Lecture Mon 28.11.

KERNELS FOR GRAPHS

Kernels for non-vectorial data

Examples of data that is originally not in feature vector form:

- Sequences
- Graphs (e.g .molecular graphs)
- Images

How to compute kernels for them (efficiently)?

Kernels on Graphs

Graphs are everywhere . . .

Graphs in Reality

- Graphs model objects and their relationships.
- Also referred to as *networks*.
- All common data structures can be modelled as graphs.

Graphs in Bioinformatics

- Molecular biology studies relationships between molecular components.
- Graphs are ideal to model:
  - Molecules
  - Protein-protein interaction networks
  - Metabolic networks

Central Questions

How similar are two graphs?

- Graph similarity is the central problem for all learning tasks such as clustering and classification on graphs.

Applications

- Function prediction for molecules, in particular, proteins
- Comparison of protein-protein interaction networks

Challenges

- Subgraph isomorphism is NP-complete.
- Comparing graphs via isomorphism checking is thus prohibitively expensive!
- Graph kernels offer a faster, yet one based on sound principles.

From the beginning ...

Definition of a Graph

- A *graph* G is a set of nodes (or vertices) V and edges E, where $E \subset V^2$.
- An *attributed* graph is a graph with labels on nodes and/or edges; we refer to labels as *attributes*.
- The *adjacency matrix* A of G is defined as

$$[A]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases},$$

  where v_i and v_j are nodes in G.
- A *walk* w of length $k-1$ in a graph is a sequence of nodes $w = (v_1, v_2, \cdots, v_k)$ where $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq k$.
- w is a *path* if $v_i \neq v_j$ for $i \neq j$.

Graph Isomorphism

Graph isomorphism (cf. Skiena, 1998)

- Find a mapping f of the vertices of G to the vertices of H such that G and H are identical; i.e. (x, y) is an edge of G iff $(f(x), f(y))$ is an edge of H. Then f is an isomorphism, and G and F are called isomorphic.
- No polynomial-time algorithm is known for graph isomorphism
- Neither is it known to be NP-complete

Subgraph isomorphism

- Subgraph isomorpism asks if there is a subset of edges and vertices of G that is isomorphic to a smaller graph H.
- Subgraph isomorphism is NP-complete

Subgraph Isomorphism

NP-completeness A decision problem C is NP-complete, iff

- C is in NP
- C is NP-hard, i.e. every other problem in NP is reducible to it

Problems for the practitioner

- Characterization of NP-complete problems: (thought to be) hard to solve, easy (polynomial-time) to verify
- Excessive runtime in worst case: Runtime may grow exponentially with number of nodes
- For large graphs with many nodes, and for large datasets of graphs, this is an enormous problem

Wanted Polynomial-time similarity measure for graphs

[Gärtner et al., 2003]

Graph kernels

- Compare substructures of graphs
- Example substructures:
  - Walks
  - Paths
  - Cyclic patterns
  - Tree-shaped subgraphs
  - (small) General subgraphs

Criteria for a good graph kernel

- Expressive
- Efficient to compute
- Positive definite
- Applicable to wide range of graphs

Random Walks

Principle

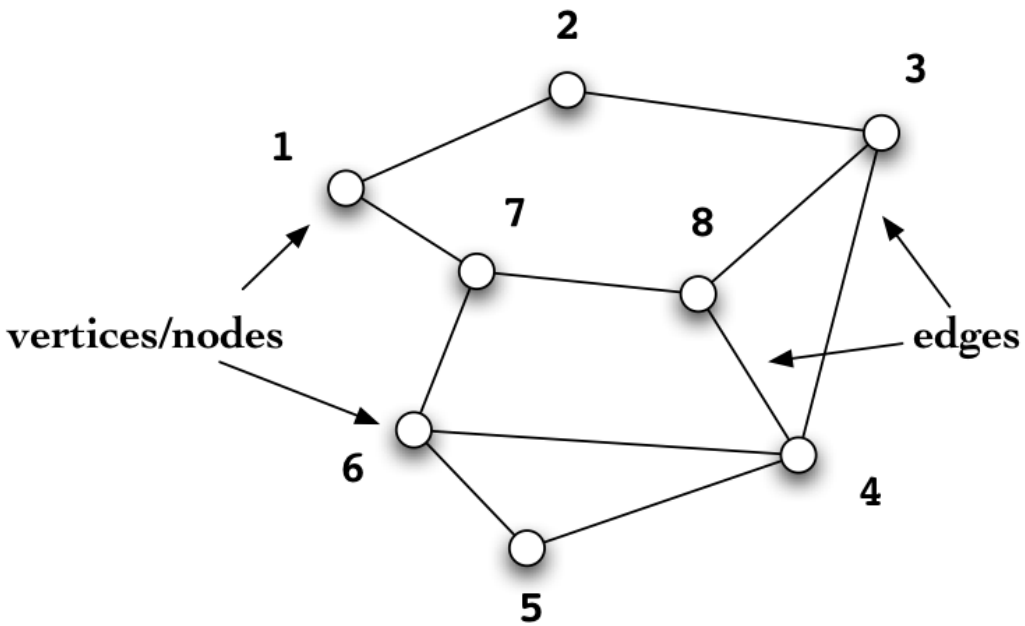
- Compare walks in two input graphs
- Walks are sequences of nodes that allow repetitions of nodes

Important trick

- Walks of length k can be computed by taking the adjacency matrix A to the power of k
- $A^k(i,j) = c$ means that c walks of length k exist between vertex i and vertex j

From adjacency matrix to walks

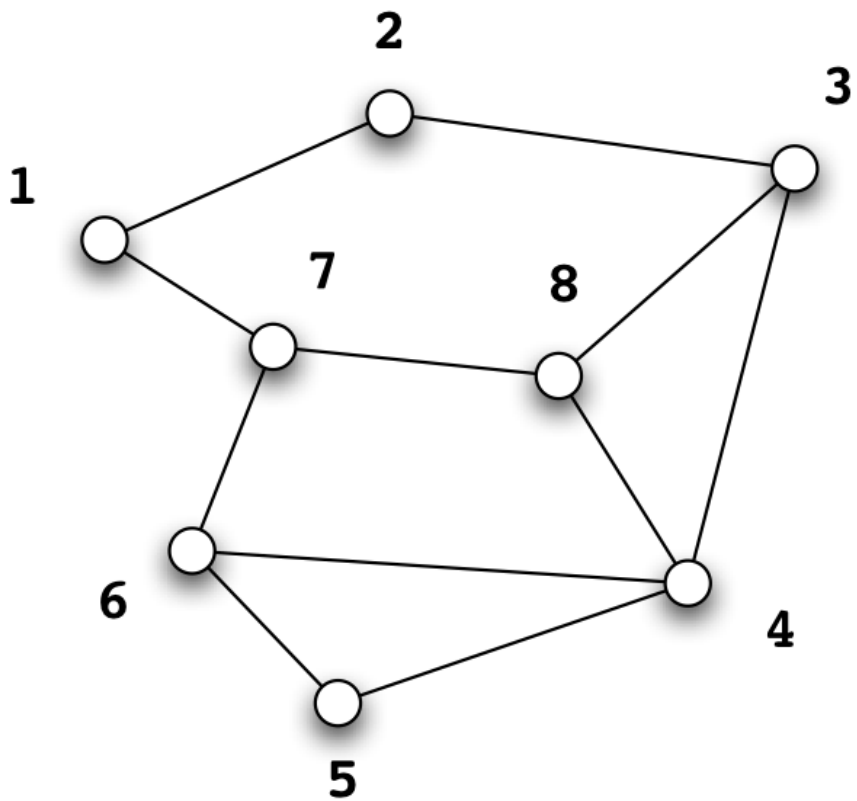
- The adjacency matrix denotes the number of length 1 walks = edges between two nodes



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

From adjacency matrix to walks

- Matrix multiplication $A^2 = AA$ reveals the number of length 2 walks $A^2_{ij} = \sum_h A_{ih} A_{hj}$



$$A^2 = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 3 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 4 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 3 & 0 & 2 \\ 0 & 1 & 1 & 2 & 1 & 0 & 3 & 0 \\ 1 & 1 & 1 & 1 & 1 & 2 & 0 & 3 \end{bmatrix}$$

Product Graph

How to find common walks in **two** graphs?

- Use the product graph of G_1 and G_2

Definition

- $G_\times = (V_\times, E_\times)$, defined via

$$V_\times(G_1 \times G_2) = \{(v_1, w_1) \in V_1 \times V_2 : label(v_1) = label(w_1)\}$$

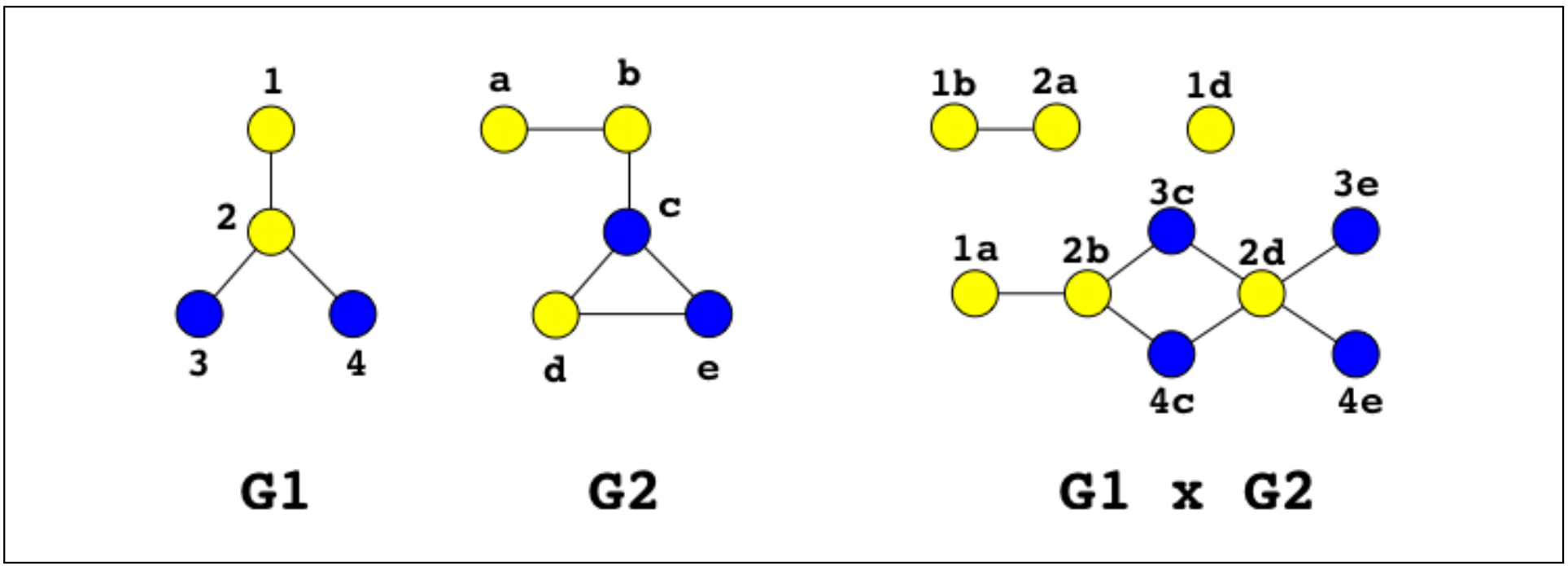
$$E_\times(G_1 \times G_2) = \{((v_1, w_1), (v_2, w_2)) \in V^2(G_1 \times G_2) : (v_1, v_2) \in E_1 \wedge (w_1, w_2) \in E_2 \wedge (label(v_1, v_2) = label(w_1, w_2))\}$$

Meaning

- Product graph consists of pairs of identically labeled nodes and edges from G_1 and G_2

Product graph

- Tracing a walk in the product graph corresponds to simultaneously tracing common walks in the two original graphs



Random Walk Kernel

The trick

- Common walks can now be computed from $A_\times^k$

Definition of random walk kernel

- $$k_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} [\sum_{n=0}^{\infty} \lambda^n A_\times^n]_{ij},$$

Meaning

- Random walk kernel counts all pairs of matching walks
- λ is decaying factor for the sum to converge (components of A^n tend to infinity as $n \to \infty$)

Runtime of Random Walk Kernels

Notation

- given two graphs G_1 and G_2
- n is the number of nodes in G_1 and G_2

Computing product graph

- requires comparison of all pairs of edges in G_1 and G_2
- runtime $O(n^4)$

Powers of adjacency matrix

- matrix multiplication or inversion for $n^2 * n^2$ matrix
- runtime $O(n^6)$

Total runtime

- $O(n^6)$

Tottering

Artificially high similarity scores

- Walk kernels allow walks to visit same edges and nodes multiple times $\rightarrow$ artificially high similarity scores by repeated visits to same two nodes

Additional node labels

- Mahé et al. [2004] add additional node labels to reduce number of matching nodes $\rightarrow$ improved classification accuracy

Forbidding cycles with 2 nodes

- Mahé et al. [2004] redefine walk kernel to forbid subcycles consisting of two nodes $\rightarrow$ no practical improvement

All-paths Kernel?

Idea

- Determine all paths from two graphs
- Compare paths pairwise to yield kernel

Advantage

- No tottering

Problem

- All-paths kernel is NP-hard to compute.
- Requires the set of graphs to be small enough, such as metabolite molecules in biochemical reactions
- Feasible to be compute for metabolites and reactions in KEGG database [Heinonen et al., 2012]

Longest paths?

- Also NP-hard – same reason as for all paths

Shortest Paths!

- computable in $O(n^3)$ by the classic Floyd-Warshall algorithm 'all-pairs shortest paths'

Shortest-path Kernels

Kernel computation

- Determine all shortest paths in two input graphs
- Compare all shortest distances in G_1 to all shortest distances in G_2
- Sum over kernels on all pairs of shortest distances gives shortest-path kernel

Runtime

- Given two graphs G_1 and G_2
- n is the number of nodes in G_1 and G_2
- Determine shortest paths in G_1 and G_2 separately: $O(n^3)$
- Compare these pairwise: $O(n^4)$
- Hence: Total runtime complexity $O(n^4)$

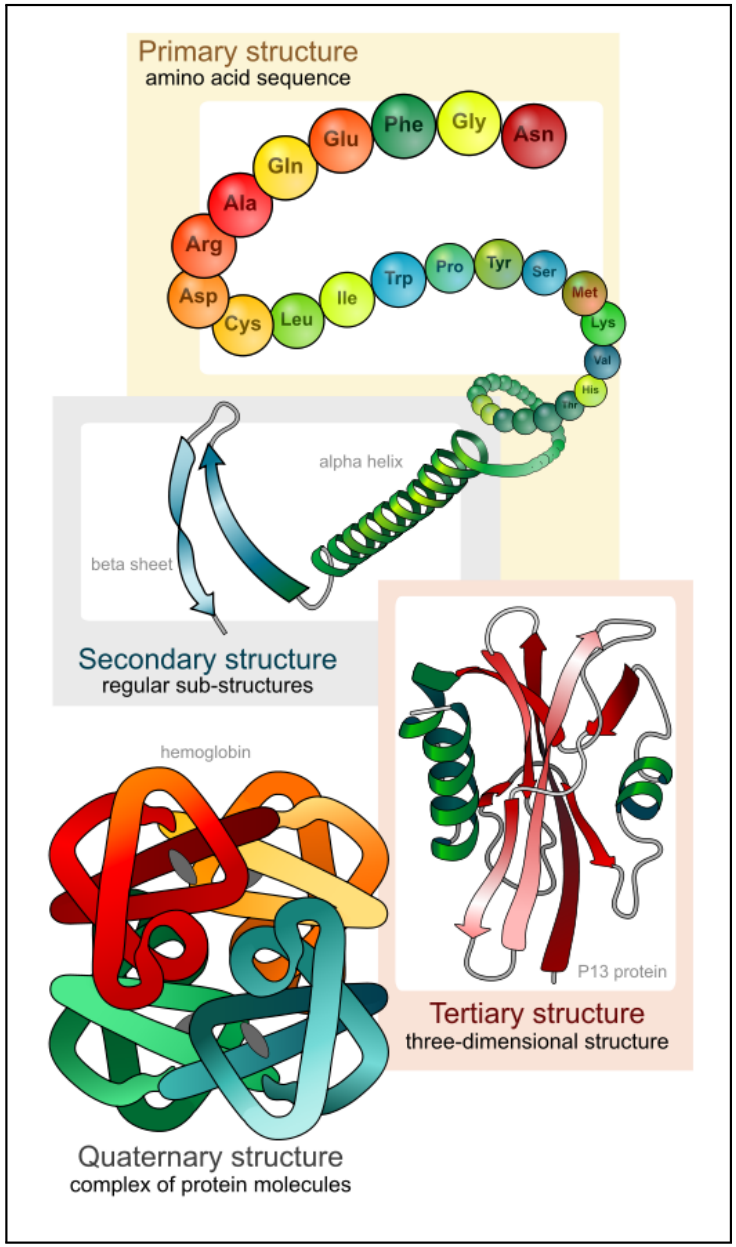
[Borgwardt and Kriegel, 2005]

Protein function prediction with graph kernels

- Motivation: protein 3D structure determines its function in principle
- But: experimental determination of the function of a protein remains a difficult, time- and cost-intensive task.
- In silico verification of the protein function via molecular simulation is extremely demanding computationally
- Can we learn to predict the function from the structure (and sequence) making use of known 3D structures and associated biological functions?

Levels of protein structure

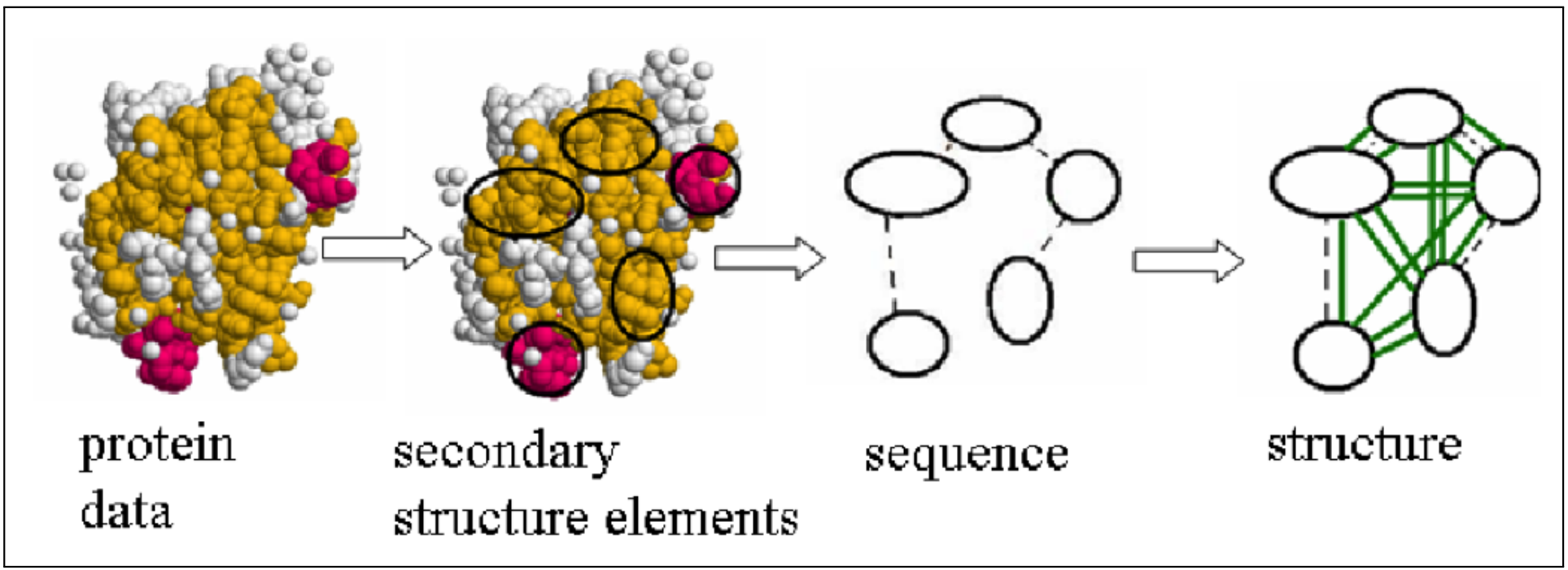
1. Primary structure: amino acid sequence
2. Secondary structure: local organization of the sequence to α helices, β strands and unorganized coils
3. Tertiary structure: 3D organization of the protein
4. Quaternary structure: protein complexes



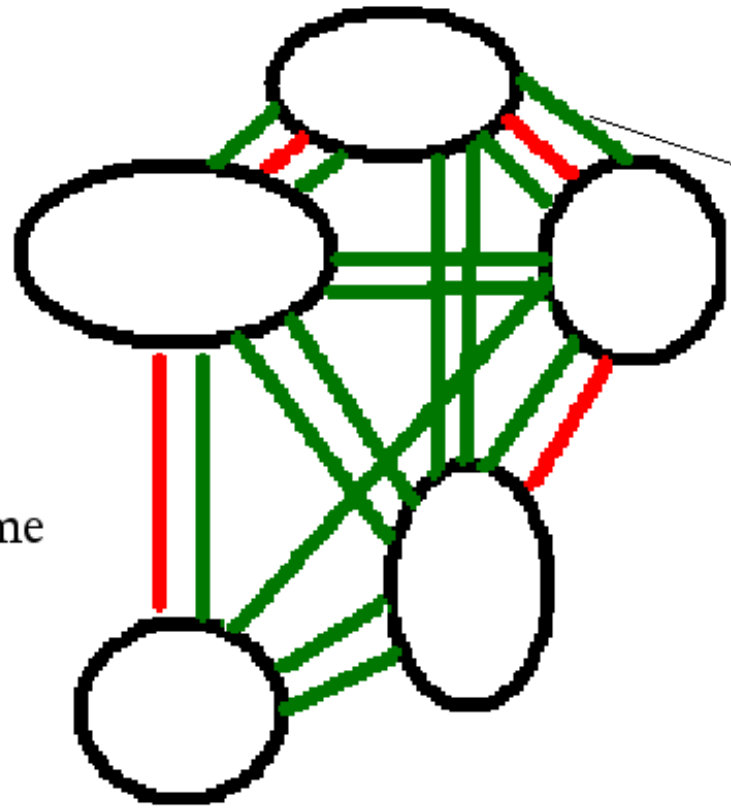
Representing protein structures via kernels

Represent proteins as protein graphs:

- Secondary structure elements (SSE) as nodes: e.g. helix of length 30 amino acids
- Edges between elements that are adjacent in sequence (sequence edge) or close in 3D space (structural edge)
- Compare the protein graphs by graph kernels



Types of data for edges and nodes



Node attributes

- hydrophobicity
- polarity
- polarizability
- van der Waals volume
- length
- helix, sheet, loop

Edge attributes

- type (sequence, structure)
- length

Karsten Borgwardt et al. - Classifying proteins into functional classes via graph kernels 11

Protein graph kernel [Borgwardt et al., 2005]

Protein graph kernel is defiend as a kernel over walks

$$k_{walk}(walk_1, walk_2) = \prod_{i=1}^{n-1} k_{step}((v_i, v_{i+1}), (w_i, w_{i+1}))$$

with steps in the walk given by kernel over node and edge similarities

$$k_{step}((v_i, v_{i+1}), (w_i, w_{i+1})) = k_{node}(v_i, w_i) * k_{node}(v_{i+1}, w_{i+1}) * k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1}))$$

Protein graph kernel [Borgwardt et al., 2005]

Node kernel

$$k_{node}(v_i, w_i) = k_{type}(v_i, w_i) \cdot k_{nodelabel}(v_i, w_i) \cdot k_{length}(v_i, w_i)$$

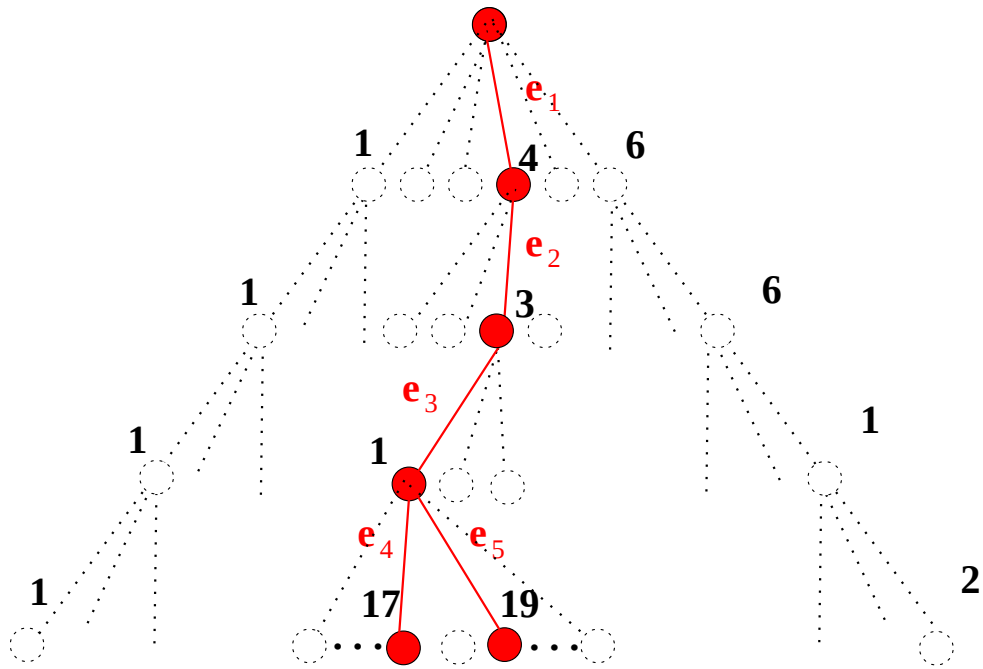
Edge kernel

$$k_{edge}((v_i, v_{i+1}), (w_i, w_{i+1})) = k_{type}((v_i, v_{i+1}), (w_i, w_{i+1})) \cdot k_{length}((v_i, v_{i+1}), (w_i, w_{i+1}))$$

- Type kernel k_{type}: forces matched edges to have the same type (structural edges or sequence edges) and the same types of SSEs as end points (helices, strands)
- Length kernel k_{length}: only allow matching SSEs that have close to same length in amino acids, for structural edges close to same distance in 3D space
- Node label kernel: $k_{nodelabels}$: compare the physico-chemical features of the SSEs

Representing enzyme function: EC classification

- Proteins catalyzing biochemical reactions are called enzymes
- Enzymatic functions are classified in the Enzyme Commission (EC) system
- Hierachy of four levels
  1. Main class (1-6): specifies the general type of reaction
  2. Levels 2-3: specify the reaction mechanism
  3. Level 4: specify the substrate molecules



Experiments [Borgwardt et al., 2005]

- Prediction of the first EC digit (main class)
- 600 enzymes with functional classification from BRENDA database (100 for each main EC class)
- protein structures from PDB
- SVM used as the classifier
- Different data used as node labels

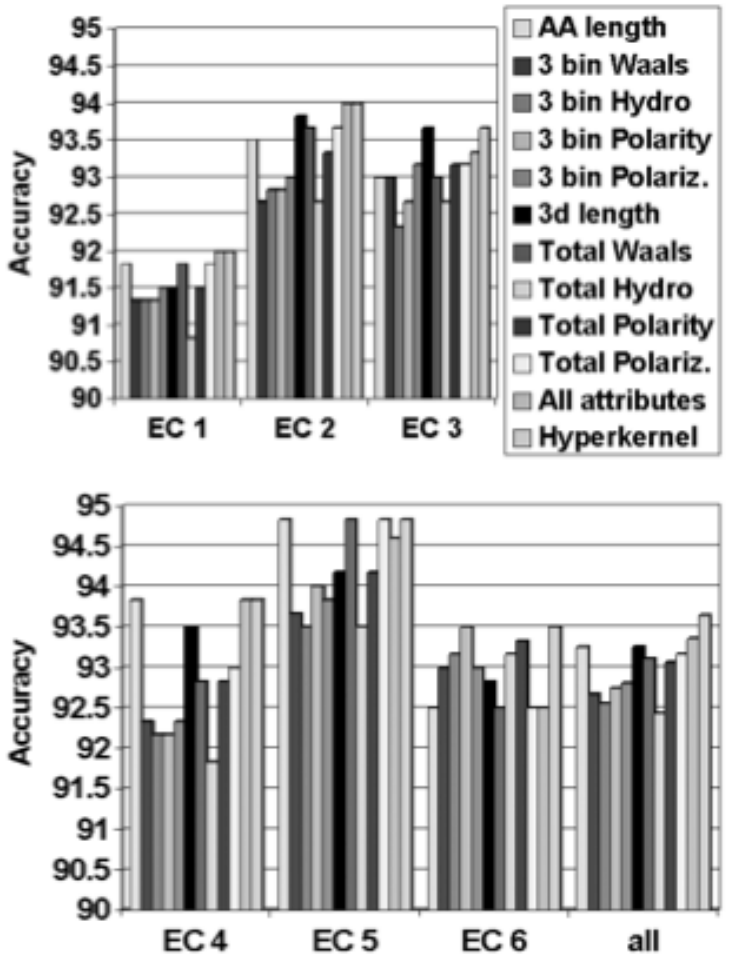
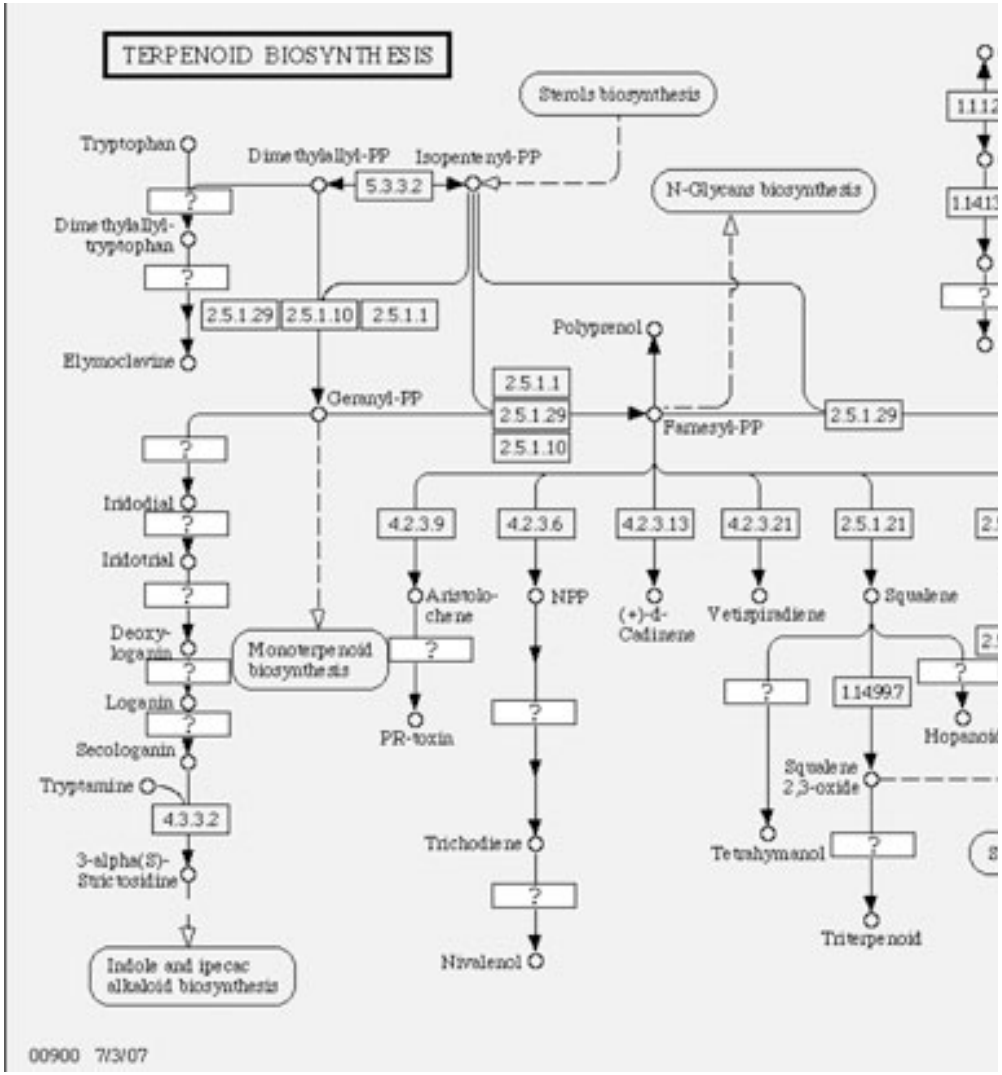


Fig. 3. Prediction accuracy using kernel matrices on individual attributes, one kernel on all attributes and the hyperkernel (Example 1) in 6-fold cross-validation on 600 enzymes from 6 EC top level classes (AA, amino acid; Waals, van der Waals volume; Hydro, Hydrophobicity; Polariz, Polarizability).

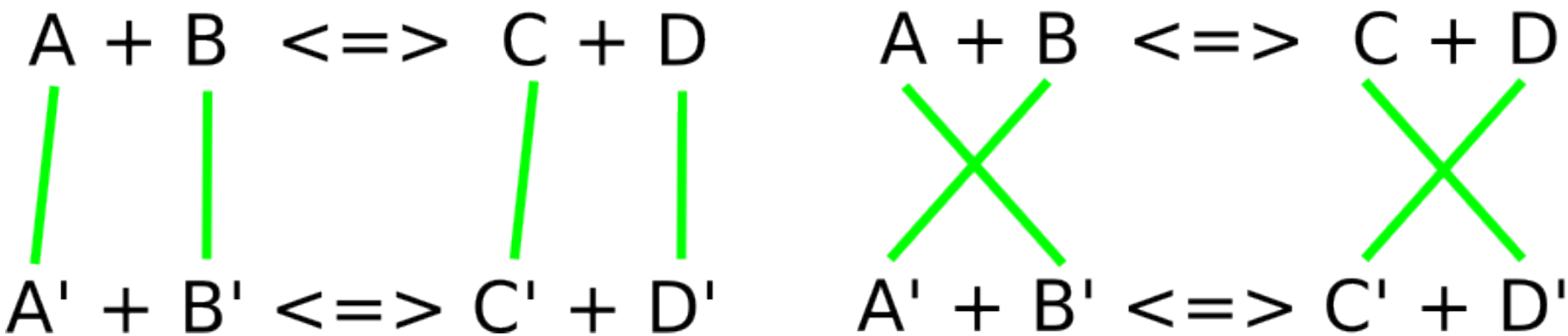
Classifying biochemical reactions

- Metabolic pathways are composed of biochemical reactions catalyzed by enzymes
- The catalyzing enzyme is not known for many metabolic reaction steps (e.g. '?'s in the picture)
- Can we predict automatically the functional classification?



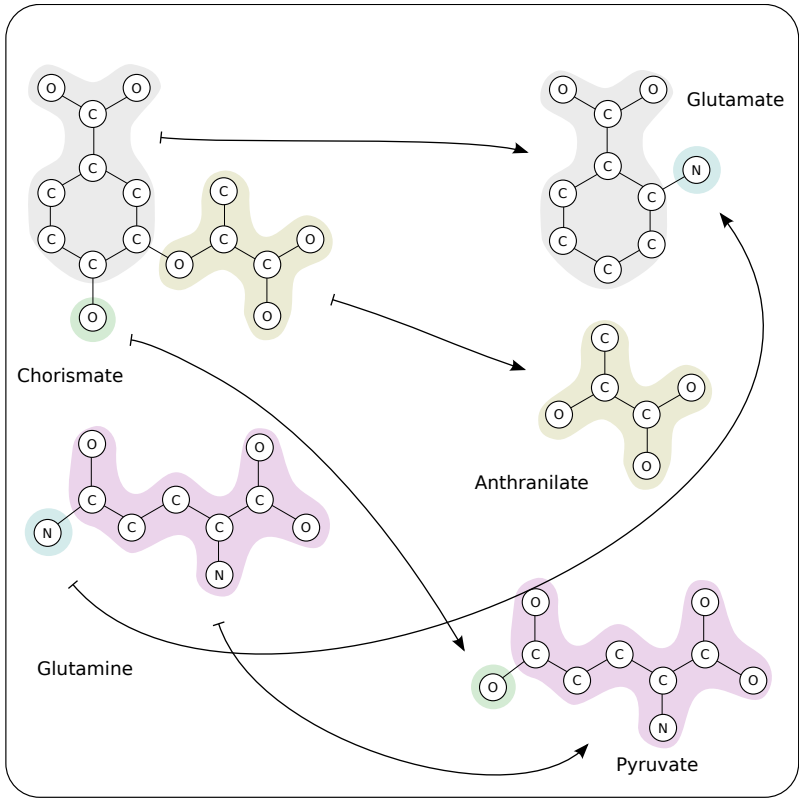
Representing reactions via graphs

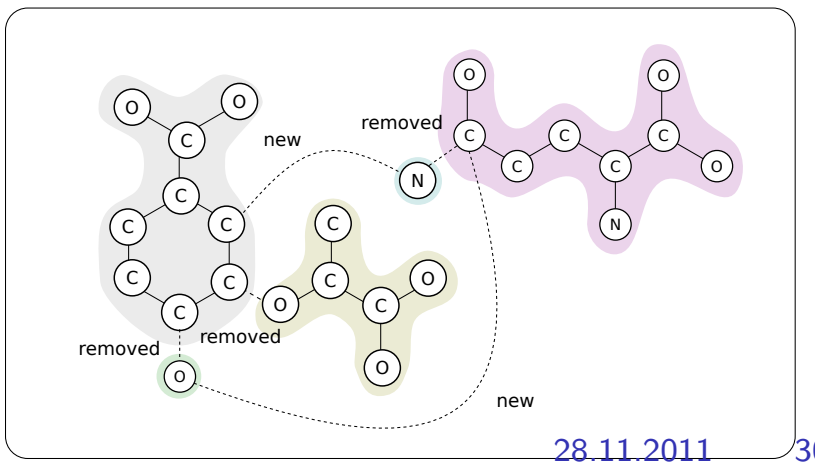
- How to represent similarity of reactions
- A biochemical reaction involves a set of molecules
- Pairwise comparison of substrates and products in two reactions with graph kernels
- Combine the molecule kernels into a reaction kernel (e.g. sum up the kernels)
- But can we do this more elegantly?



Reaction graph

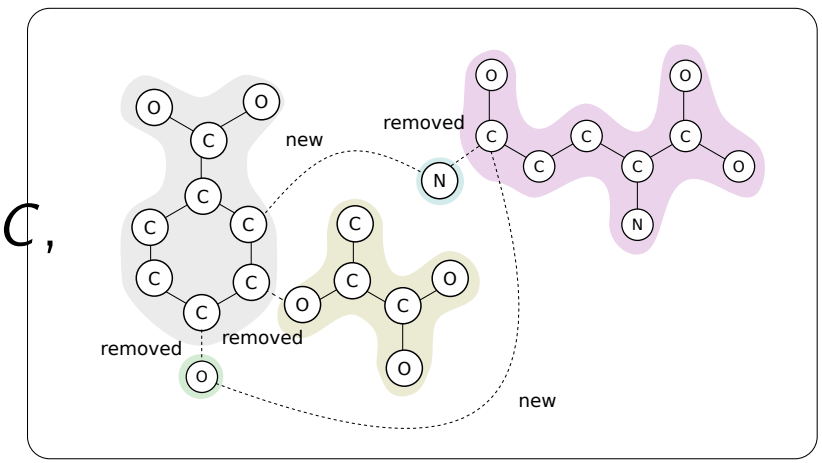
- Key principle: one to one correspondence of atoms in substrate and product molecules
- A node in reaction graph represents the corresponding atoms in substrates and products
- Labeled edges corresponding to bonds::
  - New: those that exist in products but not in substrates
  - Removed: those that exist in substrates but not in products
  - Intact: those that exist
  () in both





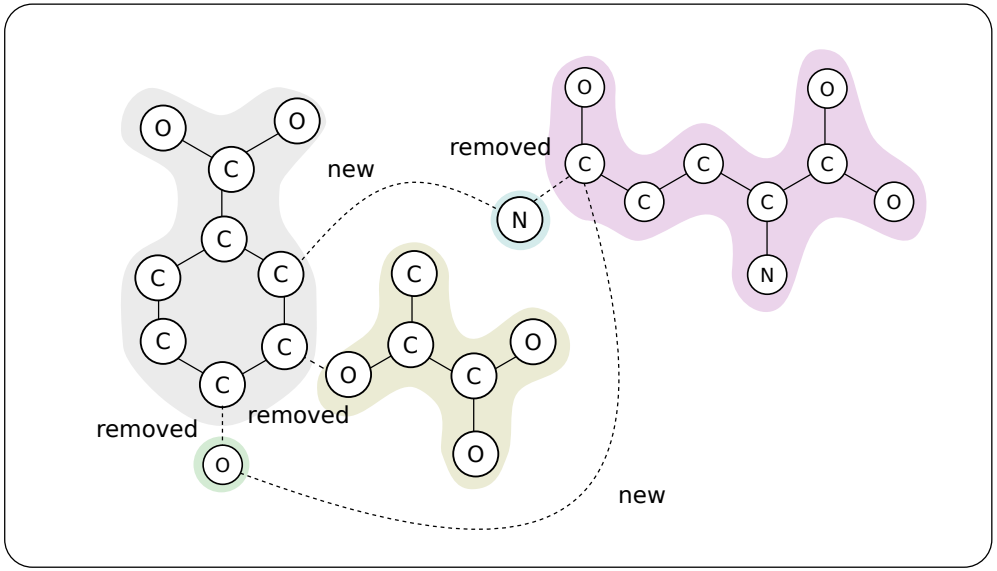
Kernels from reaction graphs

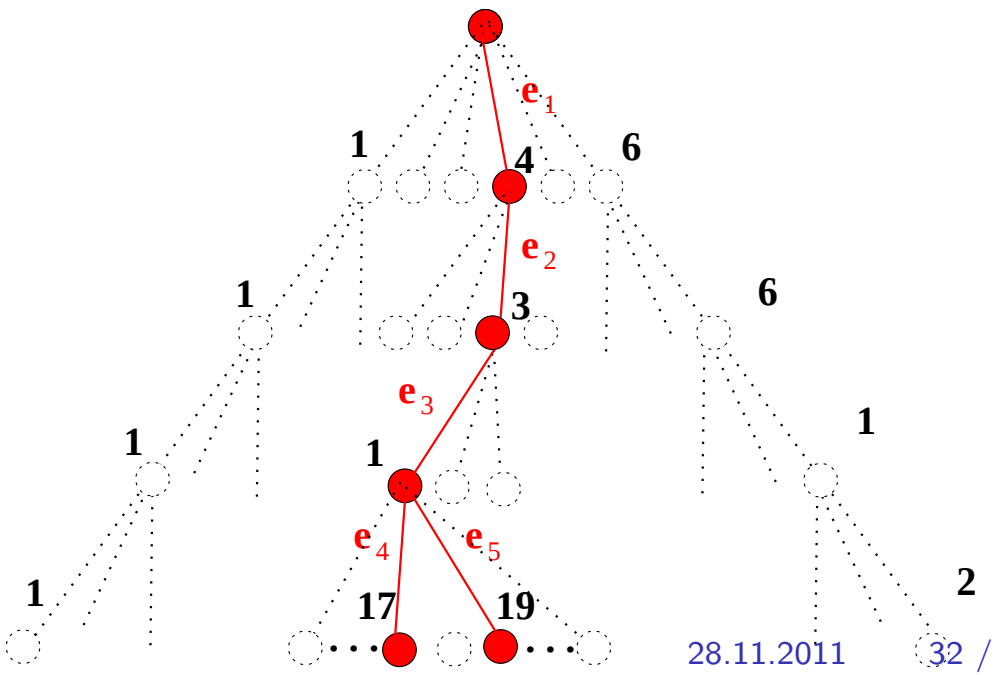
- Any graph kernel for labeled graphs will do
- Walks and paths will contain reaction information: e.g. $C(Intact)C(New)N(Removed)C$, $C(Removed)O(New)C$
- Walk/Path kernel will represent the similarity of reactions in terms of paths that are altered in similar manner



Experiments in reaction function prediction

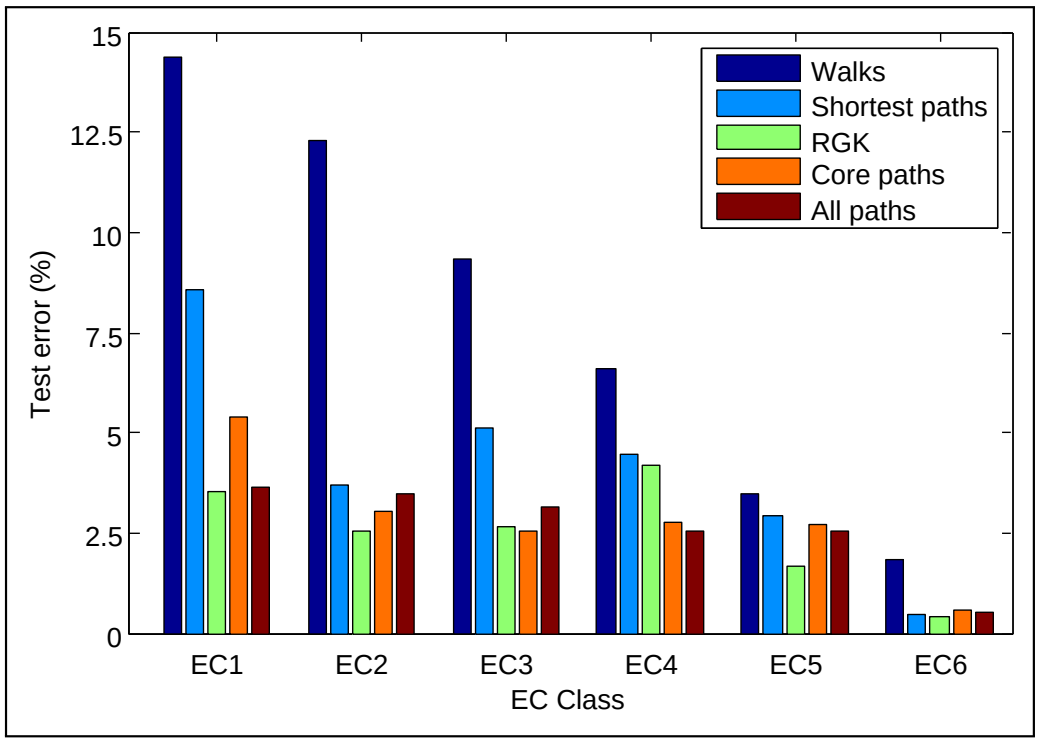
- Given a reaction, predict its functional class
- Data: ca. 17000 biochemical reactions from KEGG database
- Input: graph kernel on reaction graphs
- Output: 3 first digits of the EC code(s) corresponding to the reaction





Results [Heinonen et al., 2012]

- Reporting test errors from 5-fold cross-validation
- Correct predictions are those that have three first EC digits correct
- Walk kernel clearly inferior to others
- All paths kernel better than shortest paths
- RGK kernel [Saigo et al., 2010] on average the best (specially developed for this task)



References I

K.M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proc. of International Conference on Data Mining (ICDM 2005)*, pages 74–81, Houston, USA, 2005.

K.M. Borgwardt, C.S. Ong, S. Schonauer, S.V.N. Vishwanathan, A.J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21 Suppl 1: i47–56, Jun 2005. ISSN 1367-4803 (Print). doi: 10.1093/bioinformatics/bti1007.

T. Gärtner, P.A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proc. Annual Conf. Computational Learning Theory*. Springer, 2003.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

Markus Heinonen, Niko Välimäki, Veli Mäkinen, and Juho Rousu. Efficient path kernels for reaction function prediction. In *International Conference on Bioinformatics Models, Methods and Algorithms*, page to appear. INSTICC, February 2012.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. Intl. Conf. Machine Learning*, Washington, DC, United States, 2003.

P. Mahé, N. Ueda, T. Akutsu, Perret J.-L, and J.-P. Vert. Extensions of marginalized graph kernels. In *Proc. ICML*, 2004.

References II

L. Ralaivola, S. J Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw*, 18(8):1093–1110, Oct 2005. ISSN 0893-6080 (Print). doi: 10.1016/j.neunet.2005.07.009.

J. Ramon and T. Gärtner. Expressivity versus efficiency of graph kernels. In *First International Workshop on Mining Graphs, Trees and Sequences*, 2003.

H. Saigo, M. Hattori, H. Kashima, and K. Tsuda. Reaction graph kernels predict ec numbers of unknown enzymatic reactions in plant secondary metabolism. *BMC bioinformatics*, 11(Suppl 1):S31, 2010.

N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proc. AISTATS*, 2009. Accepted.

S.V.N. Vishwanathan, K.M. Borgwardt, and N. Schraudolph. Fast computation of graph kernels. In *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2006. MIT Press.