Axiomatic Kernels on Graphs for Support Vector Machines

Conference Paper · September 2019

DOI: 10.1007/978-3-030-30493-5_62

2 authors:

Marcin Orchel
AGH University of Science and Technology in Kraków
31 PUBLICATIONS **51** CITATIONS
SEE PROFILE

Johan A.K. Suykens
www.esat.kuleuven.be/stadius
743 PUBLICATIONS **34,100** CITATIONS
SEE PROFILE

Axiomatic Kernels on Graphs for Support Vector Machines

Marcin Orchel[1,2] [0000−0002−1081−7626]* and Johan A.K. Suykens[1] [0000−0002−8846−6352]

[1] ESAT-STADIUS, KU Leuven, Leuven (Heverlee) 3001, Belgium
{marcin.orchel,johan.suykens}@esat.kuleuven.be
[2] Department of Computer Science, AGH University of Science and Technology, Kraków, Poland

Abstract. We solve the problem of classification on graphs by generating a similarity matrix from a graph with virtual edges created using predefined rules. The rules are defined based on axioms for similarity spaces. Virtual edges are generated by solving the problem of computing paths with maximal fixed length. We perform experiments by using the similarity matrix as a kernel matrix in support vector machines (SVM). We consider two versions of SVM: for inductive and transductive learning. The experiments show that virtual edges reduce the number of support vectors. When comparing to kernels on graphs, the SVM method with virtual edges is faster while preserving similar generalization performance.

Keywords: Support vector machines, Graph kernels

We solve a problem of collective classification [11] also known as iterative classification or link-based classification where the goal is to determine correct label assignments of all objects in the network. One of the approaches is to use attributes of neighbors' examples, which is called relational classification. Another approach is to use class labels assigned to neighbor instances. This approach is called iterative collective classification. The iterative classification algorithm uses a local classifier that takes class labels of neighbors and return a label value and repeat the process. The collective classification has been applied to a number of real-world problems [11], for example document classification. Specifically, we solve a problem of classification without features. It is called similarity-based classification [4]. The example for such problem is fraud detection for anonymized cell phone network [6]. It is also known as graph-based semi-supervised learning when a graph has unlabeled nodes and weighted edges [9]. The problem of collective classification for partially labeled data is also known as within-network classification [6].

We focus on using SVM for solving the collective classification problem. The requirements for using SVM is to define a kernel matrix, which is a similarity matrix with an additional property of positive semi-definiteness. For networks, we usually have similarities only between connected objects, which leads to a

problem of generating a kernel matrix for a graph. The technique of generating kernels for networks is called "kernels on graphs". This technique has been already applied for recommendation tasks [20, 5] and for semi-supervised classification [5]. The general issue of kernels on graphs is high computational cost (usually $O(n^3)$). The alternative approach is to map the data to the Euclidean space using spectral embedding. The disadvantage is high computational cost and forcing data to be in a specific space. So discrete kernels may be preferable [7]. We focus on another workaround using a similarity matrix with SVM which may not be positive semi-definite. Recently, an efficient solver for SVM has been proposed [10], which does not use a regularization term. It solves a convex optimization problem, regardless of the positive semi-definiteness of a kernel matrix. The straightforward approach to define a similarity matrix is to use value 0 for disconnected vertices meaning no similarity at all. However, this assumption may not be met in collective classification when relations are supposed to exist also for disconnected vertices. Another problem with the straightforward approach specific to SVM is the unknown classification when the example is not connected with a support vector. One of the approaches to solve these problems is to generate additional edges in a graph with provided similarities. This idea has been proposed in [6]. The authors use "ghost edges" with proximities generated by a Random Walk method using specific measure based on a Laplacian matrix. The potential problems with this measure are related to nonstability of the randomness process. The measure is used in a supervised manner by adding ghost edges only between labeled and unlabeled pair of nodes. The supervised setting is prone to cascading errors for iterative methods [13]. Moreover, the measure for ghost edges is defined for a classification on graphs, while it is unclear how to define the measure on weighted graphs which is a more general problem considered in this paper. The weighted graphs has been mentioned in [9] with the example of a weight being the number of hyperlinks between websites.

One of the approaches to define a kernel on graphs is to compute the shortest path distance between nodes [14]. The potential problems have been mentioned like constructing positive definite function and sensitiveness to the insertion/deletion of individual edges. Computing the shortest path between all examples is also expensive, the Floyd-Warshall algorithm has complexity $O(n^3)$. The potential problem with the idea of using the shortest path is that it operates on distances instead of similarities between examples. So the similarities need to be converted to distances and vice versa. The idea in this paper is to generate "virtual edges" by finding paths with maximal similarity and a fixed size. Another idea is to compute similarities for "virtual edges" by using axioms defined for similarity spaces. Such an approach has the advantage of simple interpretation of particular added weights and can be regarded as prior knowledge for similarities. It uses local information instead of global, so it can be used for streaming graphs.

Related work There are two approaches for using a Non-Positive Semidefinite Similarity Matrix for kernel machines [19]: algorithmic and spectrum-transformation.

In the algorithmic approach, one uses the NPSD similarity matrix as a kernel. We need a special formulation of SVM, or a heuristic to find a local solution. In the spectrum-transformation methods the kernel matrix is generated. The representative of the second approach is a diffusion kernel [7] based on a diffusion equation, that considers the data distribution when computing pairwise similarity. We use a novel type of solver for SVM without regularization term which solves a convex optimization problem with any similarity matrix [10].

The approach of using generated PSD kernels for graph data has been investigated in [5]. The authors investigated nine kernels on graphs: the exponential diffusion kernel, the Laplacian exponential diffusion kernel, the von Neumann diffusion kernel, the regularized Laplacian kernel, the commute-time kernel, the random-walk-with-restart similarity matrix, the regularized commute-time kernel, the Markov diffusion kernel and the relative-entropy diffusion matrix. The graph kernels based on normalized Laplacian, mainly Regularized Laplacian, Diffusion Process, p-step Random Walk and Inverse Cosine have been investigated in [7, 16]. In [14], the authors mentioned a random walk kernel, where multiple paths are created with the size T. Potential problems include choosing suitable T and inability to reach a vertex due to cycles. Recently, a random walk method has been used for learning representations in the deep learning framework for classification of graph data [13]. In [1], authors proposed shortest path kernels in a different problem of comparing two graphs with each other. Some kernels have been proposed to improve the computational performance for comparing graphs like Weisfeiler-Lehman Graph Kernels [15]. However, it is not clear whether they can be used to classification on graphs. The diffusion kernel requires diagonalizing the Laplacian which is of order n^3. Moreover, it may have problems with accuracy [3].

The outline of the paper is as follows. First, we define a problem, then the methods and rules, then we derive rules from axioms. After that, we show experiments on real world data sets.

1 Problem definition

We consider the following problem.

Definition 1 (Classification space $\mathfrak{C}$)**.** For a universe X of objects, we have a set C of classes and a set of mappings $M_T : X_T \subset X \to C$ called a *training set* T, $X_{Te} \subseteq X$, where X_{Te} is a set called a *test set*.

For example in the Fig. 1, $\{x_1, x_2, x_6\}$ is a training set, $\{x_3, x_4, x_5, x_7, x_8\}$ is a test set.

Definition 2 (Classification space on a graph $\mathfrak{G}$)**.** We define a classification space on a graph, as a classification space with a graph $G = (X_T \cup X_{Te}, E)$ with weights for each $e \in E$.

We need to know X_{Te} for a graph. We interpret the weights as similarities between examples. The graph without weights can be represented by a weighted graph with binary weights. The example graph is depicted in the Fig. 1.

Problem 1 (Semi-supervised classification problem). A *semi-supervised classification problem* is to find a class for each element of X_{Te} given M_T and X_{Te} on $\mathfrak{C}$.

Problem 2 (Semi-supervised classification problem on a graph). A *semi-supervised classification problem on a graph* is to find a class for each element of $x \in X_{Te}$ given M_T and X_{Te} on $(\mathfrak{C}, \mathfrak{G})$.

For example in the Fig. 1, we need to find a class for all test examples x_3, x_4, x_5, x_7, x_8.

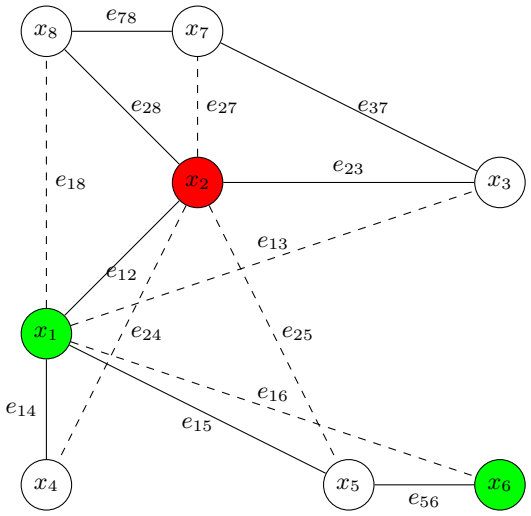

Fig. 1: Problem definition. Filled circles – training data examples, unfilled circles – test data examples, solid lines – edges in a given graph, dashed lines – virtual edges added to a graph.

2 Main contribution

The idea of a method is to generate "virtual edges" between some disconnected examples. In the Fig. 1, the virtual edges are depicted with a dashed line, and these are $e_{16}, e_{13}, e_{45}, e_{25}, e_{24}, e_{18}, e_{27}$.

Another idea in the proposed approach is to generate virtual edges for all examples for which there exists a path of a length 2. For example in the Fig. 1, there is a path from x_1 to x_3 of length 2 through the example x_2, so we generate the edge e_{13}. There is no path of a length 2 between x_5 and x_3, so there is no virtual edge between them. We generate paths in a semi-supervised way based on both training and test examples when both are available as in Prob. 2. For

example in order to generate a path between x_1 and x_6, we use connections with a test example x_5. We do not need to generate edges between test examples, like e_{45} or e_{38}. The generation of virtual edges can be regarded as an unsupervised setting in the sense, that we do not use labels. We do not consider as a path, a sequence of vertices with possible duplicates, which is called a walk, for example x_1 to x_1.

The next idea is to compute the maximal similarity for the virtual edge. Sometimes, there are two paths of a length 2 between examples. In the Fig. 1, there are two paths between x_2 and x_7, that are $x_2 \to x_3 \to x_7$ and $x_2 \to x_8 \to x_7$. We compute the similarity for the virtual edge e_{27} based on each path and we choose the maximal similarity.

We propose three rules for generating weights for virtual edges being similarities between examples. These are

$$\tilde{s}\left(x_1, x_3\right) \leftarrow \frac{s\left(x_1, x_2\right) s\left(x_2, x_3\right)}{s\left(x_1, x_2\right) + s\left(x_2, x_3\right)} \ , \tag{1}$$

where s is a similarity measure between two examples. For example, in the Fig. 1, when $s(x_1, x_5) = 0.6$, $s(x_5, x_6) = 0.9$, we induce similarity $\tilde{s}(x_1, x_6) = 0.36$. The next rule is

$$\tilde{s}\left(x_1, x_3\right) \leftarrow \exp\left(-\left(\sqrt{-\log s\left(x_2, x_3\right)} + \sqrt{-\log s\left(x_1, x_2\right)}\right)^2\right) \ . \tag{2}$$

For example, in the Fig. 1, when $s(x_1, x_5) = 0.6$, $s(x_5, x_6) = 0.9$, we induce similarity $\tilde{s}(x_1, x_6) \approx 0.34$. The third rule is

$$\tilde{s}\left(x_1, x_3\right) \leftarrow s_{\max} - \frac{1}{2}\left(\sqrt{2s_{\max} - 2s\left(x_2, x_3\right)} + \sqrt{2s_{\max} - 2s\left(x_1, x_2\right)}\right)^2 \ , \tag{3}$$

where $s_{\max}$ is a maximal possible value of a similarity. For example, in the Fig. 1, assuming $s_{\max} = 1.0$, when $s(x_1, x_5) = 0.6$, $s(x_5, x_6) = 0.9$, we induce similarity $\tilde{s}(x_1, x_6) \approx 0.1$.

We propose three methods for solving a semi-supervised classification on graphs by using SVM.

Method 1 (Reference Graph Support Vector Machines (RGSVM))**.** *Train a model with SVM with all training examples with a similarity matrix defined as* $s\left(x_i, x_j\right) = w\left(x_i, x_j\right)$ *for* $\left(x_i, x_j\right) \in E$*, otherwise* $s\left(x_i, x_j\right) = 0$*.*

For the example in the Fig. 1 the training set for SVM is x_1, x_2, x_6. The similarity matrix is

$$\begin{bmatrix} s\left(x_1, x_1\right) & s\left(x_1, x_2\right) & 0 \\ s\left(x_2, x_1\right) & s\left(x_2, x_2\right) & 0 \\ 0 & 0 & s\left(x_6, x_6\right) \end{bmatrix} \ . \tag{4}$$

The decision boundary is

$$y_1 \alpha_1 s\left(x_1, x\right) + y_2 \alpha_2 s\left(x_2, x\right) + y_6 \alpha_6 s\left(x_6, x\right) + b = 0 \ , \tag{5}$$

where y_i is a label of the ith example, α_i and b are parameters computed by SVM. The next method uses virtual edges.

Method 2 (Axiomatic Kernel Graph Support Vector Machines (AKGSVM))**.** *Train a model with SVM with all training examples with a similarity matrix defined as* $s(x_i, x_j) = w(x_i, x_j)$ *for* $(x_i, x_j) \in E$*, otherwise for each two not connected vertices* x_1*,* x_2 *find a path of length 2 between them with maximal induced similarity. So when* $(x_1, x_3) \notin E$*, find all* x_i*, such as* $(x_1, x_i) \in E$ *and* $(x_i, x_3) \in E$*, and*

$$\max_{x_i} \tilde{s}(x_1, x_3) \tag{6}$$

and then set $s(x_1, x_3) = \tilde{s}(x_1, x_3)$ *for a path through the optimal* x_i^**, where the induced similarity is computed based on* (1) *or* (2) *or* (3)*, otherwise* $s(x_i, x_j) = 0$*. For the* (1) *the method is called AKGSVM1, for* (2) *AKGSVM2, for* (3) *AKGSVM3.*

For the AKGSVM3, we may have a negative bound (3), so we compute the maximum of the bound (3) and (2). For the example in the Fig. 1, the training set for SVM is x_1, x_2, x_6. The similarity matrix is

$$\begin{bmatrix} s(x_1, x_1) & s(x_1, x_2) & \max(\tilde{s}_b(x_1, x_6), \tilde{s}(x_1, x_6)) \\ s(x_2, x_1) & s(x_2, x_2) & 0 \\ \max(\tilde{s}_b(x_6, x_1), \tilde{s}(x_6, x_1)) & 0 & s(x_6, x_6) \end{bmatrix} , \tag{7}$$

where $\tilde{s}_b$ is the similarity induced by (2). The decision boundary is

$$y_1 \alpha_1 s(x_1, x) + y_2 \alpha_2 s(x_2, x) + y_6 \alpha_6 s(x_6, x) + b = 0 \ , \tag{8}$$

where s is the original similarity when exist or the induced similarity otherwise. The third method is based on creating a local model for each test example.

Method 3 (Reduced Transductive Graph Support Vector Machines (RTGSVM))**.** *Perform AKGSVM for each test example* x_p *separately given a subset of training data with the* r *most similar examples connected with* x_p*, where* r *is some parameter.*

The property of creating local models to particular test examples which are not designed to generalize to other test examples is called transductive learning. When there is no enough connected examples to match the r value, the subset is smaller than r. For the example in the Fig. 1, for $r = 2$, the training set for $x_p = x_5$ is a subset of training data with two nearest training examples to x_5 computed based on similarities. It may be $\{x_1, x_6\}$ or $\{x_2, x_1\}$ or $\{x_2, x_6\}$. For the first case the similarity matrix is

$$\begin{bmatrix} s(x_1, x_1) & \tilde{s}(x_1, x_6) \\ \tilde{s}(x_6, x_1) & s(x_2, x_2) \end{bmatrix} . \tag{9}$$

The decision boundary is

$$y_1 \alpha_1 \tilde{s}(x_1, x) + y_6 \alpha_6 \tilde{s}(x_6, x) + b = 0 \ , \tag{10}$$

where $\tilde{s}$ is the original similarity when exist, otherwise it is the induced similarity. Overall, we have 5 separate models one per each test example. In the last two methods, virtual edges are used also when classifying test data. The RTGSVM method can be used with any type of virtual edges either based on (1) or (2) or (3).

3 Analysis of rules

The rules are generated based on defined similarity spaces. First, we define a similarity space corresponding to (1)

Definition 3 (Similarity space $\mathfrak{S}$)**.** For a binary relation R on X, we define a *similarity measure* $s_R : X \times X \to \mathbb{R}$ where s_R is a restriction of the function s to a binary relation R, which is a subset of the Cartesian product that is $R \subseteq X \times X$, shortly we note s_R as s. Similarity measure fulfills the similarity axioms: $0 < s(x_1, x_2) \leq s_{\max}$, where $s_{\max} \geq 0$, $s(x_1, x_2) = s_{\max} \iff x_1 = x_2$, $s(x_1, x_2) = s(x_2, x_1)$ and

Axiom 1.

$$s(x_1, x_3) \geq \frac{s(x_1, x_2)\, s(x_2, x_3)}{s(x_1, x_2) + s(x_2, x_3)} \quad . \tag{11}$$

We call the assumptions axioms following [2]. The Ax. 1 gives a lower bound for similarity. We define additionally a *pseudosimilarity*, when the second axiom is replaced with $s(x_1, x_1) = s_{\max}$. The similarity concept is related to the distance. The distance is also called a metric and is part of a definition of a metric space. That is the metric or distance is defined as a function $d : X \times X \to \mathbb{R}_+$ satisfying the following properties: $d(x_1, x_2) \geq 0$ and $d(x_1, x_2) = 0 \iff x_1 = x_2$ (non-negativity), $d(x_1, x_2) = d(x_2, x_1)$ (symmetry) and

Axiom 2.

$$d(x_1, x_3) \leq d(x_2, x_3) + d(x_1, x_2) \quad . \tag{12}$$

One difference between the definition of similarity and the distance is that the distance is defined for all $x \in X$, while the similarity we define only on some subset of $X \times X$, that is the binary relation R. We derive axioms for similarity from axioms for the distance. In particular, when we replace similarity by the inverse of a distance assuming $d(x, y) > 0$ in (11), so we substitute

$$s(x_1, x_2) = 1/d(x_1, x_2) \ , \tag{13}$$

we get the distance axiom Ax. 2. The kernel matrix for some kernel functions can be interpreted as similarities between all examples. One notable example is the radial basis function (RBF) kernel, for which the kernel value is defined in terms of distance. By using this property, we can get alternative triangle inequality for the similarity (1). We can convert it to similarity by using $\log s(x_1, x_2) = -d(x_1, x_2)^2$, so $d(x_1, x_2) = \pm\sqrt{-\log s(x_1, x_2)}$, d is nonnegative so

$$d(x_1, x_2) = \sqrt{-\log s(x_1, x_2)} \ . \tag{14}$$

We additionally assume that $s_{\max} = 1$. Then after substituting (14) to (12), we get

Axiom 3.

$$s\left(x_1, x_3\right) \geq \exp\left(-\left(\sqrt{-\log s\left(x_2, x_3\right)} + \sqrt{-\log s\left(x_1, x_2\right)}\right)^2\right) \quad . \tag{15}$$

Based on this axiom, we created a rule (2). It gives another possible lower bound on the similarity. It holds that

$$s\left(x_1, x_2\right) > \frac{s\left(x_1, x_2\right) s\left(x_2, x_3\right)}{s\left(x_1, x_2\right) + s\left(x_2, x_3\right)} \quad . \tag{16}$$

The same holds for the left side being $s(x_2, x_3)$. So the lower bound is smaller than any similarity involved. The maximal possible induced similarity for $s_{\max} = 1$ is 0.5, and it is achieved for $s_1 = 1$ and $s_2 = 1$. For any $s_{\max}$ the maximal possible similarity is $s_{\max}/2$ and is achieved for $s_1 = s_{\max}$ and $s_2 = s_{\max}$.

When generating similarities using the RBF kernel for some given particular distances, the question is about satisfying the triangle inequality for similarity Ax. 1. So after substitution $s(x_1, x_2) = \exp\left(-d(x_1, x_2)^2\right)$ to Ax. 1, we get

$$\begin{gathered} d^2\left(x_1, x_3\right) \leq \\ \log\left(\exp\left(-d^2\left(x_1, x_2\right)\right) + \exp\left(-d^2\left(x_2, x_3\right)\right)\right) + d^2\left(x_1, x_2\right) + d^2\left(x_2, x_3\right) \quad . \end{gathered} \tag{17}$$

This is alternative triangle inequality for a distance which can be used for defining alternative metric space. We can also substitute (12) to Ax. 3 and we get

$$d\left(x_1, x_3\right) \leq \frac{1}{\exp\left(-\left(\sqrt{\log d\left(x_2, x_3\right)} + \sqrt{\log d\left(x_1, x_2\right)}\right)^2\right)} \quad . \tag{18}$$

This is another way of defining triangle inequality for a metric space. For vector spaces, the feature map for kernel machines exists only when the kernel is positive-definite. The similarity is a broader concept and this condition may not be met. The relation between metric and the positive-definite kernels extended to the concept of similarity is as follows

$$\tilde{d}\left(x_1, x_2\right) = \sqrt{2s_{\max} - 2s\left(x_1, x_2\right)} \quad , \tag{19}$$

where

$$2s_{\max} - 2s\left(x_1, x_2\right) \geq 0 \quad , \tag{20}$$

$\tilde{d}$ is a pseudometric. The disadvantage of such definition is that a distance is bounded by a value $\sqrt{2s_{\max}}$. For a pseudometric the second axiom is replaced by $d(x, x) = 0$, so the distance between different examples can be 0. The pseudometric is related to pseudosimilarity. The condition (20) is the existence of $s_{\max}$. The relation can be reformulated as

$$s\left(x_1, x_2\right) = s_{\max} - \frac{1}{2}\tilde{d}\left(x_1, x_2\right)^2 \quad . \tag{21}$$

So positive definiteness is related to a specific assumption about connection between similarity and metric. For such definition, we can see how the triangle inequality axiom for similarities Ax. 1 relates to distances by substituting (21) to Ax. 1 and we get

$$\tilde{d}\left(x_1, x_3\right)^2 \leq \frac{1}{s_{\max}} \frac{1}{4} \tilde{d}\left(x_1, x_2\right)^2 \tilde{d}\left(x_2, x_3\right)^2 - \frac{1}{s_{\max}} \frac{1}{4} \tilde{d}\left(x_1, x_2\right)^2 \tilde{d}\left(x_1, x_3\right)^2 - \frac{1}{4} \tilde{d}\left(x_2, x_3\right)^2 d\left(x_1, x_3\right)^2 - s_{\max} \quad . \tag{22}$$

We derive the axiom for similarity by substituting distance from (19) to Ax. 2 and we get

Axiom 4.

$$s\left(x_1, x_3\right) \geq s_{\max} - \frac{1}{2} \left(\sqrt{2 s_{\max} - 2 s\left(x_2, x_3\right)} + \sqrt{2 s_{\max} - 2 s\left(x_1, x_2\right)} \right)^2 \quad . \tag{23}$$

The problem is that sometimes we get loose negative bound for similarity, which is useless for substitution. Based on this axiom, we created a rule (3).

4 Analysis of methods

In addition to the proposed methods, we analyze the following methods: k-nearest neighbors (KNN), SVM with a diffusion kernel based on a normalized Laplacian and SVM with a shortest path kernel. The shortest path kernel minimizes distances, so given a graph with similarities, we convert each similarity on a path to a distance and vice versa by using (13).

Complexity analysis. For the SVM methods trained on the whole kernel matrix, that are SVM with a diffusion kernel and SVM with a shortest path kernel the computational complexity for the stochastic gradient solver with the worst violators [10] is $O(sn)$, where s is the number of support vectors, n is the number of examples. The maximal number of iterations is s. In each iteration, we need to update a functional margin value for each remaining parameter and find the worst violator. Computing a kernel matrix for a diffusion kernel requires eigendecomposition of a normalized Laplacian matrix which has complexity $O(n^3)$. Computing a kernel matrix for a shortest path kernel leads to all-pairs shortest paths problem. We need to compute paths between all vertices, except those already connected. The Floyd-Warshall algorithm has complexity $O(n^3)$, where n is the number of vertices. For sparse graphs with nonnegative weights which is the case here, Dijkstra's algorithm can be used which has complexity $O(|E|n + n^2 \log n)$, where $|E|$ is the number of edges. Both kernels are impractical in solving large-scale machine learning problems. Regarding the method AKGSVM1, AKGSVM2, AKGSVM3 the complexity of adding virtual edges is $O(|E||E_c|)$, where $|E_c|$ is the number of edges connected with each vertex for the algorithms which iterates over all edges and all connected edges to both vertices of the edge. The connected edges can be found in a constant time using hash structures. This algorithm is suitable for sparse graphs. The question

about solving the longest path problem for a fixed size which has linear complexity $O(l!2^l n)$, where l is the length of the path for AKGSVM1, AKGSVM2, AKGSVM3, when we do not sum weights remains open. This algorithm performs deep-first search, path decomposition and apply dynamic programming. For the KNN, there is no training phase. During testing, the nearest neighbors must be found for each test example which can done in $O(tn)$, where t is the number of test examples. For the RGSVM, the complexity is as for SVM. For the RTGSVM, the complexity is the same as for AKGSVM however computed only for nearest neighbors.

Memory complexity. For graphs, especially sparse graphs, we can store information about edges in a sparse matrix structure. The problem with the methods with global kernels that are SVM with a diffusion kernel and SVM with a shortest path kernel is that they create a dense matrix with n^2 elements (including the existing edges). So we lose sparsity of a representation for a problem. For AKGSVM1, AKGSVM2, AKGSVM3, we add only a limited number of edges. For RTGSVM, it is enough to add a virtual edge only for a pair of edges, when one of them is connected with a test example.

Generalization performance. The SVM has roots in statistical learning theory. They have been developed using generalization bounds based on Vapnik-Chervonenkis (VC) dimension. Vapnik derived generalization bounds for the transductive learning in [17], which are better than for inductive learning. The proposed method RTGSVM uses transductive learning approach. We use classification on graphs, where the assumption about independent and identically distributed (i.i.d.) data may not be met. Moreover, we deal with discrete spaces, for which we need to use combinatorial bounds. Such bounds competitive to the VC dimension bounds have been derived [18].

For the RGSVM, the kernel with 0s can be indefinite (the quadratic form is neither convex nor concave). The consequence is that the objective function of the dual problem of SVM can be non-convex. The problem that is related to indefinite kernels is that the feature map may not exist for such kernel. However, for graphs, we do not have feature representation for data examples. We have similarities that are defined a priori. Recently, in [10] the stochastic gradient method has been proposed with extreme early stopping. The method does not use a regularization term, thus the kernel values are only related to the linear term.

When we compute similarities according to (11) as in AKGSVM1, then the maximal similarity path will be equivalent to the minimal distance path (the shortest path) for a path length of 2.

For the decision boundary of SVM

$$\sum_{i=1}^{n} y_i \alpha_i K\left(\boldsymbol{x_i}, \boldsymbol{x}\right) + b = 0 \ , \tag{24}$$

where y_i is a class of the ith example, K is a kernel function for RGSVM, when the testing example is not connected with any support vector, then its classification depends on the sign of b. We expect that during tuning of hyperparameters

of SVM, the solutions with support vectors close to validation examples will be promoted. Because AKGSVM increases the number of connections, we expect that the requirement for the number of support vectors will be lower. In particular, AKGSVM increases the number of test examples connected with at least one training example. In Fig. 1, x_7 after generating a virtual edge e_{27} becomes connected to a training example x_2. In the case, when the number of test examples is much bigger than the number of training examples additional procedure might be needed for AKGSVM and KNN in order to improve classification of test examples not connected with any training examples, for example iterative labeling.

Table 1: Generalization performance. The numbers in descriptions of the columns mean the methods: 1 - KNN, 2 - RGSVM, 3 - AKGSVM1, 4 - AKGSVM2, 5 - AKGSVM3, 6 - RTGSVM, 7 - diffusion kernel SVM, 8 - the shortest path kernel SVM. Column descriptions: *no* – experiment name, *data* – data set, k_g – the number of the nearest neighbors for generating a structure, k/r – the k value for KNN and r value for RTGSVM, *train* – maximal training set size, *test* – maximal testing set size, *ra* – the average rank for the mean misclassification error; the best method is in bold.

no	data	k_g	k/r	train	test	ra1	ra2	ra3	ra4	ra5	ra6	ra7	ra8
Ex1	All	5	5	100	20	4.55	4.6	4.31	4.35	4.62	4.48	4.88	**4.2**
DBLPEx1	dblp		5	500	50	6.05	6.4	3.65	3.65	3.65	**3.1**	6.4	**3.1**

Table 3: The number of support vectors. The numbers in descriptions of the columns mean the methods: 1 - KNN, 2 - RGSVM, 3 - AKGSVM1, 4 - AKGSVM2, 5 - AKGSVM3, 6 - RTGSVM, 7 - diffusion kernel SVM, 8 - the shortest path kernel SVM. Column descriptions: *no* – experiment name, *sv* – the number of support vectors reported for inductive methods, the best method is in bold, *svB* – Bayesian signed rank probability for the number of support vectors.

no	sv2	sv3	sv4	sv5	sv7	sv8	svB23	svB38
Ex1	81	71	73	76	68	**57**	0.86	**0.95**
DBLPEx1	497	430	500	500	385	**293**	0.63	0.75

Table 5: Computational performance. The numbers in descriptions of the columns mean the methods: 1 - KNN, 2 - RGSVM, 3 - AKGSVM1, 4 - AKGSVM2, 5 - AKGSVM3, 6 - RTGSVM, 7 - diffusion kernel SVM, 8 - the shortest path kernel SVM. Column descriptions: *no* – experiment name, trT – cumulative training time in seconds, teT – cumulative testing time in seconds.

no	trT2	trT3	trT4	trT5	trT7	trT8	teT1	teT6
Ex1	**0.78**	1.9	1.93	1.86	4.66	4.61	**13.0**	65.0
DBLPEx1	**0.11**	0.18	0.2	0.2	5.97	3.31	2.0	**0.0**

5 Experiments

We perform two types of experiments. The first type is on generated weighted graphs with distances from standard classification data sets. We generated graphs by connecting each example with k_g nearest neighbors from a training data set found using the Euclidean distance. The weights in graphs are distances, which are later converted to similarities based on the RBF kernel as a function of a distance by each of the method. Local methods that are KNN and RTGSVM use the similarity to find nearest neighbors. The second type of experiment is on real world graph with similarities for a citation network DBLP. The nodes are articles indexed in the DBLP data set. We use specifically the DBLP data set v1 [12]. The edges are citations between articles. The weights for this graph are binary similarities. When there is a citation the weight is 1, otherwise it is 0. It is a special case of a weighted graph. The graph is undirected.

We use our implementation of the stochastic gradient descent (SGD) solver from [10] for all variants of SVM. For the first type of experiment, we compare all methods on data sets listed in Table 7 for binary classification. More details about data sets are on the LibSVM site [8]. We selected all data sets from this site for binary classification, except those that was too large to store and process them using dense structures due to memory limitation. We plan to implement sparse representation of data in our framework in the future. For all data sets, every feature is scaled linearly to $[0, 1]$. For SVM based methods the number of hyperparameters to tune is 2, σ and C, for KNN only σ for the experiment 1. In the second experiment, we do not tune σ, because we have already similarities. For all hyperparameters, we use a double grid search method for finding the best values – first a coarse grid search is performed, then a finer grid search. The range for σ values is from 2^{-9} to 2^{9}, for C it is from 2^{-9} to 2^{14} on the first level. The β parameter for a diffusion kernel is set to 0.3. We use the procedure similar to repeated double (nested) cross validation for performance comparison. For the outer loop, we run a modified k-fold cross validation for $k = 10$, with the training set size set to 80% of all examples. When it is not possible to create the next fold, we shuffle data and start from the beginning. For the DBLP data set,

we shuffle data with balancing classes only during preprocessing graph data to the internal format. There is no need to shuffle data for our experiments due to limited size of a training data set. We use additional automated standardization of a training matrix after dividing data to folds. We use the 5-fold cross validation for the inner loop for finding optimal values of the hyperparameters. In the first experiment, we generate a graph for each iteration of the inner loop. After that, we run the method on a training set, and we report results on a test set. Here, we also generate a graph in the first experiment. We additionally limit the size of a training set to speed up the experiments. The limitation for a training set is listed in Table 1 for each experiment. We also limit the number of test examples in experiments with the RTGSVM method to speed up the experiments, because for this method, a new model must be trained for each test example. In the future, we plan to implement aggregation of test examples to limit the number of models. We also use Bayesian statistical tests which are preferred over null hypothesis significance testing (NHST). In particular, we use Bayesian signed rank test implemented in R. For the RTGSVM method, we generate virtual edges as in AKGSVM1.

The overall results for generalization performance are in Table 1. The results for the number of support vectors are in Table 3. The results for computational performance are in Table 5. The example of results for particular data sets are in Table 7. The observations are as follows.

- The generalization performance of local models like KNN and RTGSVM are competitive to inductive SVM. For standard classification SVM has small advantage in terms of accuracy over KNN [10]. For graph data, this difference is even less noticeable. However, the requirement for local models is to get enough connections to the nearest neighbors to match the expected number of neighbors k for KNN, and r for RTGSVM. For the increased number of connections to other test examples, local models may start to degrade. Moreover, in the Ex1, the number of the nearest neighbors matches the number of neighbors used during graph generation. In real world graphs the number of neighbors in a graph varies, thus choosing the optimal value of k/r may require some tuning, and even then the optimal value can be different for different test examples. The generalization performance of RGSVM is slightly worse than AKGSVM which is noticeable for the DBLP data set, but without statistical significance. Thus, reducing sparsity of a similarity matrix is beneficial for SVM. However, generating the full kernel matrix does little noticeable improvement in the generalization performance over AKGSVM, for the shortest path kernel. Specifically, for the DBLP data set, AKGSVM is little better than KNN and RGSVM. The most competitive for the shortest path kernel is RTGSVM with almost the same performance.
- The number of support vectors is much better for the AKGSVM than for RGSVM which is almost statistically significant for combined results (column svB23 in Table 3). It is almost statistically significant for particular data sets (column svB23 in Table 7). The global kernels have still advantage in terms of the number of support vectors compared to other methods (column svB39

in Table 3). The potential reason for this is that introducing sparsity in the kernel matrix by putting 0s leads to non-smoothness, which requires more complex functions. For the DBLP data sets, the number of support vectors is rather high due to the limited number of edges, which is affected by the size of a training data set and connections with other test examples.
– Training time for SVM with global kernels that are SVM with a diffusion kernel, and with the shortest path kernel are considerably higher than for other methods due to computational complexity. We also lose advantage of sparsity of a kernel matrix, so memory consumption may be an issue. Testing time for local models can be greater than training time for inductive models, which depends on the number of test examples.

Overall, from the practical point of view, the local models like KNN can be used for data sets when there are enough neighbor connections and we have a graph with connections between similar examples and relatively small number of test examples. If the weighted connections are between nonsimilar examples, the local classifiers may not work properly. For bigger number of connections, SVM methods may have advantage of taking into account structure of the data. Due to the computational performance and memory requirements, the proposed axiomatic kernels are a better choice than global kernels, especially for big data sets.

6 Summary

We proposed a novel idea of generating rules based on axioms for generating virtual edges, which are used in SVM for classification on graphs. Potentially, virtual edges can be used with any other method for classifying graphs. The results are promising in terms of computational performance when compared to kernels on graphs. The proposed methods may be preferable in some scenarios over simple local models like KNN. The framework of axiomatic rules can be extended by introducing combination of rules and tuning of rules by incorporating uncertainty in the form of hyperparameters.

Acknowledgments

The theoretical analysis and the method design are financed by the National Science Centre in Poland, project id 289884, UMO-2015/17/D/ST6/04010, titled "Development of Models and Methods for Incorporating Knowledge to Support Vector Machines" and the data driven method is supported by the European Research Council under the European Unions Seventh Framework Programme. Johan Suykens acknowledges support by ERC Advanced Grant E-DUALITY (787960), KU Leuven C1, FWO G0A4917N. This paper reflects only the authors views, the Union is not liable for any use that may be made of the contained information.

Table 7: Results per data set. The numbers in descriptions of the columns mean the methods: 1 - KNN, 2 - RGSVM, 3 - AKGSVM1, 4 - AKGSVM2, 5 - AKGSVM3. Column descriptions: *no* – experiment name, *dn* – the name of a data set, *s* – the number of all examples, *d* – the dimension of a problem, *ce* – the mean misclassification error; the best method is in bold, *sv* – the number of support vectors reported for inductive methods, *svB* – Bayesian signed rank probability for the number of support vectors.

no	dn	s	d	err1	err2	err3	err4	err5	sv2	sv3	svB23
Ex1	a1a	24947	123	**0.34**	0.39	0.385	0.39	0.36	81	**53**	0.79
Ex1	australian	690	14	0.205	0.205	0.21	0.195	**0.17**	92	**89**	0.41
Ex1	breast-cancer	675	10	0.065	0.06	0.07	0.07	**0.05**	98	**58**	0.83
Ex1	cod-rna	100000	8	0.24	0.195	**0.17**	0.18	0.21	79	**55**	0.75
Ex1	colon-cancer	62	2000	0.25	0.26	**0.22**	0.245	0.255	31	**27**	0.53
Ex1	covtype	100000	54	**0.305**	0.33	0.355	0.34	0.355	**69**	76	0.24
Ex1	diabetes	768	8	**0.215**	0.235	0.27	0.245	0.26	82	**78**	0.54
Ex1	fourclass	862	2	0.05	**0.03**	0.035	**0.03**	0.035	74	**53**	0.63
Ex1	german_numer	1000	24	**0.3**	0.365	0.35	0.335	0.33	79	**74**	0.53
Ex1	heart	270	13	0.19	0.2	0.16	**0.155**	0.18	99	**92**	0.53
Ex1	HIGGS	100000	28	**0.345**	0.355	0.43	0.395	0.4	91	**87**	0.36
Ex1	ijcnn1	100000	22	0.015	**0.0**	**0.0**	0.01	0.01	**94**	100	0.0
Ex1	ionosphere_sc	350	33	0.26	0.195	0.1	**0.075**	0.145	57	**55**	0.5
Ex1	liver-disorders	341	5	0.5	**0.48**	0.485	0.485	0.49	**88**	96	0.13
Ex1	madelon	2600	500	**0.415**	0.43	0.44	0.47	0.455	86	**77**	0.6
Ex1	mushrooms	8124	111	0.05	**0.025**	0.05	0.045	0.035	79	**44**	0.85
Ex1	phishing	5785	68	**0.185**	0.225	0.2	0.21	0.205	98	**87**	0.61
Ex1	skin_nonskin	51432	3	0.07	**0.035**	0.04	0.04	**0.035**	85	**82**	0.36
Ex1	splice	2990	60	0.345	0.355	0.375	**0.34**	0.365	85	**70**	0.73
Ex1	sonar_scale	208	60	0.29	0.23	0.215	**0.205**	0.25	69	**66**	0.52
Ex1	SUSY	100000	18	0.365	**0.355**	0.38	**0.355**	0.41	90	**85**	0.61
Ex1	svmguide1	6910	4	**0.08**	0.1	0.09	0.09	0.085	80	**63**	0.57
Ex1	svmguide3	1243	21	**0.25**	0.285	0.26	0.295	0.275	83	**70**	0.66
Ex1	w1a	34703	300	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	91	**75**	0.42
Ex1	websam_unigr	100000	134	0.155	0.175	**0.1**	0.135	0.185	71	**56**	0.72
DBLPEx1	dblp	7967		0.162	0.164	**0.144**	**0.144**	**0.144**	497	**430**	0.63

References

[1] Borgwardt, K.M., Kriegel, H.: Shortest-path kernels on graphs. In: Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA. pp. 74–81 (2005). https://doi.org/10.1109/ICDM.2005.132

[2] Bronshtein, I.N., Semendyayev, K., Musiol, G., Muehlig, H.: Handbook of Mathematics, chap. Functional Analysis, pp. 596–641. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72122-2_12

[3] Can, T., Çamoglu, O., Singh, A.K.: Analysis of protein-protein interaction networks using random walks. In: Proceedings of the 5th international workshop on Bioinformatics, BIOKDD 2005, Chicago, Illinois, USA, August 21, 2005. pp. 61–68 (2005). https://doi.org/10.1145/1134030.1134042

[4] Chen, Y., Garcia, E.K., Gupta, M.R., Rahimi, A., Cazzanti, L.: Similarity-based classification: Concepts and algorithms. J. Mach. Learn. Res. **10**, 747–776 (2009),

[5] Fouss, F., Françoisse, K., Yen, L., Pirotte, A., Saerens, M.: An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. Neural Networks **31**, 53–72 (2012). https://doi.org/10.1016/j.neunet.2012.03.001

[6] Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008. pp. 256–264 (2008). https://doi.org/10.1145/1401890.1401925

[7] Kondor, R., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002. pp. 315–322 (2002)

[8] Libsvm data sets. (06 2011)

[9] Lin, F., Cohen, W.W.: Semi-supervised classification of network data using very few labels. In: International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2010, Odense, Denmark, August 9-11, 2010. pp. 192–199 (2010). https://doi.org/10.1109/ASONAM.2010.19

[10] Melki, G., Kecman, V., Ventura, S., Cano, A.: OLLAWV: online learning algorithm using worst-violators. Appl. Soft Comput. **66**, 384–393 (2018)

[11] Namata, G., Sen, P., Bilgic, M., Getoor, L.: Collective classification. In: Encyclopedia of Machine Learning and Data Mining, pp. 238–242. Springer US (2017). https://doi.org/10.1007/978-1-4899-7687-1_44

[12] Pan, S., Zhu, X., Zhang, C., Yu, P.S.: Graph stream classification using labeled and unlabeled graphs. In: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013. pp. 398–409 (2013). https://doi.org/10.1109/ICDE.2013.6544842

[13] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New

York, NY, USA - August 24 - 27, 2014. pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
[14] Schölkopf, B., Tsuda, K., Vert, J.P.: Kernel Methods in Computational Biology, chap. Diffusion Kernels, pp. 171–192. The MIT Press (July 16, 2004) (01 2003)
[15] Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**, 2539–2561 (2011)
[16] Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings. pp. 144–158 (2003)
[17] Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (September 1998)
[18] Vorontsov, K., Ivahnenko, A.: Tight combinatorial generalization bounds for threshold conjunction rules. In: Pattern Recognition and Machine Intelligence - 4th International Conference, PReMI 2011, Moscow, Russia, June 27 - July 1, 2011. Proceedings. pp. 66–73 (2011). https://doi.org/10.1007/978-3-642-21786-9_13
[19] Wu, G., Chang, E.Y., Zhang, Z.: An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In: Proceedings of the 22nd International Conference on Machine Learning (2005)
[20] Yajima, Y., Kuo, T.: Efficient formulations for 1-svm and their application to recommendation tasks. JCP **1**(3), 27–34 (2006). https://doi.org/10.4304/jcp.1.3.27-34