ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

Exploiting the ODD framework to define a novel effective graph kernel

Giovanni Da San Martino[2], Nicolò Navarin[1] and Alessandro Sperduti[1] *

1- University of Padova - Department of Mathematics
via Trieste 63, Padova - Italy

2- ALT Research Group
Qatar Computing Research Institute

Abstract. In this paper, we show how the Ordered Decomposition DAGs kernel framework, a framework that allows the definition of graph kernels from tree kernels, allows to easily define new state-of-the-art graph kernels. Here we consider a quite fast graph kernel based on the Subtree kernel (ST), and we improve it by increasing its expressivity by adding new features involving partial tree features. While the worst-case complexity of the new obtained graph kernel does not increase, its effectiveness is improved, as shown on several chemical datasets, reaching state-of-the-art performances.

1 Introduction

The study of machine learning for structured domains has received increasing attention in the last years due to the high availability of data in stuctured form [1]. Among the different types of structured data, the most studied are undoubtedly sequences, trees and graphs [2]. Among the different techniques for dealing with structured data, kernel methods have state of the art results on many benchmark problems. A kernel method represents the data via a similarity function $K()$ which corresponds to the dot product of the mapping $\phi()$ in a feature space of the structures x_i and x_j: $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. In particular, the work in [3] presents a framework for graph kernels that can be instantiated with several kernels for Directed Acyclic Graphs (DAGs). Such kernels can be conveniently defined extending kernels for trees, thus reconducting the problem of the definition of a graph kernel to the definition of a tree kernel. The original paper proposes one instantiation of the framework with an extension to DAGs of the efficient Subtree (ST) kernel for trees. In this paper, we propose a new instance of the Ordered Decompositional DAGs graph kernel framework that adopts a new kernel for trees, referred to as ST_+, which enlarges the feature space of the ST kernel. As a side contribution, we propose a new approach for feature weighting that considers the frequency of all the generated features, in opposition to the classical feature weighting approach that considers only the size of each feature to determine their weight. Finally we show experimental results of the three new instances of the framework, comparing them to the original one and other state-of-the-art kernels on real-world datasets.

*This work was supported by the University of Padova under the strategic project BIOINFOGEN.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

2 Ordered Decomposition DAG Kernels for Graphs

This section briefly describes the framework of ODD-Kernels for graphs, proposed in [3]. Let us first introduce some notation. A graph $G = (V_G, E_G, L_G)$ is a triplet where V_G is the set of vertices, E_G the set of edges and $L_G()$ a function mapping nodes to labels. A graph is undirected if $(v_i, v_j) \in E_G \Leftrightarrow (v_j, v_i) \in E_G$, otherwise it is directed. A path $p(v_i, v_j)$ of length n in a graph G is a sequence of nodes $v_1, \ldots, v_n$, where $v_1 = v_i$, $v_n = v_j$ and $(v_k, v_{k+1}) \in E_G$ for $1 \leq k < n$. A cycle is a path for which $v_1 = v_n$. A graph is acyclic if it has no cycles. A tree is a directed acyclic graph where each node has at most one incoming edge. The root of a tree T is represented by $r(T)$. The children of a node $v \in V_T$ are all the nodes v' s.t. $(v, v') \in E_T$. ρ is the maximum number of children in a tree. The symbol $\overset{v}{\triangle}$ is used to denote the proper subtree of T rooted at v, that is the tree that comprises v and all its descendants. Moreover, we denote as $\overset{v}{\triangle}|_h$ the subtree of T resulting from a breadth-first visit starting at v and limited to h levels of depth. The idea of the ODD kernel framework is to decompose the input graph into a set of substructures for which the isomorphism can be computed efficiently, i.e. subtrees.

In order to map the graphs into trees, two intermediate steps are needed:

1. map the graph G into a multiset of DAGs $\{DD_G^{v_i} | v_i \in V_G\}$, where $DD_G^{v_i}$ is obtained by keeping each edge in the shortest path(s) connecting v_i with any $v_j \in V_G$. The Decomposition DAGs kernel for graphs can be defined as $DDK_{K_{DAG}}(G_1, G_2) = \sum_{D_1 \in DD_{G_1}} \sum_{D_2 \in DD_{G_2}} K_{DAG}(D_1, D_2)$.

2. Since the vast majority of DAG kernels are extensions of kernels for ordered trees a strict partial order between DAG nodes in $DD_G^{v_i}$ has been defined yielding Ordered Decomposition DAGs $ODD_G^{v_i}$. The ordering function considers, in order: *i)* the vertex label $L(v)$, *ii)* $\rho(v)$ and *iii)* the recursive application of the ordering function to the children of v [3].

3. Finally, any Ordered DAG (ODD) is mapped into a multiset of trees. Let us define $T(v_i)$ as the tree resulting from the visit of $ODD_G^{v_i}$ starting from node v_i: the visit returns the nodes reachable from v_i in $ODD_G^{v_i}$. Note that if a node v_j can be reached more than once, more occurrences of v_j will appear in $T(v_i)$. Notice that the tree visit $T(v_i)$ can be stopped when the tree $T(v_i)$ reaches a maximum depth h. Such tree is referred to as $T_h(v_i)$. In [3] the Ordered Decomposition DAGs kernel is defined as:

$$ODDK(G_1, G_2) = \sum_{\substack{OD_1 \in ODD_{G_1} \\ OD_2 \in ODD_{G_2}}} \sum_{j=1}^{h} \sum_{\substack{v_1 \in V_{OD_1} \\ v_2 \in V_{OD_2}}} C(r(T_j(v_1)), r(T_j(v_2))) \quad (1)$$

where $C()$ is a function defining a tree kernel. Among the kernels for trees defined in literature, the one employed in the paper is the Subtree Kernel [4],

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

which counts the number of shared proper subtrees between the two input trees and have complexity of $O(|V_T| \log |V_T|)$. We recall that the tree visits can be limited to a depth h and that, consequently, the size of the tree visits is constant if we assume ρ constant. Therefore the resulting ODD kernel with ST has a complexity of $O(|V_G| \log |V_G|)$ [3]. Another kernel worth citing is the Partial Tree kernel [5], that considers all the possible subtrees of the given trees. Its complexity is $O(\rho^3 |V_T|^2)$ while the complexity of the associated graph kernel is $O(|V_T|^2 \log |V_T|)$, if we consider ρ constant.

3 A novel tree kernel

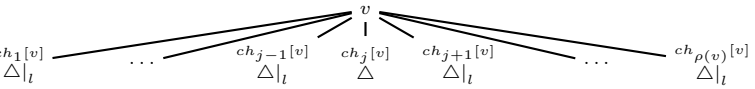
The kernel we introduce in this section enlarges the feature space of the ST kernel, with a modest increase in computational burden, and is referred to as ST_+. In Alg. 1 we define a procedure to compute the explicit feature space representation $\phi()$ of ST_+, thus making it trivial to show that the kernel is positive semidefinite. The set of features related to the ST_+ kernel is a superset of the features of ST and a subset of the features of PT. Line 6 of Alg. 1 depicts a generic feature introduced by ST_+. Given the node v and the index j, the feature is composed by v, the proper subtree rooted at the j-th child and the subtrees resulting from a limited visit of l levels for the other children. Notice that it depends on $v \in T$, the index of a child j and a limit l on the depth of the visits. The function $\pi(f)$ returns the index of the feature f in $\phi()$. Fig. 1 depicts a partial feature space representation of a tree according to ST_+. While for the ST kernel there is one feature for each $v \in T$, ST_+ associates at most $(\rho(v) \cdot h) + 1$ features for any $v \in T$. For each node $v \in T$, for example the node highlighted in Fig. 1-a, the algorithm inserts the following features: 1) the proper subtree rooted at v, which in our example is the one in Fig. 1-b; 2) given $ch_j[v]$, i.e. the j-th child of v, the subtree composed by: *i)* v; *ii)* the proper subtree rooted at the j-th child of v; *iii)* the subtrees resulting from a visit limited to $1 \leq l \leq h$ levels starting from the other children of v is added as feature. As l ranges from 0 to h, the features/subtrees from Fig. 1-c to Fig. 1-e are added. Assuming ρ constant, any tree visit with limited depth h yields a tree with a constant number of nodes H. Then the complexity of Alg. 1 is $O(Hh^2\rho^2 \log \rho)$. Alg. 1 can easily be extended for the application to ordered DAGs (substituting $\overset{v}{\triangle}$ with $T(v)$). The complexity of the ODDK kernel instantiated with ST_+ as base kernel is $|V_G| \log |V_G|$. The kernel can be written as:

$$ODDK_{ST_+}(G_1, G_2) = \sum_{\substack{OD_1 \in ODD_{G_1} \\ OD_2 \in ODD_{G_2}}} \sum_{j=1}^{h} \langle \phi_{ST_+}(OD_1, j), \phi_{ST_+}(OD_2, j) \rangle.$$

Let $|f|$ be the number of nodes in the subtree encoded by a feature f. It is possible to see from lines 3 and 8 of Alg.1 that the weight of each feature is $\lambda^{\frac{|f|}{2}}$, meaning that the weight of matching features (calculated via dot product) is $\lambda^{|f|}$. This is the usual downweight scheme used in tree kernels for avoiding

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Algorithm 1 Sketch of an algorithm to compute the features of the ST_+ kernel.

Input: a tree T, h
for each $v \in T$ **do**
  $\phi_{\pi(\overset{v}{\triangle})} = \phi_{\pi(\overset{v}{\triangle})} + \lambda^{\frac{|\overset{v}{\triangle}|}{2}}$ // add the proper subtree rooted at v as a feature.
  // If the feature is first encountered, it is assumed $\phi_{\pi(\overset{v}{\triangle})} = 0$
  for $0 \leq l < \min(h, depth(\overset{v}{\triangle}))$ **do**
    for $1 \leq j \leq \rho(v)$ **do**
      $t =$ 
      $\phi_{\pi(t)} = \phi_{\pi(t)} + \lambda^{\frac{|t|}{2}}$ // add the subtree t as a feature.
    end for
  end for
end for
Output: ϕ, the set of features of T

to overweight the contribution of a feature. Following the same principle we observe that, by construction, a node of the graph may appear in multiple tree visits. In order to control the contribution of the features related to such nodes, we propose to replace the weight $\lambda^{|f|}$ with $\tanh(|f| * freq(f) * \sigma)$, where $freq(f)$ is the frequency of the feature in the particular example and σ is a parameter determining the smoothness of the sigmoid function. The new weighting scheme is applied to the ST kernel, obtaining a variant of the kernel proposed in [3], and to the ST+ kernel first proposed in this paper.

4 Experimental results

The experimentation has been performed on five datasets: CAS[1], CPDB [6], AIDS [7], NCI1 [8] and GDD [9]. All datasets involve chemical compounds and represent binary classification problems. In all data sets nodes are labeled and there are no self-loops. We compare the predictive abilities of the $ODDK_{ST_+}$ kernel and the two proposed variants $ODDK^{\text{TANH}}_{\text{ST}_h}$ and $ODDK^{\text{TANH}}_{\text{ST}_+}$ to the original $ODDK_{ST_h}$ kernel [3], the Fast Subtree Kernel (FS) [10] and the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [11]. The experiments are performed using the *nested* k-fold cross validation: for each of the K folds another *inner* K-fold cross validation, in which we select the best parameters for that particular fold, is performed. The K parameter (number of folds) has been fixed to 10; all the experiments have been repeated 10 times using different splits for the cross validation, and the average results (with standard deviation) are reported. For all the experiments, the values of the parameters of the $ODDK_{ST_h}$ and $ODDK_{ST_+}$ kernels have been restricted to: $\lambda = \{0.1, 0.2, \ldots, 2.0\}$, $h = \{1, 2, \ldots, 10\}$. For the variants using *tanh*, the σ parameter is optimized (instead of λ) in the range $\{\frac{1}{0.00001}, 1, \frac{1}{3}, \frac{1}{5}, \frac{1}{10}, \frac{1}{15}\}$. For the Fast Subtree kernel only the parameter $h = \{1, 2, \ldots, 10\}$ is optimized. For the

[1] http://www.cheminformatics.org/datasets/bursi

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8.
Available from http://www.i6doc.com/en/.

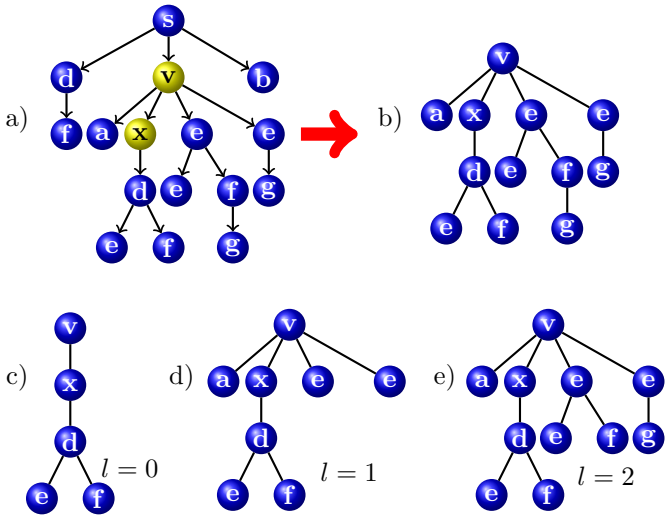


Fig. 1: Feature space representation related to the kernel ST_+ for an example tree: a) the input tree; b) the proper subtree rooted at the node labelled as **v**; c)-e) given the child **x** of **v**, the features related to visits limited to l levels.

NSPDK, the parameters $h = \{1, 2, \ldots, 8\}$ and $d = \{1, 2, \ldots, 7\}$ are optimized. Table 1 reports the average accuracy and the ranking obtained by the considered methods. The proposed kernels, $ODDK^{\text{TANH}}_{\text{ST}_h}$, $ODDK_{ST_+}$, $ODDK^{\text{TANH}}_{\text{ST}_+}$ together have best accuracy on three out of five datasets, and the second best accuracy on the other two. The variant employing the hyperbolic tangent is always useful for the ST kernel, making it the best performing kernel on GDD, and is able to boost the accuracy performance of $ODDK_{ST_+}$ on AIDS and CPDB. The generally good results of the ODDK kernels, with respect to FS and NSPDK, may be attributed to the fact that they have associated a large feature space, which makes them more adaptable to different tasks.

5 Conclusions

The paper presents a novel instance of the ODDK graph kernel based on a novel tree kernel, ST_+. Moreover we defined a novel feature weighting scheme for the ODDK kernels. The experimental results show that the proposed kernels have state of the art performances on five benchmark graph datasets. This constitutes an example of how the generality of the framework can potentially lead to the definition of novel graph kernels that can significantly improve the state-of-the-art. This work suggests that the way to follow to get better graph kernels is to move from hard to soft substructure matching.

ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22-24 April 2015, i6doc.com publ., ISBN 978-287587014-8. Available from http://www.i6doc.com/en/.

Kernel	CAS	CPDB	AIDS	NCI1	GDD
FS	83.32 (6) ±0.37	76.36 (4) ±1.48	82.02 (5) ±0.4	84.41 (3) ±0.49	75.46 (2) ±0.98
NSPDK	83.6 (2) ±0.34	**76.99** (1) ±1.15	**82.71** (1) ±0.66	83.45 (5) ±0.43	74.09 (6) ±0.91
$\text{ODDK}_{\text{ST}_h}$	83.34 (5) ±0.31	76.44 (3) ±0.62	81.51 (6) ±0.74	82.10 (6) ±0.42	75.27 (4) ±0.68
$\text{ODDK}^{\text{TANH}}_{\text{ST}_h}$	83.37 (4) ±0.5	76.45 (2) ±1.13	82.54 (2) ±0.57	84.35 (4) ±0.44	**75.62** (1) ±1.1
$\text{ODDK}_{\text{ST}_+}$	**83.90** (1) ±0.33	76.3 (6) ±0.23	82.06 (4) ±0.70	**84.97** (1) ±0.47	75.33 (3) ±0.81
$\text{ODDK}^{\text{TANH}}_{\text{ST}_+}$	83.39 (3) ±0.34	76.31 (5) ±1.72	82.54 (2) ±0.90	84.69 (2) ±0.34	74.6 (5) ±0.68

Table 1: Average accuracy results ± standard deviation in nested 10-fold cross validation for the Fast Subtree, the Neighborhood Subgraph Pairwise Distance, the $ODDK_{ST_h}$ and the $ODDK_{ST_+}$ kernels obtained on CAS, CPDB, AIDS, NCI1 and GDD datasets. The rank of the kernel is reported between brackets.

References

[1] Giovanni Da San Martino and Alessandro Sperduti. Mining Structured Data. *IEEE Computational Intelligence Magazine*, 5(1):42–49, February 2010.

[2] Thomas Gärtner. *Kernels for Structured Data*. World Scientific Publishing Co Pte Ltd, 2008.

[3] Giovanni Da San Martino, Nicolò Navarin, and Alessandro Sperduti. A Tree-Based Kernel for Graphs. In *Proceedings of the Twelfth SIAM International Conference on Data Mining*, pages 975–986, 2012.

[4] S V N Vishwanathan and Alexander J Smola. Fast Kernels for String and Tree Matching. In *NIPS*, pages 569–576, 2002.

[5] Alessandro Moschitti. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329, September 2006.

[6] Christoph Helma, Tobias Cramer, Stefan Kramer, and Luc De Raedt. Data mining and machine learning techniques for the identification of mutagenicity inducing substructures and structure activity relationships of noncongeneric compounds. *Journal of chemical information and computer sciences*, 44(4):1402–11, 2004.

[7] Owen S Weislow, Rebecca Kiser, Donald L Fine, John Bader, Robert H Shoemaker, and Michael R Boyd. New soluble-formazan assay for HIV-1 cytopathic effects: application to high-flux screening of synthetic and natural products for AIDS-antiviral activity. *Journal of the National Cancer Institute*, 81(8):577–586, 1989.

[8] Nikil Wale, Ian Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[9] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330:771–783, 2003.

[10] Nino Shervashidze and Karsten M. Borgwardt. Fast subtree kernels on graphs. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *NIPS*, pages 1660–1668. Curran Associates, Inc., 2009.

[11] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *ICML*, 2010.