A Random Walk Kernel Derived from Graph Edit Distance

Michel Neuhaus* and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{mneuhaus, bunke}@iam.unibe.ch

Abstract. Random walk kernels in conjunction with Support Vector Machines are powerful methods for error-tolerant graph matching. Because of their local definition, however, the applicability of random walk kernels strongly depends on the characteristics of the underlying graph representation. In this paper, we describe a simple extension to the standard random walk kernel based on graph edit distance. The idea is to include global matching information in the local similarity evaluation of random walks in graphs. The proposed extension allows us to improve the performance of the random walk kernel significantly. We present an experimental evaluation of our method on three difficult graph datasets.

1 Introduction

For more than thirty years, a huge variety of methods have been developed addressing the problem of graph matching [1]. In recent years, a novel class of algorithms based on kernel machines has gained a significant amount of interest in the pattern recognition community. The basic idea of kernel machines is to map the classification problem from the pattern domain to a vector space implicitly defined in terms of a kernel function [2]. In the context of graph matching, kernel machines allow us to apply vector space operations to graphs by embedding the space of graphs in a vector space. Provided that the definition of a kernel function that is suitable for the pattern matching problem under consideration is given, a large number of algorithms for pattern analysis and recognition can readily be applied, including principal component analysis, Fisher discrimination analysis, and Support Vector Machines [2].

Various kernel functions have been proposed to solve the graph matching problem as well as the related string matching problem. In a common approach, the similarity of patterns is defined in terms of similar substructures they contain [3,4]. Another approach employs the definition of a Schur-Hadamard inner product on graphs [5]. Based on the notion of random walks in graphs, several kernels have been developed [6,7]. While kernel methods provide a powerful way

* Supported by the Swiss National Science Foundation NCCR program *Interactive Multimodal Information Management (IM)2* in the Individual Project *Multimedia Information Access and Content Protection*.

D.-Y. Yeung et al. (Eds.): SSPR&SPR 2006, LNCS 4109, pp. 191–199, 2006.

to analyse and classify graphs, they are, in some cases, limited in terms of the flexibility of their structural matching process.

In this paper we aim at enhancing a standard random walk kernel by information derived from the well-established error-tolerant graph edit distance measure [8,9]. In the remainder of this paper, we will briefly introduce graph edit distance and the random walk kernel, describe the extension we propose, and demonstrate the usefulness of our method in classification experiments.

2 Graph Edit Distance

Graph edit distance is one of the most universal graph matching methods in the sense that edit distance is not restricted to special classes of graphs, such as planar graphs, bounded valence graphs, or graphs labeled with discrete attributes. The key idea is to measure the structural dissimilarity of two graphs by the minimal amount of distortion that is needed to transform one graph into the other [8,9]. The only requirement for graph edit distance to be applicable is that an underlying distortion model must be given such that the strength of distortions can be measured. Hence, graph edit distance can be computed for graphs with arbitrary node and edge relations and any kind of node and edge labels.

More formally, let $g = (V, E, \mu, \nu)$ denote a graph g consisting of a finite set of nodes V, a set of directed edges $E \subseteq V \times V$, a node labeling function $\mu : V \rightarrow L$ assigning an attribute from L to each node, and an edge labeling function $\nu : E \rightarrow L$. The label alphabet L is often defined as a finite set of labels, $L = \{\alpha, \beta, \gamma, \ldots\}$, or a Euclidean vector space, $L = \mathbb{R}^n$. We then define a number of distortion, or edit, operations on graphs. A standard set of graph edit operations consists of an insertion, a deletion, and a substitution operation of nodes and edges. An edge deletion is equivalent to the removal of an edge from a graph, and a node substitution results in the replacement of a node label by another one. Further required is a cost function assigning each edit operation a penalty cost value, such that weak edit operations have low costs and strong edit operations have high costs. For instance, slightly changing a label should, in most cases, result in lower costs than strongly changing the same label. The key idea of graph edit distance is that for two structurally similar graphs only a few weak edit operations are needed to convert one graph into the other. By contrast, for two quite different graphs, a larger number of edit operations of greater strength are needed to make the two graphs identical to each other. Consequently, the edit distance of two graphs g and g' is defined by the minimum cost sequence of edit operations transforming g into g',

$$d(g, g') = \min_{(e_1, \ldots, e_k) \in E(g, g')} \sum_{i=1}^{k} c(e_i) \ . \tag{1}$$

A sequence of edit operations transforming one graph into the other is also called an edit path. Note that $E(g, g')$ denotes the set of edit paths from g to g', and c is a function assigning costs to edit operations.

The edit distance of graphs is usually computed by means of a tree search procedure [9]. As every node can potentially be substituted by any other node, it can be shown that the computational complexity of edit distance is exponential in terms of space and time. In practice, it turns out that the computation of exact edit distance is limited to graphs with up to 12 nodes, typically. In this paper, we therefore resort to an approximate edit distance algorithm [10] in those cases where the exact distance cannot be computed.

The edit distance of graphs is normally used in conjunction with a k-nearest-neighbor classifier. For an unknown input graph, we compute the edit distance to a number of prototype graphs and assign the input graph to the most frequent class among the k closest prototypes.

3 Random Walk Kernels

The objective in this section is to define error-tolerant graph similarity measures, or kernel functions, that can be used in conjunction with kernel machines [2]. The main advantage of kernel based classifiers for structured data is that the classification problem can be formulated in a vector space related to the original pattern space solely by definition of a kernel function. Given a valid kernel function, it can be proven that there exists a vector space with its inner product being equal to the kernel function. This allows us to run a number of algorithms for classification and pattern analysis in the implicitly existing vector space without explicitly mapping the graphs to the elements of the vector space. In our experiments, we apply the kernel functions in conjunction with one of the most prominent and best performing kernel based classifiers, the Support Vector Machine (SVM) [2].

We proceed by first describing a well-known random walk kernel for discretely labeled graphs [6] and its extension to continuously labeled graphs [7]. Then we suggest modifications to make the random walk kernel more robust.

3.1 Discretely Labeled Graphs

The original random walk kernel is defined by means of the direct product graph [6]. The direct product of two graphs $g = (V, E, \mu, \nu)$ and $g' = (V', E', \mu', \nu')$ is the graph $(g \times g') = (V_\times, E_\times, \mu_\times, \nu_\times)$ given by

$$V_\times = \{(v, v') \in V \times V' : \mu(v) = \mu'(v')\} \text{ and} \tag{2}$$
$$E_\times = \big\{((u, u'), (v, v')) \in V_\times^2 : (u, v) \in E \wedge (u', v') \in E' \wedge \nu(u, v) = \nu'(u', v')\big\} \ .$$

The labeling functions of the product graph are defined by $\mu_\times(v, v') = \mu(v) = \mu'(v')$ and $\nu_\times((u, u'), (v, v')) = \nu(u, v) = \nu'(u', v')$. In other words, in the direct product graph $(g \times g')$, we simply identify pairs of nodes of both graphs with identical labels and pairs of edges with identical labels, constituting the nodes and edges of the product graph. The adjacency matrix $A_\times$ of $(g \times g')$ is then defined as

$$[A_\times]_{(u,u'),(v,v')} = \begin{cases} 1 & \text{if } ((u, u'), (v, v')) \in E_\times \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{3}$$

Note that the adjacency matrix is a $|V_\times| \cdot |V_\times|$-matrix containing at position (i, j) value 1 if node i is connected to node j by an edge in $(g \times g')$, and value 0 otherwise. From the adjacency matrix $A_\times$ of the direct product, one can then derive the graph kernel with weighting parameter $\lambda \geq 0$ according to the formula [6]

$$k_\times(g, g') = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda^n A_\times^n \right]_{ij} . \tag{4}$$

If $\lambda < 1$, it is sufficiently accurate to evaluate infinite sums by their first few dominant addends only.

The kernel can be interpreted as a measure of the number of matching labeled random walks in both graphs. That is, if the sequence of node and edge labels encountered on a random walk in g matches the sequence of node and edge labels of a random walk in g', this contributes a certain amount to the overall similarity $k_\times(g, g')$. The graph kernel reflects the intuitive understanding that two graphs are similar if there are a large number of identical random walks in both graphs.

3.2 Continuously Labeled Graphs

The main limitation of the kernel defined above is that it is only applicable to graphs with discretely labeled nodes and edges. If a random walk in g differs from a random walk in g' only in a single node label, the two walks are considered completely different and are therefore not taken into account. Unfortunately, most graphs extracted from real-world data contain a significant amount of noise, and attributes with continuous values are mostly used to describe non-discrete data. For these reasons, an extension of the original random walk kernel has been proposed [7]. The idea is not to evaluate if two walks are identical, but rather if they are similar. This modified kernel is applicable to graphs with continously labeled nodes and edges.

To obtain the modified kernel, we leave out the label equality conditions in Eq. 2, resulting in a modified direct product $(g \times g')$, and define the adjacency matrix of $(g \times g')$ by

$$[A_\times]_{(u,u'),(v,v')} = \begin{cases} k((u, u'), (v, v')) & \text{if } ((u, u'), (v, v')) \in E_\times \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{5}$$

where the kernel function k measuring the similarity of pairs of nodes (u, u') and (v, v') is given by

$$k((u, u'), (v, v')) = k_{node}(u, u') \cdot k_{edge}((u, v), (u', v')) \cdot k_{node}(v, v') \ . \tag{6}$$

This function is defined with respect to underlying kernels k_{node} evaluating the similarity of two node labels and k_{edge} evaluating the similarity of two edge labels. In our experiments, we use standard RBF kernels for this purpose. Note that the adjacency matrix defined in Eq. 5 can be interpreted as a fuzzy adjacency matrix, where the adjacency value of two nodes of the product graph is high if the corresponding pairs of nodes and pairs of edges have similar labels, and low otherwise.

Plugging the adjacency matrix from Eq. 5 into the kernel function in Eq. 4, we obtain the modified kernel that can be applied to continuously labeled graphs [7].

3.3 Edit Distance Enhancement

As will be shown in the next section, the random walk kernel defined above is very powerful on certain datasets, but may perform poorly on other data compared to a standard edit distance based nearest-neighbor classifier. In the case of the random walk kernel, the similarity of graphs is defined by accumulating the similarity of local parts of the graphs. For certain graph representations, however, there are global matching constraints that need to be taken into account. In such a case it may be more appropriate to apply other graph matching methods, such as the one based on edit distance. Experiments confirm that random walk kernels and edit distance methods address the graph matching problem in complementary ways, and one approach usually performs significantly worse or better than the other one.

The main objective in this paper is to bring together the best from both worlds: The flexibility of graph edit distance and the power of random walk kernels. The basic idea is to enhance the random walk kernel with an edit distance matching at the global level. This allows us to integrate global information into the local random walk matching process. To this end, let us assume that an optimal edit path from g to g' has been computed, and let $S = \{v_1 \to v_1', v_2 \to v_2', \ldots\}$ denote the set of node substitutions present in the optimal path. We then proceed by defining the adjacency matrix of the direct product graph $(g \times g')$ by

$$[A_\times]_{(u,u'),(v,v')} = \begin{cases} k((u,u'),(v,v')) & \text{if } ((u,u'),(v,v')) \in E_\times \text{ and} \\ & u \to u' \in S \text{ and } v \to v' \in S \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{7}$$

In other words, we restrict the random walks to nodes that satisy the optimal node-to-node correspondences identified by the edit distance computation. This adjacency matrix is then used with the kernel function given in Eq. 4.

4 Experimental Results

In this section, we offer an evaluation of the proposed enhanced random walk kernel in comparison to two baseline systems. In the first baseline system, the edit distance of graphs is computed (see Sec. 2), and test graphs are classified according to the k most similar graphs from a labeled training set. In the second baseline system, the similarity of graphs is evaluated by means of the traditional random walk kernel (see Sec. 3.2), and an SVM is used for classification. The third system, our proposed method, is based on the enhanced random walk kernel defined in Sec. 3.3.

The first database consists of line drawings representing capital letters. To obtain a noisy sample set of letters, we iteratively apply distortions to clean letter prototypes. The distorted line drawings are then converted into graphs by

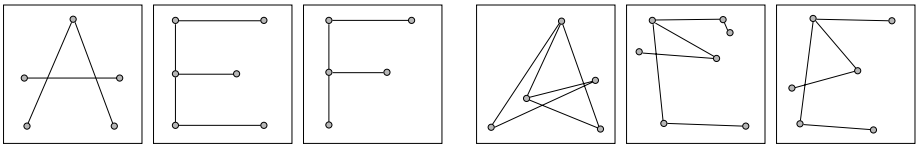

Fig. 1. Illustration of a) three clean letters and b) three distorted letters *A, E, F*

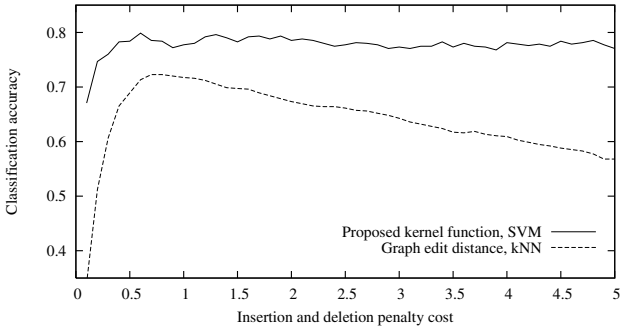

Fig. 2. Influence of insertion and deletion penalty cost on classification accuracy

representing end points of lines by nodes and lines by edges. Nodes are labeled with the two-dimensional position of the corresponding end point. Following this procedure, we construct a training set and validation set of size 150 each, and a test set of size 750. The database consists of 15 classes of letters (*A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z*). An illustration is provided in Fig. 1.

In a first experiment, we focus on the influence of edit costs on the classification accuracy. For this purpose we only consider the k-nearest-neighbor edit distance based classifier and the SVM with the edit distance enhanced kernel. The edit costs of node (or edge) insertions and deletions essentially determine how likely a node (edge) is to be substituted by another node (edge). If insertion and deletion costs are low, only a few inexpensive substitutions will occur in an optimal edit path. Conversely, if insertion and deletion costs are high, the edit distance algorithm will tend to substitute as many nodes (edges) as possible. For an edit distance based nearest-neighbor classifier, the resulting cost of an optimal edit path is crucial for the performance. It is therefore important to carefully adjust insertion and deletion costs, as well as any other edit cost parameter. In the case of the random walk kernel proposed in this paper, on the other hand, we are interested in promising node-to-node correspondences, rather than a particular distance value. This means that in the case of the proposed method there is no need for an extensive optimization of the edit cost parameters. This issue can very well be observed in Fig. 2, where the classification accuracy of an edit distance based nearest-neighbor classifier and an SVM based on the proposed kernel function is shown for various insertion and deletion penalty costs. As

a) b)

Fig. 3. Example images from the Lesaux database, a) *city* and b) *countryside*

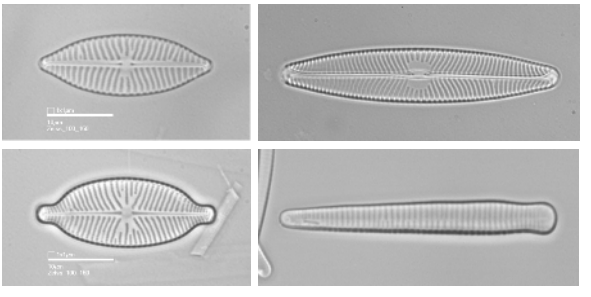

Fig. 4. Example images from the Diatom database (four different classes)

expected, the accuracy of the traditional edit distance classifier strongly depends on the actual edit costs, while the proposed method exhibits a roughly constant behavior for penalty costs above a certain threshold. It should also be noted that the proposed method clearly outperforms the nearest-neighbor classifier.

We next compare the classification accuracy of the two baseline classifiers with the proposed method. To this end, we classify graphs from the Letter database described above, from the Lesaux database, and the Diatom database. The Lesaux database [11] consists of graphs representing images from five classes (*city, countryside, people, snowy, streets*). Graphs are extracted from images by running a region segmentation process and removing those segments that are deemed irrelevant for classification. The remaining regions are then turned into a region adjacency graph with labels describing the dominant colors of the region. We use a training set, validation set, and test set of size 54 each. For two example images, see Fig. 3. The Diatom database [12] consists of 110 microscopic images of diatoms, evenly split into training set, validation set, and test set. The recognition task is to classify diatoms from the test set according to 22 classes. The images are represented by attributed region adjacency graphs. Four example diatom images from different classes are shown in Fig. 4. The various parameters of the classifiers (such as edit cost parameters and weighting factor λ) are first optimized on the validation set and then applied to the independent test set.

The classification accuracy of the three methods under consideration determined on the independent test set is given in Table 1. There are two entries in

Table 1. Comparison of classification accuracy

	Letter database	Lesaux database	Diatom database
Edit distance, kNN	69.3	48.2*	63.9*
Random walk, SVM	75.7*	33.3	44.4
Proposed, SVM	74.7*	51.9*	58.3*

* Marked classification rates do not differ significantly. Unmarked classification rates are significantly lower than marked ones ($\alpha = 0.05$).

each column of this table marked with an asterisk. These two entries are, in each column, not significantly different from each other (on a statistical significance level of $\alpha = 0.05$). However, the unmarked entry in each column is significantly smaller than the two marked ones. It can clearly be observed that the two traditional methods — the edit distance based k-nearest-neighbor classifier and the standard random walk kernel — perform quite different on all datasets. One of the two methods is always significantly better than the other one. The proposed random walk kernel enhanced by edit distance information, on the other hand, performs as good as the better method throughout our experiments. That is, while the traditional edit distance method and random walk kernel method emphasize a certain aspect of the graph matching problem, the proposed kernel function combines the information in an advantageous manner. By applying the method proposed in this paper, we obtain a robust classifier that succeeds well on all tested datasets without recourse to the characteristics of the underlying graphs. Our method can be regarded as an extension to the standard random walk kernel that leads to a statistically significant improvement of the graph matching performance on the Lesaux database and the Diatom database.

5 Conclusions

In this paper, we propose an extension of a standard random walk kernel for graphs. It can be observed, on graphs extracted from real-world data, that random walk kernels offer an interesting alternative to traditional edit distance based graph classifiers in the sense that they address the graph matching problem in a different way. One some datasets, the edit distance measure is the most suitable method for graph matching; on other datasets, edit distance is outperformed by random walk kernels and Support Vector Machines. The method we propose is based on the idea that it is advantageous to include graph matching information from the global level in the random walk kernel defined locally based on the similarity of walks in graphs. By constraining the random walk kernel to pairs of nodes that satisfy the global node-to-node correspondence, instead of any pairs of nodes, we obtain a system that combines the flexibility of graph edit distance with the classification power of the random walk kernel. The proposed kernel offers a classification accuracy that is at least as good as the better one of the two baseline methods — graph edit distance and standard random walk kernels — and significantly better than the other one. The performance

is evaluated on a semi-artificial line drawing dataset and two real-world image datasets.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence **18** (2004) 265–298
2. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
3. Watkins, C.: Dynamic alignment kernels. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds.: Advances in Large Margin Classifiers. MIT Press (2000) 39–50
4. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz (1999)
5. Jain, B., Geibel, P., Wysotzki, F.: SVM learning with the Schur-Hadamard inner product for graphs. Neurocomputing **64** (2005) 93–105
6. Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In Schölkopf, B., Warmuth, M., eds.: Proc. of the 16th Annual Conf. on Learning Theory. (2003) 129–143
7. Borgwardt, K., Ong, C., Schönauer, S., Vishwanathan, S., Smola, A., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics **21** (2005) 47–56
8. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics (Part B) **13** (1983) 353–363
9. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1** (1983) 245–253
10. Neuhaus, M., Bunke, H.: An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In: Proc. 10th Int. Workshop on Structural and Syntactic Pattern Recognition. LNCS 3138, Springer (2004) 180–189
11. Le Saux, B., Bunke, H.: Feature selection for graph-based image classifiers. In: Proc. 2nd Iberian Conf. on Pattern Recognition and Image Analysis. LNCS 3523, Springer (2005) 147–154
12. Ambauen, R., Fischer, S., Bunke, H.: Graph edit distance with node splitting and merging and its application to diatom identification. In Hancock, E., Vento, M., eds.: Proc. 4th Int. Workshop on Graph Based Representations in Pattern Recognition. LNCS 2726, Springer (2003) 95–106