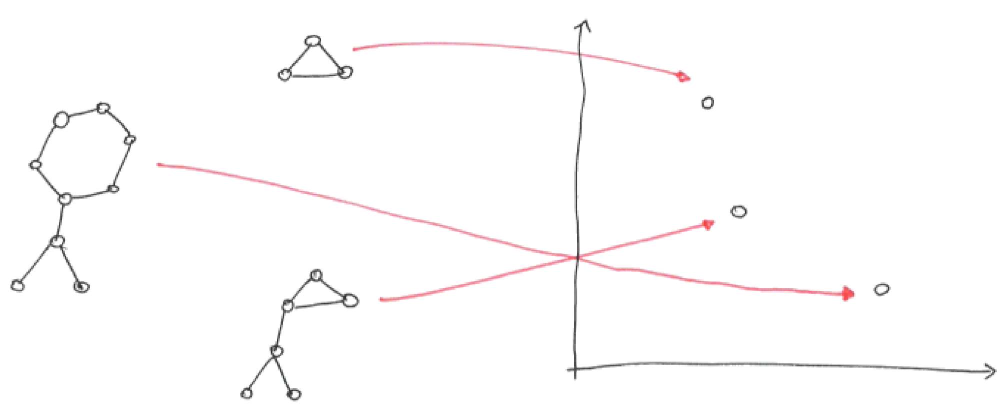
Min-Hashing for Probabilistic Frequent Subtree Feature Spaces

Pascal Welke, Tamás Horváth, Stefan Wrobel

Graph Kernels

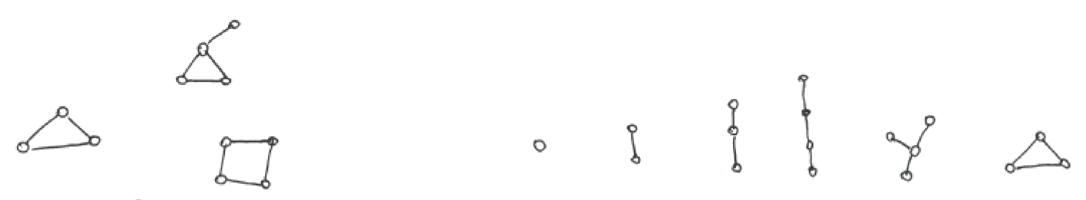
- Measure the similarity between graphs
- Enable us to learn models on graphs with generic learners
  - e.g. support vector machines, kernel PCA, …
- Expressive graph kernels suffer from their severe computational complexity
  - Most are NP-hard to compute



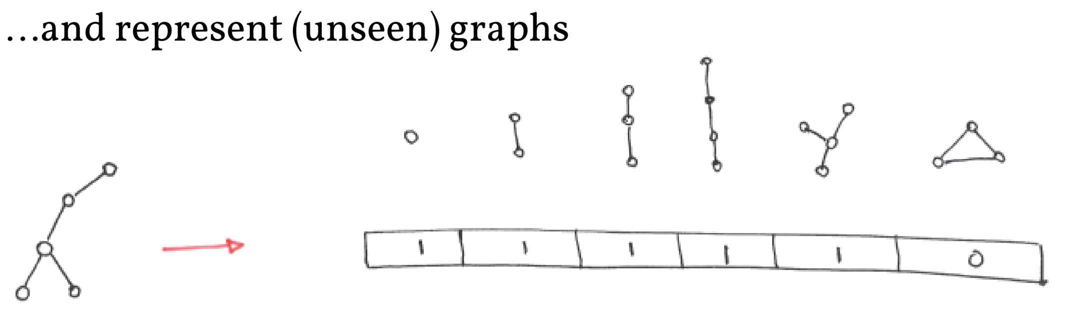
A kernel behaves like a scalar product in some real vector space

Frequent Subgraph Mining

- We can learn a representation of a graph dataset by mining the frequent connected subgraphs



- …and represent (unseen) graphs



+ gives quite good results

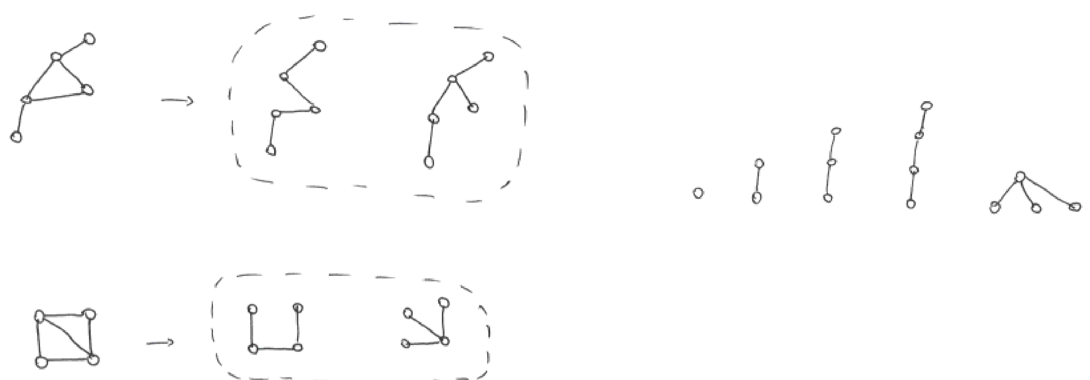
- computationally intractable
  - mining cannot be done in output polynomial time
  - computing the embedding is NP-hard

Can we speed things up both theoretically and practically?

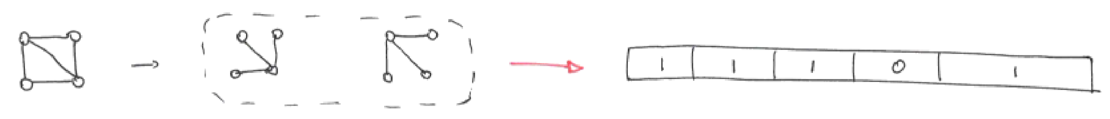
Probabilistic Subtree Kernels [1]

- Don't even try to mine cyclic patterns
- Forget about being exact

Mining:



Embedding:

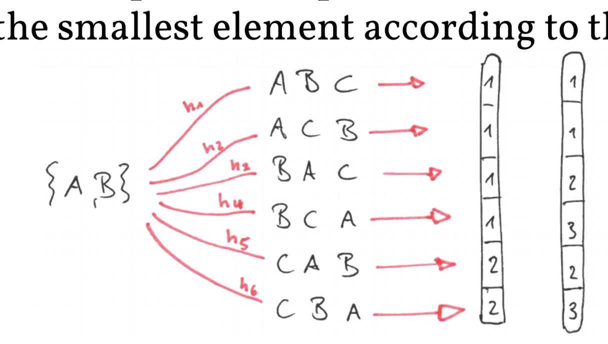


Jaccard Similarity and Min-Hashing [2]

- Jaccard Similarity (aka. Tanimoto Kernel) can be approximated using Min-Hashing

$$Jacc(A,B) = \frac{A \cap B}{A \cup B} = Prob_{h \in H}(h(A) = h(B))$$

- Each h in H corresponds to a permutation of the full feature set
- It returns the smallest element according to the permutation



Small sketch vectors containing Min-Hashes suffice

+ saves space
+ kernel can be computed fast

- normally, we need to know embedding for Min-Hashing to work
  → lots of subgraph isomorphism tests to run…

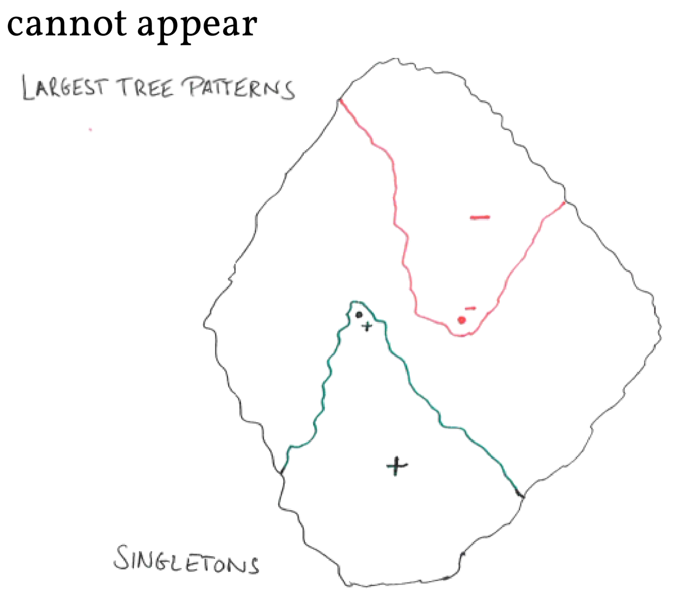
Can we do the embedding for unseen graphs more elegantly?

How can we use this algorithmically?

And does it work?

Additional Structure on Tree Patterns

If a subgraph of a pattern does not appear in a graph, then the pattern itself cannot appear ⇔ If a pattern appears in a graph, all its subgraphs must appear.



→ When computing the embedding of a graph, we do not need to test all patterns for subgraph isomorphism.

Min-Hash Sketching Algorithm

Input: graph G, directed graph $F = (\mathcal{F}, E)$ representing a poset $(\mathcal{F}, \preccurlyeq)$ and K permutations $\sigma_1, \ldots, \sigma_K$ of $\mathcal{F}$

Output: $\text{Sketch}_{\pi_1,\ldots,\pi_K}(G)$

init $sketch := [\perp, \ldots, \perp]$
init $state(T) := 0$ for all $T \in \mathcal{F}$
for $i = 1$ to $|\mathcal{F}|$ **do**
  for $j = 1$ to K **do**
    if $|\sigma_j| \geq i \wedge sketch[j] = \perp$ **then**
      if $state[\sigma_j[i]] \neq 0$ **then**
        if $state[\sigma_j[i]] = 1$ **then** $sketch[j] = \sigma_j[i]$
      else if $\sigma_j[i] \preccurlyeq G$ **then**
        $sketch[j] = \sigma_j[i]$
        for all $T' \in \mathcal{F}$ (including T) that can reach T in F **do**
          set $state(T') := 1$
      else
        for all $T' \in \mathcal{F}$ (including T) that are reachable from T in F **do**
          set $state(T') := -1$
return $sketch$

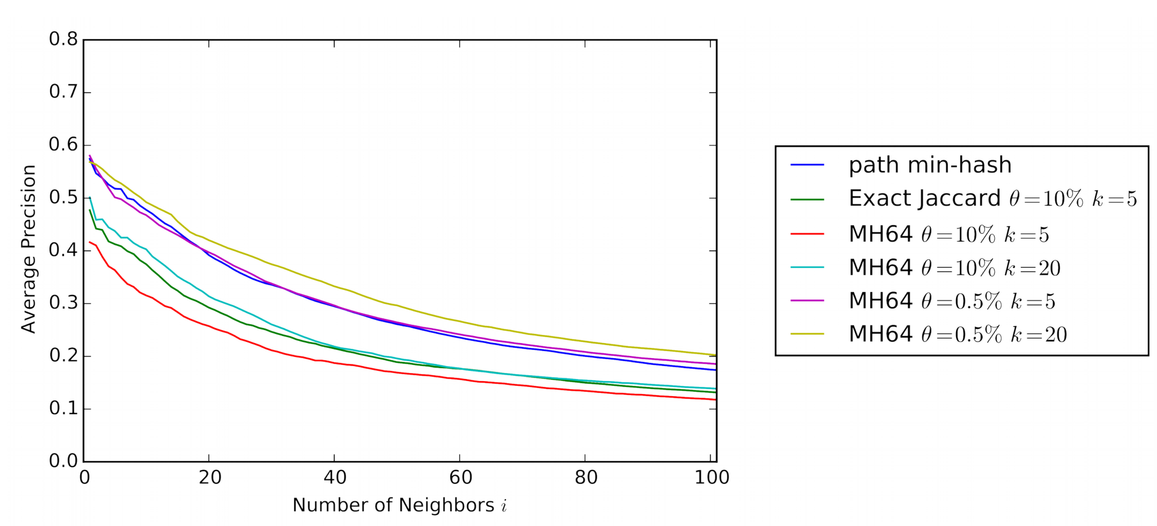
Reduction of Embedding Costs

Dataset	k	θ	$size(\mathcal{F})$	naive	MH32	MH64	MH128	MH256
MUTAG	5	10%	452	206.38	49.93	68.24	96.12	127.42
MUTAG	10	10%	543	244.11	42.77	63.77	90.57	125.39
MUTAG	15	10%	562	254.86	45.39	65.96	94.87	133.91
MUTAG	20	10%	573	260.18	55.34	76.32	105.15	135.11
PTC	5	10%	1,430	321.04	70.07	102.62	121.12	156.12
PTC	5	1%	9,619	734.79	236.31	327.27	475.35	611.92
PTC	10	10%	1,566	354.20	79.63	108.59	109.44	147.91
PTC	20	10%	1,712	376.65	17.60	25.81	31.49	39.62
DD	5	10%	8,111	3,547.22	260.47	486.09	846.09	1,374.76
DD	10	10%	18,137	6,670.93	317.82	568.23	1,072.58	1,936.42
DD	20	10%	33,100	11,005.49	344.59	653.66	1,242.03	2,190.15
NCI1	5	10%	1,819	431.19	89.12	137.75	185.22	221.21
NCI1	5	1%	21,306	900.68	615.62	920.17	1,227.52	1,378.18
NCI1	20	10%	2,441	557.70	115.07	183.54	220.14	255.58
NCI109	5	10%	2,182	462.62	115.62	170.43	206.23	254.70
NCI109	5	1%	19,099	886.06	532.38	727.15	1057.18	1,348.27
NCI109	20	10%	2,907	598.36	110.42	175.76	226.07	284.92

Table 1: Average number of subtree isomorphism test per graph for several datasets with varying number k of sampled spanning trees and frequency thresholds θ. The table reports $size(\mathcal{F})$ and the average number of subtree isomorphism tests evaluated by the naive method and by our algorithm for K =32,64,128,256 (last four columns).

Active Molecule Retrieval on NCI-HIV

- On a highly imbalanced dataset, we want to retrieve examples of the smaller class
- We are given a positive example as query



Predictive Performance

θ	Method	MUTAG	PTC	DD	NCI1	NCI109
10%	MH32	87.84	58.97	77.58	77.36	77.48
10%	MH64	87.73	58.68	79.91	78.04	79.54
10%	MH128	87.59	56.97	82.07	79.94	79.94
10%	MH256	87.78	57.18	83.58	80.76	81.72
10%	Jaccard	89.04	57.72	85.38	82.28	82.41
10%	PSK	84.22	54.17	84.67	79.09	78.05
10%	FSG	87.34	56.76	82.20	81.66	81.55
	HK	93.00	62.70	81.00	n/a	n/a

Table 2: AUC values for our method (MH) for sketch sizes K =32,64,128,256, k =5 spanning trees per graph, and frequency threshold θ=10% to obtain the feature set. "n/a" indicates that the autors of [3] did not provide results for the respective datasets.

[1] P. Welke, T. Horváth, and S. Wrobel. Probabilistic frequent subtree kernels. In NFMCP 2015, Springer LNCS 9607, pages 179–193, 2015.
[2] A. Z. Broder. On the resemblance and containment of documents. In Compression and Complexity of Sequences 1997. Proceedings, pages 21–29. IEEE, 1997.
[3] Q. Shi, J. Petterson, G. Dror, J. Langford, A. J. Smola, and S. V. N. Vishwanathan. Hash kernels for structured data. J. Mach. Learn. Res., 10:2615–2637, 2009.

universitätbonn