}}$ was set equal to the average number of vertices in a graph for GSGK and to the average number of vertices in a 3-neighborhood subgraph for LSGK. For FRANKENSTEIN we set $\gamma = 0.0073$, this value was taken from [Sevakula and Verma, 2012], while for QC we set $\gamma = 1$. This value was found during preliminary experiments in which we used the n-grams of word vectors ($n = 1, 2, 3$) matched with the RBF kernel.

KERNEL		ACCURACY $T = 1$	$T = 2$	$T = 3$
WL		89.2	89.0	88.0
PROP	(w = 0.01)	86.8	88.0	85.4
PROP	(w = 0.1)	77.4	78.2	79.8
PROP	(w = 1.0)	58.0	58.0	57.2
PROP$_{\text{TV}}$	(w = 0.01)	87.8	89.0	88.6
PROP$_{\text{TV}}$	(w = 0.1)	87.8	88.6	88.4
PROP$_{\text{TV}}$	(w = 1.0)	79.0	81.8	81.0

Table 3: QC dataset using words as features

6.3 Experiments

E1 In Table 1 we show the classification accuracy that we achieved without ($k_{\text{ATTR}} = 1$) and with ($k_{\text{ATTR}} = \text{RBF}(\gamma)$) kernel on continuous attributes. We use the bold font for the best results of each column. For FRANKENSTEIN we also measured the area under the roc curve (AUROC) which is 0.74 for GRAPHHOPPER with continuous attributes. With GWL$_{\text{WL}}$ we obtained 0.82 AUROC without continuous attributes and 0.86 with continuous attributes. For all datasets we used SVM-classifiers. Except for QC and WEASEL the accuracy was estimated by 10-times 10-fold cross-validation reporting means and standard deviations [Feragen *et al.*, 2013]. QC and WEASEL have predefined train-test splits. The SVM regularization parameter was selected with an internal k-fold cross-validation on the training data ($k = 3$ except $k = 10$ for QC and WEASEL).

E2 In Table 3 we report the accuracies obtained on the original implementations of WL and PROP using words instead of word vectors. The symbol T denotes the number of iterations and TV and w are parameters (see [Neumann *et al.*, 2012]).

These results can be compared with Table 1.

E3 In Table 4 we show the results on WEASEL for LWL_{V} and GRAPHHOPPER. We used word vectors and no task specific knowledge. We compare to the state of the art which exploits task specific knowledge encoded in word lists. We set the radius of the r-neighborhood subgraphs to $r = 0, 1$ as done by Verbeke *et al.* (2012).

METHOD	F1-SCORE
LWL_{V} (r = 0, 1)	55.7%
LWL_{V} (r = 0)	41.9%
LWL_{V} (r = 1)	58.3%
GRAPHHOPPER	48.8%
[Verbeke *et al.*, 2012]	61.5%

Table 4: WEASEL sentence classification

The experiments were run on a 16 cores machine (Intel Xeon CPU E5-2665@2.40GHz and 96GB of RAM). The code of GIKs was written in the Python programming language, while for GRAPHHOPPER we used the MATLAB implementation from Feragen *et al.* The difference between the programming languages makes hard to perform fine-grained time measurements. We measured the wall-clock execution time.

DATASET	MOST ACCURATE AND FAST GIK		GRAPH-HOPPER
$\text{ENZYMES}_{\text{SYMM}}$	NSK_{WL}	2' 36"	4' 53"
PROTEIN	NSK_{WL}	11' 00"	35' 27"
$\text{SYNTHETIC}_{\text{NEW}}$	GWL_{V}	15' 22"	9' 48"
FRANKENSTEIN	GWL_{WL}	1h 20'	2h 5'
QC	LWL_{V}	2h 30'	1h 30'

Table 5: runtime of GIKs vs GRAPHHOPPER

In Table 5 we show the runtime of our most accurate and fast GIK selected from Table 1 compared to the runtime of GRAPHHOPPER. The experiment that had the highest runtime was on the dataset WEASEL: LWL_{V} terminated in 73h 47' while GRAPHHOPPER terminated in 75h 43'. We also mention that the spectral kernels tend to be slower and in some cases (e.g. on FRANKENSTEIN) can have a slowdown of a factor of 2 with respect to GRAPHHOPPER.

6.4 Discussion

A1 Our experiments show that we obtained state-of-the-art results for all the datasets in Table 1. $\text{ENZYMES}_{\text{SYMM}}$, PROTEINS, QC and WEASEL are are real-world datasets. On $\text{SYNTHETIC}_{\text{NEW}}$ we verified that its random nature combined with the graph invariant $\mathcal{I}_{\text{WL}}$ leads to diagonal kernel matrices and this explains the poor classification performance. On the other hand if we use the subgraph invariant $\mathcal{I}_{\text{V}}$, we outperform GRAPHHOPPER on its own artificial benchmark. In our experiments on FRANKENSTEIN (see Table 1), in some cases (GWL_{WL}) we have an increase of 0.12 points of AUROC with respect to GRAPHHOPPER. This performance mismatch is also due to the nature of the dataset from which FRANKENSTEIN was originated. Indeed neighborhood subgraphs are known to perform well on BURSI [Costa and De Grave, 2010], and shortest paths are probably not the best pattern for this kind of dataset. We consider as upper bound the result obtained with NSPDK ($R = 3$, $D = 0$) on BURSI, which is 0.91 AUROC score. The best result obtained on FRANKENSTEIN with continuous attributes (0.86 AUROC score) is significantly higher than the best result obtained without (0.82 AUROC score). On QC we obtain the 94.6% (see Table 1) which is the state of the art. Our result can be compared to the 94.8% obtained by Croce *et al.* (2011), when they combine kernels on lexical centered trees LCT with a word vector representation obtained with latent semantic analysis. Nevertheless GIKs are a generic tool, not specialized for natural language as LCT and we did not rely on manually built word lists. The 58.3% f1-score (see Table 4) obtained on WEASEL with $r = 1$ is comparable with the top five results of the CoNNL-2010 shared task. The state of the art for the task is 61.5% and was obtained by Verbeke *et al.* (2012) also exploiting task specific word lists not used by GIKS. The runtime measurements in Table 5 show that we could always instantiate GIKs obtaining comparable or higher accuracy with the same or inferior runtime except for $\text{SYNTHETIC}_{\text{NEW}}$ and QC in which the higher runtime is compensated by higher accuracies.

A2 Comparing the results in Table 1 we see that it is always the case that continuous attributes increase the accuracy.

A3 On QC WL and PROP use words as discrete attributes and cannot obtain more than 89% of accuracy (see Table 3). GIKs successfully use word vectors, indeed LWL_{V} achieves state-of-the-art results (see Table 1).

A4 If we consider the results in Table 1 we notice that LWL_{V} generally tends to give the best results, indeed local versions favor spurious matches and also the subgraph invariant $\mathcal{I}_{\text{V}}$ is weaker than $\mathcal{I}_{\text{WL}}$. Weaker invariants lower down the dimensionality of the kernel and thus make the learning system less prone to overfitting. This effect was probably beneficial due to the relatively small size of some of the datasets that we used. The spectral color invariant can obtain good results, but other GIKs yield equally or more accurate classifiers and have better runtime. What makes spectral color invariants interesting is the nice property of providing vertex invariants embedded in the euclidean space. This makes Spectral Graph Kernel amenable for future work on improving scalability using sketching techniques.

7 Conclusion

The GIK reformulation of well known graph kernels allows to obtain more insights in the exploration of graph kernels with continuous attributes. The underlying idea was to employ vertex invariants for soft subgraph matching. We contributed new insights into graph kernels and to upgrade existing ones for use with continuous attributes. Several graph-kernel instances were then empirically evaluated on a number of new and existing benchmark datasets. The results showed that some combinations of graph and vertex invariants with continuous attributes lead to excellent performance. For future work we plan to improve runtime and scalability focusing on the local GIKs and spectral coloring which are more appealing for sketching.

References

[Babai *et al.*, 1982] L. Babai, D. Yu. Grigoryev, and D. M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proc. of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 310–324, New York, NY, USA, 1982. ACM.

[Borgwardt and Kriegel, 2005] K. M. Borgwardt and H-P Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

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

[Costa and De Grave, 2010] F. Costa and K. De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proc. of the 26th ICML*, pages 255–262, 2010.

[Croce *et al.*, 2011] D. Croce, A. Moschitti, and R. Basili. Structured lexical similarity via convolution kernels on dependency trees. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics, 2011.

[De Marneffe *et al.*, 2006] M. C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*, volume 6, pages 449–454, 2006.

[Deshpande *et al.*, 2005] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. on Knowledge and Data Engineering*, 17(8):1036–1050, 2005.

[Dobson and Doig, 2003] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[Feragen *et al.*, 2013] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, and K. Borgwardt. Scalable kernels for graphs with continuous attributes (see also erratum). In *Advances in NIPS*, pages 216–224, 2013.

[Haussler, 1999] D. Haussler. Convolution kernels on discrete structures. Technical report, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.

[Horváth *et al.*, 2004] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proc. of the tenth ACM SIGKDD international conference on KDDM*, pages 158–167. ACM, 2004.

[Kashima *et al.*, 2003] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. ICML*, volume 3, pages 321–328, 2003.

[Kazius *et al.*, 2005] J. Kazius, R. McGuire, and R. Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005.

[Kriege and Mutzel, 2012] N. Kriege and P. Mutzel. Subgraph matching kernels for attributed graphs. In *Proc. of the 29th ICML*, 2012.

[Li and Roth, 2002] Xin Li and Dan Roth. Learning question classifiers. In *Proc. of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

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

[McKay and Piperno, 2014] B. D. McKay and A. Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

[Mikolov *et al.*, 2013] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pages 3111–3119, 2013.

[Miller and Širán, 2005] M. Miller and J. Širán. Moore graphs and beyond: A survey of the degree/diameter problem. *Electronic Journal of Combinatorics*, 61:1–63, 2005.

[Mladenov *et al.*, 2012] M. Mladenov, B. Ahmadi, and K. Kersting. Lifted linear programming. In *AISTATS*, pages 788–797, 2012.

[Neumann *et al.*, 2012] M. Neumann, N. Patricia, R. Garnett, and K. Kersting. Efficient graph kernels by randomization. In *MLKDD*, pages 378–393. Springer, 2012.

[Ramon and Gärtner, 2003] J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74. Citeseer, 2003.

[Schomburg *et al.*, 2004] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl 1):D431–D433, 2004.

[Sevakula and Verma, 2012] R. K. Sevakula and N. K. Verma. Support vector machine for large databases as classifier. In *Swarm, Evolutionary, and Memetic Computing*, pages 303–313. Springer, 2012.

[Shervashidze *et al.*, 2011] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 12:2539–2561, 2011.

[Umeyama, 1988] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *PAMI, IEEE Transactions on*, 10(5):695–703, 1988.

[Verbeke *et al.*, 2012] M. Verbeke, P. Frasconi, V. Van Asch, R. Morante, W. Daelemans, and L. De Raedt. Kernel-based logical and relational learning with klog for hedge cue detection. In *Inductive Logic Programming*, pages 347–357. Springer, 2012.

[Weisfeiler, 1976] B. Weisfeiler. On construction and identification of graphs. In *Lecture Notes in Mathematics*. Citeseer, 1976.