The Weisfeiler-Lehman Kernel

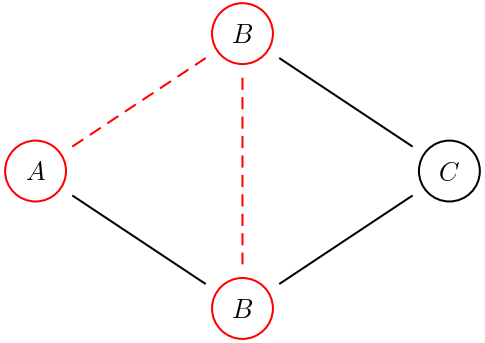
Karsten Borgwardt and Nino Shervashidze

Machine Learning and
Computational Biology Research Group,
Max Planck Institute for Biological Cybernetics and
Max Planck Institute for Developmental Biology, Tübingen

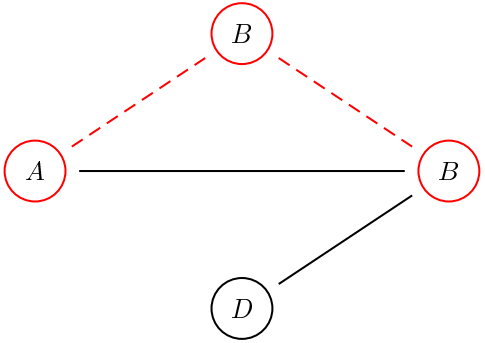
Graph Kernels Are All About...

How similar are two graphs?

- How similar is their structure?
- How similar are their node labels and edge labels?



G



G'

- Applications
  - Function prediction for molecules and proteins
  - Comparison of parts of images in computer vision
  - Comparison of semantic structures in NLP

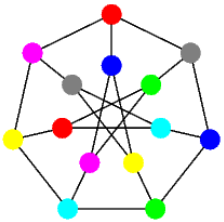
Today's Talk

- Introduction: What are graph kernels?
- Motivation: Why is there a need for scalable graph kernels?
- Past: Fast computation of random walk graph kernels
- Present: Scalable hash-based graph kernels
- Future: Scalable interpretable graph kernels

Graph Mining

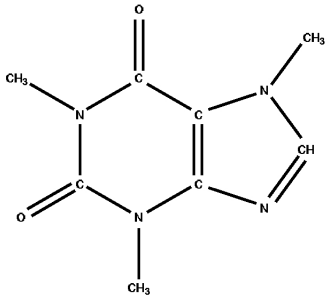
Graphs are everywhere

- Bioinformatics
- Computer Vision
- Natural Language Processing



Hot topics in databases/data mining

- Frequent subgraph mining
- Dense subgraph mining
- Graph indexing and search



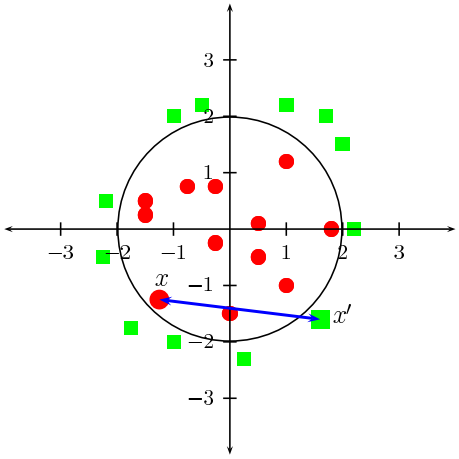
Recent trends

- Data: Growing size of graphs is a challenge for classic approaches
- Methods: Machine Learning approaches to graph mining

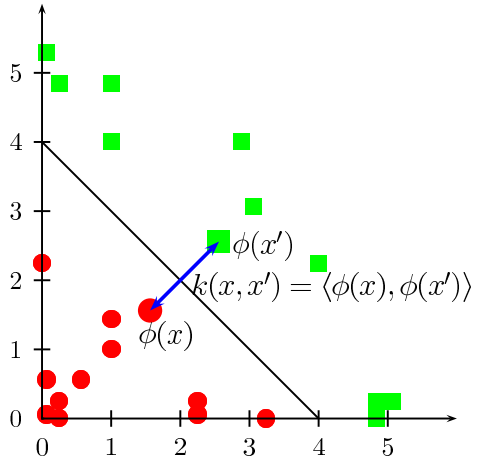
Kernels in a Nutshell I

Kernels

- Key concept: Move learning task to feature space $\mathcal{H}$.
- Naive explicit approach:
  - Map objects x and x' via mapping ϕ to $\mathcal{H}$.
  - Measure their similarity in $\mathcal{H}$ as $\langle \phi(x), \phi(x') \rangle$.
- **Kernel Trick**: Compute inner product in $\mathcal{H}$ as kernel in input space $k(x, x') = \langle \phi(x), \phi(x') \rangle$.



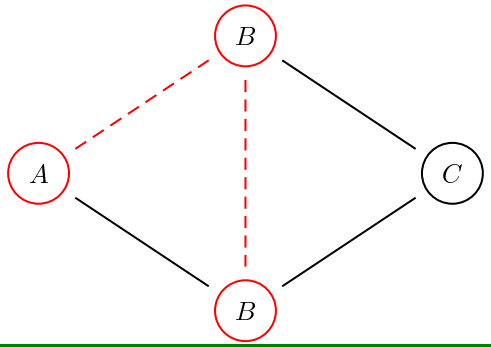
$\mathbb{R}^2 \Rightarrow \mathcal{H}$



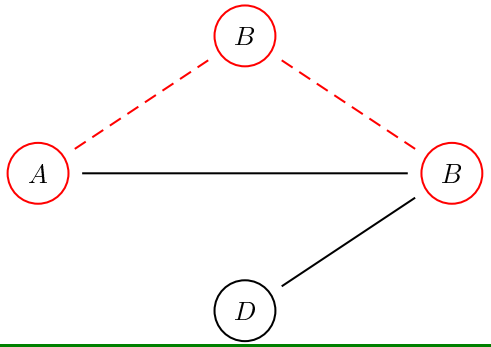
Kernels in a Nutshell II

Graph kernels

- Kernels on pairs of graphs
  (**not** pairs of nodes)
- Instance of R-Convolution kernels (Haussler, 1999):
  - Decompose objects x and x' into substructures.
  - Pairwise comparison of substructures via kernels to compare x and x'.
- A graph kernel makes the whole family of kernel methods applicable to graphs.



G



G'

Learning with Graph Kernels

Define a graph kernel

- Lots of attention (decades of research in chemoinformatics, early work on graph kernels)

Compute this graph kernel

- Little attention on efficiency on large graphs (ICDM 2005, NIPS 2006c, AISTATS 2009a, NIPS 2009, JMLR 2010)

Plug kernel values into a kernel method

- Lots of interest in developing new kernel methods

Apply kernel method to an application problem

- Bioinformatics

Complete Graph Kernels

Complete graph kernels (Gärtner et al., 2003)

- If the mapping ϕ is injective, then computing the corresponding kernel k is as hard as deciding graph isomorphism.
- Wanted: Polynomial-time kernel (non-injective) which combines expressivity with efficiency

Walk-based graph kernels (Kashima et al., 2003)

- Count common walks in two graphs G and G'.
- The more common walks, the more similar G and G'.
- Common walk means series of nodes and edges with identical labels.

Product Graph Kernels

Elegant computation (Gärtner et al., 2003)

- Form a direct product graph of G and G'.
- Count walks in this product graph.
- Each walk in product graph corresponds to one walk in each of the two input graphs:

$$k_\times(G,G') = \sum_{i,j=1}^{|V_\times|} [\sum_{k=0}^{\infty} \lambda^k A_\times^k]_{ij} = \mathbf{e}^\top \underbrace{(\mathbf{1} - \lambda A_\times)^{-1}}_{n^2 \times n^2} \mathbf{e}$$

Setbacks

- Lack of efficiency: scales as $O(n^6)$
- Tottering: walks allow to go back and forth between the same two nodes
- Halting: short walks are overweighted

Speeding Up Graph Kernels

Fast product graph kernel

- Compute product graph kernel via Sylvester Equations and Kronecker Products in $O(n^3)$ (NIPS 2006c).

Shortest path kernel

- Avoid tottering and halting by a graph kernel based on shortest path distances (ICDM 2005); scales as $O(n^4)$.

Graphlet kernel

- Enumerate or sample unlabeled subgraphs of limited size g from each graph (AISTATS 2009); scales as $O(nd^{g-1})$.

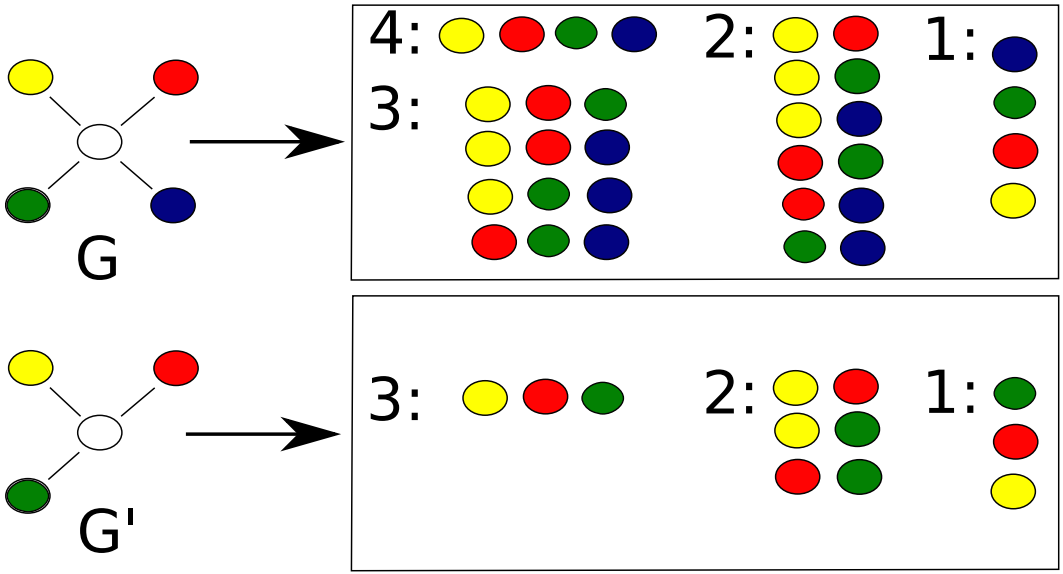
Unresolved problem

- How do scale up graph kernels to large, labeled graphs?

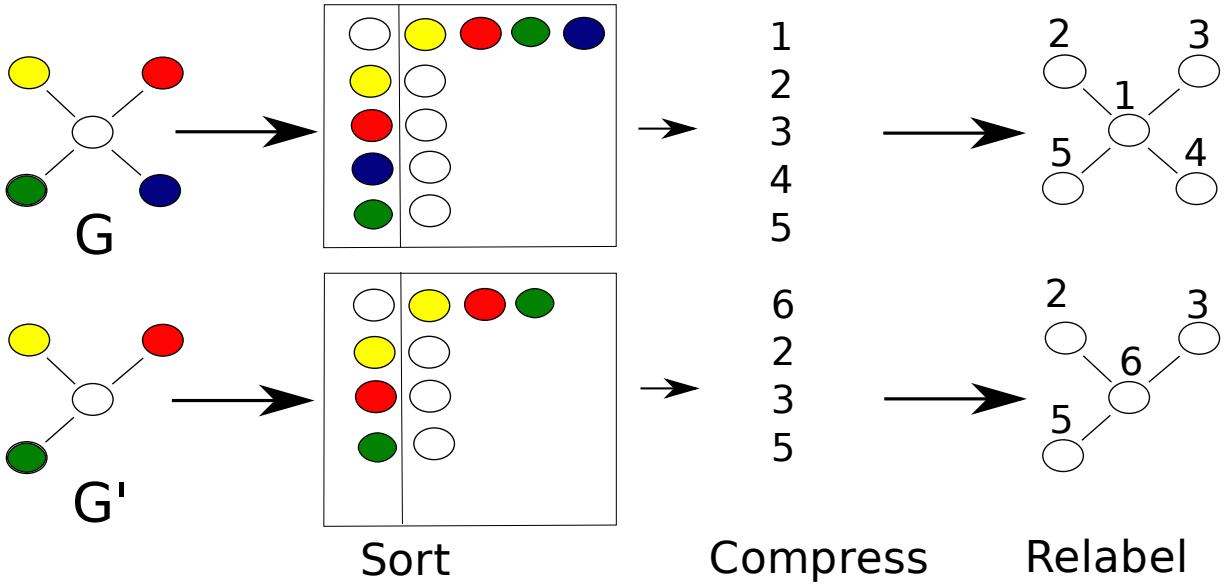
Subtree Kernels

Classic 'subtree kernel' (Ramon et al., 2003)

- Given two graphs G and G' with n nodes each.
- Compare all pairs of nodes v and v' from G and G'
- Compare all subsets of their neighbours recursively
- Runtime $O(n^2\ 4^d\ h)$



Weisfeiler-Lehman Procedure



A new subtree kernel

- Count common labels after each iteration
- Process N graphs simultaneously
- Use a global hash function for compression

Fast Subtree Kernels

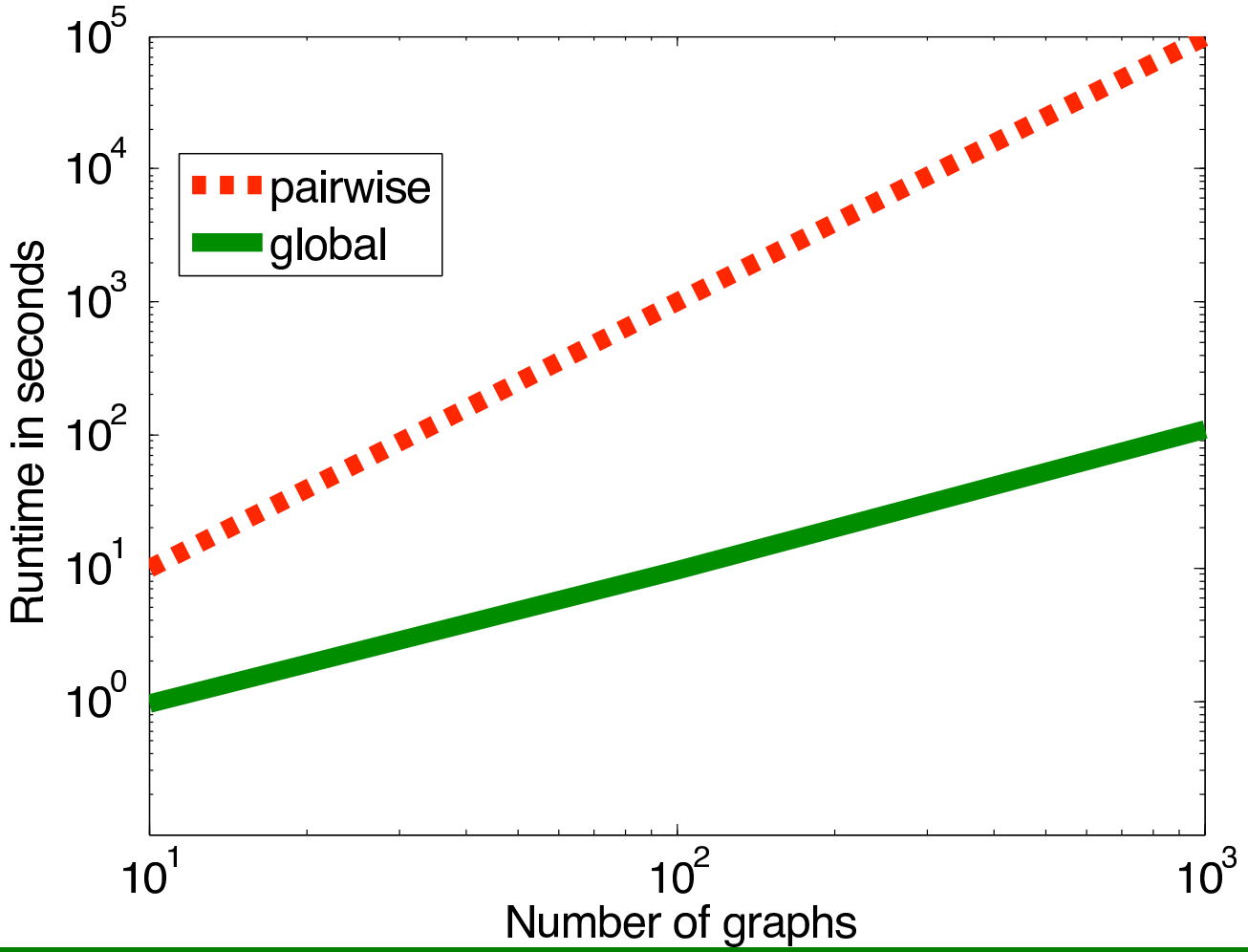
Weisfeiler-Lehman Kernel (NIPS 2009)

- **Algorithm**: Perform the following three steps h times:
  1. Sorting: Represent each node v as a sorted list L_v of its neighbours ($O(m)$)
  2. Compression: Compress this list into a hash value $h(L_v)$ ($O(m)$)
  3. Relabeling: Relabel v with $h(L_v)$ as its new node label ($O(n)$)

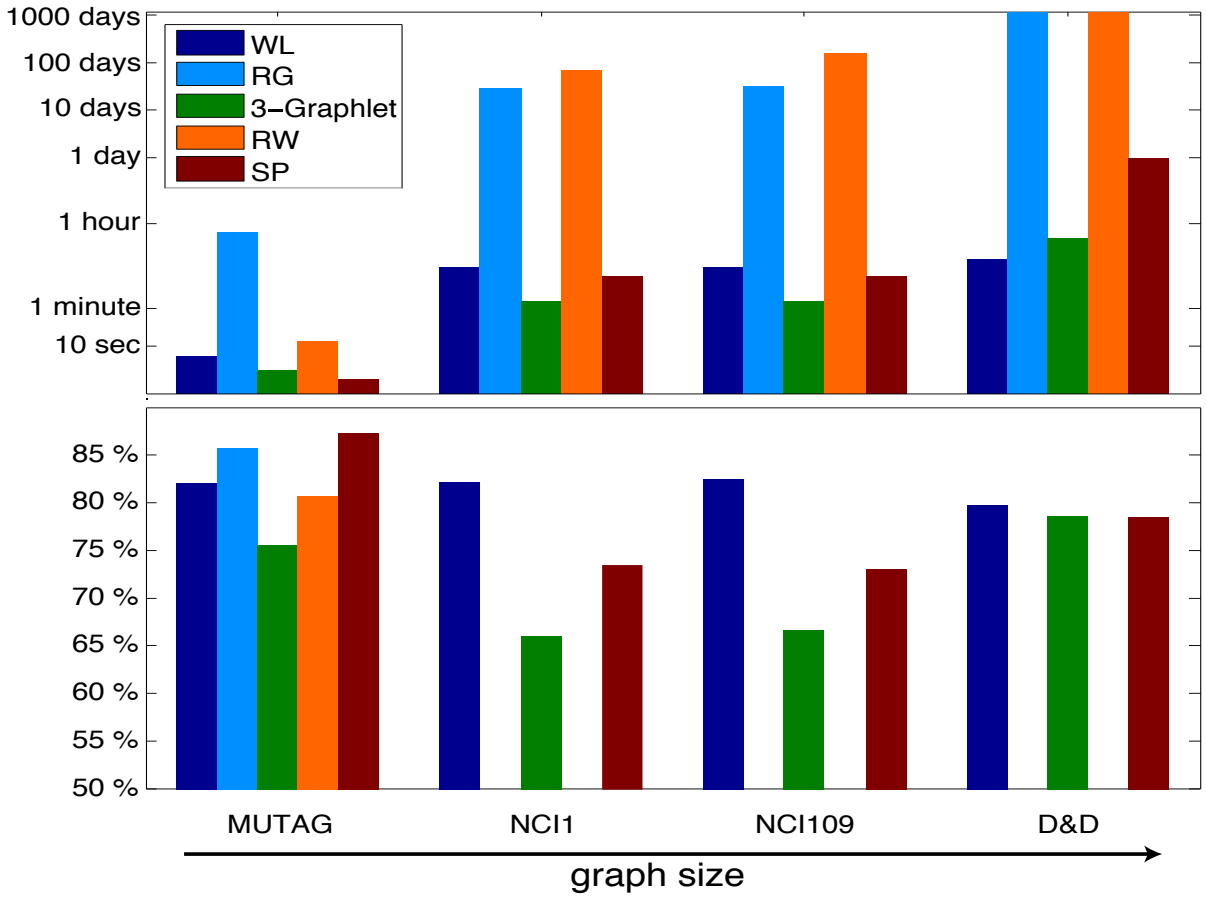
Analysis

- per pair of graphs: Runtime $O(m\ h)$ (versus $O(n^2\ 4^d\ h)$ for the classic kernel)
- for N graphs: Runtime $O(N\ m\ h + N^2\ n\ h)$ (naive $O(N^2\ m\ h)$)

WL Kernel: Pairwise vs. Global



WL Kernel: Runtime & Accuracy



Outlook: Feature Selection

Interpretability

- **Why** are two graphs similar? $\rightarrow$ feature selection among graphs, that is selecting discriminative subgraphs
- Their number may grow exponentially with graph size.
- In general, selecting an optimal group of k features requires a runtime which is exponential in k.

Roadmap

- Our subtree kernels operate in a feature space of size $N\ n\ h$ — not exponential in n and hence more feasible to search.
- Alternatively, explore theoretically-founded strategies for finding discriminative subgraphs of arbitrary shape (Thoma et al., SDM 2009; Thoma et al., SADM 2010).

Summary

Today's talk

- Motivation
  - Graph kernels allow to apply kernel methods to graph data.
  - Efficient graph kernels are the key to exploit this in practice.
- Past: Speed-up of random walk kernels to $O(n^3)$
- Present: Hash-based subtree kernels in $O(m\ h)$
- Future: Feature selection on graphs using kernels