SSPR 2008

GRAPH CLASSIFICATION BASED ON DISSIMILARITY SPACE EMBEDDING

Horst Bunke and Kaspar Riesen
{bunke,riesen}@iam.unibe.ch
Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, CH-3012 Bern (Switzerland)
http://www.iam.unibe.ch/∼fki/

Acknowledgments

- Michel Neuhaus
- Swiss National Science Foundation
- Swiss National Science Foundation ’National Centers of Competence in Research’ Program
- Bob Duin, Ela Pekalska

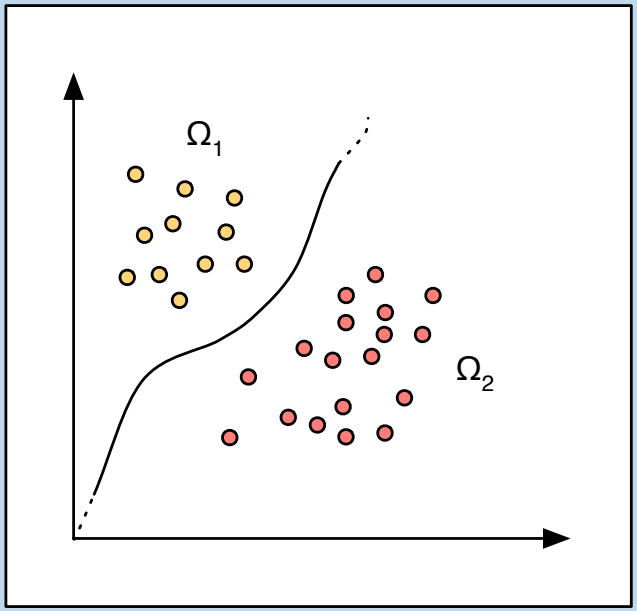
Contents

- Introduction
- Graph Embedding
- Dissimilarity Space Embedding of Graphs
- Multiple Classifier Systems
- Further Refinements
- Efficient Graph Edit Distance
- Summary and Conclusions

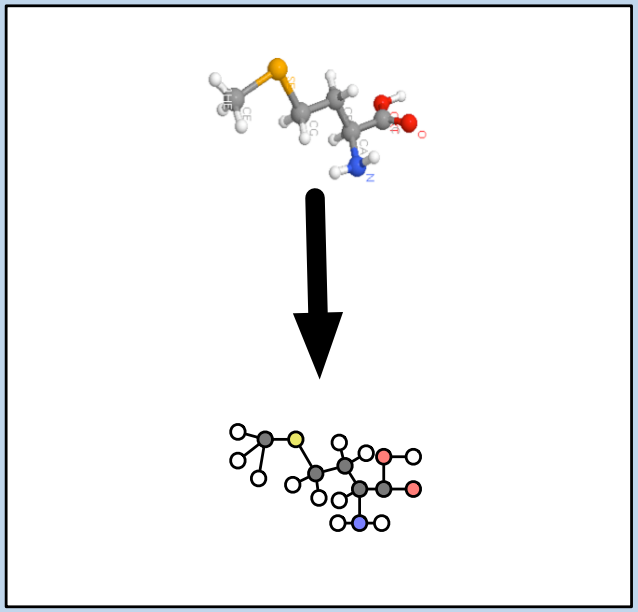
Introduction

- Traditional subdivision of Pattern Recognition:

Statistical approach



Structural approach (graphs)



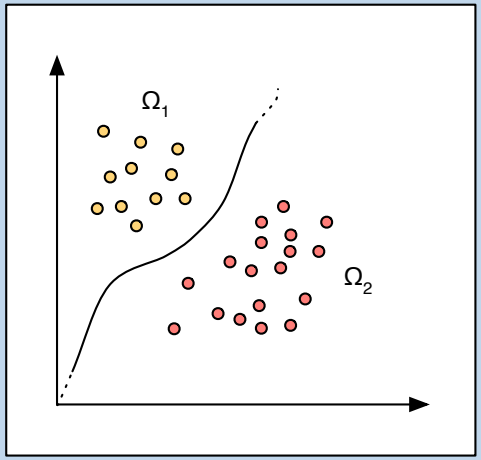
Statistical Approach

Advantages:

- Theoretically well founded
- Many powerful algorithms available

Disadvantages:

- Dimension of feature vectors fixed
- Only unary feature values, but no relations can be modeled



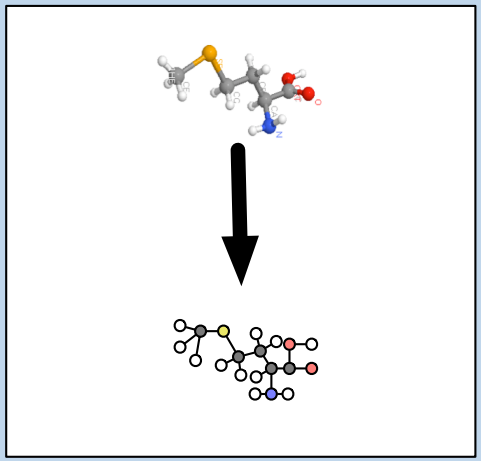
Structural Approach

Advantages:

- Representation size is variable
- Higher representational power (structural relationships)

Disadvantages:

- Lack of mathematical structure in the graph domain
- Lack of algorithmic tools



Vectors vs. Graphs

	vectors	graphs
representational power	-	+
available tools	+	-

Vectors vs. Graphs

	vectors	graphs
representational power	-	+
available tools	+	+

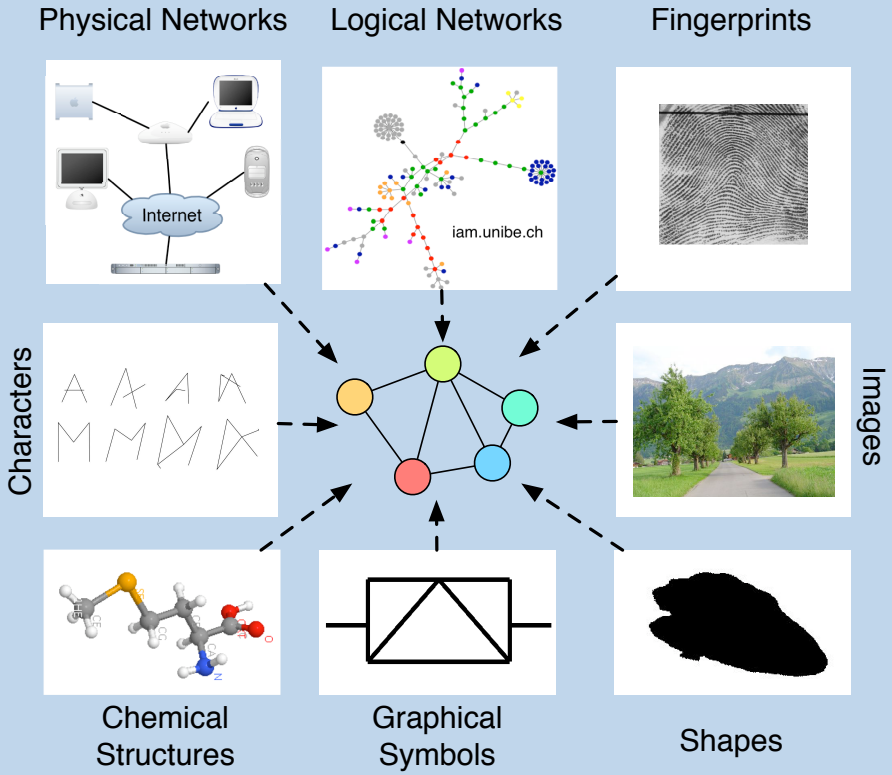
- Embedding graphs in vector spaces makes available all algorithmic tools developed for feature based object representations
- Close relationship to graph kernels

Contents

- Introduction
- **Graph Embedding**
- Dissimilarity Space Embedding of Graphs
- Multiple Classifier Systems
- Further Refinements
- Efficient Graph Edit Distance
- Summary and Conclusions

Graph Based Representation

- A graph is defined by $g = (V, E, \mu, \nu)$, where
  - V is the finite set of nodes
  - $E \subseteq V \times V$ is the set of edges
  - $\mu : V \rightarrow L$ is the node labeling function
  - $\nu : E \rightarrow L$ is the edge labeling function



Graph Embedding

- Definition: Let $\mathcal{G}$ be a set of graphs. A **graph embedding** is a function

$$\varphi : \mathcal{G} \rightarrow \mathbb{R}^n$$

mapping graphs to n-dimensional vectors, i.e.,

$$\varphi(g) = (x_1, \ldots, x_n)'$$

- Previous work in graph embedding:
  - Spectral methods [Kosinov, Caelli, 2004], [Luo, Wilson and Hancock, 2005], [Shokoufandeh et al. 2005]
  - Heat kernel [Bai and Hancock, 2005]
  - Riemannian approach [Robles-Kelly and Hancock, 2007]
  - Quantum commute times [Qiu and Hancock, 2007]

Graph Embedding

Characteristics of previous works:

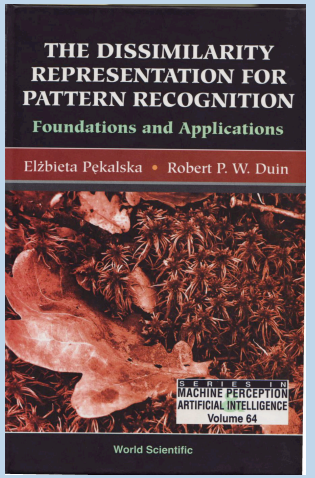
- Solid methodological foundation
- Often restricted to special classes of graphs
- Robustness to noise may be an issue

Contribution of this work:

- Applicable to any type of graphs
- Robust against various types of distortions, due to the use of graph edit distance

Background

- Method described in this paper is inspired by dissimilarity space embedding proposed by Duin and Pekalska
- Power of this approach mainly demonstrated on feature vector representations, although much more generally applicable
- Later investigated for string representations
- In this paper extension to graphs



Contents

- Introduction
- Graph Embedding
- **Dissimilarity Space Embedding of Graphs**
- Multiple Classifier Systems
- Further Refinements
- Efficient Graph Edit Distance
- Summary and Conclusions

Graph Edit Distance

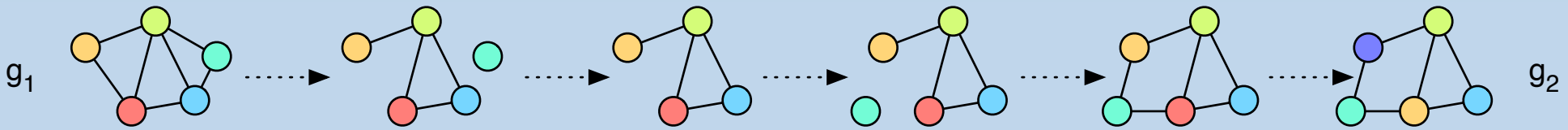
Define the dissimilarity of graphs by the minimum amount of distortion that is needed to transform one graph into another

- **Distortions** e_i**:** deletions, insertions, substitutions of nodes and edges
- **Edit path** $S = e_1, \ldots, e_n$**:** a sequence of edit operations that transform g_1 into g_2
- **Cost function:** Measuring the strength of a given distortion
- **Edit distance** $d(g_1, g_2)$**:** Minimum cost edit path between two graphs

Graph Edit Distance

Define the dissimilarity of graphs by the minimum amount of distortion that is needed to transform one graph into another

- **Distortions** e_i**:** deletions, insertions, substitutions of nodes and edges
- **Edit path** $S = e_1, \ldots, e_n$**:** a sequence of edit operations that transform g_1 into g_2
- **Cost function:** Measuring the strength of a given distortion
- **Edit distance** $d(g_1, g_2)$**:** Minimum cost edit path between two graphs



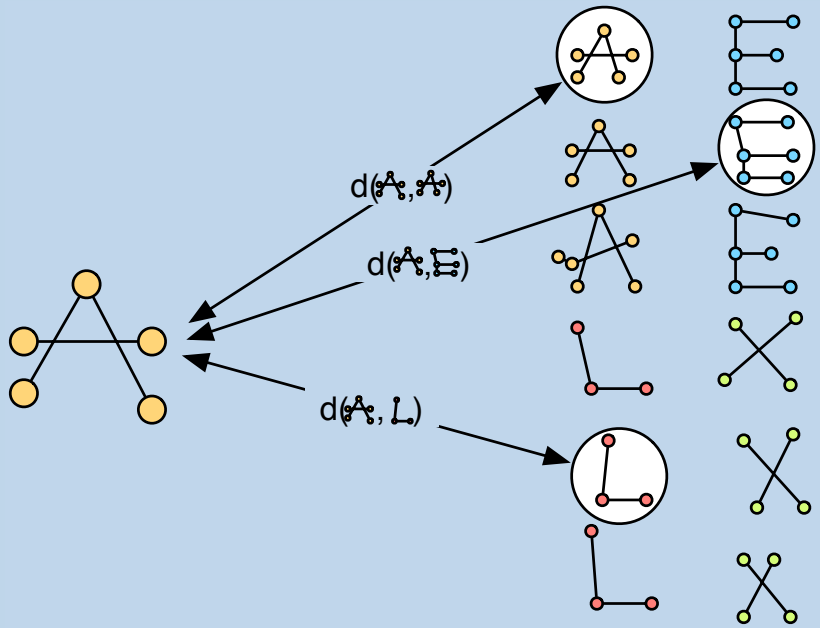
Graph Embedding

- Graph set: $\mathcal{G} = \{g_1, \ldots, g_t\}$
- Graph edit distance: $d(g_i, g_j)$
- Prototype set: $P = \{p_1, \ldots, p_n\}$
- The mapping

$$\varphi_n^P : \mathcal{G} \rightarrow \mathbb{R}^n$$

is defined as the function

$$\varphi_n^P(g) \mapsto (d(g, p_1), \ldots, d(g, p_n))$$



Distance in the Embedding Space

$$
\begin{aligned}
||\varphi_n^P(g) - \varphi_n^P(g')|| &= \langle \varphi_n^P(g), \varphi_n^P(g) \rangle + \langle \varphi_n^P(g'), \varphi_n^P(g') \rangle - 2\langle \varphi_n^P(g), \varphi_n^P(g') \rangle \\
&= \sum_{i=1}^{n} d(g, p_i)^2 + \sum_{i=1}^{n} d(g', p_i)^2 - 2\sum_{i=1}^{n} d(g, p_i) d(g', p_i) \\
&= \sum_{i=1}^{n} (d(g, p_i) - d(g', p_i))^2
\end{aligned}
$$

- The Euclidean distance of a pair of graphs g and g' in the vector space is equal to the sum of squared differences between the edit distances of g and g' to the prototype graphs

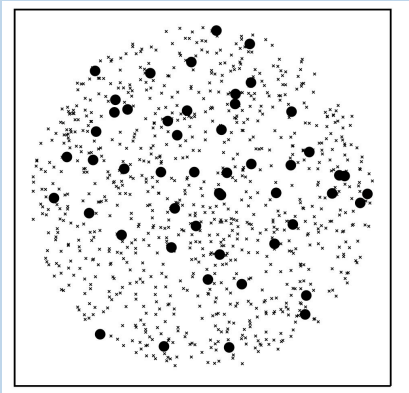
Prototype Selectors

UNIVERSITÄT BERN

- Problem: Appropriate choice of the prototype set $P = \{p_1, \ldots, p_n\}$ that leads to a good performance of the resulting classifier in the vector space
- We distinguish between **class-wise** and **class-independent** selection:
  - class-independent: Selection is executed over the whole graph set to get n prototypes
  - class-wise: Selection is performed individually for each of the k different classes (l_i prototypes per class c_i such that $\sum_{i=1}^{k} l_i = n$)

Random Prototype Selector (*rps*)

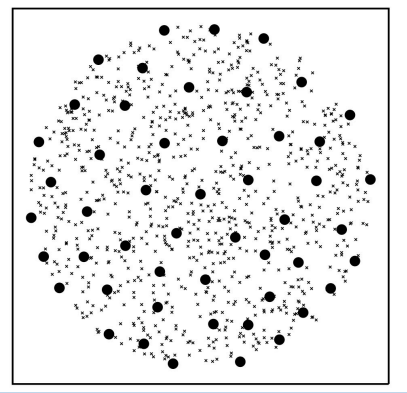
- Selects randomly n prototypes from the graph set $\mathcal{G}$



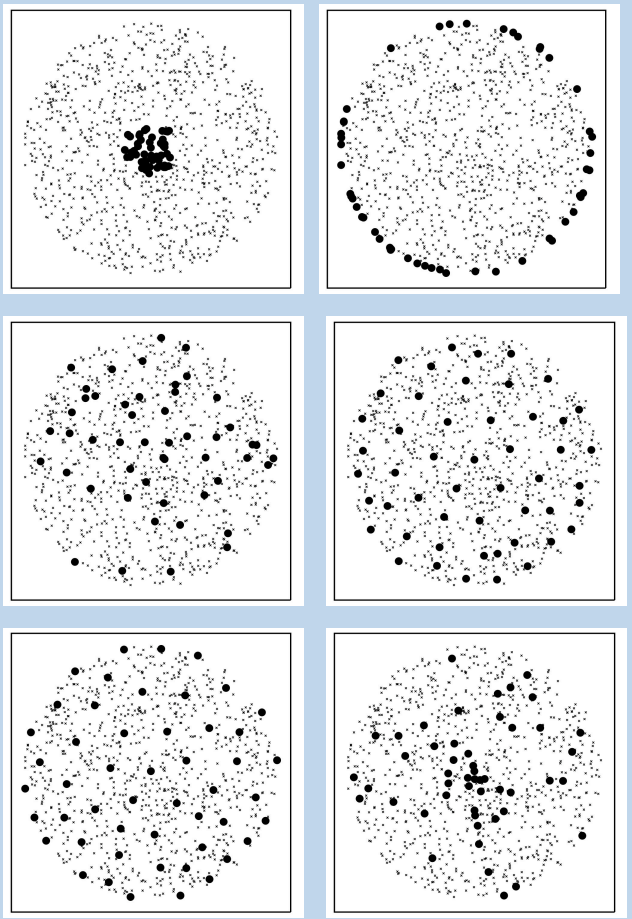
Spanning Prototype Selector (*sps*)

- The first prototype selected is the set median graph of $\mathcal{G}$
- Each additional prototype selected by the spanning prototype selector is the graph the furthest away from the already selected prototype graphs

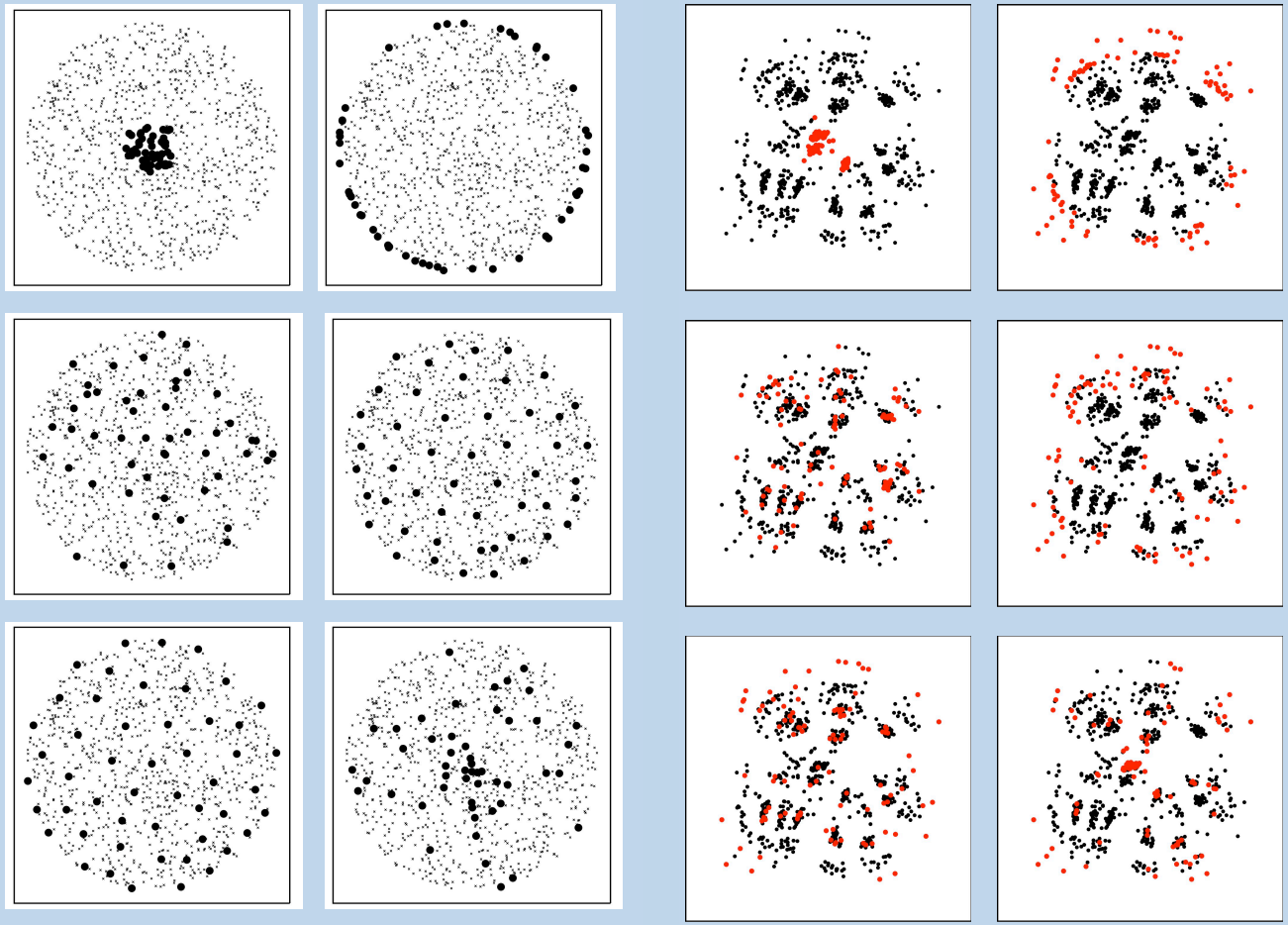
$$P_i = \begin{cases} median(\mathcal{G}) & \text{if } i = 1 \\ P_{i-1} \cup \{p_i\} & \text{if } 1 < i \leqslant n \text{ , where } p_i = \underset{g \in \mathcal{G} \setminus P_{i-1}}{\operatorname{argmax}} \min_{p \in P_{i-1}} d(g,p) \end{cases}$$



Prototype Selection



Prototype Selection on Real Data

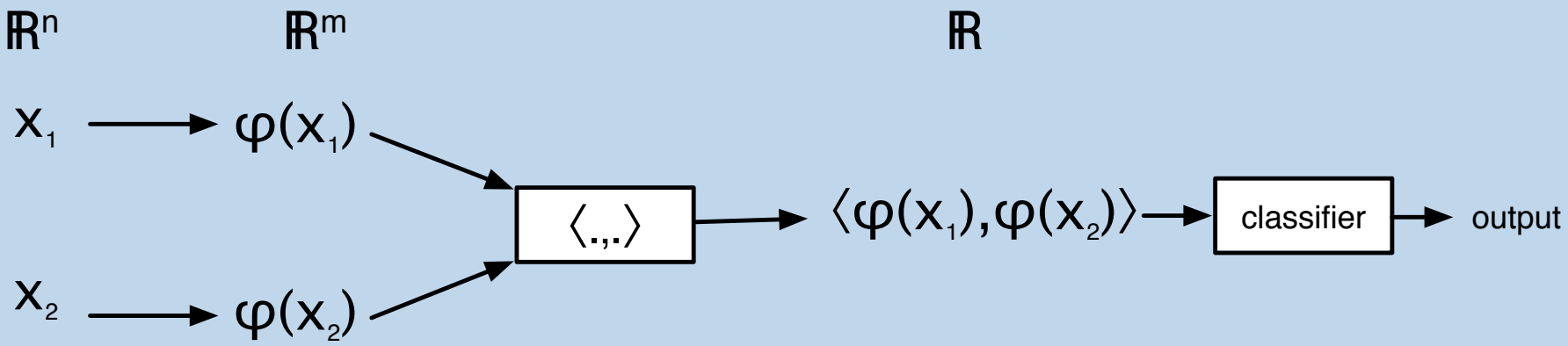


Relation to Graph Kernels

- Kernel methods have emerged as a promising new research direction
- Kernel methods have been known for long, but their potential for intelligent data analysis has been discovered only about 10 years ago
- Originally kernel methods have been developed for feature vectors
- Recently it has been demonstrated that they are applicable to graphs as well
- Graph kernel methods make available a large spectrum of methods from computational intelligence, pattern recognition, machine learning, etc.

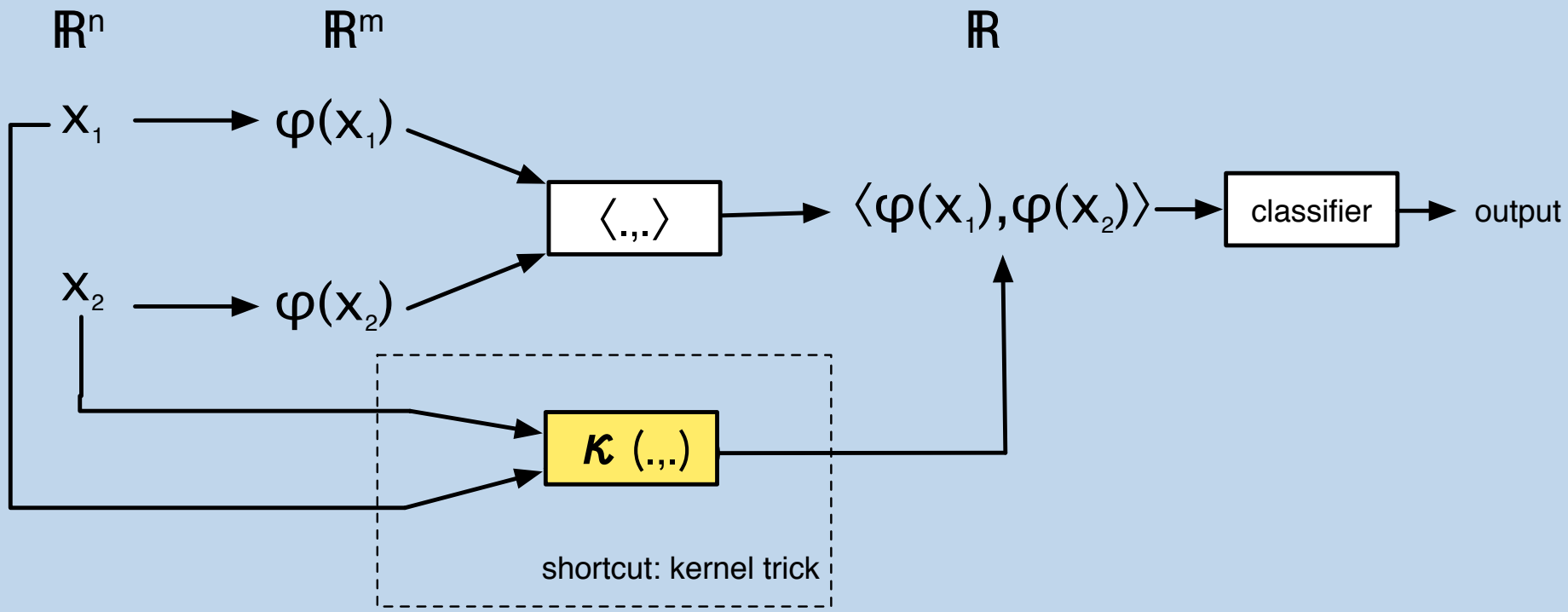
Kernel Trick

- Illustration of the *kernel trick*:



Kernel Trick

- Illustration of the *kernel trick*:



Graph Kernel

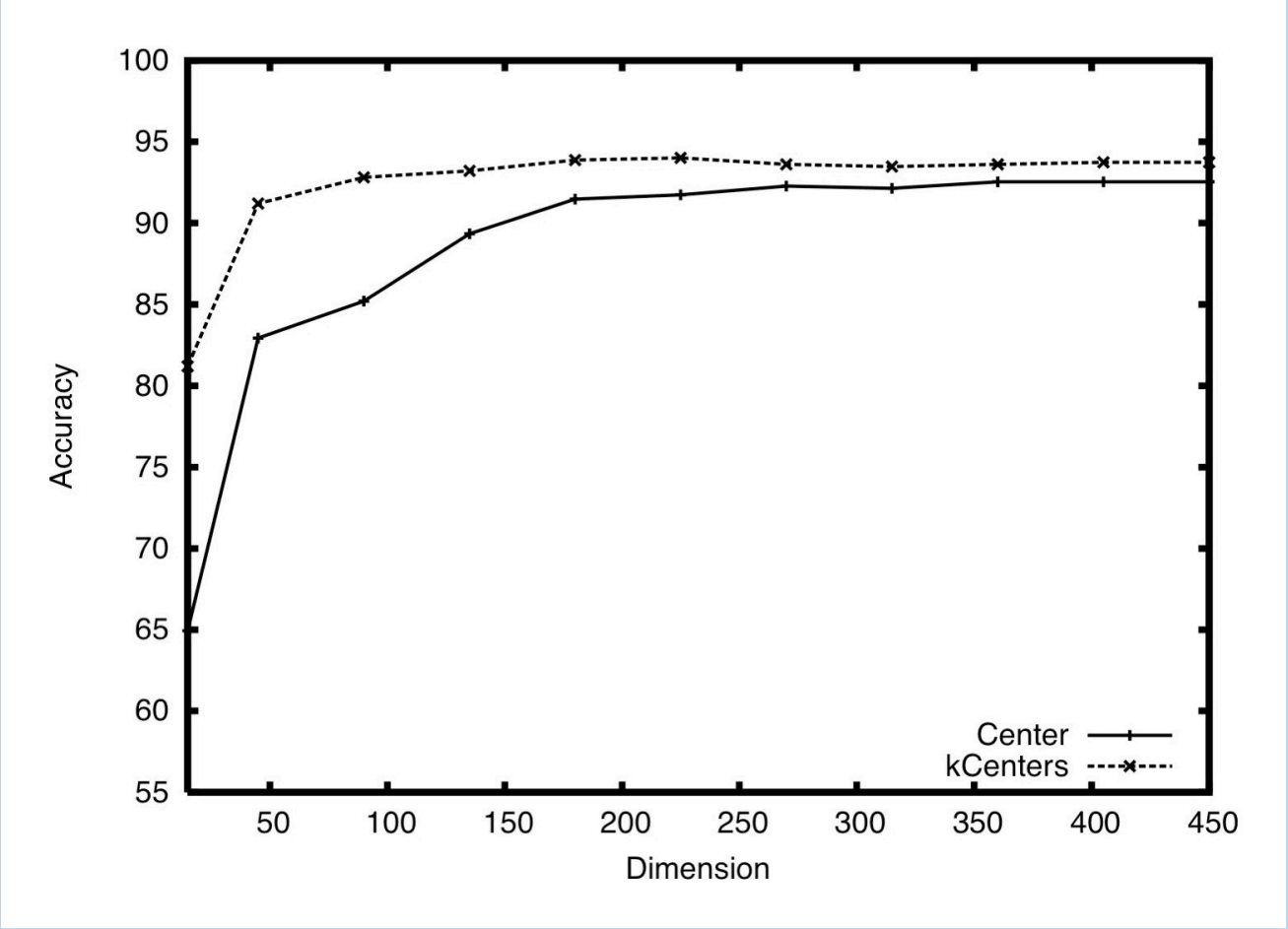
- **Definition**: Let $g, g' \in \mathcal{G}$ be graphs and $\varphi : \mathcal{G} \to \mathbb{R}^n$ a function with $n \in \mathbb{N}$. A graph kernel function is a mapping $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that $\kappa(g, g') = \langle \varphi(g), \varphi(g') \rangle$. □
- If a classifier can be kernelized, it can be run with scalar products resulting from graphs
- Consequently, all kernel machines can be applied to graphs
- The proposed graph embedding is even more general: it makes not only all kernel machines, but all algorithms from statistical pattern recognition available for graphs

Experimental Evaluation

UNIVERSITÄT BERN

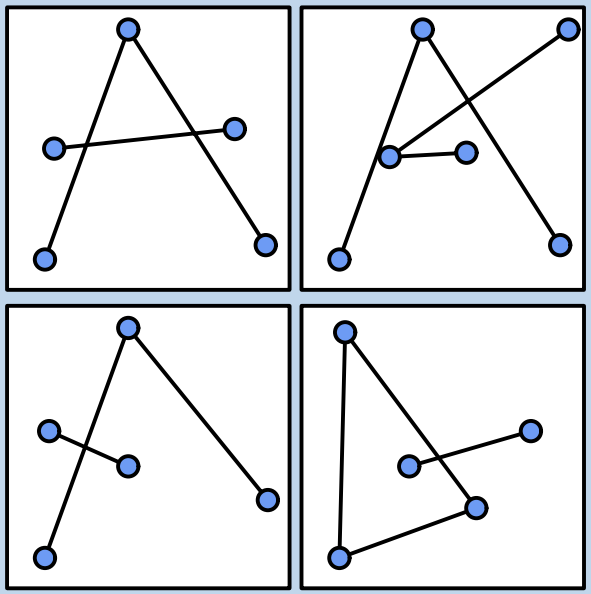
- Aim: compare classifiers in the embedding space with classifiers in the graph domain
- Use data sets with a wide range of different characteristics (number of classes, size of graphs, type of labels, . . .)
- Reference classifiers:
  - k-NN classifier in graph domain (k-NN)
  - SVM on graph edit distances: $\kappa(g, g') = -d(g, g')^2$ (Sim)
- Classifier in the vector space:
  - SVM
- Meta parameters of the vector space embedding optimized on a validation set: number of prototypes, prototype selection procedure

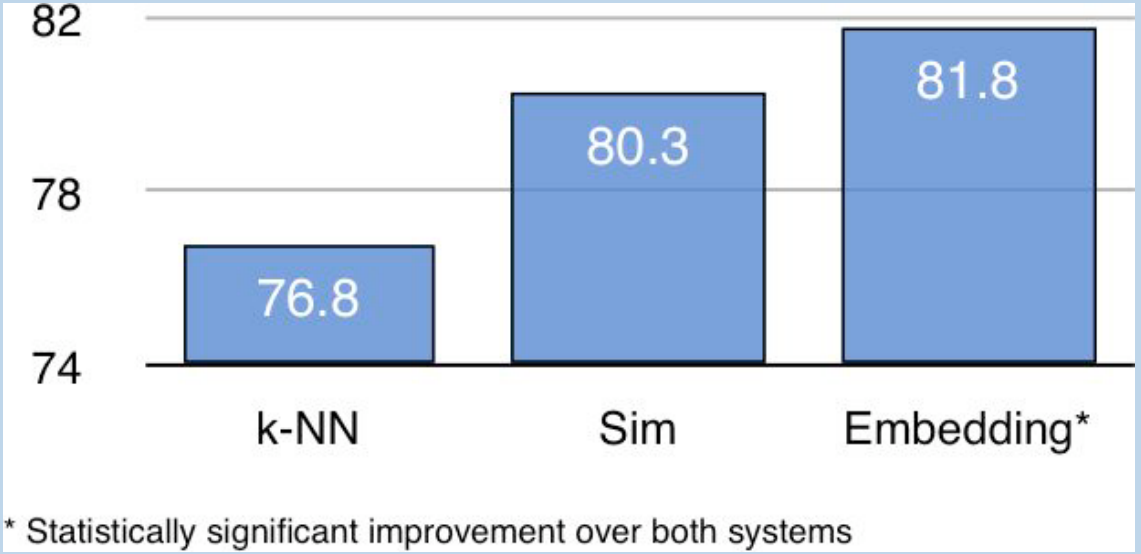
Validation Results (Letter Data Set)



Letter Data Set

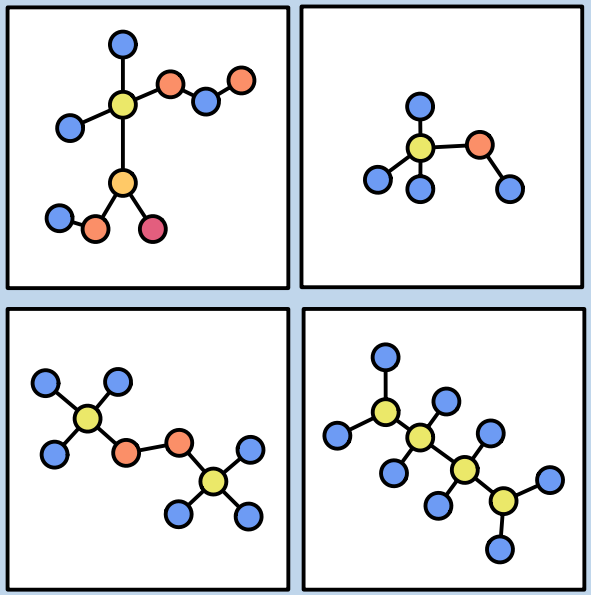
- Graphs representing capital letter line drawings, 15 classes, tr 750, va 750, te 750

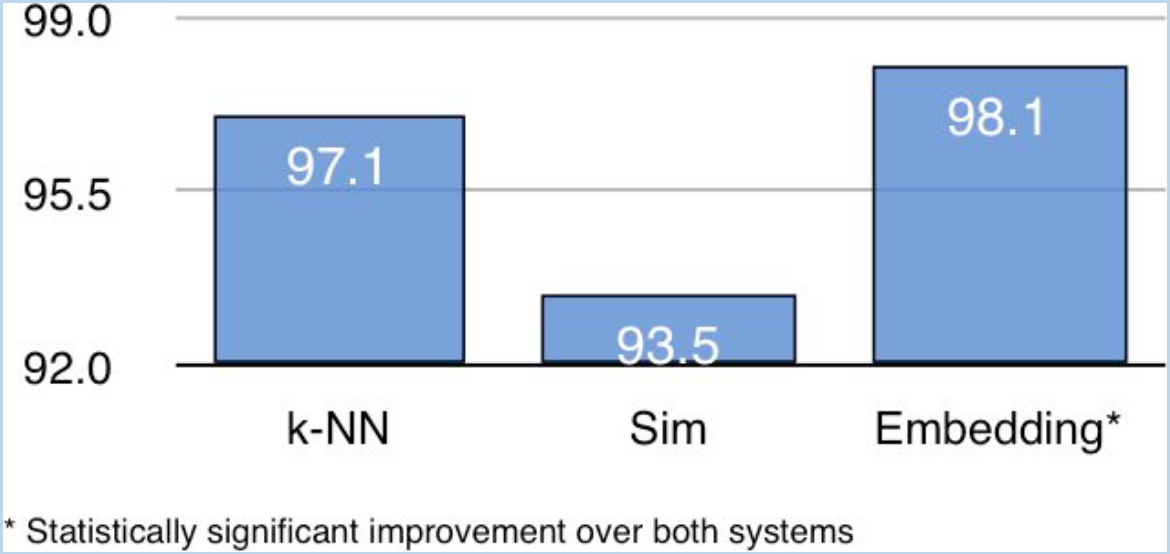




Molecule Data Set

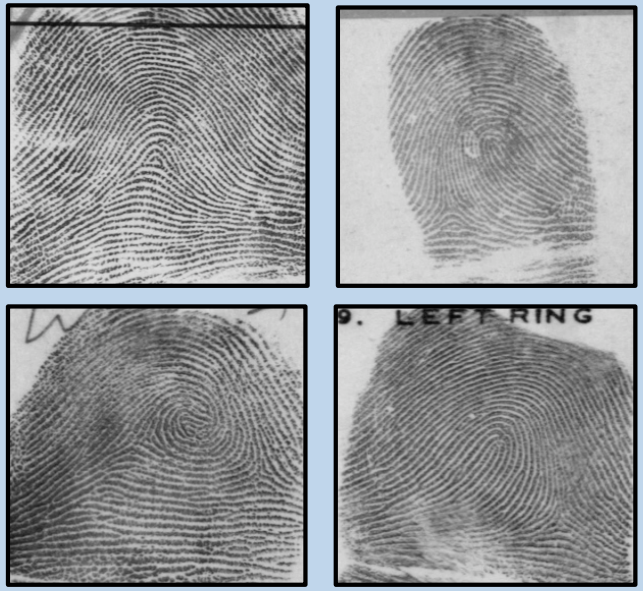
- Graphs representing molecules, 2 classes (active and inactive), tr 250, va 250, te 1500

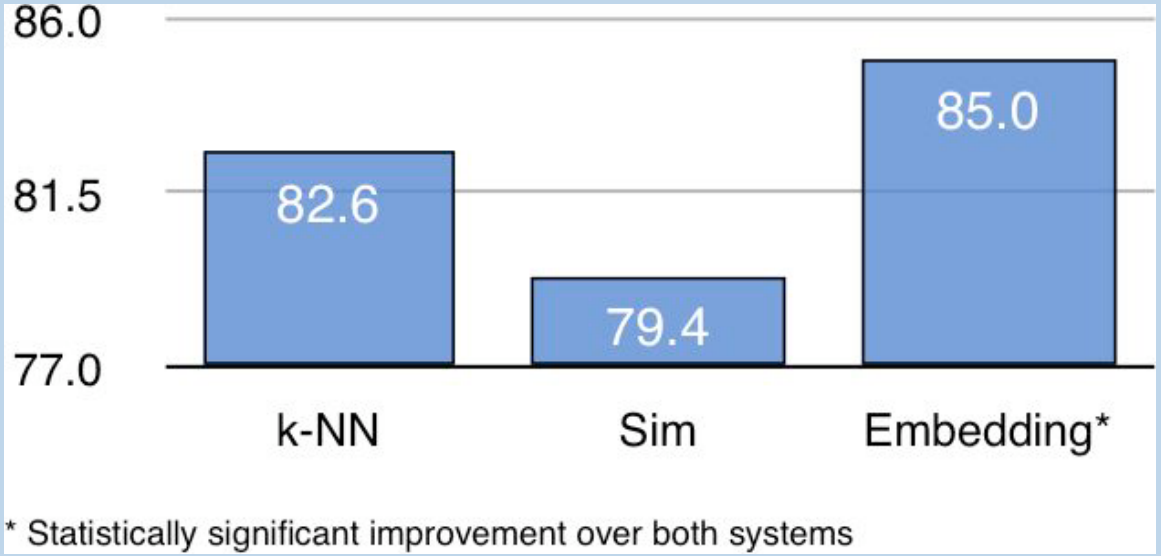




Fingerprint Data Set

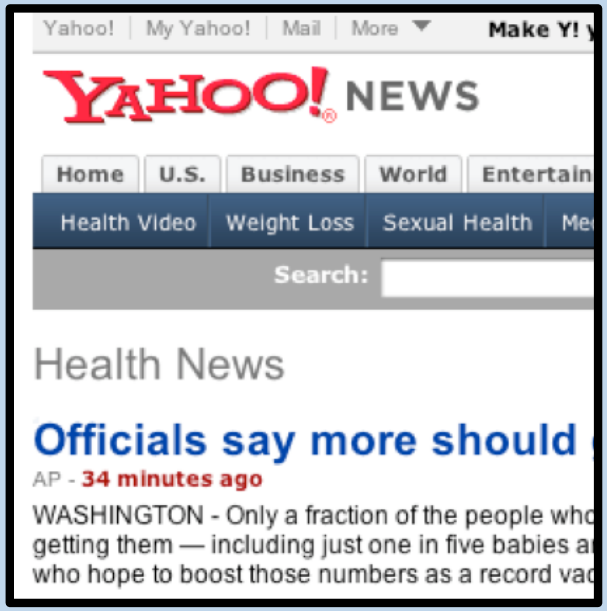
- Graphs representing fingerprint images, 4 classes (arch, left loop, right loop, whorl), tr 500, va 300, te 500

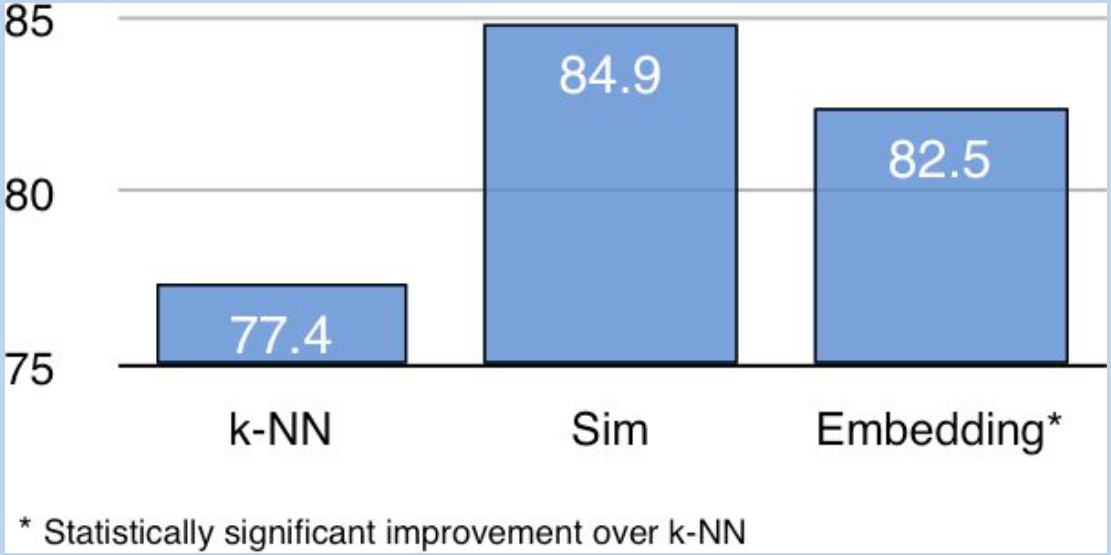




Webpage Data Set

- Graphs representing webpages, 22 classes (Business, Health, Politics, . . .), tr 780, va 780, te 780





Additional Data Sets

- Digits
- GREC
- COIL
- Protein
- Mutagenicity

- See poster presentation and `www.iam.unibe.ch/fki/databases/iam-graph-database`

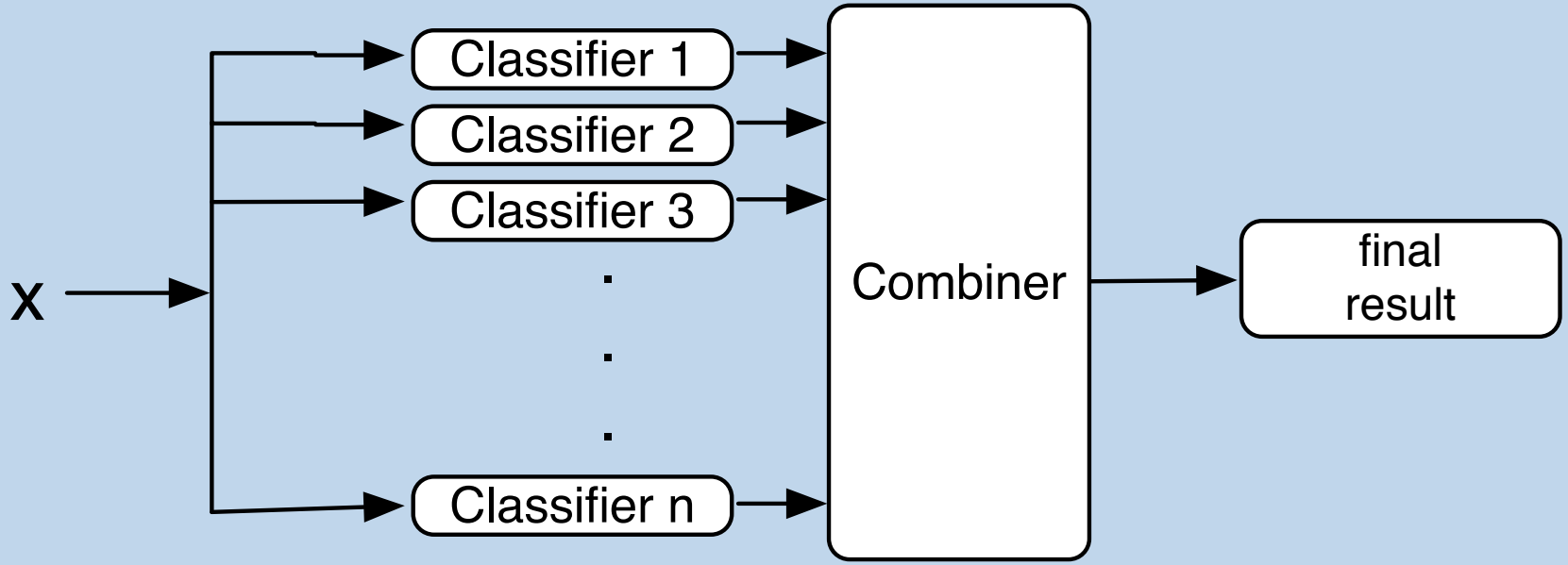
Contents

- Introduction
- Graph Embedding
- Dissimilarity Space Embedding of Graphs
- **Multiple Classifier Systems**
- Further Refinements
- Efficient Graph Edit Distance
- Summary and Conclusions

Multiple Classifier Systems

- Multiple Classifier Systems (MCS), also called Ensemble Methods, have become a focus of intensive research
- They are based on the idea that the errors committed by one classifier can be corrected by the other ensemble members
- Many methods for building MCS have been proposed recently
- However, the majority of work is based on feature vector representations; there is almost no work on MCS for graph based representations

Multiple Classifier Systems



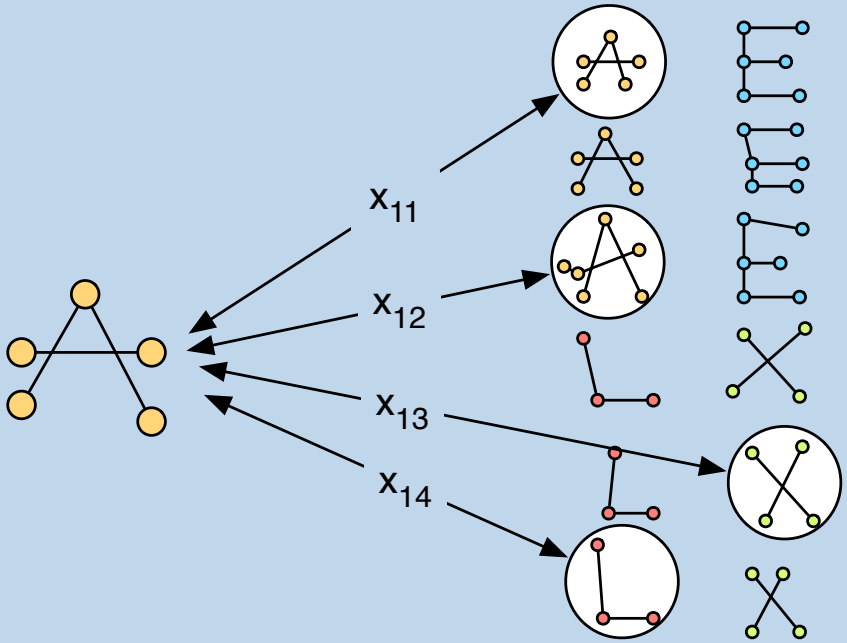
Standard Methods for MCS Creation

- Bagging (change the training set)
- Boosting (change the training set)
- Random feature subspace (change the features)
- Architecture variation (change the classifier)

Ensemble Generation

- Use random prototype selection
- Train classifier using the whole training set
- Repeat random prototype selection until desired number of classifiers is obtained

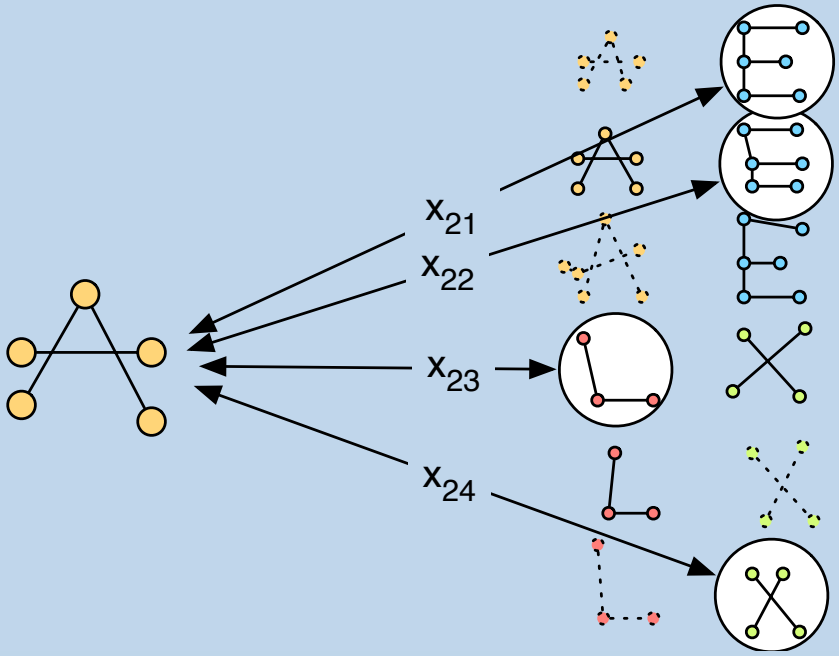
Randomized Graph Embedding



$v_1 = (x_{11}, x_{12}, x_{13}, x_{14})$

$$\varphi_n^{\mathcal{P}}(g) \mapsto (d(g, p_1), \ldots, d(g, p_n))$$

Randomized Graph Embedding

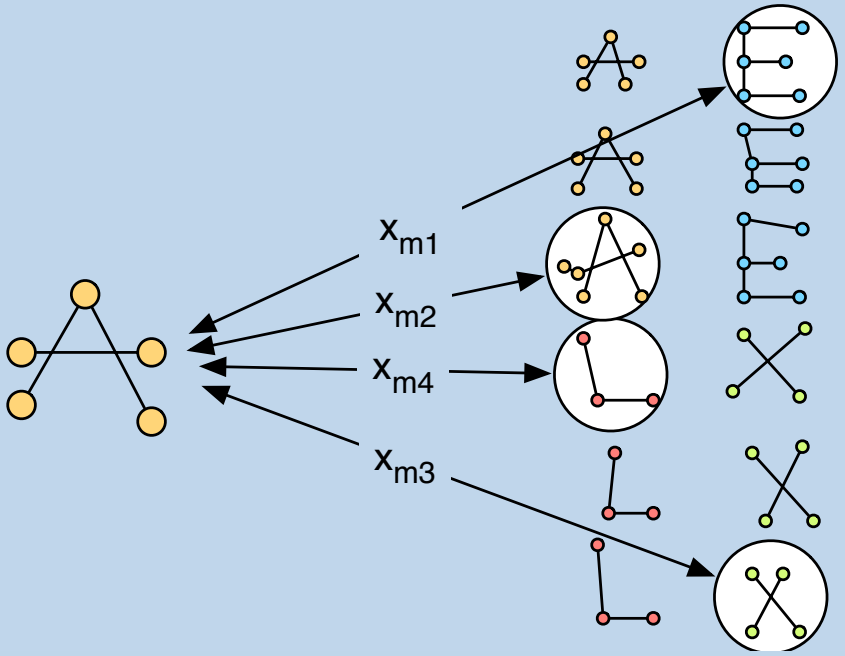


$v_1 = (x_{11}, x_{12}, x_{13}, x_{14})$

$v_2 = (x_{21}, x_{22}, x_{23}, x_{24})$

The selected graphs become temporarily unavailable for further selection

Randomized Graph Embedding



$v_1 = (x_{11}, x_{12}, x_{13}, x_{14})$

$v_2 = (x_{21}, x_{22}, x_{23}, x_{24})$

.
.
.

$v_m = (x_{m1}, x_{m2}, x_{m3}, x_{m4})$

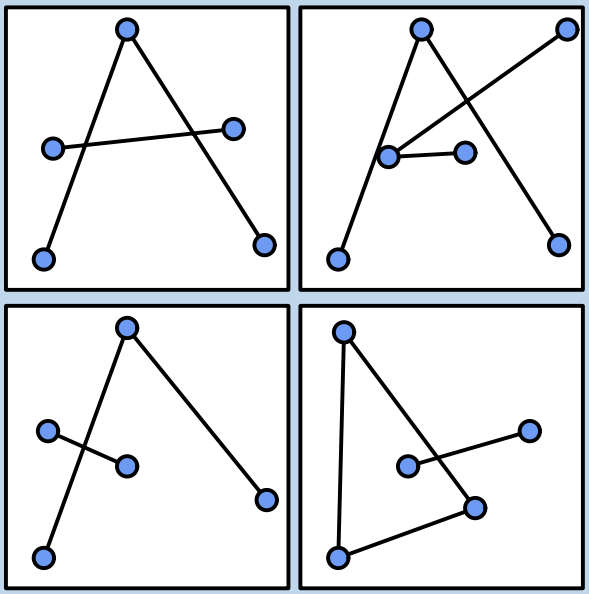
Applying randomized graph embedding m times leads to m different vectorial descriptions of the same underlying graph

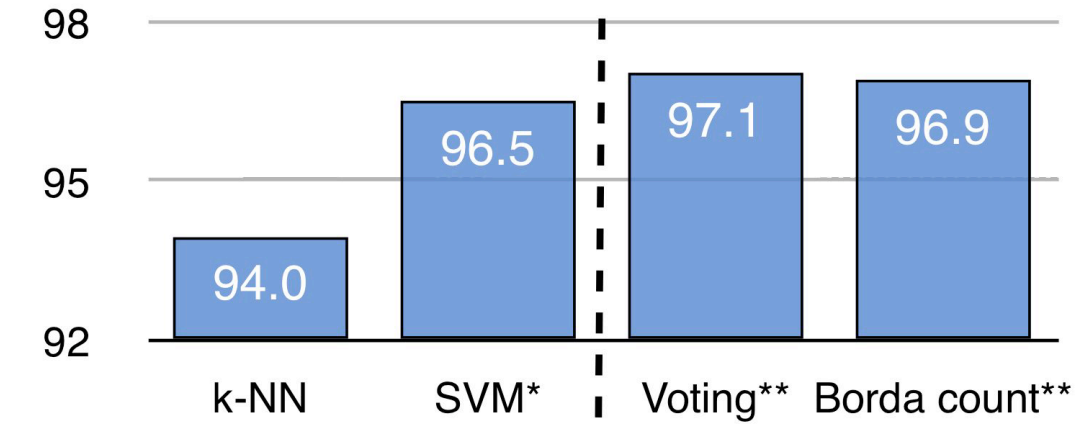
Experimental Evaluation

- Aim: compare classifiers in the embedding space with classifiers in the graph domain; compare the best individual SVM with the ensemble
- Combination rules: voting and Borda count

Letter Dataset

- Graph representing capital letter line drawings, 15 classes, tr 750, va 750, te 750



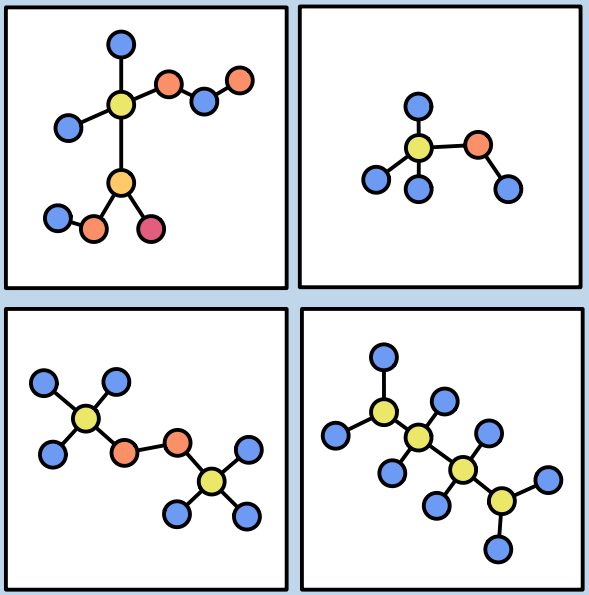


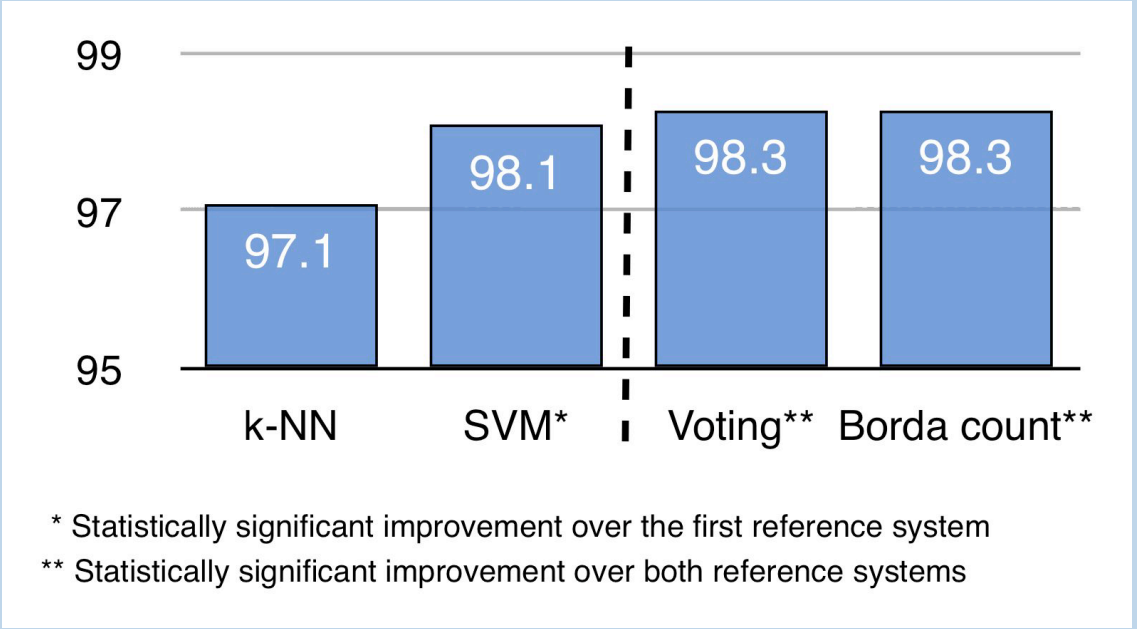
* Statistically significant improvement over the first reference system

** Statistically significant improvement over both reference systems

Molecule Dataset

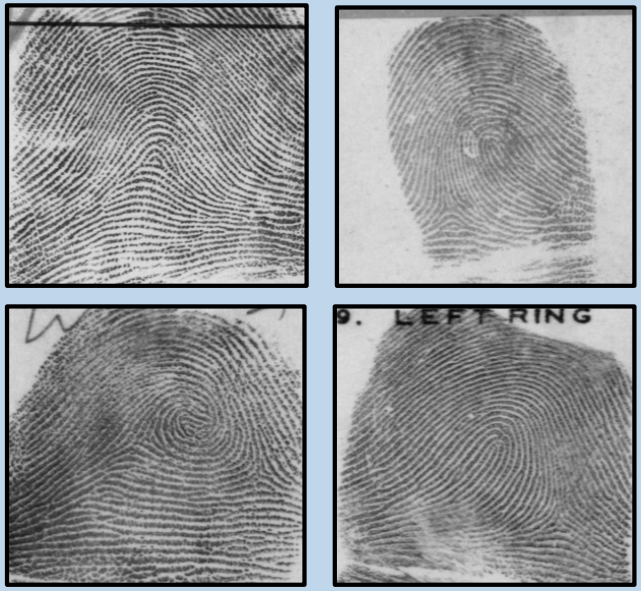
- Graphs representing molecules, 2 classes (active and inactive), tr 250, va 250, te 1500

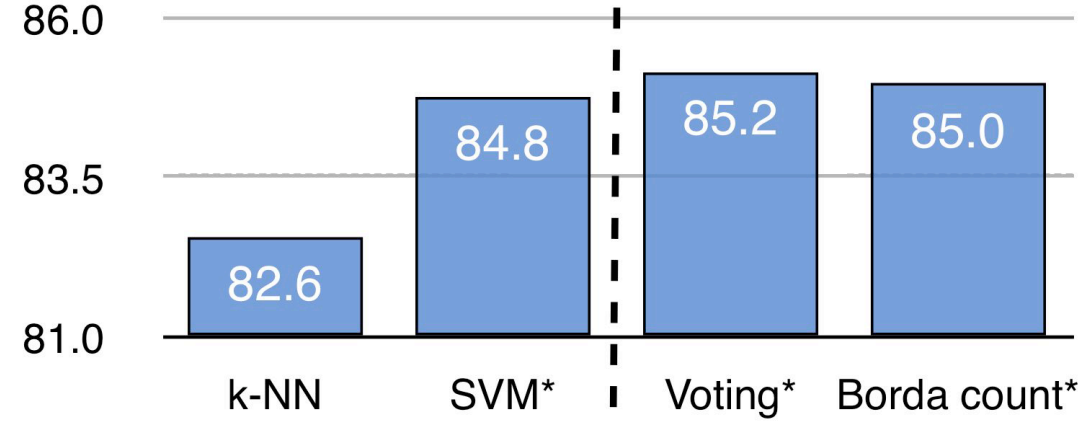




Fingerprint Dataset

- Graphs representing fingerprint images, 4 classes (arch, left loop, right loop, whorl), tr 500, va 300, te 500



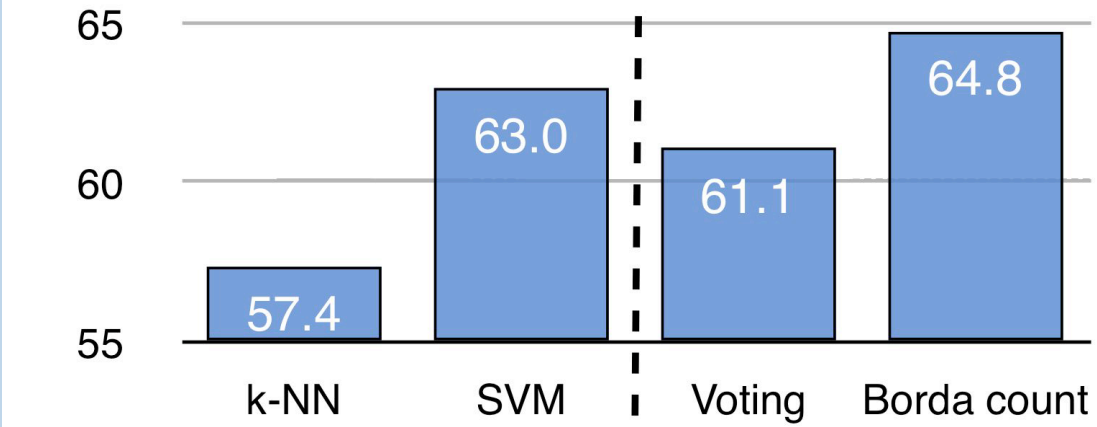


* Statistically significant improvement over the first reference system

** Statistically significant improvement over both reference systems

Image Dataset

- Graphs representing images, 5 classes (city, countryside, people, snowy, streets), tr 54, va 54, te 54



* Statistically significant improvement over the first reference system

** Statistically significant improvement over both reference systems

Contents

- Introduction
- Graph Embedding
- Dissimilarity Space Embedding of Graphs
- Multiple Classifier Systems
- **Further Refinements**
- Efficient Graph Edit Distance
- Summary and Conclusions

Further Refinements

- Lipschitz embedding:
  Instead of individual prototypes $p_1, \ldots, p_n$, use sets $P_1, \ldots, P_n$ of prototypes and define the embedding as

$$\varphi_n^{P_1,\ldots,P_n}(g) = (d(g, P_1), \ldots, d(g, P_n))$$

- PCA and kernel PCA:
  Use all available graphs for the embedding and apply some dimensionality reduction method; alternatively, the dimensionality reduction problem can be viewed as a feature selection problem
- Prototype reduction methods for nearest neighbor classification:
  Use editing, condensing, and similar procedures

Further Refinements 2/2

- These methods have the potential of further improvements over the basic methods discussed so far
- Graph clustering:
  Map graphs to a vector space and apply clustering methods in the vector space rather than in the original graph domain

Contents

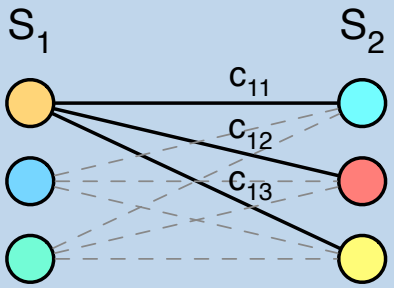
- Introduction
- Graph Embedding
- Dissimilarity Space Embedding of Graphs
- Multiple Classifier Systems
- Further Refinements
- **Efficient Graph Edit Distance**
- Summary and Conclusions

Complexity of GED

- The novel graph embedding methods depend on the graph edit distance
- However, the complexity of graph edit distance computation is exponential
- Possible solutions:
  - Restriction to special classes of graphs, e.g., graphs with unique node labels; complexity becomes linear; applications in web text mining and computer network analysis
  - **Use of approximate (suboptimal) methods**
- In our case:
  - Suboptimal versions of the A* algorithm
  - **Bipartite graph matching**

Bipartite Graph Matching

- Bipartite graph matching, also known as the Hungarian method or Munkres' algorithm, provides us with a method to solve the assignment problem



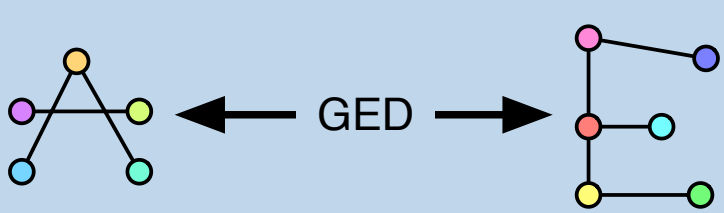
optimization problem:

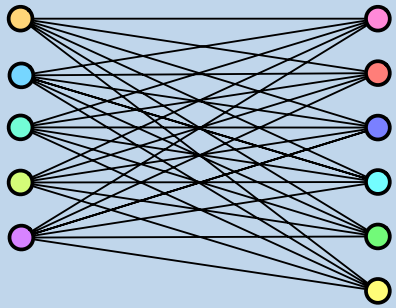
$$\sum_{i=1}^{n} c_{i\pi_i} \stackrel{!}{=} \min_{\pi}$$

- Typical applications in operations research (assignment of tasks to machines)
- The method is optimal for the assignment problem
- It has a time complexity of $O(n^3)$

Bipartite Graph Matching for GED

- The assignment problem has nothing to do with GED computation
- However GED can be reformulated (simplified) such that Munkres' algorithm becomes applicable



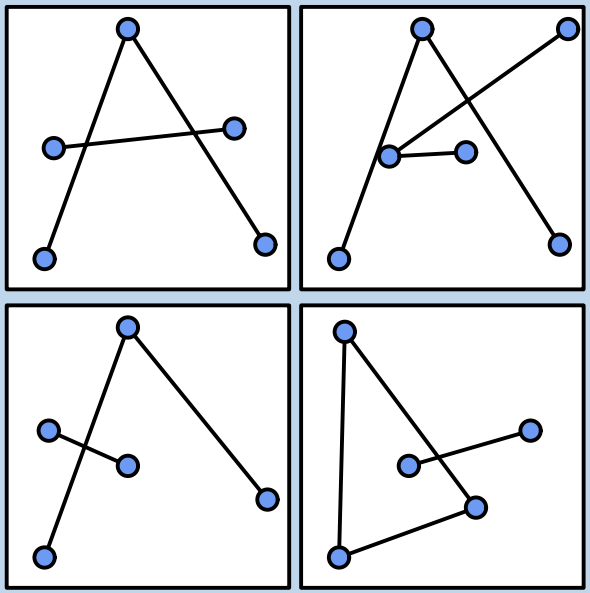


- Different reformulations are possible (considering only nodes, or nodes plus their local edge neighborhood)
- Munkres' algorithm returns an optimal solution for the assignment problem
- However, we get only a suboptimal solution for GED

Experimental Evaluation

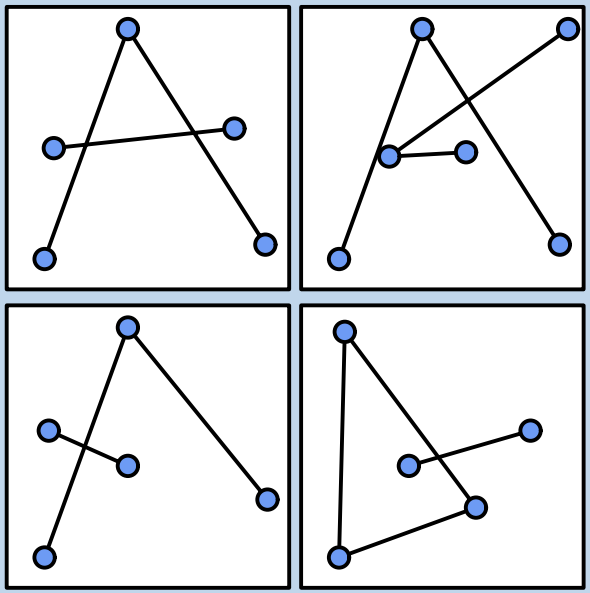
- There are two questions to be answered:
  - How much computation time can be saved by the suboptimal algorithms?
  - How much classification accuracy do we loose by applying suboptimal rather than optimal algorithms?

Letter Data Set



Method	Time [ms]
Tree Search	468.0
Plain	0.2
Adjacency	2.8

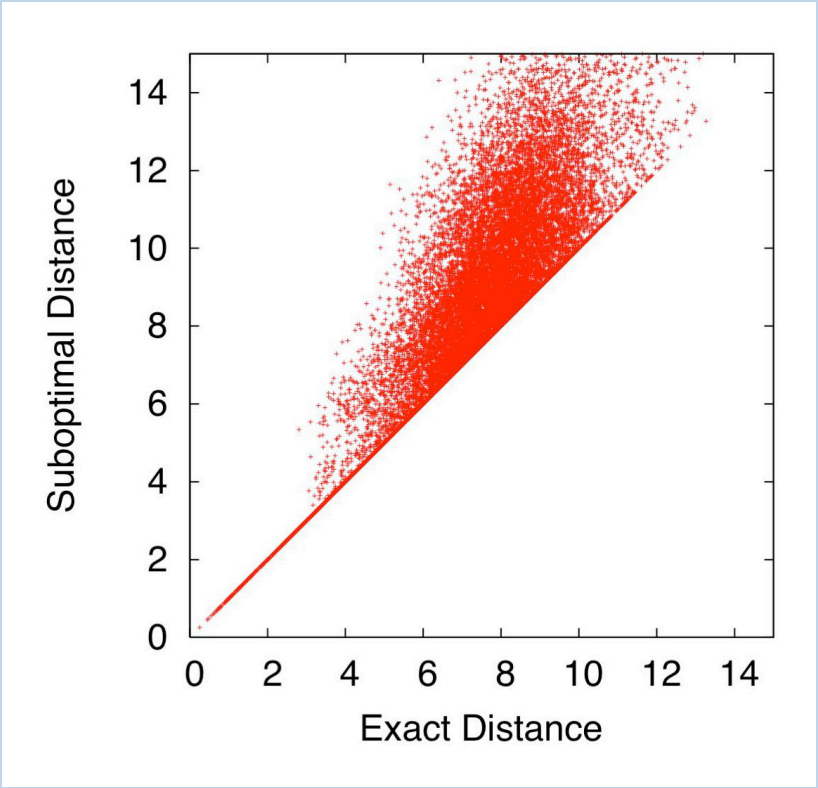
Letter Data Set



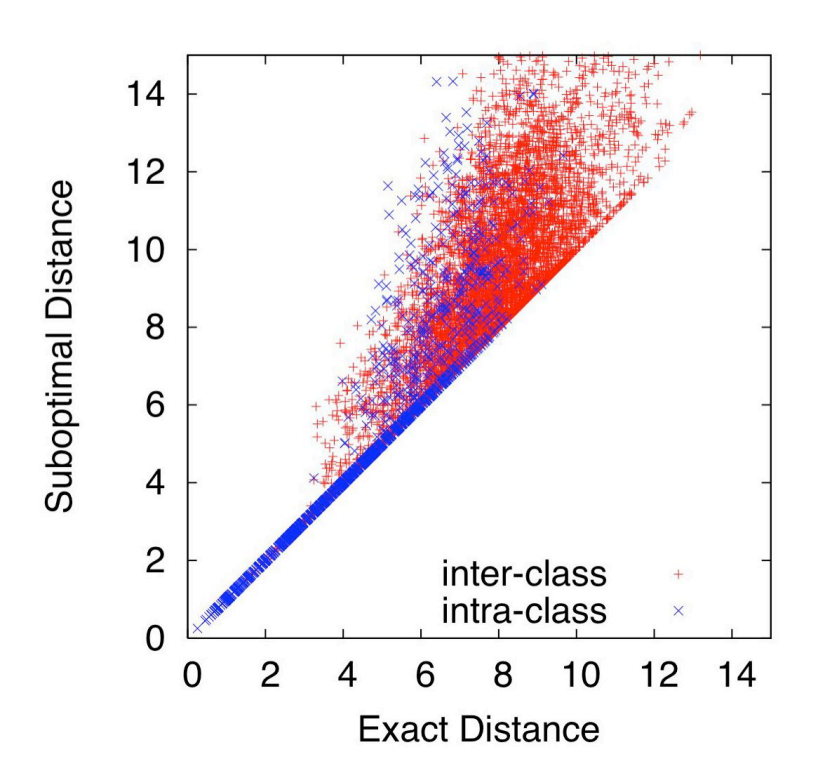
Method	Time [ms]	Accuracy
Tree Search	468.0	80.7
Plain	0.2	84.0 ○
Adjacency	2.8	84.0 ○

○ Statistically significantly better than the reference system

Suboptimality



Suboptimality



- Suboptimality mainly leads to an increase of inter-class distances, while most of the intra-class distances are not strongly affected

Contents

- Introduction
- Graph Embedding
- Dissimilarity Space Embedding of Graphs
- Multiple Classifier Systems
- Further Refinements
- Efficient Graph Edit Distance
- **Summary and Conclusions**

Summary and Conclusions

- Graphs are embedded in vector spaces by means of prototype selection and graph edit distance
- In this way we can utilize the high representational power of graphs and make available all pattern recognition methods that have been developed for feature based object representations
- The proposed method can be applied to classification and clustering tasks
- It is applicable to any type of graph
- Furthermore, it is applicable not only in conjunction with kernel methods, but to any other type of algorithm that operates in a feature space

Summary and Conclusions

- To avoid computational complexity problems, approximate methods for edit distance computation can be used; they lead to a substantial gain in computational speed, but don't compromise classification performance
- The experimental results show superior performance over reference systems over a spectrum of different applications and graphs with different characteristics

Related Publications 1/4

UNIVERSITÄT BERN

- Efficient Graph Edit Distance
  - Riesen, K. and Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. To be published in IMAVIS.
  - Riesen, K. and Neuhaus, M. and Bunke, H.: Bipartite Graph Matching for Computing the Edit Distance of Graphs. In Proc. 6th Int. Workshop on GbR, 2007.
  - Neuhaus, M. and Riesen, K. and Bunke, H.: Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In Proc. 11th int. Workshop on SSPR, 2006.
  - Riesen, K. and Fankhauser, S. and Bunke, H.: Speeding up Graph Edit Distance Computation with a Bipartite Heuristic. In Proc. 5th. Int. Workshop on MLG

Related Publications 2/4

- Dissimilarity Space Embedding
  - Riesen, K. and Bunke, H.: Graph Classification Based on Vector Space Embedding. To be published in IJPRAI.
  - Riesen, K. and Neuhaus, M. and Bunke, H.: Graph Embedding in Vector Spaces by Means of Prototype Selection. In Proc. 6th Int. Workshop on GbR, 2007.
  - Riesen, K. and Bunke, H.: Reducing the Dimensionality of Dissimilarity Space Embedding Graph Kernels. To be published in AAI.
  - Riesen, K. and Kilcherr, V. and Bunke, H.: Reducing the Dimensionality of Vector Space Embeddings of Graphs. In Proc. 5th Int. Conf. on MLDM, 2007.
  - Riesen, K. and Bunke, H.: Classifier Ensembles for Vector Space Embedding of Graphs. In Proc. 7th Int. Workshop on MCS, 2007.

Related Publications 3/4

- Dissimilarity Space Embedding
  - Riesen, K. and Bunke, H.: Structural Classifier Ensembles for Vector Space Embedded Graphs. In Proc. 20th Int. Joint Conf. on NN, 2007
  - Riesen, K. and Bunke, H.: Non-linear Transformations of Vector Space Embedded Graphs. In Proc. 8th Int. Workshop on PRIS, 2008.
  - Riesen, K. and Bunke, H.: On Lipschitz Embeddings of Graphs. In Proc. 12th International Conference on KES, 2008.
  - Riesen, K. and Bunke, H.: Kernel k-Means Clustering Applied to Vector Space Embeddings of Graphs. In Proc. 3rd IAPR Workshop ANNPR, 2008

Related Publications 4/4

- Databases
  - Riesen, K. and Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In Proc. 12th int. Workshop on SSPR, 2008.
  - `www.iam.unibe.ch/fki/databases/iam-graph-database`